MINISTÉRIO DA CIÊNCIA, TECNOLOGIA, INOVAÇÕES E COMUNICAÇÕES
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

sid.inpe.br/mtc-m21b/2017/07.28.18.52-TDI

SENTINEL: AMBIENTE E-SCIENCE DE ALTO DESEMPENHO PARA GERENCIAMENTO E MONITORAMENTO DE APLICAÇÕES CIENTÍFICAS

Bruno Bustamante Ferreira Leonor

Tese de Doutorado do Curso de Pós-Graduação em Computação Aplicada, orientada pelos Drs. Reinaldo Roberto Rosa, e Walter Abrahão dos Santos, aprovada em 11 de agosto de 2017.

URL do documento original:
<http://urlib.net/8JMKD3MGP3W34P/3PBTFSH>

INPE
São José dos Campos
2017

PUBLICADO POR:

Instituto Nacional de Pesquisas Espaciais - INPE
Gabinete do Diretor (GB)
Serviço de Informação e Documentação (SID)
Caixa Postal 515 - CEP 12.245-970
São José dos Campos - SP - Brasil
Tel.:(012) 3208-6923/6921
E-mail: pubtc@inpe.br

COMISSÃO DO CONSELHO DE EDITORAÇÃO E PRESERVAÇÃO DA PRODUÇÃO INTELECTUAL DO INPE (DE/DIR-544):
Presidente:
Maria do Carmo de Andrade Nono - Conselho de Pós-Graduação (CPG)
Membros:
Dr. Plínio Carlos Alvalá - Centro de Ciência do Sistema Terrestre (CST)
Dr. André de Castro Milone - Coordenação de Ciências Espaciais e Atmosféricas (CEA)
Dra. Carina de Barros Melo - Coordenação de Laboratórios Associados (CTE)
Dr. Evandro Marconi Rocco - Coordenação de Engenharia e Tecnologia Espacial (ETE)
Dr. Hermann Johann Heinrich Kux - Coordenação de Observação da Terra (OBT)
Dr. Marley Cavalcante de Lima Moscati - Centro de Previsão de Tempo e Estudos Climáticos (CPT)
Silvia Castro Marcelino - Serviço de Informação e Documentação (SID)
BIBLIOTECA DIGITAL:
Dr. Gerald Jean Francis Banon
Clayton Martins Pereira - Serviço de Informação e Documentação (SID)
REVISÃO E NORMALIZAÇÃO DOCUMENTÁRIA:
Simone Angélica Del Ducca Barbedo - Serviço de Informação e Documentação (SID)
Yolanda Ribeiro da Silva Souza - Serviço de Informação e Documentação (SID)
EDITORAÇÃO ELETRÔNICA:
Marcelo de Castro Pazos - Serviço de Informação e Documentação (SID)
André Luis Dias Fernandes - Serviço de Informação e Documentação (SID)

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA, INOVAÇÕES E COMUNICAÇÕES
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

sid.inpe.br/mtc-m21b/2017/07.28.18.52-TDI

SENTINEL: AMBIENTE E-SCIENCE DE ALTO DESEMPENHO PARA GERENCIAMENTO E MONITORAMENTO DE APLICAÇÕES CIENTÍFICAS

Bruno Bustamante Ferreira Leonor

Tese de Doutorado do Curso de Pós-Graduação em Computação Aplicada, orientada pelos Drs. Reinaldo Roberto Rosa, e Walter Abrahão dos Santos, aprovada em 11 de agosto de 2017.

URL do documento original:
<http://urlib.net/8JMKD3MGP3W34P/3PBTFSH>

INPE
São José dos Campos
2017

Dados Internacionais de Catalogação na Publicação (CIP)

Leonor, Bruno Bustamante Ferreira.

L554s Sentinel: ambiente e-science de alto desempenho para gerenciamento e monitoramento de aplicações científicas / Bruno Bustamante Ferreira Leonor. – São José dos Campos : INPE, 2017.

xxiv + 175 p. ; (sid.inpe.br/mtc-m21b/2017/07.28.18.52-TDI)

Tese (Doutorado em Computação Aplicada) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2017.

Orientadores : Drs. Reinaldo Roberto Rosa, e Walter Abrahão dos Santos.

1. E-science. 2. HPC. 3. Containers. 4. Workflow científico. 5. Clima espacial. I.Título.

CDU 004.82:001.1

Esta obra foi licenciada sob uma Licença Creative Commons Atribuição-NãoComercial 3.0 Não Adaptada.

This work is licensed under a Creative Commons Attribution-NonCommercial 3.0 Unported License.

Aluno (a): ***Bruno Bustamante Ferreira Leonor***

Título: **"SENTINEL: AMBIENTE E-SCIENCE DE ALTO DESEMPENHO PARA GERENCIAMENTO E MONITORAMENTO DE APLICAÇÕES CIENTÍFICAS"**

Aprovado (a) pela Banca Examinadora em cumprimento ao requisito exigido para obtenção do Título de ***Doutor(a)*** em

Computação Aplicada

Dr. **Stephan Stephany**

Presidente / INPE / SJCampos - SP

Dr. **Reinaldo Roberto Rosa**

Orientador(a) / INPE / SJCampos - SP

Dr. **Walter Abrahão dos Santos**

Orientador(a) / INPE / São José dos Campos - SP

Dr. **Maurício Gonçalves Vieira Ferreira**

Membro da Banca / INPE / SJCampos - SP

Dr. **Eugênio Sper de Almeida**

Membro da Banca / INPE / São José dos Campos - SP

Dr. **Luis Fernando de Almeida**

Convidado(a) / UNITAU / Taubaté - SP

Dr. **Murilo da Silva Dantas**

Convidado(a) / IFSP.. / Jacareí - SP

Este trabalho foi aprovado por:

( ) maioria simples

(X) unanimidade

São José dos Campos, 11 de agosto de 2017

"Eu ainda estou aprendendo".

MICHELANGELO BUONARROTI

A minha família, namorada e amigos
companheiros de todas
as horas.

AGRADECIMENTOS

A meus pais João e Carmen Cinira, a minha namorada Paula, a minha avó Anna Maria e a minha irmã Anna Luiza por todo incentivo, apoio e compreensão pelo tempo de convívio muitas vezes sacrificado durante o desenvolvimento deste trabalho.

Aos meus orientadores e amigos Reinaldo Roberto Rosa e Walter Abrahão dos Santos pelo incentivo e conhecimentos compartilhados, pelos quais sou muito grato.

Ao Instituto Nacional de Pesquisas Espaciais (INPE) pela infraestrutura e a oportunidade de realizar meus estudos.

Aos colegas da CAP, principalmente aos que se tornaram amigos, pelo companheirismo durante esses anos.

Aos amigos do CPTEC pelo apoio incondicional.

A todas as pessoas que, direta ou indiretamente, contribuíram para a realização deste trabalho durante esses anos.

Aos membros da banca examinadora pelas considerações na defesa e consequentemente pela melhoria desta tese.

RESUMO

Mediante um paradigma de *e-Science*, a computação vem revolucionando o modo como diversas áreas científicas conduzem suas pesquisas. Para realizar simulações/pesquisas, existem diversos sistemas que lidam com uma enorme quantidade de variáveis que podem exigir vários cálculos e recursos computacionais de alto desempenho. Entretanto, há a necessidade de se lidar eventualmente com questões de interoperabilidade e execução de *workflows* científicos caso a caso. Motivada pelo crescimento explosivo do volume de dados, abordagens em *e-Science* vêm sendo adotadas como apoio na resolução de problemas práticos e de pesquisa. A proposta desta pesquisa é a criação de uma solução de *e-Science* chamada Sentinel que permite a pesquisadores o gerenciamento e monitoramento de experimentos científicos. Isto fornece mecanismos que possibilitam a inclusão de novas aplicações e total controle sobre todos os seus experimentos cadastrados, concentrando toda sua pesquisa em um só lugar. As características desejadas para o ambiente foram identificas a partir de um levantamento e comparação de diversas ferramentas utilizadas no meio científico e alguns apontamentos/críticas feitos por usuários. Através de um Portal *Web*, o Sentinel fornece um ambiente em nuvem para apoiar os pesquisadores em seus estudos que poderão ser acessados através de qualquer dispositivo habilitado para acessar a *web*. Para eximir do pesquisador a necessidade de conhecimento do ambiente computacional a ser utilizado, seus experimentos são executados em uma plataforma de conteinerização que trata os aspectos de interoperabilidade e concatenação de *workflows* científicos. A infraestrutura computacional utilizada para demonstração deste projeto é constituída basicamente de microcomputadores Raspberry Pi 3 model B. Esta solução é interessante pelo seu baixo custo e recursos oferecidos, podendo ser associados outros recursos de *hardware* sempre que necessário. Com um estudo de caso voltado ao Clima Espacial, o método DFA (Análise de Flutuação Destendenciada) foi aplicado às medidas de irradiação solar (TSI) obtidas entre 1978 e 2012. O Clima Espacial foi escolhido devido à sua relevância nas condições e nos processos que ocorrem no espaço que potencialmente afetam o ambiente próximo à Terra, seres humanos e seus ativos tecnológicos. No Brasil, os estudos nesta área são dirigidos pelo programa EMBRACE (Estudos e Monitoramento Brasileiro do Clima Espacial) do INPE (Instituto Nacional de Pesquisas Espaciais). Durante a execução dos experimentos criados dentro do estudo de caso proposto, o ambiente Sentinel apresentou-se consistente e estável entre as execuções locais e as conteinerizadas. Finalmente, a abordagem adotada é geral o bastante para ser estendida a outros domínios de aplicação uma vez que permite a definição prévia de um *workflow* científico.

Palavras-chave: *e-Science*. HPC. *Containers*. *Workflow* Científico. Clima Espacial.

SENTINEL: A HIGH PERFORMANCE E-SCIENCE ENVIRONMENT FOR MANAGEMENT AND MONITORING SCIENTIFIC APPLICATIONS

ABSTRACT

Through an e-Science paradigm computing has revolutionized the way in which diverse scientific areas conduct their research. To perform simulations/surveys there are several systems that deal with a huge amount of variables that may require various calculations and high performance computational resources. However, there is a need to eventually address issues of interoperability and execution of scientific workflows on a case-by-case. Motivated by the explosive growth of data volume, approaches in e-Science have been adopted as support in solving practical and research problems. The proposal of this research is the creation of an e-Science solution called Sentinel that allows researchers to manage and monitor scientific experiments. This provides mechanisms that allow the inclusion of new applications and total control over all your registered experiments, concentrating all your research in one place. The characteristics desired for the environment were identified from a survey and comparison of several tools used in the scientific environment and some notes/criticisms made by users. Through a Web Portal, Sentinel provides a cloud environment to support researchers in their studies that can be accessed through any device enabled to access the web. In order to exempt the researcher from the need to know the computational environment to be used, his experiments are executed in a containerization platform that deals with the interoperability and concatenation aspects of scientific workflows. The computational infrastructure used to demonstrate this project consists basically of Raspberry Pi 3 model B microcomputers. This solution is interesting because of its low cost and features offered, and other hardware resources can be associated whenever necessary. With a case study focused on the Space Weather, the DFA method was applied to the measurements of solar irradiation (TSI) obtained between 1978 and 2012. The Space Weather was chosen because of its relevance in the conditions and processes that occur in space that potentially affect the environment close to Earth, humans and their technological assets. In Brazil, studies in this area are conducted by the EMBRACE (Brazilian Space Weather Studies and Monitoring) program of INPE (National Institute of Space Research). During the execution of the experiments created within the proposed case study, the Sentinel environment was consistent and stable between local and containerized executions. Finally, the approach adopted is general enough to be extended to other application domains since it allows the prior definition of a scientific workflow.

Keywords: e-Science. HPC. Containers. Scientific Workflow. Space weather.

LISTA DE FIGURAS

Pág.

2.1 Relação existente entre Ciência da Computação X Engenharia de *Software* X *e-Science*. 5
2.2 Integração de áreas de pesquisa com *e-Science*. 6
2.3 Distribuição das pesquisas em *e-Science* no Estado de SP. 7
2.4 Visão geral de uma nuvem computacional. 10
2.5 Arquitetura em camadas de uma grade computacional. 12
2.6 Virtualização x Conteinerização. 15
2.7 *Framework* x Biblioteca. 16
2.8 Funcionamento *Framework* x Biblioteca. 17

3.1 Inovações em Comunicação Científica. 19
3.2 Estrutura padrão de diretórios de um projeto Maven. 28
3.3 Ferramentas de gerenciamento de projetos. 29
3.4 Servidores de aplicações *web*. 30
3.5 Ferramentas de conteinerização utilizadas com mais frequência. 30
3.6 Exemplo de arquitetura de um ambiente com Docker Swarm. 31
3.7 *Ranking* IEEE Spectrum das principais linguagens de programação - 2014. . . 32
3.8 Java *web frameworks* utilizados com mais frequência nos últimos anos. 34
3.9 Sistemas de Gerenciamento de Banco de Dados mais utilizados. 35
3.10 Número de *downloads* anuais do Flyway. 36
3.11 Visão superior de um Raspberry Pi 3 Model B. 42

4.1 Interface gráfica padrão do ambiente Sentinel. 45
4.2 Diagrama de Casos de Uso - Funcionalidades do pesquisador. 46
4.3 Fluxo de trabalho do Sentinel. 46
4.4 Interface gráfica `Nova Aplicação`. 47
4.5 Interface gráfica `Nova Aplicação` - guia `Argumentos`. 48
4.6 Interface gráfica `Nova Aplicação` - guia `Variáveis de Ambiente`. 49
4.7 Interface gráfica `Nova Aplicação` - guia `Libs`. 49
4.8 Interface gráfica `Nova Aplicação` - guia `Compilação`. 50
4.9 Interface gráfica `Nova Aplicação` - guia `Execução`. 51
4.10 Estrutura de diretórios de uma aplicação. 51
4.11 Imagem docker criada para a aplicação DFA. 53
4.12 Exemplo de Relatório de Aplicações cadastradas. 54
4.13 Interface gráfica `Novo Experimento`. 54
4.14 Interface gráfica `Configuração do Experimento`. 54

4.15 Estrutura de diretórios de um experimento. 55
4.16 Exemplo de Relatório de Experimentos criados. 55
4.17 `Dashboard` - Pesquisador 56
4.18 Diagrama de Máquina de Estados - *Status* do Experimento. 57
4.19 Execução do comando "docker ps" para listar todos os *containers* em execução. 58
4.20 Exibição de informações sobre o experimento. 59
4.21 Interface gráfica `Resultado do Experimento` - guia `Argumentos`. . 59
4.22 Interface gráfica `Resultado do Experimento` - guia `Visualização`. 60
4.23 Interface gráfica `Resultado do Experimento` - guia `Saídas`. . . . 61
4.24 Interface gráfica `Resultado do Experimento` - guia `Log`. 61
4.25 Interface gráfica `Resultado do Experimento` - guia `Publicações`. 62
4.26 Interface gráfica `Resultado do Experimento` - guia `Comentários`. 62
4.27 Diagrama de Casos de Uso - Administradores. 63
4.28 Interface gráfica `Cadastro de Organização`. 63
4.29 Interface gráfica `Organizações`. 64
4.30 Exemplo de Relatório de Organizações cadastradas. 64
4.31 Interface gráfica `Cadastro de Usuário`. 65
4.32 Interface gráfica `Usuários`. 66
4.33 Exemplo de Relatório de Usuários cadastrados. 66
4.34 `Dashboard` - Administrador Geral do Sistema. 67

5.1 Equipamentos para coleta de dados. 70
5.2 Série temporal da variável TSI do experimento VIRGO/SOHO. 72
5.3 Aplicação do método DFA na série temporal da variável TSI. 72
5.4 Janelas da série temporal TSI x Resultado da aplicação do método DFA. . . . 73
5.5 Comparativo de tempo das execuções nos casos 1 e 2. 75
5.6 Comparativo de tempo das execuções nos casos 3 e 4. 75
5.7 Comparativo de tempo das execuções nos casos 5 e 6. 76
5.8 Comparativo de tempo das execuções nos casos 1, 2, 3, 4, 5 e 6. 76

A.1 Componentes utilizados na construção do *cluster*. 93
A.2 Projeto CAD do *cluster*. 94
A.3 *Cluster* construído. 95
A.4 Ferramenta de configuração do Raspberry Pi. 96

B.1 Diagrama de Casos de Uso. 101
B.2 Diagrama de Classes. 117
B.3 Diagrama de Implantação. 118

C.1 Série Temporal Total Solar Irradiance (TSI). 119
C.2 Série Temporal TSI - Janela 1978. 120

C.3 Série Temporal TSI - Janela 1979. 121
C.4 Série Temporal TSI - Janela 1980. 122
C.5 Série Temporal TSI - Janela 1981. 123
C.6 Série Temporal TSI - Janela 1982. 124
C.7 Série Temporal TSI - Janela 1983. 125
C.8 Série Temporal TSI - Janela 1984. 126
C.9 Série Temporal TSI - Janela 1985. 127
C.10 Série Temporal TSI - Janela 1986. 128
C.11 Série Temporal TSI - Janela 1987. 129
C.12 Série Temporal TSI - Janela 1988. 130
C.13 Série Temporal TSI - Janela 1989. 131
C.14 Série Temporal TSI - Janela 1990. 132
C.15 Série Temporal TSI - Janela 1991. 133
C.16 Série Temporal TSI - Janela 1992. 134
C.17 Série Temporal TSI - Janela 1993. 135
C.18 Série Temporal TSI - Janela 1994. 136
C.19 Série Temporal TSI - Janela 1995. 137
C.20 Série Temporal TSI - Janela 1996. 138
C.21 Série Temporal TSI - Janela 1997. 139
C.22 Série Temporal TSI - Janela 1998. 140
C.23 Série Temporal TSI - Janela 1999. 141
C.24 Série Temporal TSI - Janela 2000. 142
C.25 Série Temporal TSI - Janela 2001. 143
C.26 Série Temporal TSI - Janela 2002. 144
C.27 Série Temporal TSI - Janela 2003. 145
C.28 Série Temporal TSI - Janela 2004. 146
C.29 Série Temporal TSI - Janela 2005. 147
C.30 Série Temporal TSI - Janela 2006. 148
C.31 Série Temporal TSI - Janela 2007. 149
C.32 Série Temporal TSI - Janela 2008. 150
C.33 Série Temporal TSI - Janela 2009. 151
C.34 Série Temporal TSI - Janela 2010. 152
C.35 Série Temporal TSI - Janela 2011. 153
C.36 Série Temporal TSI - Janela 2012. 154

LISTA DE TABELAS

Pág.

3.1 Etapas x Tarefas. 20
3.2 *Workflow* de pesquisa x Ferramentas utilizadas. 21
3.3 Tecnologias adotadas no projeto em cada uma das categorias. 27
3.4 Relação entre Modelo Relacional e Modelo Orientado a Objetos. 37
3.5 Configuração dos modelos Raspberry Pi utilizados. 43
3.6 Configuração final dos dispositivos Raspberry Pi. 44
3.7 Equipamentos utilizados na construção do *cluster*. 44

4.1 Descrição das colunas da interface gráfica `Dashboard` - Pesquisador. 56
4.2 Descrição dos possíveis *Status* que um experimento pode assumir. 57
4.3 Comandos para gerenciar os experimentos. 58

5.1 Equipamentos para captação dados utilizados pelo EMBRACE. 70
5.2 Expoente α de cada janela. 74
5.3 Tempo de execução e *Computing Throughput* de cada um dos casos apresentados. 77

6.1 Comparação: Ferramentas identificadas x Características esperadas. 79

B.1 Descrição do Caso de Uso Registrar Aplicação 102
B.2 Descrição do Caso de Uso Registrar Experimento 103
B.3 Descrição do Caso de Uso Manter Usuário 104
B.4 Descrição do Caso de Uso Manter Organização 105
B.5 Descrição do Caso de Uso Configurar Experimento 106
B.6 Descrição do Caso de Uso Iniciar Execução do Experimento 107
B.7 Descrição do Caso de Uso Parar Execução do Experimento 108
B.8 Descrição do Caso de Uso Visualizar Experimento 109
B.9 Descrição do Caso de Uso Validar Resultado do Experimento 110
B.10 Descrição do Caso de Uso Rejeitar Resultado do Experimento 111
B.11 Descrição do Caso de Uso Compartilhar Resultado 112
B.12 Descrição do Caso de Uso Download Resultados do Experimento 113
B.13 Descrição do Caso de Uso Excluir Experimento 114
B.14 Descrição do Caso de Uso Comentar Experimento 115
B.15 Descrição do Caso de Uso Registrar Publicação 116

LISTA DE CÓDIGOS

3.1 Arquivo POM de configuração do projeto Sentinel. 27
3.2 Exemplo de mapeamento objeto relacional. 37
3.3 *Template* básico para uso do framework Bootstrap. 39
3.4 *Template* Thymeleaf para renderizar as páginas do ambiente Sentinel. . . 40
4.1 Configuração da linguagem Python3. 50
4.2 *Template* do arquivo Dockerfile para criação das imagens. 52
4.3 Dockerfile para criação da imagem da aplicação DFA. 53

LISTA DE ABREVIATURAS E SIGLAS

API	–	*Application Programming Interface*
BDA	–	*Brazilian Decimetric Array*
CSS	–	*Cascading Style Sheets*
CSV	–	*Comma-separated values*
DFA	–	*Detrended Fluctuation Analysis*
DI	–	*Dependency Injection*
DNA	–	*Deoxyribonucleic Acid*
EMBRACE	–	Estudos e Monitoramento Brasileiro do Clima Espacial
GPA	–	*Gradient Pattern Analysis*
GIC	–	*Geomagnetically Induced Currents*
GPS	–	*Global Positioning System*
HPC	–	*High-Performance Computing*
HTML	–	*HyperText Markup Language*
HTTP	–	*HyperText Transfer Protocol*
INPE	–	Instituto Nacional de Pesquisas Espaciais
IoC	–	*Inversion of Control*
Java EE	–	*Java Enterprise Edition*
JPA	–	*Java Persistence* API
MVC	–	*Model View Controller*
ORM	–	*Object Relational Mapper*
OSG	–	Open Science Grid
OV	–	Organização Virtual
PDF	–	*Portable Document Format*
POM	–	*Project Object Model*
PSD	–	*Power Spectral Density*
SaaS	–	*Software as a Service*
SGBD	–	Sistema de Gerenciamento de Banco de Dados
SMS	–	Supervisor Monitoring Scheduler
SQL	–	*Structured Query Language*
SVG	–	*Scalable Vector Graphics*
SOA	–	*Service-Oriented Architecture*
TSI	–	*Total Solar Irradiance*
UFRGS	–	Universidade Federal do Rio Grande do Sul
VBA	–	*Visual Basic for Applications*
VHF	–	*Very High Frequency*
XLS	–	*Excel Spreadsheet*
WfMS	–	*Workflow Management Systems*
XML	–	*Extensible Markup Language*
WSDL	–	*Web Service Description Language*
WWW	–	*World Wide Web*

SUMÁRIO

Pág.

1 INTRODUÇÃO 1
1.1 Objetivos 4
1.2 Organização do Documento 4

2 CONCEITUAÇÃO 5
2.1 *e-Science* 5
2.1.1 e-Infraestrutura para *e-Science* 7
2.1.1.1 Ontologias 8
2.1.1.2 *Web Service* 8
2.1.1.3 SOA - *Service-oriented architecture* 9
2.1.1.4 SaaS - *Software as a Service* 9
2.1.1.5 *Cloud Computing* 9
2.1.1.6 *Grid Computing* 11
2.2 HPC - *High-Performance Computing* 12
2.3 Ciência Aberta 13
2.4 Portal 14
2.5 Conteinerização 14
2.6 *Framework* 15

3 METODOLOGIA 19
3.1 *Workflow* de pesquisa 19
3.1.1 A Etapa de Análise de Dados 22
3.2 Estudo das ferramentas de *e-Science* 23
3.2.1 Excel 23
3.2.2 Google Drive 23
3.2.3 GridUnesp 24
3.2.4 The Jupyter Notebook 24
3.2.5 nanoHub 24
3.2.6 Open Science Grid (OSG) 24
3.2.7 Profager 24
3.2.8 R 25
3.2.9 ROpenSci 25
3.2.10 RTGrid 25

3.2.11 Supervisor Monitoring Scheduler (SMS) / ECFlow . . . 25
3.2.12 SPiNMe . . . 25
3.2.13 SPSS . . . 26
3.2.14 VLADA . . . 26
3.2.15 Zooniverse . . . 26
3.3 Infraestrutura de *Software* . . . 26
3.3.1 Maven . . . 27
3.3.2 Apache Tomcat . . . 29
3.3.3 Docker . . . 30
3.3.4 Docker Swarm . . . 31
3.3.5 Java . . . 32
3.3.6 Spring . . . 33
3.3.7 Spring Framework . . . 33
3.3.7.1 Spring MVC . . . 33
3.3.7.2 Spring Data . . . 34
3.3.7.3 Spring Security . . . 34
3.3.8 Apache Velocity . . . 35
3.3.9 MySQL . . . 35
3.3.10 Flyway . . . 36
3.3.11 Hibernate . . . 37
3.3.12 HTML5 . . . 38
3.3.13 Bootstrap . . . 38
3.3.14 Thymeleaf . . . 39
3.3.15 D3.js . . . 41
3.3.16 JasperReports . . . 42
3.4 Infraestrutura de *Hardware* . . . 42
3.4.1 Raspberry Pi . . . 42
3.4.2 *Cluster* . . . 43

4 O AMBIENTE SENTINEL . . . 45
4.1 Requisitos do Sistema . . . 45
4.2 Fluxo de Trabalho . . . 46
4.2.1 Etapa 1 - Registro da Aplicação . . . 47
4.2.2 Etapa 2 - Criação e Configuração do Experimento . . . 54
4.2.3 Etapa 3 - Gerência e Monitoramento do Experimento . . . 56
4.2.4 Etapa 4 - Visualização de Resultado . . . 58
4.3 Administração do Sistema . . . 63
4.3.1 Cadastro de Organização . . . 63

4.3.2 Cadastro de Usuário 64
4.3.3 Dashboard - Administração do Ambiente Sentinel 66

5 ESTUDO DE CASO 69
5.1 Clima Espacial 69
5.1.1 Equipamentos 70
5.2 O método DFA 71
5.3 Fonte dos Dados 71
5.4 Resultados obtidos 72

6 CONCLUSÕES 79
6.1 Contribuições 81
6.2 Trabalhos Futuros 82

REFERÊNCIAS BIBLIOGRÁFICAS 83

APÊNDICE A - MONTAGEM E CONFIGURAÇÃO DO *CLUSTER* RPi . . . 93
A.1 Montagem do *cluster* 93
A.2 Instalação do Sistema Operacional 95
A.3 Configurando o Raspbian 96
A.4 Configuração do endereço IP 96
A.5 Mapeando os *hosts* conhecidos 97
A.6 Gerando chaves públicas SSH 97
A.7 Instalação do Docker 98
A.8 Montagem do *cluster* com Docker Swarm 99

APÊNDICE B - DETALHAMENTO DA MODELAGEM DO SISTEMA. . . . 101
B.1 Diagrama de Casos de Uso 101
B.1.1 Descrição dos Casos de Uso 102
B.1.1.1 Caso de Uso Registrar Aplicação 102
B.1.1.2 Caso de Uso Registrar Experimento 103
B.1.1.3 Caso de Uso Manter Usuário 104
B.1.1.4 Caso de Uso Manter Organização 105
B.1.1.5 Caso de Uso Configurar Experimento 106
B.1.1.6 Caso de Uso Iniciar Execução do Experimento 107
B.1.1.7 Caso de Uso Parar Execução do Experimento 108
B.1.1.8 Caso de Uso Visualizar Experimento 109
B.1.1.9 Caso de Uso Validar Resultado do Experimento 110
B.1.1.10 Caso de Uso Rejeitar Resultado do Experimento 111

B.1.1.11 Caso de Uso Compartilhar Experimento 112
B.1.1.12 Caso de Uso Download Resultados do Experimento 113
B.1.1.13 Caso de Uso Excluir Experimento 114
B.1.1.14 Caso de Uso Comentar Experimento 115
B.1.1.15 Caso de Uso Registrar Publicação 116
B.2 Diagrama de Classes 117
B.3 Diagrama de Implantação 118

APÊNDICE C - RESULTADOS COMPLEMENTARES DFA 119

APÊNDICE D - ARTIGO NSC 155

APÊNDICE E - ARTIGO CCIS. 159

APÊNDICE F - ARTIGO REVISTA DNC 161

1 INTRODUÇÃO

A computação tem exercido um papel fundamental na produção científica possibilitando simulações computacionais, controle de experimentos e/ou filtragem de dados nas mais diferentes áreas de pesquisa (TOLLE et al., 2011).

Utilizadas desde 1950, as simulações computacionais têm auxiliado na exploração de áreas inacessíveis da ciência (ALISSON, 2013). No início, os cientistas não conheciam o real poder da ciência da computação e os profissionais da computação não entendiam a complexidade dos problemas científicos.

A presença cada vez mais marcante de *softwares* tem desempenhado papel fundamental na produção científica, seja transformando um modelo teórico em simulações, controlando experimentos ou filtrando dados para serem analisados (RAINERE, 2013).

Essa influência da computação na forma como pesquisas científicas vêm sendo conduzidas faz com que seja crescente a necessidade por poder de processamento e armazenamento, consequência da complexidade das atuais aplicações e sistemas.

Essa relação também tem dado espaço para o desenvolvimento de novos paradigmas na computação, o que tem possibilitado o desenvolvimento e a criação de diversas infraestruturas (também conhecidas por e-Infraestruturas) para integrar equipamentos cada vez mais poderosos com experimentos de diversas áreas da ciência. Essas infraestruturas se baseiam na utilização de computação, usualmente paralela, na resolução de problemas mais rapidamente do que seria possível através de sistemas individuais.

A evolução da computação, na sua história mais recente, não se prende somente à evolução do *hardware*, mas também ao crescimento das linguagens de programação. A evolução do *hardware* foi fundamental para o desenvolvimento, mas incapaz de justificar tamanha difusão. A evolução das linguagens de computação - do Assembler à Java - apresenta a distinção entre os paradigmas de programação imperativas e declarativas, os aprimoramentos na arquitetura, os avanços dos sistemas operacionais; enfim, a consolidação da Computação como Ciência (FILHO, 2007).

Com a constante evolução sofrida pelo *hardware*, computadores cada vez mais potentes têm surgido para auxiliar tanto nos processos de cálculos aritméticos como nas tarefas repetitivas devido à necessidade de resultados rápidos e precisos nas mais diferentes áreas da humanidade, na medida em que os cálculos foram se complicando e aumentando de tamanho (CARDI, 2002). Neste cenário, aplicações nas mais diversas áreas têm se beneficiado.

O desenvolvimento da computação vem revolucionando o modo como diversas áreas da ciência estão sendo conduzidas. A ciência tem evoluído constantemente nos dias atuais graças à sua integração com a computação, desenvolvendo pesquisas de forma mais rápida e eficiente (PALAZZI et al., 2009).

Toda essa evolução tem permitido que haja uma colaboração global no acompanhamento de pesquisas científicas, colaboração esta que recebe o nome de *e-Science*. Fundamentalmente, "um ambiente de *e-Science* deve permitir o compartilhamento de recursos coordenados em larga escala entre comunidades, indivíduos, grupos, laboratórios e instituições, permitindo o gerenciamento cooperativo de facilidades (equipamentos, instrumentos, experimentos, entre outros) e análise colaborativa de produtos (dados) oriundos dessas facilidades" (LAUSCHNER, 2005).

De acordo com Tony Hey (apud ALISSON, 2013), "algumas tecnologias utilizadas na ciência da computação vão ajudar a resolver problemas científicos. Em contrapartida, a utilização dessas ferramentas para solucionar problemas científicos também possibilitará o próprio desenvolvimento da ciência da computação".

Consequentemente, desde seu surgimento, a *World Wide Web* (www), acompanhando essa evolução, tem se desenvolvido gradualmente de um sistema de documentos em hipermídias interligados para um universo em crescimento de páginas e de aplicativos. Em Tait (2007), a autora discute a importância da *internet* nos dias atuais, que faz parte da rotina de trabalho, do estudo e da vida das pessoas.

"Com o tempo, as tecnologias da *web* evoluíram para proporcionar aos desenvolvedores a capacidade de criar novas gerações de experiências úteis e imersivas. A *web* atual é resultado dos esforços contínuos de uma comunidade aberta que ajuda a definir essas tecnologias e a garantir que elas sejam suportadas em todos os navegadores" (WEB, 2012).

Acompanhando a evolução da *web*, portais temáticos começaram a ser adotados como forma de disseminar os conhecimentos gerados pelas pesquisas desenvolvidas. Portais são *sites* que agregam informações, aplicações e serviços relevantes ao tema do Portal, por meio de uma única interface. Eles representam uma resposta direta à amplitude e complexidade das informações disponíveis *on-line* (TATNALL, 2005).

Com o crescente número de serviços informatizados que fazem parte das tarefas cotidianas oferecidos pela *web*, novas exigências vão surgindo a partir das necessidades dos usuários. Pesquisas científicas tendem cada vez mais a lidar com grandes volumes de dados ou utilizar métodos computacionais sofisticados, resultado da evolução das aplicações/experimentos

e do grande volume de dados gerados diariamente.

Atualmente várias pesquisas e *softwares* são construídos em cima de ferramentas proprietárias, são direcionados para solucionar problemas específicos e muitos dos *softwares* não se encontram disponíveis publicamente. Esses fatores podem interferir na revisão e reprodução da pesquisa por outros interessados e na divulgação/comunicação de suas descobertas para a sociedade (RAINERE, 2013).

A motivação do trabalho advém da necessidade cada vez maior de acesso a recursos computacionais de alto desempenho por parte dos pesquisadores para realizar seus estudos dado o grande volume de dados gerados e a complexidade das aplicações desenvolvidas.

Durante a execução dessa pesquisa, diversas aplicações e ambientes de pesquisa foram analisados a fim de identificar quais pontos/características deveriam ser abordados no desenvolvimento do ambiente Sentinel. Nesta fase, foram identificadas oito características desejáveis para o ambiente. Foi identificada também a falta de padrão na composição dos ambientes utilizados, sendo este, considerado outro fator a influenciar na replicabilidade de uma pesquisa científica.

A proposta deste trabalho é apresentar um ambiente de *e-Science* que permita que pesquisadores concentrem em um único local toda sua pesquisa, podendo realizar o gerenciamento e o monitoramento de cada uma das suas aplicações científicas. Esta solução se apoia na construção de um portal *web* que permita a integração de aplicações utilizadas em pesquisas nos mais diversos domínios.

A solução abordada envolve o uso de *containers* para a execução dos experimentos, buscando padronizar o ambiente de execução. *Containers* permitem portabilidade de aplicações, isolamento de processos, prevenção de violação externa e gerenciamento de consumo de recursos (VITALINO; CASTRO, 2016).

O estudo de caso produzido nessa pesquisa é voltado ao Clima Espacial. Este domínio foi adotado devido à sua relevância nas condições e nos processos que ocorrem no espaço que potencialmente afetam o ambiente próximo à Terra, seres humanos e seus ativos tecnológicos. Durante o desenvolvimento da pesquisa, inúmeros experimentos utilizando o método DFA (Análise de Flutuação Destendenciada) aplicado às medidas de irradiação solar (TSI) foram executados. No Brasil, esses estudos são coordenados pelo programa EMBRACE (Estudos e Monitoramento Brasileiro do Clima Espacial) do INPE (Instituto Nacional de Pesquisas Espaciais).

Durante as execuções criadas no estudo de caso proposto, o ambiente Sentinel apresentou-

se consistente e estável. Oferecendo aos pesquisadores liberdade na escolha das linguagens de programação e sistema operacional a ser utilizado em suas pesquisas, além de permitir que suas próprias aplicações sejam acopladas ao ambiente, o Sentinel apresentou-se como uma abordagem geral o bastante para ser estendida a outros domínios de aplicação.

1.1 Objetivos

O objetivo desta pesquisa é propor e desenvolver um ambiente capaz de:

- Oferecer um ambiente padronizado para execução de experimentos visando à portabilidade das aplicações.
- Criar um ponto único para concentrar toda a pesquisa, e
- Permitir o compartilhamento dos produtos produzidos com a execução dos experimentos científicos, buscando apoiar a disseminação do conhecimento produzido pela pesquisa realizada.

1.2 Organização do Documento

Este trabalho está organizado em seis capítulos, sendo este o primeiro. A definição dos conceitos utilizados no desenvolvimento deste trabalho é apresentada no Capítulo 2. O Capítulo 3 apresenta a metodologia utilizada na identificação das principais características desejadas para o ambiente Sentinel, assim como todas as tecnologias utilizadas no desenvolvimento e operacionalização do ambiente proposto nesta tese. O ambiente desenvolvido para gerenciar e monitorar a execução de aplicações científica, chamado Sentinel, é apresentado no Capítulo 4. O estudo de caso e os resultados obtidos com a execução dos experimentos criados são apresentados e discutidos no Capítulo 5. As conclusões e sugestões para trabalhos futuros são apresentadas no Capítulo 6.

Um roteiro para construção e configuração de um *cluster* similar ao utilizado nessa pesquisa é apresentado no Apêndice A. No Apêndice B estão os diagramas construídos durante o trabalho de engenharia de *software* para modelagem dos requisitos do ambiente Sentinel, que conta com o Diagrama de Casos de Uso e suas respectivas descrições e o Diagrama de Classes. Todos os resultados obtidos com os experimentos executados durante este trabalho são apresentados no Apêndice C. O material relacionado às publicações e participação em eventos durante o desenvolvimento desta pesquisa estão nos Apêndices D , E e F.

2 CONCEITUAÇÃO

Neste capítulo será apresentada uma revisão abordando conceitos envolvidos no desenvolvimento deste trabalho.

2.1 *e-Science*

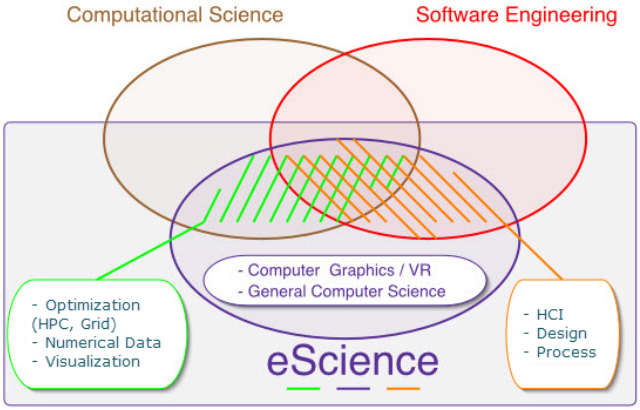
O termo *e-Science* (ou e-Ciência no Brasil) foi criado para se referir à ciência de colaboração global e às infraestruturas que permitem essa colaboração.

Pesquisas em *e-Science* estão cada vez mais ganhando importância no cenário nacional e internacional. No entanto, existem ainda poucos grupos de Engenharia de *Software* e, mais especificamente, em ontologias que focam suas pesquisas em *e-Science* (PALAZZI et al., 2009).

Algumas características de *e-Science* citadas por Cardoso (2007) são: (1) o acesso a uma vasta coleção de dados, (2) utilização de recursos computacionais em larga escala, (3) utilização de recursos de múltiplas organizações e (4) *workflows*.

Hoje existe uma estreita relação entre as áreas de ciência da computação, engenharia de *software* e *e-Science*, que pode ser vista na Figura 2.1. Algumas tecnologias utilizadas na ciência da computação têm auxiliado na resolução de problemas científicos em diversas áreas de pesquisa. A busca por resultados precisos e cada vez mais rápidos vêm contribuindo com o desenvolvimento da computação.

Figura 2.1 - Relação existente entre Ciência da Computação X Engenharia de *Software* X *e-Science*.



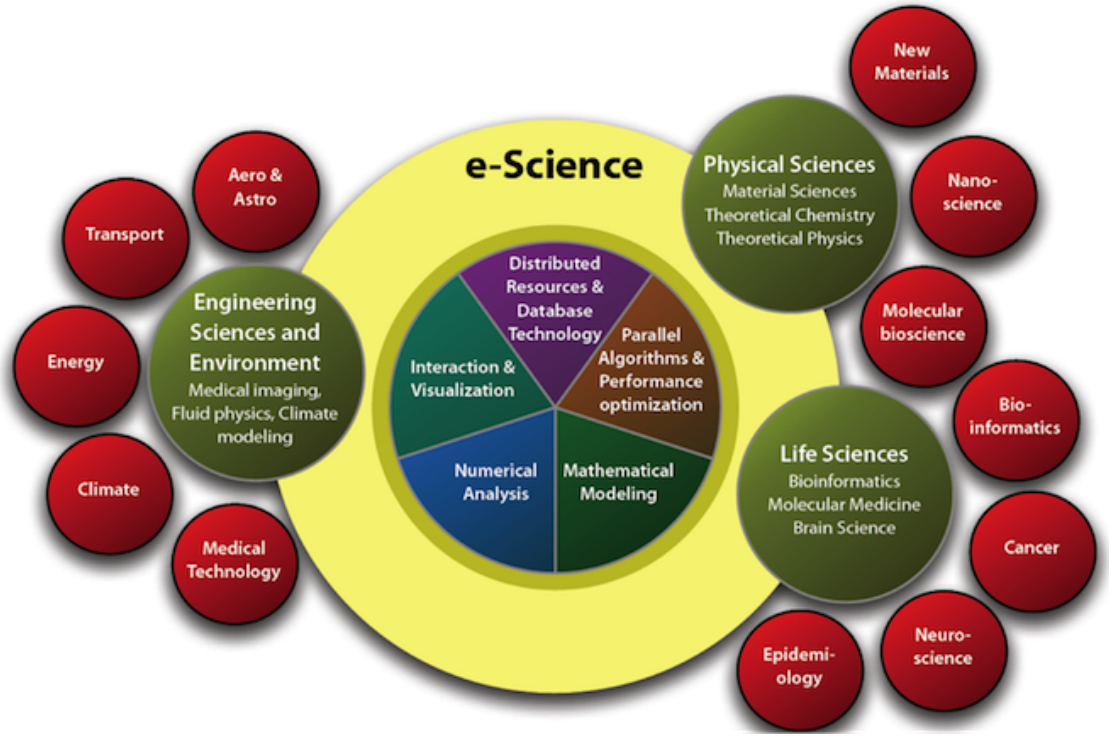
Fonte: Strazdins (2002).

Em Alisson (2013) é abordada a necessidade da realização de um trabalho conjunto, de longo prazo, para que computação e ciência entendessem qual era a real contribuição que cada uma poderia dar em suas respectivas áreas, e iniciar o desenvolvimento de novos algoritmos, *hardware*, *software* e da programação de linguagens para possibilitar a realização de experimentos em diversas áreas.

Atualmente existem grandes expectativas em relação à *e-Science*. "Se soubermos utilizá-la adequadamente, ela poderá trazer grandes avanços não só em pesquisas mas também na própria maneira de se fazer ciência" (ALISSON, 2013).

A área de *e-Science*, também conhecida como ciência orientada por dados, integra pesquisas em computação a estudos em diversas, como Física, Química, Engenharias, Humanidades e Saúde. Isso ocorre mediante o desenvolvimento conjunto de *softwares* específicos para visualização e análise de informações coletadas e armazenadas nesses projetos. A integração permite a interpretação dos dados, formulação de teorias, testes por simulação e o levantamento de novas hipóteses de pesquisa com base em eventos difíceis de serem observadas sem o apoio da tecnologia da informação (FAPESP, 2005). A Figura 2.2 pode ser vista como um exemplo da integração das mais diversas áreas de pesquisa pela *e-Science*.

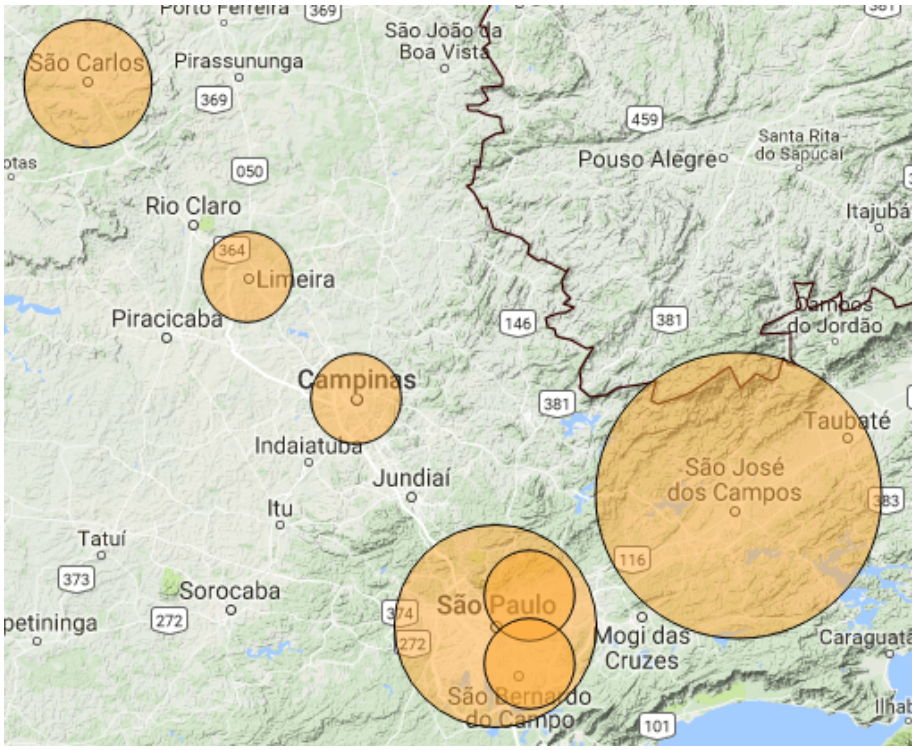
Figura 2.2 - Integração de áreas de pesquisa com *e-Science*.



Fonte: Polis (2013).

Atualmente, 7 projetos em *e-Science* estão em andamento no país (FAPESP, 2017). O mapa apresentado na Figura 2.3 representa como esses recursos para *e-Science* estão distribuídos no Estado de São Paulo.

Figura 2.3 - Distribuição das pesquisas em *e-Science* no Estado de SP.



Fonte: FAPESP (2017).

Ao longo dos anos a ciência vem buscando por tecnologias que ajudem a decifrar e dimensionar o mundo. Como resultado, diariamente são gerados *Terabytes* de dados em laboratórios, estações remotas e institutos de pesquisa. Com uma quantidade cada vez maior de dados existe a necessidade de aproveitá-los de forma efetiva, o que tem contribuído com o avanço das tecnologias existentes e com o surgimento de novas tecnologias (TOLLE et al., 2011).

2.1.1 e-Infraestrutura para *e-Science*

Diversas e-Infraestruturas têm surgido para possibilitar a realização de experimentos em diversas áreas da ciência mediante a realização de um trabalho conjunto entre profissionais de computação e cientistas. Com o entendimento de qual era a contribuição que cada um poderia oferecer em suas respectivas áreas, teve início o desenvolvimento de novos algoritmos, *hardware*, *software* e de linguagens de programação para possibilitar a realização de experimentos em diversas áreas.

Essas tecnologias têm se apresentado como uma solução viável para a execução remota de funcionalidades. As funcionalidades de um sistema são tarefas que o usuário pode executar através do sistema (registrar aplicação, configurar experimento, etc.). Algumas das propriedades que tornam isso possível são: (1) a independência dos sistemas operacionais e das linguagens de programação, (2) a interoperabilidade, (3) a ubiquidade e (4) a possibilidade de desenvolvimento de sistemas menos acoplados (PALAZZI et al., 2009).

A seguir, serão apresentadas algumas tecnologias que permitem/possibilitam a realização de experimentos nas mais diferentes áreas da ciência.

2.1.1.1 Ontologias

Foster e Kesselman (2004) definem ontologia como uma estrutura para organizar conhecimento. Em Ciência da Computação, ontologias são desenvolvidas para facilitar o compartilhamento e reuso de informações (DAVIES et al., 2003).

Uma ontologia é um modelo de dados que representa um conjunto de conceitos, relações, restrições e axiomas dentro de um domínio, utilizada para realizar inferência sobre os objetos do domínio.

Ontologias podem ser aplicadas em diversos contextos dentro da *e-Science*, tais como: (1) auxiliar na troca de informações entre aplicações científicas que trabalham em domínios correlatos, (2) permitir a descoberta semântica de serviços *web* e (3) auxiliar na composição de serviços *web* em *workflows* científicos (PALAZZI et al., 2009).

2.1.1.2 *Web Service*

Web services representam a tendência mais recente na computação distribuída sendo independente de plataforma. Eles se concentram nos protocolos utilizados para trocar mensagens e não na implementação que suporta esses protocolos. Um *web service* pode ser construído em qualquer plataforma utilizando qualquer linguagem de programação da maneira que quiser (BURKE; MONSON-HAEFEL, 2007). Um *web service* expõe sua interface para os usuários usando um documento *eXtensible Markup Language* (XML), conhecido como *Web Service Description Language* (WSDL). "Usando um WSDL é possível descobrir quais são os tipos de dados, formatos de mensagens e serviços disponibilizados por um *web service*" (SAMPAIO, 2003).

Serviços podem ser classificados como: (1) Utilitários - que fornecem uma funcionalidade de uso geral e (2) Negócios - que implementam parte de um processo de negócios ou serviços de coordenação que executam outros serviços.

2.1.1.3 SOA - *Service-oriented architecture*

As Arquiteturas Orientadas a Serviços (SOA) são uma forma de desenvolvimento de sistemas distribuídos em que os componentes de um sistema são serviços autônomos, executando em computadores geograficamente distribuídos (SOMMERVILLE, 2011).

Segundo Sommerville (2011), SOA é uma abordagem para a estruturação de um sistema de *software* como um conjunto de serviços separados, sem estado. Estes podem ser fornecidos por vários provedores e podem ser distribuídos. Normalmente, tratam-se de transações curtas, em que um serviço é chamado, faz alguma coisa e, em seguida, retorna um resultado.

2.1.1.4 SaaS - *Software as a Service*

O SaaS é uma forma de fornecer funcionalidade em um servidor remoto com acesso de clientes por meio de um *web browser*. O servidor mantém os dados e o estado do usuário durante uma sessão de interação. Geralmente, as transações são longas (SOMMERVILLE, 2011).

Os custos de gerenciamento do *software* são transferidos para o provedor, responsável pela correção de *bugs* e instalação das atualizações de *software*, pelas alterações de plataforma de sistema operacional e pela garantia de que a capacidade do *hardware* possa atender à demanda.

Ao implementar o SaaS, é preciso considerar que pode haver usuários do *software* de várias organizações diferentes. São três os fatores que precisam ser considerados (SOMMERVILLE, 2011): (1) Configurabilidade - como você configura o *software* para as necessidades específicas de cada organização?, (2) Multilocação - como você apresenta para cada usuário do *software* a impressão de que eles estão trabalhando com sua própria cópia do sistema enquanto, e ao mesmo tempo, fazem uso eficiente dos recursos do sistema? e (3) Escalabilidade - como você projeta o sistema para que ele possa ser dimensionado a fim de acomodar um número imprevisível de usuários?

2.1.1.5 *Cloud Computing*

Com o avanço da sociedade humana moderna, serviços básicos e essenciais são quase todos entregues de uma forma transparente a seus usuários.

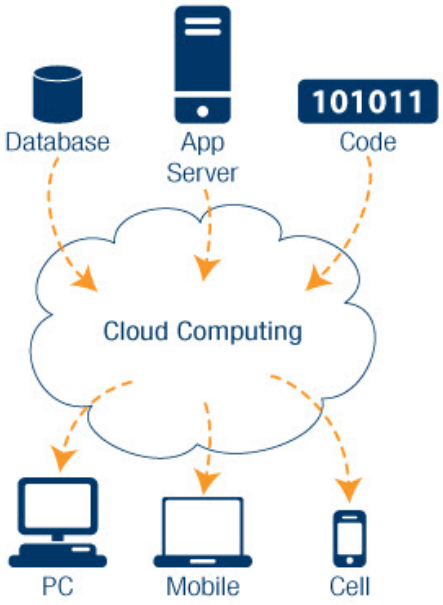
Vecchiola et al. (2009) discutem como serviços de utilidade pública como água, gás, eletricidade e telefone, explorados através de um modelo de pagamento baseado no uso, tornaram-se fundamentais para nossa vida diária.

De acordo com Sousa et al. (2009), “as infraestruturas existentes permitem entregar tais serviços em qualquer lugar e a qualquer hora, de forma que possamos simplesmente acender a luz, abrir a torneira ou usar o fogão”. O uso desses serviços é, então, cobrado de acordo com as diferentes políticas para o usuário final. Atualmente, a mesma ideia de utilidade tem sido aplicada no contexto da informática e uma mudança consistente neste sentido tem sido feita com a disseminação de *Cloud Computing* ou Computação em Nuvem.

A infraestrutura do ambiente de computação em nuvem normalmente é composta por um grande número, centenas ou milhares de máquinas físicas ou nós físicos de baixo custo, conectadas por meio de uma rede. Cada máquina física tem as mesmas configurações de *software*, mas pode ter variação na capacidade de *hardware* em termos de CPU, memória e armazenamento em disco (SOROR et al., 2008).

A Figura 2.4 apresenta uma visão geral de uma nuvem computacional.

Figura 2.4 - Visão geral de uma nuvem computacional.



Fonte: Sousa et al. (2009).

Com a *cloud computing*, muitos aplicativos, assim como arquivos e outros dados relacionados, não precisam mais estar instalados ou armazenados no computador do usuário ou em um servidor próximo. Este conteúdo passa a ficar disponível na nuvem, isto é, em

servidores compartilhados e interligados por meio da *internet*. O fornecedor da aplicação é responsável por todas as tarefas de desenvolvimento, armazenamento, manutenção, atualização, *backup*, escalonamento, etc. O usuário não precisa se preocupar com nenhum destes aspectos, apenas em acessar e utilizar a aplicação (ALECRIM, 2013).

2.1.1.6 *Grid Computing*

Em Foster e Kesselman (2004) o autor apresenta *Grid computing*, ou computação em grade: "é um conceito que explora as potencialidades das redes de computadores, provendo um ambiente computacional de alto desempenho por meio de compartilhamento de recursos geograficamente distribuídos".

O termo *grid*, também conhecido como grade, foi usado inicialmente nos anos 90 no meio acadêmico (TAGLIARI; CARVALHO, 2006). A ideia é utilizar recursos ociosos de computadores independentes, sem a preocupação de localização física e sem investimentos em novos *hardwares*, com o objetivo de auxiliar atividades de pesquisa e desenvolvimento científico.

Foster e Kesselman (2004) discutem a possibilita de uma *grid* agregar recursos computacionais variados e dispersos em um único "supercomputador virtual", acelerando a execução de várias aplicações paralelas e, ao mesmo tempo, executando-as a um custo mais baixo do que o de um supercomputador de potência semelhante, permitindo assim, o desenvolvimento de aplicações antes restritas a caríssimos supercomputadores.

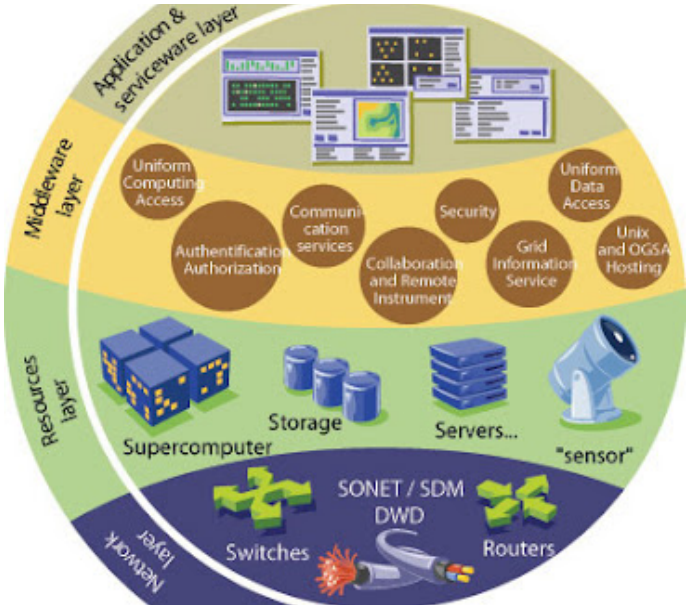
O conceito de *grid* permite que diversas entidades disponibilizem seus recursos no âmbito de uma organização virtual (OV), de forma a otimizar a sua utilização. Essa possibilidade permitiu a rentabilidade de projetos que, de outro modo, seriam inviáveis em termos financeiros ou de complexidade técnica.

Uma arquitetura de grade é geralmente descrita em camadas, onde cada camada possui uma função específica. As camadas mais altas são geralmente focadas no usuário, enquanto que as camadas inferiores focam no *hardware* (computadores e redes) (CERN, 2013).

As quatro camadas de uma grade computacional podem ser vistas na Figura 2.5, sendo a camada mais baixa a de rede, que liga os recursos da rede. Acima desta camada encontra-se a camada de recursos (computadores, sistemas de armazenamento, sensores, telescópios, etc.) conectados a rede. A terceira camada, *middleware*, fornece as ferramentas que permitem que vários elementos (servidores, armazenamento, redes, etc.) participem da grade, podendo ser considerada o "cérebro" por trás de uma grade computacional. A camada mais alta da estrutura é a camada de aplicação, que inclui aplicações em ciência, engenharia,

negócios, finanças, entre outras.

Figura 2.5 - Arquitetura em camadas de uma grade computacional.



Fonte: CERN (2013).

"A intenção do sistema de computação em grade ser dividida em camadas é que, como recursos distribuídos normalmente são heterogêneos, a arquitetura em camadas habilita interoperabilidade e modularidade" (ROCHA, 2010).

2.2 HPC - *High-Performance Computing*

O conhecimento na ciência moderna está baseado em uma grande variedade de dados obtidos com alta resolução e no desenvolvimento de técnicas matemáticas e algoritmos complexos para análise desses dados, o que requer um grande poder computacional para processamento.

"Complexidade, no contexto de algoritmos, refere-se aos requerimentos de recursos necessários para que um algoritmo possa resolver um problema sob o ponto de vista computacional, ou seja, à quantidade de trabalho despendido pelo algoritmo" (TOSCANI; VELOSO, 2012).

HPC engloba qualquer técnica computacional que resolve um grande problema mais rapidamente do que seria possível através de sistemas individuais.

A HPC permite que cientistas e engenheiros resolvam problemas complexos de ciência, de engenharia e de negócios usando aplicativos que exigem grande largura de banda, redes de baixa latência e recursos de computação muito altos. Normalmente, cientistas e engenheiros devem esperar em longas filas para acessar *clusters* compartilhados ou adquirir sistemas de *hardware* muito caros (AMAZON, 2013).

Um *cluster* é apresentado por Pitanga (2003) como um sistema que compreende dois ou mais computadores, ligados em rede, na qual trabalham em conjunto para executar aplicações ou realizar outras tarefas, de tal forma para que os usuários que os utilizam tenham a impressão que somente um único sistema responde para eles, criando assim uma ilusão de um recurso.

Eadline (2012) discute em seu trabalho a evolução da HPC desde os anos 1990, sendo considerado atualmente um modelo computacional imprescindível na infraestrutura de *data centers*. Apesar de ser uma técnica poderosa, HPC não pode resolver todos os problemas de processo.

Como características fundamentais para a construção destas plataformas incluem-se elevação da confiança, distribuição de carga e desempenho.

2.3 Ciência Aberta

Ciência (do latim *scientia*, traduzido por "conhecimento") é o esforço para descobrir e aumentar o conhecimento humano de como o Universo funciona.

Hoje em dia a ciência está fortemente ligada com a área da tecnologia. Os grandes avanços da ciência são alcançados através do desenvolvimento de novas tecnologias e do aprimoramento de tecnologias já existentes.

> A publicação de resultados de pesquisa em acesso aberto é a forma que se tornou consensual de prestar contas à sociedade sobre os recursos empregados na pesquisa, tendo em vista que boa parte dessa é financiada com recursos públicos. Além disso, a sociedade como um todo - e não apenas a comunidade acadêmica - se beneficia largamente do conhecimento gerado em todas as áreas do conhecimento humano (NASSI-CALò, 2013).

"Quando a ciência publica atrás de barreiras, ela perde seu objetivo de melhorar a condição social em que a gente vive", avalia o professor da Universidade Federal do Rio Grande do Sul (UFRGS) Rafael Pezzi em Castro (2015).

2.4 Portal

Portais são aplicações *web* que permitem fácil interação do usuário com serviços e recursos da rede (BANSAL; BAWA, 2005).

Liferay (2014) apresenta um portal como um aplicativo baseado na *web* que fornece comumente personalização, agregação de conteúdos de diferentes fontes e hospeda a camada de apresentação dos Sistemas de Informação. Agregação é a ação de integração de conteúdos de diferentes fontes dentro de uma página *web*. Alguns casos comuns para um portal são:

- *Websites* que exigem a apresentação de diferentes páginas dependendo do estado de *login* do usuário.
- *Websites* que exigem a apresentação de diferentes páginas dependendo do perfil do usuário.
- *Websites* que exigem a integração de múltiplas aplicações *web*.
- *Websites* que permitem que grupos de indivíduos colaborem por meio de aplicativos, com conteúdo ou com documentos.

Um portal pode ter recursos de personalização sofisticados para fornecer conteúdo personalizado para os usuários, permitindo a adição de várias funcionalidades que os tornaram uma boa escolha para uma ampla gama de aplicações *web*.

2.5 Conteinerização

A conteinerização é o processo de distribuição e implantação de aplicativos de uma forma portátil e previsível. Isso é feito empacotando componentes e suas dependências em um ambiente de processos padronizado, isolado e leve chamado *container* (ELLINGWOOD, 2015).

Vitalino e Castro (2016) discutem a similaridade entre *containers* e máquinas virtuais, apontando os *containers* como mais leves e mais integrados ao sistema operacional da máquina *host*, uma vez que compartilha o seu *kernel*, o que proporciona melhor desempenho por conta do gerenciamento único de recursos.

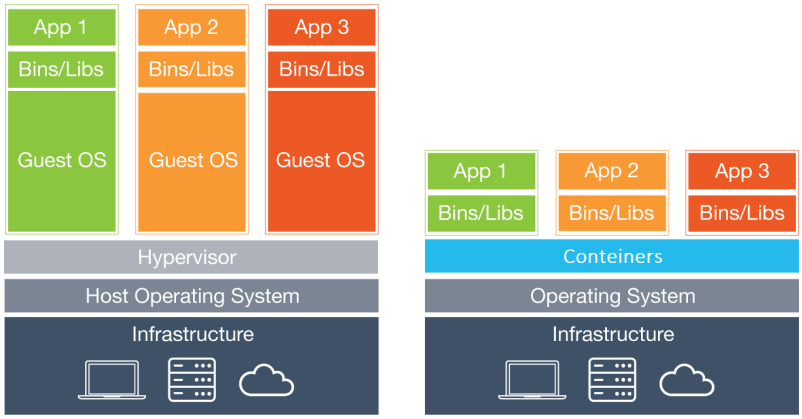
Cada *container* é iniciado de uma imagem que forma a base para outras aplicações e camadas. Elas constituem uma base sólida, consistente e confiável com tudo o que é necessário para executar as aplicações. Um ambiente autossuficiente elimina o risco de

atualizações ou modificações, em nível de sistema, influenciar no comportamento da aplicação.

"Fazendo um paralelo com o conceito de orientação a objeto, a imagem é a classe e o *container* o objeto, ou seja, a imagem é a abstração da infraestrutura em um estado somente leitura, que é de onde será instanciado o *container*" (GOMES, 2016).

Na Figura 2.6 pode-se observar um comparativo entre um sistema com Máquinas Virtuais (esquerda) e, outro sistema com *Containers* (direita).

Figura 2.6 - Virtualização x Conteinerização.



A Virtualização Tradicional demonstrada (lado esquerdo), exige uma camada adicional sobre o Sistema Operacional do *Host* (Hospedeiro), chamada de Hypervisor - que é o responsável por controlar os recursos para as Máquinas Virtuais. Além disso, cada máquina virtual possui seu próprio Sistema Operacional (*Guest OS* - Hóspede). No lado direito temos a ilustração da técnica de *Container*, que é um recurso integrado ao Sistema Operacional do *Host*. Pode-se observar que não há camadas adicionais de Sistemas Operacionais, há apenas camadas separadas de bibliotecas e binários (Bins/Libs) para cada *container*.
Fonte: Aouar (2017).

Alguns bons motivos para se utilizar *containers* para distribuição de aplicações são: (1) ambientes padronizados, (2) distribuir aplicações como pacotes, (3) replicação e (4) isolamento.

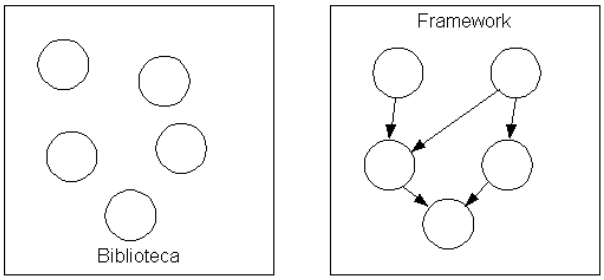
2.6 *Framework*

Em Fayad e Schmidt (1997) os autores apresentam um *framework* como uma aplicação reutilizável, "semi-completa" que pode ser especializada para produzir aplicações persona-

lizadas, servindo de *template* para a criação de aplicações derivadas de sua estrutura.

A Figura 2.7 apresenta uma simples comparação entre *framework* e biblioteca. Como pode ser observado, biblioteca é um conjunto de classes utilitárias isoladas, já um *framework* é um conjunto de classes interligadas que colaboram para um determinado fim.

Figura 2.7 - *Framework* x Biblioteca.



Fonte: Sauvé (2014).

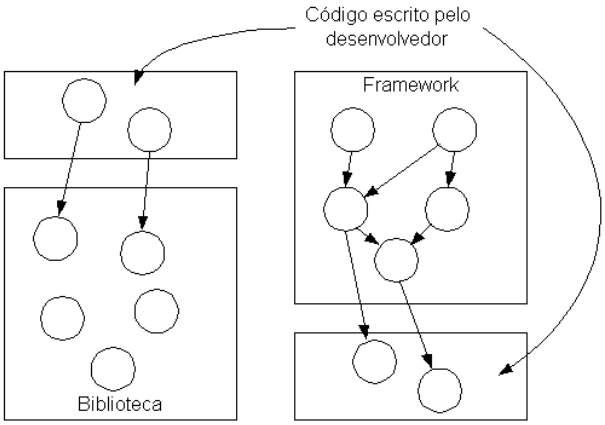
O *framework* cria a base da aplicação, ou seja, uma abstração no qual um trecho de código comum fornecendo uma funcionalidade genérica pode ser substituído ou especializado pelo código do usuário fornecendo funcionalidade específica.

Conceitualmente, os *frameworks* são compostos de duas partes: (1) o núcleo, que representa as partes fixas, ou seja, são serviços já implementados pelo *framework*, conhecidos como ***frozen spots***, e (2) os *slots*, que representam as partes flexíveis do *framework*, conhecidos como ***hot spots***.

Frozen spots são classes, bibliotecas, métodos e módulos que são iguais e que servem como a base para todas as aplicações, enquanto *hot spots* são pontos extensíveis que necessitam de complementação por funcionalidades/serviços nos quais os programadores que usam o *framework* adicionam o seu código para especificar uma funcionalidade que deve ser implementada.

A Figura 2.8 apresenta o funcionamento de um *framework*, onde o *framework* impõe um modelo de colaboração (o resultado da análise e *design*) ao qual o programador deve se adaptar já que a comunicação entre objetos já está definida. O projetista de aplicações não precisa saber quando chamar cada método, é o *framework* que faz isso. No caso das bibliotecas é o programador que deve saber a hora de realizar a chamada a um determinado método e relacionar os recursos disponibilizados por ela.

Figura 2.8 - Funcionamento *Framework* x Biblioteca.



Fonte: Sauvé (2014).

Sauvé (2014) apresenta as características básicas de um *framework*, que são: (1) deve ser reusável e para isso, deve ser bem documentado e fácil de usar, (2) deve ser extensível contendo funcionalidade abstrata (sem implementação), (3) deve ser de uso seguro não podendo ser destruído pelo desenvolvedor, (4) deve ser eficiente dado seu uso em muitas situações, e (5) deve ser completo para contemplar o domínio pretendido.

Além da redução no tempo de criação e facilidade de manutenção da aplicação, com um *framework* busca-se também flexibilidade e reusabilidade, entre outros aspectos. Por outro lado, a construção de um *framework* é complexa pois deve ser bem planejada e testada exaustivamente para garantir seu funcionamento.

3 METODOLOGIA

A pesquisa desse trabalho teve início com a identificação de etapas desenvolvidas durante uma pesquisa científica, onde pode-se destacar o *workflow* genérico apresentado por Bosman e Kramer (2017). Identificadas essas etapas, foram identificados *softwares* para uso em cada uma delas. Após a análise desses *softwares*, chegou-se a um conjunto de características desejáveis para o ambiente proposto, chamado Sentinel. Finalizando, serão expostas as tecnologias adotadas no desenvolvimento do ambiente Sentinel, seguidas da apresentação da infraestrutura de *hardware* utilizada para sua operacionalização.

3.1 *Workflow* de pesquisa

Atualmente, muitas ferramentas estão disponíveis aos autores, pesquisadores e estudantes na *web*. Em uma iniciativa que tem como objetivo apresentar inovações em comunicação científica e explicitar como essas inovações estão mudando o *workflow* de pesquisa, Bosman e Kramer (2017) mapearam etapas e ferramentas que são adotadas em cada uma delas, como pode ser observado na Figura 3.1.

Figura 3.1 - Inovações em Comunicação Científica.

Fonte: Bosman e Kramer (2017).

Para cada uma das etapas definidas por Bosman e Kramer (2017) existe um conjunto de tarefas que podem ser executadas pelos pesquisadores para cumprir com suas atividades, apresentadas na Tabela 3.1.

Tabela 3.1 - Etapas x Tarefas.

Etapa	Descrição	Tarefas
Descoberta	Iterações de busca e leitura.	Pesquisa Obter acesso Recomendações de leitura Gerenciar referências Leitura Anotações
Análise	Gerar e analisar conjuntos de dados.	Coletar/extrair dados Compartilhar notas Analisar
Redação	Redação e recebimento de comentários.	Visualizar Escrever Citar Traduzir
Publicação	Submissão, revisão pelos pares, rejeição, reenvio.	Compartilhar código Compartilhar dados Compartilhar publicações Compartilhar pôsteres Compartilhar apresentações Apresentar os resultados Comentário (pré-publicação) Selecionar onde publicar
Divulgação	Divulgação dos resultados obtidos.	Valorização Perfil do pesquisador
Avaliação	Analisar o impacto do trabalho.	Revisão dos pares (pós-publicação) Contagem de citações Avaliação

Fonte: Adaptado de Bosman e Kramer (2017).

Trabalhos similares ao de Bosman e Kramer (2017) foram produzidos por Crouzier (2016), Armoza et al. (2015), Ebermann (2017), Force11 (2017), Stash (2017) e SciCrunch (2017).

Atualmente, existem várias ferramentas ligadas à gestão das atividades de pesquisa, em *workflows* que vão do tradicional, moderno, inovador ao experimental, dependendo do perfil do pesquisador. As ferramentas mais utilizadas em cada etapa dos *workflows* apresentados por Bosman e Kramer (2017) são apresentadas na Tabela 3.2.

Tabela 3.2 - *Workflow* de pesquisa x Ferramentas utilizadas.

	Descoberta	**Análise**	**Escrita**	**Publicação**	**Divulgação**	**Avaliação**
Tradicional	Web of Science	SPSS	MS Word EndNote	Nature	ResearcherID	Journal Impact Factor
Moderno	Google Scholar	R	Google Drive RefWorks	ArXiv PLoS One	Repositório Institucional	Eigenfactor ALM SNIP
Inovador	Mendeley	Zooniverse Hivebench	Authorea Colwiz	Figshare PeerJ	ResearchGate ORCiD	Peerage of Science Altmetric
Experimental	Sparrho Utopiadocs	ROpen-Sci IPython Notebooks	Overleaf Docear	The Winnower	Kudos	Publons PubPeer
Popular	Google Scholar	Excel	Mendeley	PLoS One	ResearchGate	Web of Science/JCR h-index
Open Science	Paperity	ROpenSci Science Exchange Zooniverse	Hypothes.is Google Drive Zotero	F1000Research PLoS One Figshare Dryad	ORCiD ResearchGate SlideShare	Publons ImpactStory

Fonte: Adaptado de Bosman e Kramer (2017).

3.1.1 A Etapa de Análise de Dados

É na etapa de análise onde são executados os experimentos científicos para obtenção e análise de dados. A execução de um experimento pode ser definida por uma execução única ou por uma sequência de tarefas chamada de *workflow*, que pode envolver vários experimentos com execuções encadeadas. De acordo com Digiampietri et al. (2004), um *workflow* consiste de um modelo para a execução controlada de múltiplas tarefas.

Desde os anos 90, *workflows* vêm sendo usados em grande escala para automatizar processos industriais e gerenciais, comumente designados como *workflows* de negócio. Já o uso em domínios científicos se popularizou nos anos 2000 (BRAGHETTO; CORDEIRO, 2014).

Apesar das muitas semelhanças que podem ser estabelecidas entre os *workflows* de negócio e os científicos, no meio científico não foi adotado um padrão como no meio industrial. Isso ocorre porque o domínio científico possui particularidades que justificam a criação de técnicas e ferramentas específicas (BRAGA; GANNON, 2014).

Existem diversos *softwares* computacionais, que utilizam uma gama enorme de tecnologias diferentes, que possuem a tarefa de gerenciar o ciclo de vida de um *workflow*. Esses sistemas são geralmente conhecidos como Sistemas Gerenciadores de *Workflow* (WfMS - *Workflow Management Systems*), e garantem que os passos que automatizam os processos ocorram na sequência correta (RAMOS, 2016).

No desenvolvimento dessa pesquisa foram identificadas 314 ferramentas *on-line* aplicadas a pesquisas científicas, sendo 72 delas aplicadas na análise de dados. Com base no estudo dessas ferramentas e em algumas críticas e necessidades apontadas pelos usuários, foram identificadas algumas características para o ambiente Sentinel:

Ambiente multidisciplinar Possibilitar o desenvolvimento de pesquisas em diversos domínios científicos. Essa característica busca reduzir os custos de operação e de manutenção do ambiente, pois recursos podem ser compartilhados entre projetos e/ou instituições associadas.

Conjunto personalizado de aplicações Permitir aos pesquisadores acoplarem, sem restrição de linguagens de programação, suas próprias aplicações ao ambiente para concentrar em um só ambiente todas as suas ferramentas de pesquisa.

Controle Oferecer mecanismos que admitam total controle ao pesquisador para poder gerenciar e monitorar cada um dos seus experimentos tomando decisões sobre sua execução.

Escalonamento e orquestração dos recursos Coordenar a execução dos experimentos nos recursos apropriados não sobrecarregando os recursos computacionais disponíveis no ambiente.

Compartilhamento de resultados Compartilhar os produtos (resultados, publicações, apresentações, ...) originários da pesquisa com outros pesquisadores ou grupos de pesquisa adotando práticas científicas abertas.

Colaboração Oferecer um ambiente de colaboração entre pesquisadores e instituições onde estes poderão colaborar com as pesquisas de terceiros, contribuindo no aprimoramento dos trabalhos desenvolvidos, seja através de melhorias em algoritmos, obtenção de resultados mais precisos, geração de publicações, etc.

Status Emitir alertas sobre o atual andamento da execução dos experimentos mantendo o pesquisador atualizado para poder tomar decisões sobre seu experimento.

Composição de *workflow* Oferecer ao pesquisador uma ferramenta que permita mapear uma sequência de tarefas a ser executada a fim de automatizar suas pesquisas.

3.2 Estudo das ferramentas de *e-Science*

Após a análise inicial realizada nas ferramentas e a definição das características desejadas para a construção do ambiente Sentinel, estas informações foram confrontadas a fim de identificar quais das ferramentas possuíam ao menos uma das características desejadas. Desta etapa, resultou uma lista de 15 ferramentas, que serão brevemente apresentadas nas subseções que seguem.

3.2.1 Excel

O Excel[1] é um editor de planilhas produzido pela Microsoft. Seus recursos incluem uma interface intuitiva, modernas ferramentas de cálculo e de construção de gráficos, tabelas dinâmicas e uma linguagem de programação macro chamada *Visual Basic for Applications* (VBA).

3.2.2 Google Drive

O Google Drive[2] é um pacote de aplicativos da Google que funciona totalmente *on-line* diretamente no *browser*. Os arquivos produzidos podem ser compartilhados, oferecendo

[1] products.office.com/excel
[2] https://drive.google.com/

serviço de edição colaborativa entre os usuários dependendo do nível de permissão atribuído a cada um.

3.2.3 GridUnesp

O GridUnesp[3] disponibiliza recursos computacionais para projetos científicos destinados a todos os pesquisadores da Universidade Estadual Paulista (Unesp). Os recursos estão disponíveis através de *clusters* HPC, armazenamento e processamento em grade (*grid computing*).

3.2.4 The Jupyter Notebook

O projeto Jupyter[4] é uma aplicação *web* de código aberto que permite criar e compartilhar documentos que contenham códigos, equações, visualizações e texto explicativo com suporte a mais de 40 linguagens de programação. Seu uso inclue: limpeza e transformação de dados, simulação numérica, modelagem estatística, *machine learning* entre outras.

3.2.5 nanoHub

O nanoHub[5] é um portal para a pesquisa, educação e colaboração computacional em nanotecnologia que hospeda uma coleção de ferramentas de simulação para fenômenos de nanoescala que são executados na nuvem e acessíveis através de um navegador da *web*. Além das simulações, o nanoHUB oferece também apresentações *on-line*, cursos, animações e materiais didáticos que ajudam os usuários a aprender sobre as ferramentas de simulação disponíveis e sobre a nanotecnologia em geral.

3.2.6 Open Science Grid (OSG)

O OSG[6] fornece serviços e suporte para provedores de recursos e instituições científicas usando uma rede distribuída de serviços computacionais de alto desempenho. O OSG não possui recursos, eles são disponibilizados pela comunidade, organizados e gerenciados pelo consórcio OSG.

3.2.7 Profager

O portal ProFrager[7] é uma ferramenta de geração de bibliotecas de fragmentos de proteínas onde os usuários podem escolher diversos parâmetros que alteram a composição das

[3] http://www.ncc.unesp.br/
[4] https://jupyter.org/
[5] nanoHUB.org
[6] https://www.opensciencegrid.org/
[7] https://www.lncc.br/sinapad/Profrager/

bibliotecas de maneira significativa. O portal oferece ao usuário diversas opções para criação de suas próprias bibliotecas.

3.2.8 R

R[8] é uma linguagem de programação e também um ambiente de desenvolvimento integrado para cálculos estatísticos e gráficos largamente utilizada por estatísticos e analistas de dados para desenvolver *software* de estatística e análise de dados.

3.2.9 ROpenSci

ROpenSci[9] são pacotes de código aberto desenvolvidos na linguagem R que fornecem acesso a uma variedade de dados científicos, texto completo de artigos de revistas e repositórios, fornecendo métricas em tempo real e apoiando uma ciência aberta.

3.2.10 RTGrid

O projeto RTGrid[10] é uma colaboração de pesquisa multidisciplinar que investiga o uso de recursos de computação distribuída, como *clusters* e *grid*, para reduzir o tempo de computação de simulações de Monte Carlo quando usado para cálculos de dose de radioterapia.

3.2.11 Supervisor Monitoring Scheduler (SMS) / ECFlow

O ecFlow[11] é um pacote que permite aos usuários executar uma grande quantidade de programas em um ambiente controlado com tolerância para falhas de *hardware* e *software*, combinadas com recursos de reinício das execuções. O ecFlow permite a criação de um *workflow* para controlar dependências entre as execuções de tarefas (com dependências umas sobre as outras e no tempo) através de um mecanismo de controle de mudança de *status*.

3.2.12 SPiNMe

SPiNMe[12] é um ambiente computacional em que a especificação e prototipagem de métodos numéricos podem ser perpetuadas e a experimentação com eles pode ser feita de forma mais facilmente reproduzível e comparável com outros métodos. Métodos numéricos são em geral de difícil implementação o que torna o processo de comparação com outros métodos bastante custoso do ponto de vista de tempo de desenvolvimento.

[8] www.r-project.org
[9] http://ropensci.org/
[10] http://www.cs.cf.ac.uk/gridprojects/rtgrid/pages/Overview.html
[11] https://software.ecmwf.int/wiki/display/ECFLOW/
[12] http://www.lncc.br/sinapad/SPiNMe/

3.2.13 SPSS

SPSS[13] é um *software* estatístico de apoio à tomada de decisão usado para solucionar uma ampla variedade de problemas de negócios e de pesquisas. Inclui entre as suas ferramentas: aplicação analítica, *data mining*, *text mining* e estatística que transformam os dados em informações importantes para auxiliar na tomada de decisões de forma mais rápida proporcionando redução de custos e aumento de lucratividade.

3.2.14 VLADA

VLADA[14] é o acrônimo de *Virtual Lab for Advanced Data Analysis*, um ambiente de análise colaborativa de dados dedicado à análise de séries temporais projetado como o primeiro ambiente virtual de análise de dados para a comunidade científica na *web*.

3.2.15 Zooniverse

Zooniverse[15] é uma plataforma colaborativa onde estão concentrados os mais populares e bem-sucedidos projetos de ciência cidadã, com mais de um milhão de voluntários cadastrados em 2014.

3.3 Infraestrutura de *Software*

Nesta seção serão apresentadas as tecnologias adotadas no desenvolvimento e operacionalização do ambiente Sentinel. Os principais fatores considerados na escolha das tecnologias foram: (1) comunidade de usuários, (2) curva de aprendizagem, (3) licença de uso gratuito e (4) conhecimento prévio da tecnologia.

As tecnologias adotadas foram categorizadas da seguinte forma:

Infraestrutura Adotadas para disponibilizar e manter os serviços que são utilizados pelos usuários.

Back-end Responsáveis pela aplicação das regras de negócio.

Persistência de dados Utilizadas para armazenamento e recuperação de dados.

Front-end Empregues na construção das interfaces gráficas.

Visualização Usada na apresentação dos dados produzidos.

[13] https://www.ibm.com/analytics/us/en/technology/spss/
[14] www.inpe.br
[15] https://www.zooniverse.org/

Na Tabela 3.3 são apresentadas as tecnologias adotadas em cada categoria.

Tabela 3.3 - Tecnologias adotadas no projeto em cada uma das categorias.

Categoria	Tecnologias
Infraestrutura	Maven Apache Tomcat Docker Docker Swarm
Back-end	Java Spring Framework Spring MVC Spring Data JPA Spring Security Apache Velocity
Persistência de Dados	MySQL Flyway Hibernate
Front-end	HTML5 Thymeleaf Bootstrap
Visualização	D3.js JasperReports

Fonte: Produção do autor (2017).

Cada uma das tecnologias adotadas no desenvolvimento desse projeto será brevemente apresentada nas subseções que seguem.

3.3.1 Maven

Maven é uma ferramenta da Apache Software Foundation para gerenciamento de dependências e automação de *build*, principalmente em projetos Java (FARIA, 2013).

Um projeto Maven utiliza um arquivo XML conhecido como POM (*Project Object Model*) para descrever o projeto, suas dependências sobre módulos e componentes externos, a ordem de compilação, estrutura de diretórios e *plugins* necessários (O´BRIEN et al., 2013).

Uma versão simplificada do arquivo POM criado para o projeto Sentinel é apresentado no Código 3.1.

Código 3.1 - Arquivo POM de configuração do projeto Sentinel.

```
<project>
  <!-- model version is always 4.0.0 for Maven 2.x POMs -->
  <modelVersion>4.0.0</modelVersion>

  <!-- project coordinates -->
  <groupId>com.inpe</groupId>
  <artifactId>sentinel</artifactId>
  <version>1.0</version>

  <!-- library dependencies -->
  <dependencies>
    <dependency>
      <!-- coordinates of the required library -->
    </dependency>
  </dependencies>
</project>
```

Um projeto Maven possui uma estrutura padrão de diretórios, conforme apresentada na Figura 3.2. Essa estrutura permite que novos usuários se familiarizem imediatamente com o projeto.

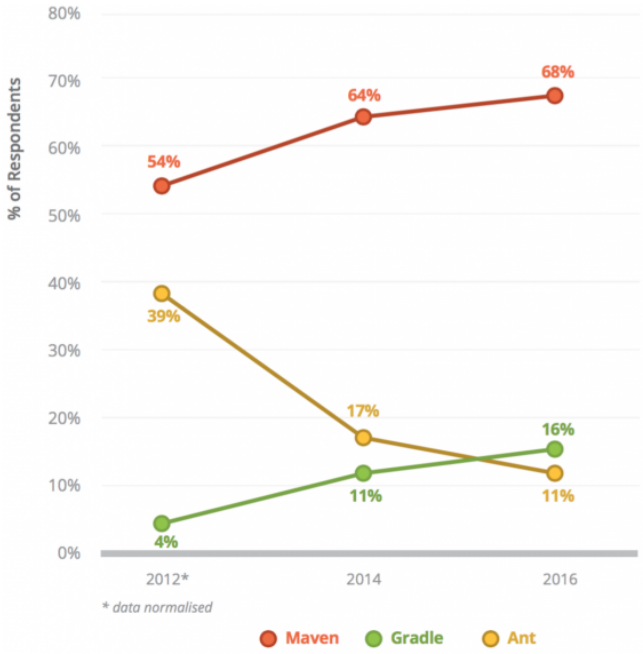
Figura 3.2 - Estrutura padrão de diretórios de um projeto Maven.

```
/
├── src/
│   ├── main/
│   │   ├── java/ ... Contém o código-fonte Java da aplicação.
│   │   └── resources/ ... Contém os recursos entregáveis
│   │                      para o projeto.
│   └── test/
│       ├── java/ ... Contém os arquivos de testes unitários.
│       └── resources/ ... Contém os arquivos que serão
│                          utilizados pelas classes de
│                          testes unitários.
├── target/ ... Contém toda a saída da compilação.
└── pom.xml
```

Como apresentado na Figura 3.3, Maven é apontada como a ferramenta de gerenciamento

mais utilizada nos últimos anos em projetos de *software*.

Figura 3.3 - Ferramentas de gerenciamento de projetos.



Fonte: Maple e Shelajev (2016).

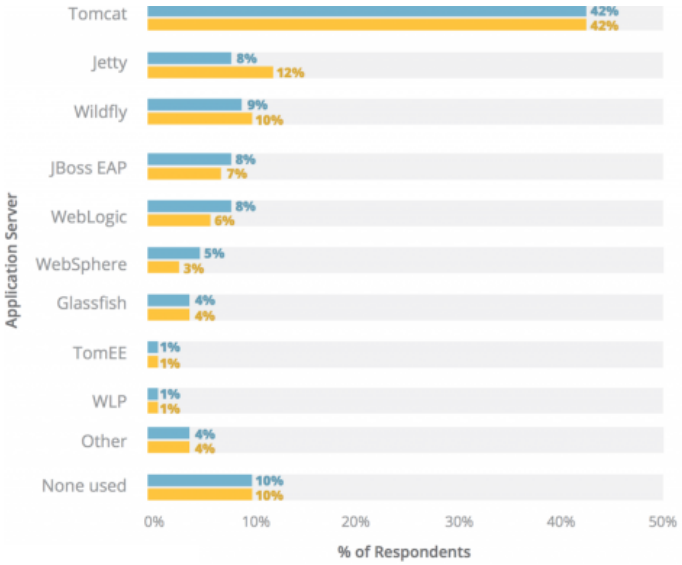
O gerenciador Maven foi utilizada neste projeto para tratar exclusivamente do gerenciamento das dependências como poderá ser observado na apresentação das demais tecnologias.

3.3.2 Apache Tomcat

O Apache Tomcat é um servidor *web* Java, mais especificamente, um *container* de *servlets*. *Containers* são interfaces entre componentes e funcionalidades de baixo nível específicas de uma plataforma. Para uma aplicação *web* desenvolvida em Java ou um componente corporativo ser executado, eles precisam ser implantados em um *container*. Os *containers* também são chamados de servidores de objetos, ou servidores de aplicação, pois oferecem serviços de infraestrutura para execução de componentes (FARIA, 2013).

Vukotic e Goodwill (2011) apontam o Apache Tomcat como um servidor *web* muito estável e que possui todas as características de um servidor comercial. Já Maple e Shelajev (2016), apontam o Apache Tomcat como o servidor de aplicações *web* mais utilizado entre os desenvolvedores tanto em desenvolvimento quanto em produção, ver Figura 3.4.

Figura 3.4 - Servidores de aplicações *web*.



A barra azul representa a porcentagem de uso em Produção, enquanto a barra amarela representa o uso em Desenvolvimento.

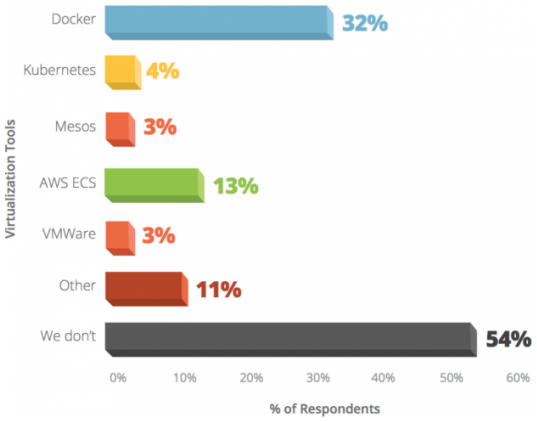
Fonte: Maple e Shelajev (2016).

3.3.3 Docker

Docker é uma ferramenta que automatiza a implantação de aplicativos dentro de *containers*, fornecendo uma camada adicional de abstração e automação de virtualização a nível de sistema operacional (MATTHIAS; KANE, 2015).

Como apresentado na Figura 3.5, Docker é a ferramenta de conteinerização mais utilizada nos dias atuais.

Figura 3.5 - Ferramentas de conteinerização utilizadas com mais frequência.



Fonte: Maple e Shelajev (2016).

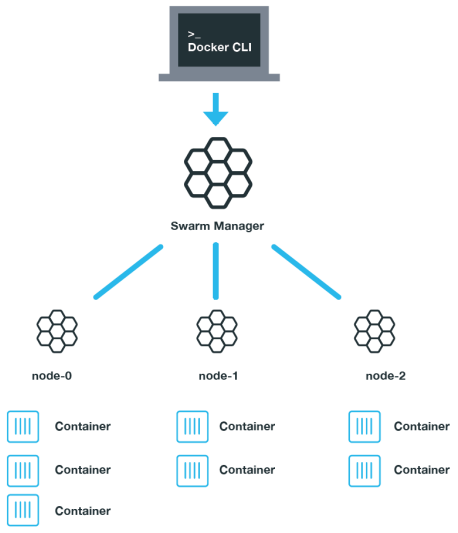
Usando *containers*, os recursos podem ser isolados, os serviços restritos e os processos provisionados para ter uma visão quase totalmente privada do sistema operacional com sua própria estrutura do sistema de arquivos e interfaces de rede, o que possibilita uma implementação segura das aplicações sem danos ao ambiente operacional (*host*) (NICKOLOFF, 2016). Estes são os principais fatores que levaram a adoção do Docker.

3.3.4 Docker Swarm

O Docker Swarm é uma ferramenta de orquestração que permite a criação de *clusters* de Docker e possui políticas flexíveis para agendamento de recursos permitindo controle sobre a melhor maneira de gerenciar os recursos do *host* disponíveis para os *containers* de aplicativos em execução. Em vez de ter que se comunicar com cada Docker Engine diretamente para criar e executar *containers*, Docker Engines podem ser reunidos em um único "mecanismo virtual" que agrupa seus recursos e se comunica com um único gerenciador de Swarm para executar comandos (FARCIC, 2017).

A Figura 3.6 mostra como funciona uma infraestrutura utilizando o Docker Swarm. Ela é composta basicamente de duas camadas: *workers* (responsáveis pela execução das tarefas) e *manager nodes* (ponto central de administração).

Figura 3.6 - Exemplo de arquitetura de um ambiente com Docker Swarm.



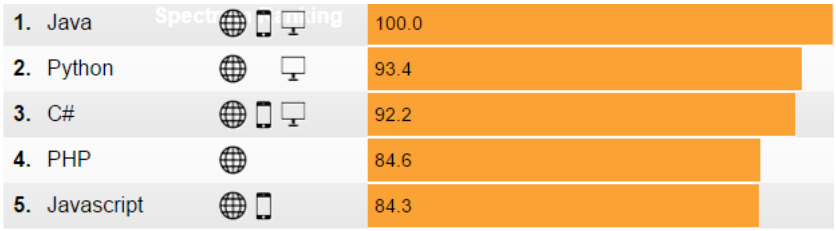
Fonte: Docker (2016).

Docker Swarm foi a opção adotada para a gestão do *cluster* Docker utilizado na execução dos experimentos por ser uma aplicação nativa do Docker.

3.3.5 Java

A linguagem de programação Java é uma linguagem de programação interpretada orientada a objetos desenvolvida na década de 90 (SCHILDT, 2007). No início dessa pesquisa de doutorado, a linguagem era apontada no *ranking* IEEE Spectrum [16] como a linguagem de programação mais utilizada no desenvolvimento de *websites* e aplicações como pode ser visto na Figura 3.7.

Figura 3.7 - *Ranking* IEEE Spectrum das principais linguagens de programação - 2014.



Ranking IEEE Spectrum das linguagens de programação mais utilizadas em 2013, ano de início das pesquisas para desenvolvimento do ambiente Sentinel. Atualmente a linguagem Java ocupa a posição 3 com 99.5 pontos (CASS, 2017).
Fonte: Cass et al. (2014).

O *ranking* IEEE Spectrum sintetiza 12 métricas de 10 fontes para chegar a um *ranking* de popularidade daw linguagens. As fontes abrangem contextos que incluem "*social chatter*", produção de código *open-source* e postagem de trabalhos (CASS, 2016).

Faria e Junior (2015a) abordam os vários usos da plataforma Java. Ela pode ser utilizada para criar páginas da *web* com conteúdo interativo e dinâmico, para desenvolver aplicativos corporativos de grande porte, aprimorar a funcionalidade de servidores da www, fornecer aplicativos para dispositivos destinados ao consumidor final e para muitas outras finalidades.

Um dos fatores que motivou o emprego da linguagem Java no desenvolvimento do ambiente Sentinel foi sua posição no *ranking* IEEE Spectrum das principais linguagens de progamação para *websites* e aplicações apresentado na Figura 3.7. O fato da linguagem se encontrar no topo da lista aponta sua popularidade entre desenvolvedores, o que pode contribuir para futuras manutenções e evolução do ambiente.

[16] http://spectrum.ieee.org/

3.3.6 Spring

Criado por Rob Johnson em meados de 2002, o Spring surgiu como uma alternativa ao Java EE (*Java Enterprise Edition*), e seus criadores sempre se preocuparam para que ele fosse o mais simples e leve possível (WALLS, 2015).

O trabalho de Junior e Afonso (2017) aponta o Spring não como um *framework* apenas, mas um conjunto de projetos que resolvem várias situações do cotidiano de um programador, ajudando a criar aplicações Java com simplicidade e flexibilidade.

O Spring oferece diversos módulos que podem ser utilizados de acordo com as necessidades do projeto, como Spring Data para acesso a banco de dados, Spring Security para prover segurança, Spring Web para aplicativos da *web* e diversos outros projetos que vão de *cloud computing* até *big data* (GENTIL, 2015). Cabe ao desenvolvedor dizer ao Spring quais são os módulos que deseja utilizar.

Alguns dos módulos oferecidos pelo Spring foram adotados nesse projeto para aumentar a produtividade e facilitar o desenvolvimento do ambiente.

3.3.7 Spring Framework

O Spring Framework é o projeto do Spring que serve de base para todos os outros. “Ele foi pensado para que as aplicações pudessem focar mais na regra de negócio e menos na infraestrutura” (JUNIOR; AFONSO, 2017).

Spring Framework é voltado para o desenvolvimento de aplicações corporativas para a plataforma Java, baseado nos conceitos de inversão de controle (IoC) e injeção de dependências (DI). De uma forma simplificada, pode-se pensar o seguinte: a IoC controla o ciclo de vida das aplicações, a DI define quais classes serão instanciadas e em quais lugares serão injetadas (WEISSMANN, 2012).

3.3.7.1 Spring MVC

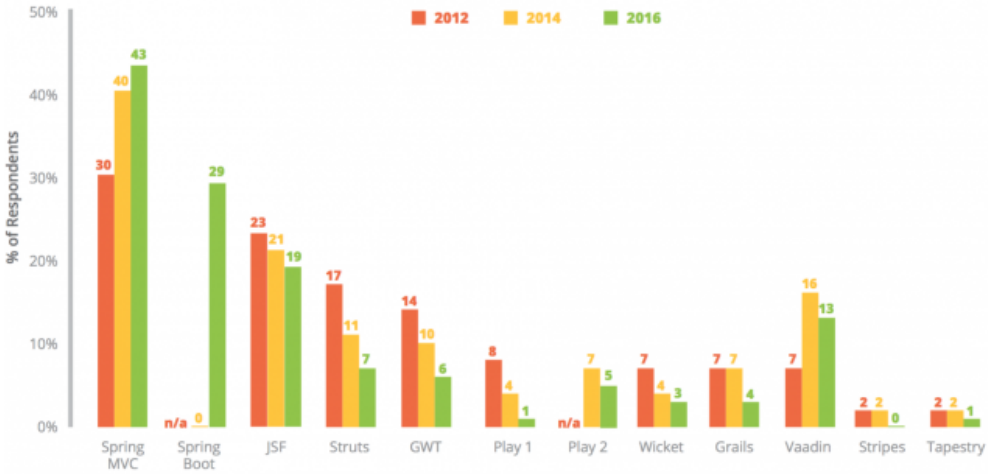
Spring MVC é o *framework* que permite implementar os “fluxos” de uma aplicação *web*. Um fluxo encapsula uma sequência de etapas que orientam o usuário através da execução de alguma tarefa. Ele abrange várias solicitações HTTP (*Hyper Text Transfer Protocol*), tem estado, lida com dados transacionais, é reutilizável e pode ser dinâmico (SPRING, 20–).

MVC é acrônimo de *Model*, *View* e *Controller*, e entender bem o que cada um deve fazer na aplicação é importante para se ter uma aplicação bem escrita e de fácil manutenção. O *Controller* é o responsável por tratar as requisições dos clientes. No *Model* é onde se aplica

as regras de negócio. A *View* irá transformar em HTML (*HyperText Markup Language*) os dados para que o usuário consiga visualizar a informação. Essa é a ideia do MVC, separar claramente a responsabilidade de cada componente dentro de uma aplicação.

Maple e Shelajev (2016) apontam o Spring MVC como o *framework* para desenvolvimento Java *web* mais utilizado entre os desenvolvedores, como pode ser visto na Figura 3.8.

Figura 3.8 - Java *web frameworks* utilizados com mais frequência nos últimos anos.



Fonte: Maple e Shelajev (2016).

Spring MVC é mais que um *framework web*: é também a aplicação de todas as boas práticas de projeto, o que o torna uma alternativa bastante viável na criação de novas aplicações *web* para a plataforma Java EE (WEISSMANN, 2012).

3.3.7.2 Spring Data

O Spring Data é o projeto que tem como objetivo simplificar o acesso a tecnologias de armazenamento de dados. Sejam elas relacionais (MySQL, PostgreSQL, etc.) ou não (MongoDB, Redis, etc.) (JUNIOR; AFONSO, 2017).

O módulo Spring Data JPA tem como objetivo melhorar significativamente a implementação de repositórios baseados em *Java Persistence API* (JPA) (GIERKE et al., 2017).

3.3.7.3 Spring Security

Spring Security é uma estrutura que se concentra em fornecer autenticação e autorização para aplicativos Java, que nos permite definir de forma declarativa quem acessará o quê em nossos sistemas (ALEX et al., 2004).

3.3.8 Apache Velocity

Apache Velocity é um *template engine* baseado em Java que fornece uma linguagem projetada para referenciar objetos definidos no código Java. O objetivo é garantir uma separação limpa entre a camada de apresentação (*view*) e os níveis de negócios (*model*) em um aplicativo da *web* (o padrão de *design* do MVC) (GRADECKI; COLE, 2003).

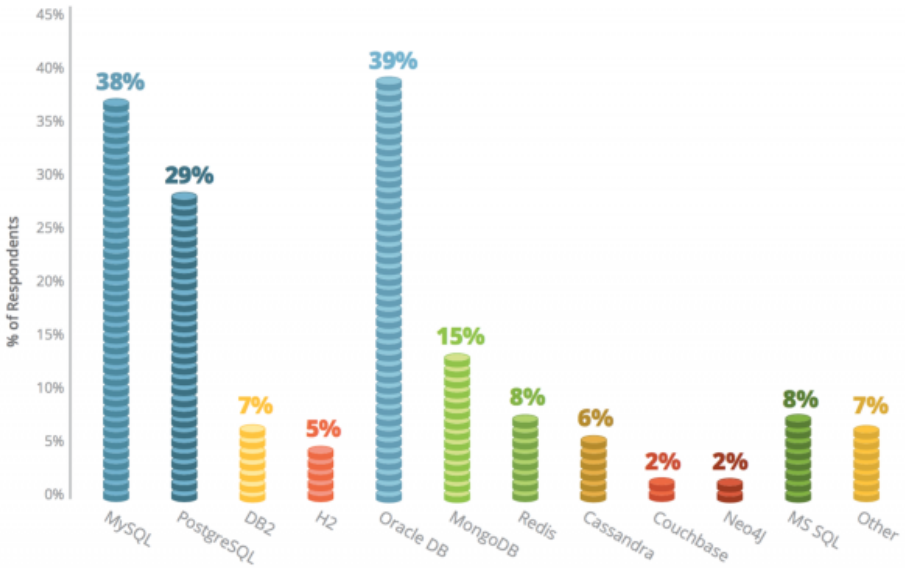
Quando o Velocity é usado no desenvolvimento *web*, *designers* podem trabalhar em paralelo com programadores para desenvolver *sites* de acordo com o modelo MVC, o que significa que os *designers* podem se concentrar unicamente na criação da interface gráfica, e os programadores podem se concentrar unicamente na escrita de código (APACHE, 201-).

Velocity pode ser usado para gerar SQL (*Structured Query Language*), PostScript, XML, código fonte e relatórios a partir de *templates*, sendo utilizado neste projeto para auxiliar na construção do arquivo de configuração que serve de base para criação da imagem das aplicações, chamado Dockerfile.

3.3.9 MySQL

O MySQL é um Sistema de Gerenciamento de Banco de Dados (SGBD) que utiliza a linguagem SQL como interface. É atualmente um dos bancos de dados mais populares como mostra a Figura 3.9.

Figura 3.9 - Sistemas de Gerenciamento de Banco de Dados mais utilizados.



Fonte: Maple e Shelajev (2016).

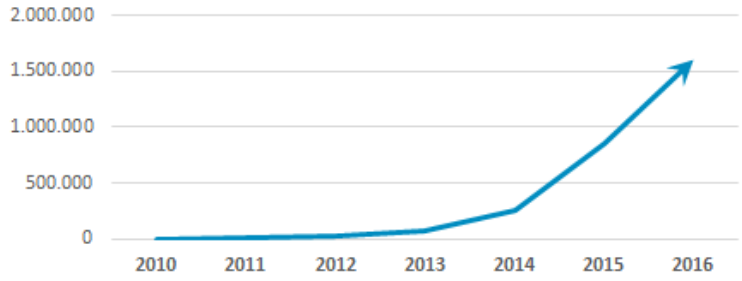
Tahaghoghi e Williams (2006) sugerem que alguns fatores que tornaram o MySQL tão popular entre usuários iniciantes e grandes indústrias são: (1) tamanho e velocidade, (2) facilidade de instalação, (3) atenção às normas, (4) capacidade de resposta à comunidade e (5) interface fácil para outros *softwares*. Esses fatores foram considerados para sua adoção no desenvolvimento desse projeto.

3.3.10 Flyway

Flyway é um gerenciador de migrações de banco de dados, descrito em seu *site*[17] como uma estrutura popular de migração de banco de dados de código aberto para Java. Ele traz estrutura e confiança para a evolução do seu esquema de banco de dados.

Sua popularidade pode ser justificada pelo gráfico apresentado na Figura 3.10 que apresenta um aumento considerável no número de *downloads* nos últimos anos.

Figura 3.10 - Número de *downloads* anuais do Flyway.



Fonte: Boxfuse (200-?).

As migrações podem ser escritas em SQL ou Java e podem ser executadas através da API (*Application Programming Interface*) do plugin Maven (BOXFUSE, 200-?).

Os principais motivos que levaram a adoção do Flyway são os mesmos apontados por Parsons et al. (2015): (1) possui uma abordagem simplificada, (2) tem uma comunidade de código aberto ativa, e (3) suporte tanto para bancos de dados tradicionais quanto baseados em nuvem como o Amazon Redshift e o Google Cloud SQL.

[17] https://flywaydb.org/

3.3.11 Hibernate

Hibernate é uma biblioteca *Object-Relational Mapping* (ORM) para Java que implementa a especificação JPA.

JPA é uma API padrão do Java para persistência de dados, que usa um conceito de ORM. O mapeamento objeto relacional é uma técnica de programação para conversão de dados entre banco de dados relacionais e linguagens de programação orientada a objetos. Podemos comparar o modelo relacional com o modelo orientado a objetos conforme mostra a Tabela 3.4.

Tabela 3.4 - Relação entre Modelo Relacional e Modelo Orientado a Objetos.

Modelo relacional	Modelo OO
Tabela	Classe
Linha	Objeto
Coluna	Atributo
-	Método
Chave estrangeira	Associação

Fonte: Produção do autor (2017).

O Hibernate é a implementação padrão do JPA no projeto Spring Data-JPA, graças a isso tem integração perfeita com o Spring. "O projeto do Hibernate ORM possui alguns módulos, sendo que o Hibernate EntityManager é a implementação da JPA que encapsula o Hibernate Core, base para o funcionamento da persistência" (FARIA; JUNIOR, 2015b).

Um exemplo de mapeamento objeto relacional pode ser visto no Código 3.2, que apresenta o mapeamento da classe `Organization.java`.

Código 3.2 - Exemplo de mapeamento objeto relacional.

```
1  package lac.cap.br.inpe.sentinel.model;
2
3  // imports omitidos
4
5  @Entity
6  @Table(name = "organizations")
7  public class Organization extends AbstractPersistable<Long> {
8
9    @NotBlank(message = "{orgName.notBlank}")
10   @Size(max = 80, message = "{orgName.size}")
```

```
11    @Column(nullable = false, unique=true)
12    private String name;
13
14    @NotBlank(message = "{orgEmail.notBlank}")
15    @Size(max = 80, message = "{orgEmail.size}")
16    private String email;
17
18    @NotBlank(message = "{orgCountry.notBlank}")
19    private String country;
20
21    @OneToMany(mappedBy = "organization")
22    private List<User> users;
23
24    @Override
25    public void setId(Long id) {
26      super.setId(id);
27    }
28
29    // getters e setters omitidos
30  }
```

3.3.12 HTML5

A função do HTML na programação *web* sofreu alterações ao longo dos anos e hoje essa linguagem deve ser utilizada para estruturar o conteúdo das páginas e descrever o significado do conteúdo presente nesses documentos por meio de *tags* semânticas. Características visuais e comportamentos devem ser feitos usando CSS (*Cascading Style Sheets*) e JavaScript.

O uso das *tags* semânticas descrevem aos *user-agents* (*web browser*) o significado dos conteúdos que estão sendo inseridos em cada uma das divisões da página, organizando e padronizando o desenvolvimento. Como exemplos podem ser citados o uso da *tag* <main> que irá definir o conteúdo principal da página e a *tag* <nav> para indicar que o conteúdo é uma seção de navegação.

3.3.13 Bootstrap

> Bootstrap é o mais popular *framework* JavaScript, HTML e CSS voltado para o desenvolvimento de *sites* e aplicações *web* responsivas e alinhadas com a filosofia *mobile first*. Torna o desenvolvimento *front-end* muito mais rápido e fácil. Indicado para desenvolvedores de todos os níveis de conhecimento, dispositivos de todos os tipos e projetos de todos os tamanhos (SILVA, 2015).

O *site* do Bootstrap[18] o define como um poderoso, elegante e intuitivo *framework front-end* que possibilita um desenvolvimento *web* de modo ágil e fácil.

Um *template* básico para construção de uma página HTML utilizando o *framework* Bootstrap é apresentado no Código 3.3.

Código 3.3 - *Template* básico para uso do framework Bootstrap.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Bootstrap Template</title>

    <!-- Bootstrap -->
    <link href="css/bootstrap.min.css" rel="stylesheet">
  </head>
  <body>

    <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/
       jquery.min.js"></script>
    <!-- Include all compiled plugins (below), or include individual
       files as needed -->
    <script src="js/bootstrap.min.js"></script>
  </body>
</html>
```

3.3.14 Thymeleaf

O Thymeleaf é um moderno *template engine* Java para ambientes *web* e *standalone*, capaz de processar HTML, XML, JavaScript, CSS e até texto simples (THYMELEAF, 2017).

"Essa biblioteca foi criada para facilitar a criação da camada de *view* com uma forte integração com o Spring, e uma boa alternativa ao JSP. O principal objetivo do Thymeleaf é prover uma forma elegante e bem formatada para criarmos nossas páginas" (JUNIOR, 2016).

O dialeto do Thymeleaf é bem poderoso e baseia-se no conceito de *natural templates* para injetar sua lógica nos *templates* de forma a não afetá-los. *Natural templates* são *templates*

[18] http://getbootstrap.com/

que podem ser abertos diretamente nos navegadores e que ainda são exibidos corretamente como páginas da *web* (THYMELEAF, 2017).

O Thymeleaf foi utilizado na construção das interfaces gráficas do ambiente Sentinel por sua proximidade com HTML como pode ser visto no Código 3.4, que apresenta o *template* utilizado para construção das páginas.

Código 3.4 - *Template* Thymeleaf para renderizar as páginas do ambiente Sentinel.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
  xmlns:th="http://www.thymeleaf.org"
  xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
  xmlns:sec="http://www.thymeleaf.org/extras/spring-security">

<head>
<meta charset="UTF-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="viewport" content="width=device-width, initial-scale=1" />

<title>#</title>

<link rel="stylesheet" type="text/css" th:href="@{/layout/vendors/
    bootstrap/css/bootstrap.min.css}"/>
<link rel="stylesheet" type="text/css" th:href="@{/layout/css/vendors.
    min.css}" />
<link rel="stylesheet" type="text/css" th:href="@{/layout/css/sentinel.
    css}"/>
<link rel="stylesheet" type="text/css" th:href="@{/layout/css/
    application.css}"/>
<link rel="stylesheet" type="text/css" th:href="@{/layout/vendors/
    font-awesome/css/font-awesome.min.css}" />

<!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and
    media queries -->
<!-- WARNING: Respond.js doesn't work if you view the page via file://
    -->
<!--[if lt IE 9]>
    <script src="https://oss.maxcdn.com/html5shiv/3.7.2/html5shiv.min.js
        "></script>
    <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></
        script>
  <![endif]-->
</head>
<body>

```

```
29    <input type="hidden" name="_csrf" th:value="${_csrf.token}" />
30    <input type="hidden" name="_csrf_header" th:value="${_csrf.headerName}
          " />
31
32    <header layout:include="layout/Header">#</header>
33
34    <section class="aw-layout-content js-content" layout:fragment="
          content"></section>
35
36    <!-- footer class="aw-layout-footer js-content">
37      <div class="container-fluid">
38        <span class="aw-footer-disclaimer">&copy; 2016 Sentinel.</span>
39      </div>
40    </footer -->
41
42    <script th:src="@{/layout/vendors/jquery/jquery-3.1.1.min.js}"></s
          cript>
43    <script th:src="@{/layout/vendors/bootstrap/js/bootstrap.min.js}"></s
          cript>
44    <!-- script th:src="@{/layout/js/vendors.min.js}"></script -->
45    <script th:src="@{/layout/js/sentinel.js}"></script>
46    <script th:src="@{/layout/js/application.js}"></script>
47
48  </body>
49  </html>
```

3.3.15 D3.js

D3.js é uma biblioteca JavaScript que permite manipular elementos de uma página da *web* no contexto de um conjunto de dados. Esses elementos podem ser introduzidos, removidos ou editados de acordo com o conteúdo do conjunto de dados (DEWAR, 2012).

Por não ser uma estrutura monolítica, que fornece todos os recursos concebíveis, D3.js oferece uma flexibilidade extraordinária, expondo todas as capacidades dos padrões da *web*, como HTML, SVG (*Scalable Vector Graphics*) e CSS. Com sobrecarga mínima, D3.js é extremamente rápido, suportando grandes conjuntos de dados e comportamentos dinâmicos para interação e animação Bostock (201-?).

Os resultados produzidos com a execução dos experimentos dentro do ambiente Sentinel podem ser visualizados através de gráficos gerados com a biblioteca D3.js sempre que o formato dos arquivos de saída for suportado.

3.3.16 JasperReports

JasperReports é uma biblioteca de renderização de conteúdo que deve ser incorporada em outro aplicativo Java do lado do cliente ou do servidor (JASPERSOFT, 2011).

A biblioteca permite exportar relatórios em diversos formatos, como PDF (*Portable Document Format*), HTML, XML, XLS (*EXceL Spreadsheet*), etc. e aceita diversas formas de entrada de dados, tais como um arquivo XML ou CSV (*Comma-separated values*), banco de dados, uma coleção de objetos em memória, etc. (MARTINS, 2010).

JasperReports foi utilizado na construção de relatórios gerenciais que podem ser utilizados pelos administradores para auxiliar na gestão do ambiente.

3.4 Infraestrutura de *Hardware*

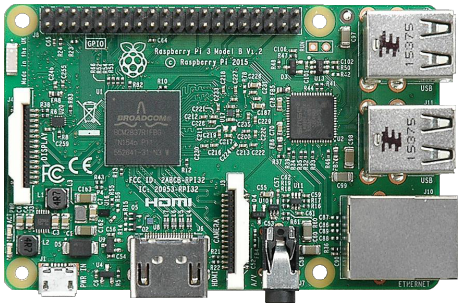
Nesta seção será apresentada a infraestrutura de *hardware* utilizada para operacionalização do ambiente, que consiste basicamente de microcomputadores Raspberry Pi.

3.4.1 Raspberry Pi

O Raspberry Pi, também conhecido como RPi, é um computador do tamanho de um cartão de crédito, que se conecta a um monitor de computador ou TV, e usa um teclado e um mouse padrão. Desenvolvido no Reino Unido pela Fundação Raspberry Pi tem todo o *hardware* integrado numa única placa (FOUNDATION, 201-?b).

Esse pequeno dispositivo é capaz de fazer tudo que você espera de um computador *desktop*, como navegar na *internet*, reproduzir vídeo em alta definição, processamento de texto, entre outros. O Raspberry Pi tem sido utilizado atualmente em uma ampla gama de projetos. A Figura 3.11 apresenta a visão superior de um microcomputador Raspberry Pi 3 Model B onde é possível observar a sua estrutura e componentes.

Figura 3.11 - Visão superior de um Raspberry Pi 3 Model B.



Fonte: Foundation (201-?a).

Os motivos que levaram a adoção do Raspberry Pi para compor a infraestrutura de *hardware* do ambiente foram basicamente quatro: (1) baixo custo, (2) baixo consumo de energia, (3) dimensões e (4) peso.

Nesta pesquisa, foram utilizados dois modelos do Raspberry Pi, o RPi 2 Model B e o RPi 3 Model B. Um breve comparativo da configuração dos modelos é apresentado na Tabela 3.5.

Tabela 3.5 - Configuração dos modelos Raspberry Pi utilizados.

Característica	**Raspberry Pi 2 B**	**Raspberry Pi 3**
Chip:	Broadcom BCM2836	Broadcom BCM2837
Arquitetura:	ARMv7-A	ARMv8-A
CPU Clock:	0.9GHz	1.2GHz
Tipo Core:	ARM Cortex A7	ARM Cortex A53 64-bit
Nº Core;	4	
GPU:	Broadcom VideoCore IV	
GPU Freq:	250MHz	400 MHz
Memória (SDRAM):	1Gb DDR2 compartilhada com GPU	
Memory Freq:	400MHz	
USB (2.0):	4	
Saída de Vídeo:	HDMI	
Saída de Audio:	Conector de 3.5mm, HDMI	
Rede:	10/100 Mbit/s	
Wireless:	-	802.11n
Bluetooth:	-	4.1
Armazenamento:	MicroSD 2.0	
Dimensões:	85.6mm X 56.5mm	
Peso:	45 g	
Fonte de energia:	5V2A	5V2.5A
Consumo de energia:	800 mA	

Fonte: Produção do autor (2017).

Não é proposta deste trabalho analisar o desempenho do Raspberry Pi na operacionalização de um ambiente de *e-Science* dadas as suas limitações. Seu uso se deu exclusivamente com o intuito de viabilizar a implementação do ambiente, reduzindo o custo de construção de uma infraestrutura deste tipo.

3.4.2 *Cluster*

Para operacionalização do ambiente Sentinel foi construído um *cluster* contendo 5 (cinco) nós, sendo 2 (dois) nós constituídos pelo RPi 2 Model B e 3 (três) nós pelo RPi 3 Model B.

Cada um dos nós do *cluster* opera com o Raspbian, uma versão otimizada do sistema operacional Debian para uso no Raspberry Pi. Os nós contam ainda com um cartão microSD de 16Gb cada para armazenamento local. A configuração final de cada um dos Raspberry Pi que compõe a infraestrutura é apresentada na Tabela 3.6.

Tabela 3.6 - Configuração final dos dispositivos Raspberry Pi.

Nó	**rpi01**	**rpi02**	**rpi03**	**rpi04**	**rpi05**
Raspberry Pi 2	x	x			
Raspberry Pi 3			x	x	x
Sistema Operacional	Raspbian				
Armazenamento local	microSD - 16Gb				
SGBD	x				
HDD (Storage)	x				
Servidor Web	x				
Docker Engine	x	x	x	x	x
Docker Swarm (Manager)	x				

Fonte: Produção do autor (2017).

Os equipamentos utilizados na construção do *cluster*, assim como a quantidade utilizada de cada equipamento, são apresentados na Tabela 3.7.

Tabela 3.7 - Equipamentos utilizados na construção do *cluster*.

Equipamento	**Quantidade**	**Descrição**
Fonte	1x	5.0V 300W
HDD	1x	100Gb
Roteador	1x	DLink 10/100Mbps - 8 portas
Cartão SD	5x	16Gb
Cooler	3x	5.0V 0.14A
Conector microUSB	5x	
Cabos de rede	5x	vários tamanhos

Fonte: Produção do autor (2017).

O passo a passo para a construção e configuração do *cluster* utilizado é abordado com detalhes no Apêndice A.

4 O AMBIENTE SENTINEL

Sentinel é um ambiente de *e-Science* projetado para oferecer a pesquisadores uma forma de gerenciar e monitorar seus experimentos científicos de maneira prática, onde poderão concentrar em um único ambiente toda sua pesquisa e produção científica sendo possível acessá-la de qualquer dispositivo conectado à *internet*.

Todas as interfaces gráficas do ambiente Sentinel seguem o mesmo padrão apresentado na Figura 4.1. Na parte superior da interface são exibidas informações do usuário que está conectado ao ambiente. Já na lateral esquerda, encontra-se o menu que dá acesso as funcionalidades do ambiente, personalizado para cada usuário de acordo com o seu perfil de acesso. Todo o conteúdo é apresentado aos usuários na parte central da interface.

Figura 4.1 - Interface gráfica padrão do ambiente Sentinel.

Fonte: Produção do autor (2017).

Por se tratar de uma aplicação *web* responsiva toda a interface gráfico do ambiente Sentinel se adapta ao dispositivo utilizado pelo usuário para acessá-lo.

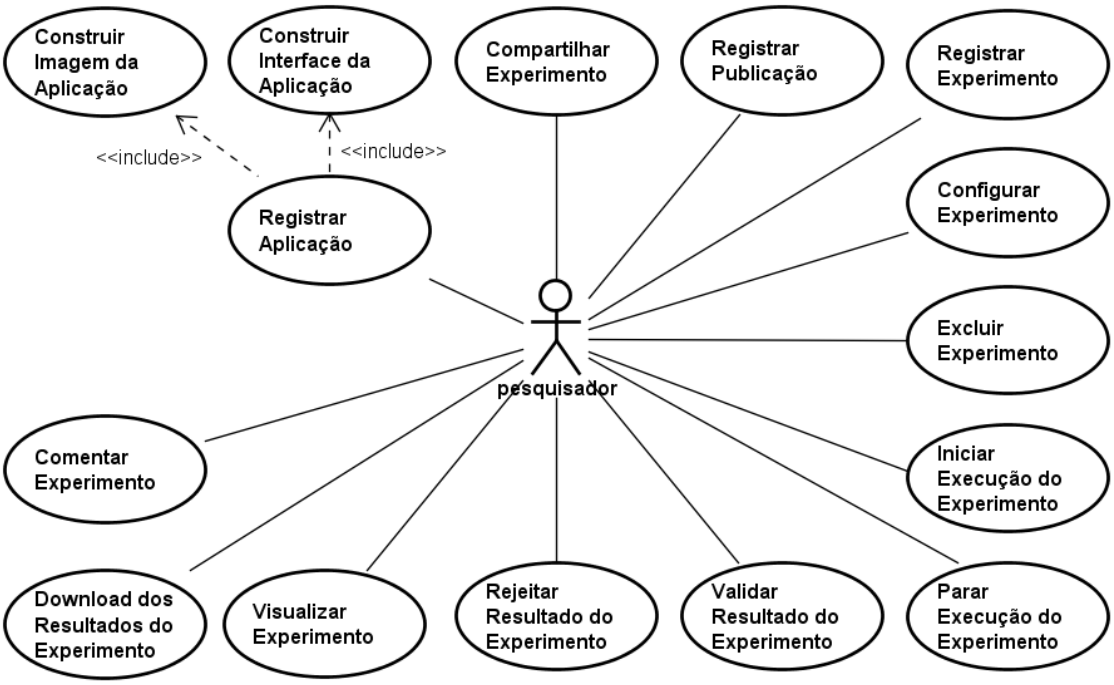
4.1 Requisitos do Sistema

Os requisitos do sistema foram levantados através da análise realizada nas ferramentas identificadas e apresentadas na Seção 3.2.

A Figura 4.2 apresenta o diagrama de Casos de Uso que representa as funcionalidades que podem ser requisitadas pelo Pesquisador dentro do ambiente Sentinel. O diagrama

completo pode ser visto na Figura B.1 e a descrição de cada um dos casos de uso é apresentada na Subseção B.1.1.

Figura 4.2 - Diagrama de Casos de Uso - Funcionalidades do pesquisador.

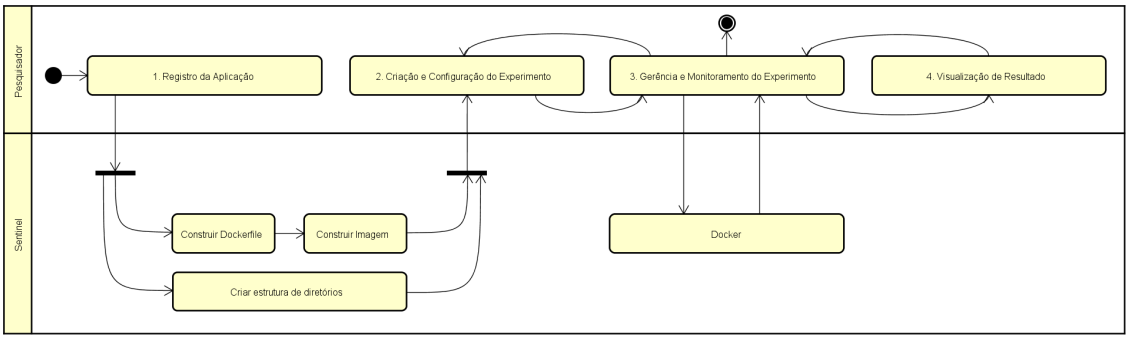


Fonte: Produção do autor (2017).

4.2 Fluxo de Trabalho

O fluxo de trabalho do ambiente Sentinel está dividido em quatro etapas, como apresentado no diagrama de Atividades da Figura 4.3. Cada etapa deste fluxo segue descrita nas subseções seguintes.

Figura 4.3 - Fluxo de trabalho do Sentinel.



Fonte: Produção do autor (2017).

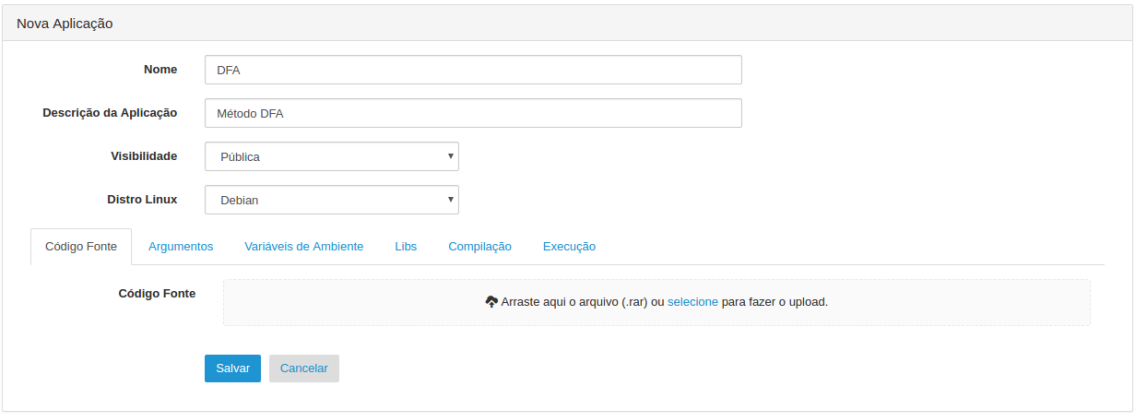
4.2.1 Etapa 1 - Registro da Aplicação

A primeira etapa é a inclusão da aplicação no ambiente. Isso possibilita ao pesquisador ter seu próprio conjunto de ferramentas para realizar suas pesquisas em sua área de trabalho.

O ambiente Sentinel oferece suporte a execução de aplicações, baseadas em Linux, desenvolvidas pelos pesquisadores. Por definição todas as aplicações devem ler seus arquivos de entrada a partir da pasta `INPUTS` e os arquivos de saída devem ser escritos na pasta `OUTPUTS`.

A inclusão da aplicação acontece através da interface gráfica `Nova Aplicação`, mostrada na Figura 4.4, onde o pesquisador informa todos os dados necessários para a implantação da aplicação no ambiente. Estas informações estão agrupadas pelas guias `Código Fonte`, `Argumentos`, `Variáveis de Ambiente`, `Libs`, `Compilação` e `Execução`.

Figura 4.4 - Interface gráfica `Nova Aplicação`.



Fonte: Produção do autor (2017).

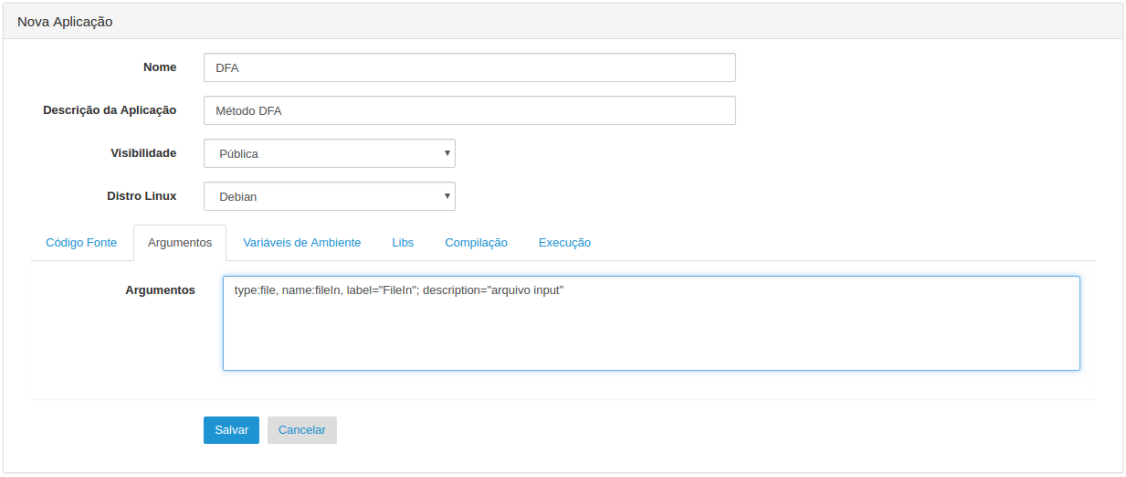
Para cada aplicação registrada no ambiente, o pesquisador deve obrigatoriamente informar um nome, definir sua visibilidade como pública ou privada e selecionar em qual distribuição Linux a aplicação será executada. Se uma aplicação é definida como pública outros pesquisadores podem utilizá-la em suas áreas de trabalho. Opcionalmente pode ser fornecida uma descrição sobre a aplicação que está sendo cadastrada.

O *upload* do código fonte para os servidores do ambiente é feito através da guia `Código`

`Fonte`, onde o pesquisador pode selecionar ou arrastar um arquivo `.rar` contendo o código da aplicação.

Na guia `Argumentos`, apresentada na Figura 4.5, são definidos os argumentos de entrada para a execução da aplicação. Os argumentos devem ser inseridos um por linha e para cada argumento podem ser informados os atributos `type`, `description`, `label`, `default` e `name` que devem ser separados por " , " ou " ; " não importando a ordem de inclusão. O atribuído `type`, obrigatório assim como o atributo `name`, pode assumir um dos seguintes valores: `text`, `number`, `date`, `file` ou `options`.

Figura 4.5 - Interface gráfica `Nova Aplicação` - guia `Argumentos`.

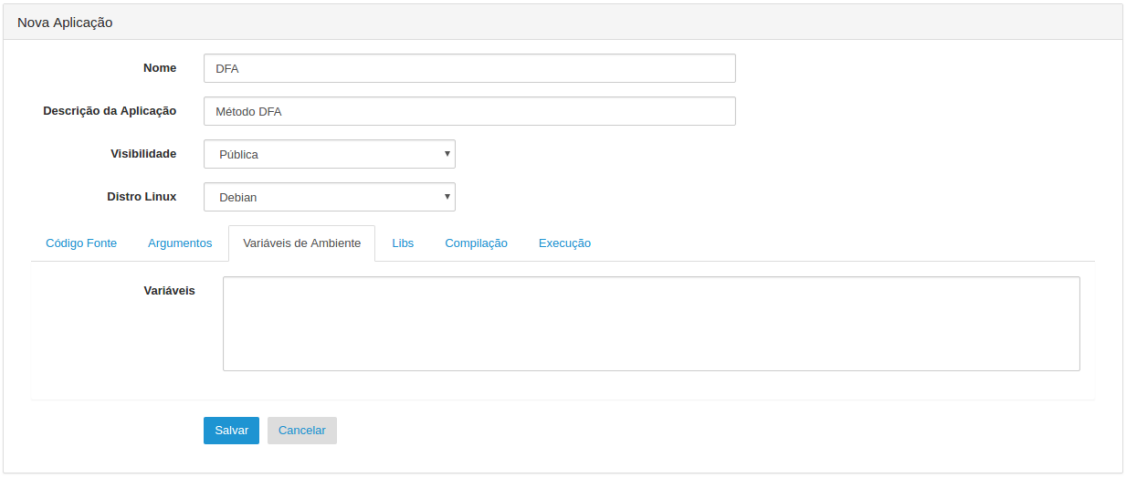


Fonte: Produção do autor (2017).

A aplicação DFA (*Detrended Fluctuation Analysis*), cujo registro é apresentado na Figura 4.5, necessita de apenas um argumento para sua execução. Neste caso, o argumento cadastrado é do tipo `file`, recebeu o nome `fileIn`, rótulo `FileIn` e a descrição "`arquivo input`".

Informações relacionadas às configurações do sistema operacional necessárias para a execução da aplicação são informadas através da guia `Variáveis de Ambiente`, apresentada na Figura 4.6. Assim como os argumentos, as variáveis de ambiente devem ser inseridas uma por linha. No caso da aplicação DFA não é necessária nenhuma variável de ambiente.

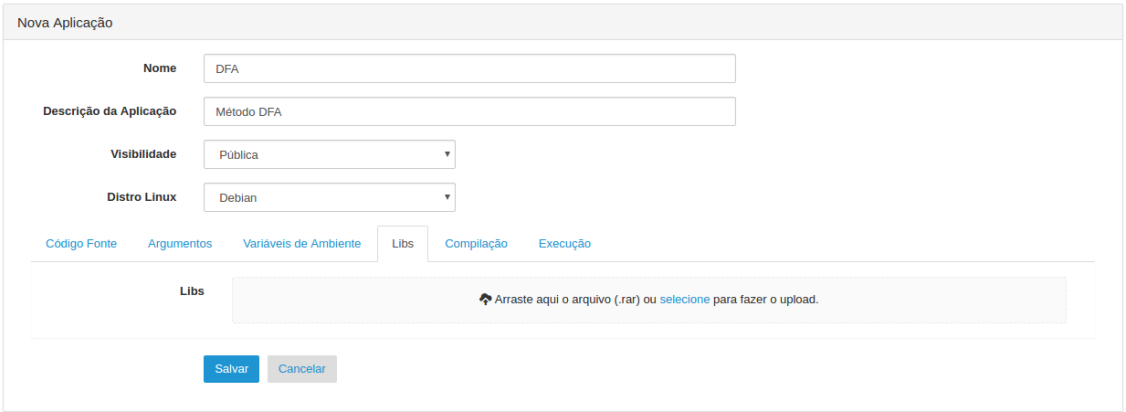
Figura 4.6 - Interface gráfica `Nova Aplicação` - guia `Variáveis de Ambiente`.



Fonte: Produção do autor (2017).

Se existir a necessidade de incluir alguma biblioteca especial para a execução da aplicação esta pode ser incorporada ao ambiente através da guia `Libs`, apresentada na Figura 4.7.

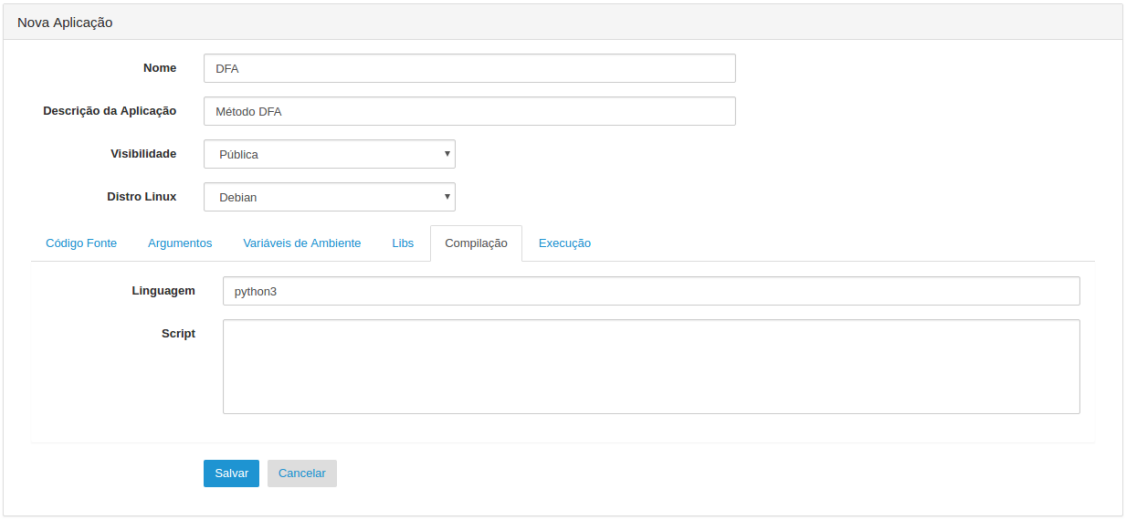
Figura 4.7 - Interface gráfica `Nova Aplicação` - guia `Libs`.



Fonte: Produção do autor (2017).

A guia `Compilação` é utilizada para informar a linguagem de programação utilizada na construção da aplicação e o *script* de compilação da aplicação, conforme apresentada na Figura 4.8. No caso da aplicação DFA a linguagem utilizada é a Python, uma linguagem interpretada que não necessita de um *script* de compilação.

Figura 4.8 - Interface gráfica `Nova Aplicação` - guia `Compilação`.



Fonte: Produção do autor (2017).

O ambiente Sentinel oferece suporte a diversas linguagens de programação, pois cada pesquisador com acesso ao ambiente tem suas preferências de linguagem que podem ser influenciadas pela sua formação, conhecimento ou área de atuação. Restringir o uso a uma ou outra linguagem de programação poderia interferir na possibilidade de novas aplicações serem acopladas ao ambiente por não oferecer suporte a linguagem escolhida pelo pesquisador.

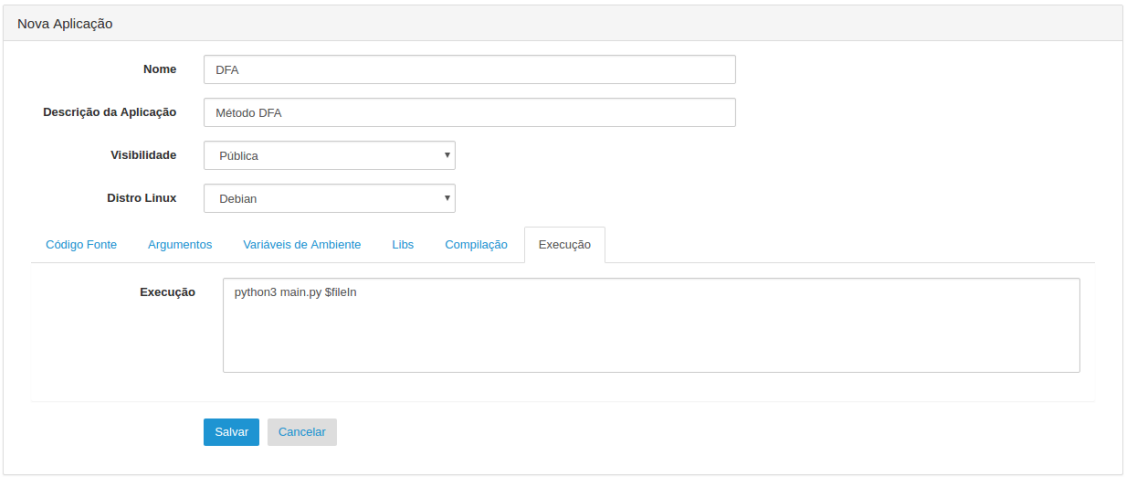
Para cada linguagem de programação suportada pelo ambiente, existe um arquivo de configuração como o apresentado no Código 4.1, que representa como a linguagem deve ser instalada e configurada na imagem que será criada.

Código 4.1 - Configuração da linguagem Python3.

```
1  RUN apt-get update &&\
2     apt-get -y upgrade
3
4  RUN apt-get -y install python3
5  RUN apt-get -y install python3-dev
6  RUN apt-get -y install python3-numpy
7  RUN apt-get -y install python3-matplotlib
8  RUN apt-get -y install python3-scipy
9  RUN apt-get -y install python3-tk
10 RUN apt-get -y install ipython
```

Na guia `Execução`, o pesquisador pode inserir um *script* de execução para a aplicação, conforme apresentado na Figura 4.9. Para a execução da aplicação pode ser necessário que alguns argumentos sejam informados. Neste caso, estes argumentos devem estar relacionados aos que foram informados na guia `argumentos`. Cada argumento criado anteriormente deve ser relacionado através do uso de um " $ " seguido do valor do atributo `name` dado ao argumento, como apresentado para o argumento `fileIn`.

Figura 4.9 - Interface gráfica `Nova Aplicação` - guia `Execução`.



Fonte: Produção do autor (2017).

Todas as aplicações criadas possuem uma estrutura de diretórios como a apresentada na Figura 4.10. Na pasta `config`, será criado um arquivo de configuração chamado Dockerfile, a pasta `source` onde são armazenados os arquivos de código fonte da aplicação e a pasta `libs` para armazenar bibliotecas extras necessárias. A pasta `experiments` é utilizada para armazenar todos os dados relacionados aos experimentos criados utilizando esta aplicação.

Figura 4.10 - Estrutura de diretórios de uma aplicação.

```
/
└── app_VJ234PKQD34/
    ├── config/
    │   ├── libs/
    │   ├── source/
    │   │   └── DFA.rar
    │   └── Dockerfile
    └── experiments/
```

Para a criação do arquivo Dockerfile, é utilizado o *template* apresentado no Código 4.2. Este arquivo é a base para a construção da imagem utilizada na execução dos experimentos e contém todas as instruções necessárias para a construção da aplicação a partir dos dados informados pelo pesquisador.

Código 4.2 - *Template* do arquivo Dockerfile para criação das imagens.

```
## SET DISTRO
FROM $app.distro

## SET LABELS
LABEL maintainer="Sentinel"
LABEL application="$app.name"
LABEL description="$app.description"

## SET WORKDIR
WORKDIR /code

RUN mkdir /code/inputs
RUN mkdir /code/outputs

## SET LANGUAGE PROPERTIES
include ("$path/$app.language")

## SET ENV_VARS
ENV $envVar

## COPY SOURCE
ADD $source /code

## COPY LIBS
ADD $libs /code

## COMPILE APPLICATION
RUN $command

RUN apt-get clean && rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*
```

O resultado final da criação do arquivo Dockerfile para a aplicação DFA pode ser visto no Código 4.3.

Código 4.3 - Dockerfile para criação da imagem da aplicação DFA.

```
FROM debian

LABEL maintainer="Sentinel"
LABEL application="DFA"
LABEL description="DFA"

WORKDIR /code

RUN mkdir /code/inputs
RUN mkdir /code/outputs

RUN apt-get update &&\
   apt-get -y upgrade

RUN apt-get -y install python3
RUN apt-get -y install python3-dev
RUN apt-get -y install python3-numpy
RUN apt-get -y install python3-matplotlib
RUN apt-get -y install python3-scipy
RUN apt-get -y install python3-tk
RUN apt-get -y install ipython

ADD source/a48f4706-1637-46c1-af13-871a582f637d_dfa.tar /code

RUN apt-get clean && rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*
```

A Figura 4.11 apresenta o resultado da execução do Dockerfile com a imagem `app_dfa` criada. O nome da imagem é gerado a partir do nome dado à aplicação pelo pesquisador com o acréscimo do prefixo `app_`.

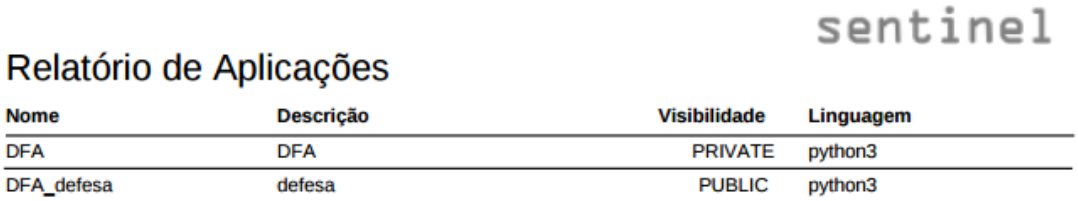
Figura 4.11 - Imagem docker criada para a aplicação DFA.

```
sentinel@yoda:~/sentinel/app_dfa$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
app_dfa             latest              3c71a709a361        13 seconds ago      1.302 GB
```

Fonte: Produção do autor (2017).

As aplicações cadastradas pelo pesquisador podem ser listadas a partir de um relatório. Um exemplo do Relatório de Aplicações é apresentado na Figura 4.12.

Figura 4.12 - Exemplo de Relatório de Aplicações cadastradas.



Fonte: Produção do autor (2017).

4.2.2 Etapa 2 - Criação e Configuração do Experimento

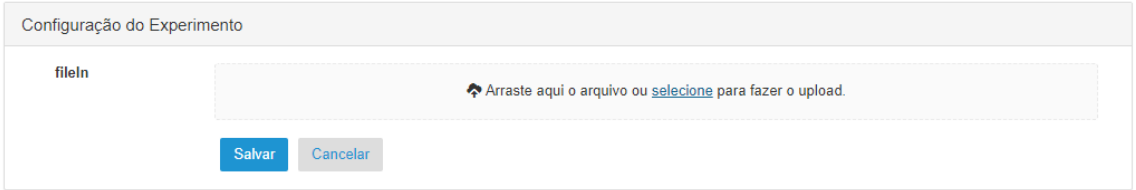
Após registrada pelo pesquisador a aplicação está pronta para ser executada. As execuções criadas nesta etapa são chamadas de experimentos e são criadas a partir da interface gráfica `Novo Experimento`, apresentada na Figura 4.13.

Figura 4.13 - Interface gráfica `Novo Experimento`.

Fonte: Produção do autor (2017).

Todos os experimentos criados são exibidos na interface `Dashboard` do Pesquisador, Figura 4.17, onde eles poderão ser configurados. A interface de configuração, Figura 4.14, é criada automaticamente pelo ambiente com informações inseridas pelo pesquisador no cadastro da aplicação. Nessa interface, o pesquisador configura o experimento atribuindo valor aos argumentos de entrada. No caso dos experimentos criados para a aplicação DFA é necessário somente informar o arquivo que será processado.

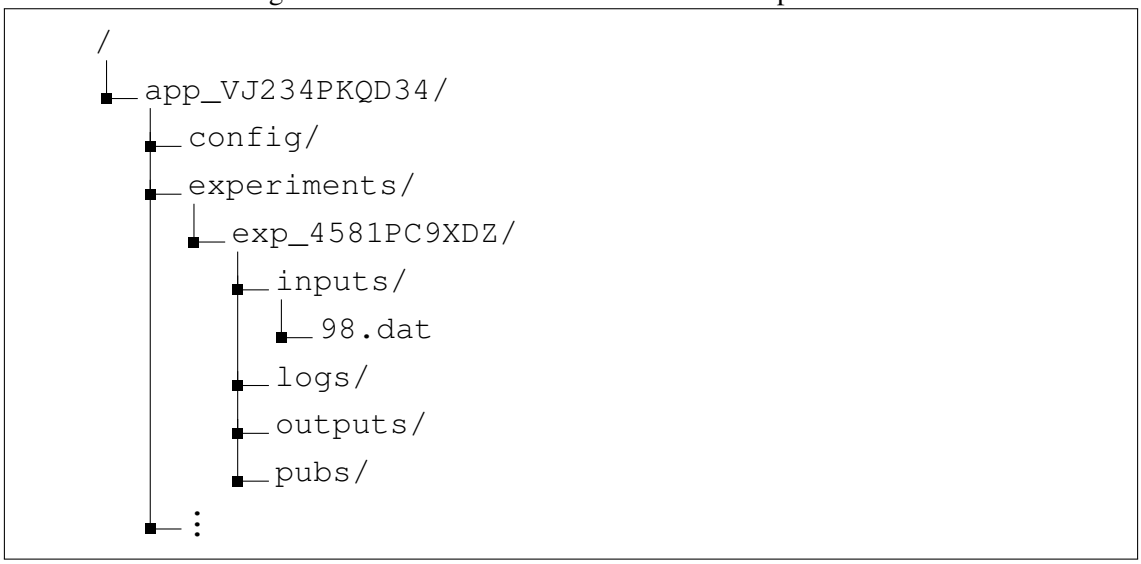
Figura 4.14 - Interface gráfica `Configuração do Experimento`.



Fonte: Produção do autor (2017).

Com a criação dos experimentos novos diretórios são criados na estrutura apresentada na Figura 4.10, que ficará como apresenta a Figura 4.15. Na pasta `inputs` serão armazenados todos os arquivos necessários para a execução do experimento. Os *logs* da execução são mantidos na pasta `logs`. A pasta *outputs* é destinada aos arquivos resultantes da execução do experimento. Publicações oriundas do experimento podem ser importadas para o ambiente e ficarão reunidas na pasta `pubs`.

Figura 4.15 - Estrutura de diretórios de um experimento.



O pesquisador pode ter acesso a informações sobre todos os seus experimentos através de um relatório. A Figura 4.16 apresenta um exemplo do Relatório de Experimentos.

Figura 4.16 - Exemplo de Relatório de Experimentos criados.

sentinel

Relatório de Experimentos

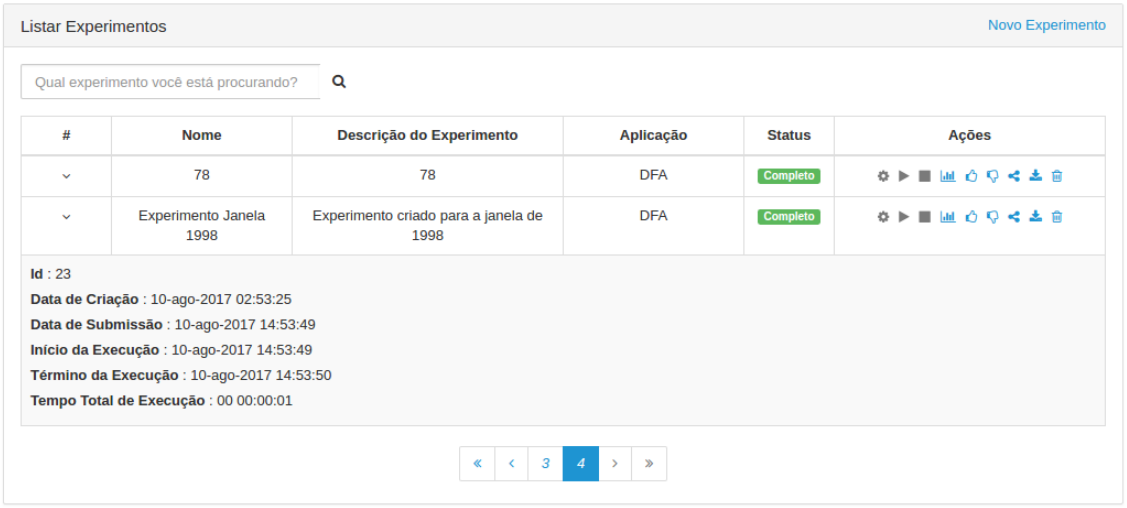
Nome Data de Criação	Descrição Data de	Início da Execução	Fim da Execução	Aplicação Status
experimento 11/08/17 02:52	experimento defesa 11/08/17 14:54	11/08/17 14:54	11/08/17 14:54	DFA COMPLETE
92 10/08/17 10:26	92 10/08/17 22:27	10/08/17 22:27	10/08/17 22:27	DFA COMPLETE
90 10/08/17 22:26	90 null	null	null	DFA QUEUED

Fonte: Produção do autor (2017).

4.2.3 Etapa 3 - Gerência e Monitoramento do Experimento

A gerência e o monitoramento dos experimentos criados são feitos pelo pesquisador através da interface gráfica `Dashboard`, mostrada na Figura 4.17. Detalhes sobre cada uma das etapas de execução do experimento são exibidos selecionando a coluna # na linha referente ao experimento desejado.

Figura 4.17 - `Dashboard` - Pesquisador



Fonte: Produção do autor (2017).

Através desta interface gráfica, o pesquisador tem acesso ainda a um conjunto de informações sobre cada um dos experimentos criados. Para monitorar seus experimentos, o pesquisador tem à sua disposição as informações apresentadas na Tabela 4.1.

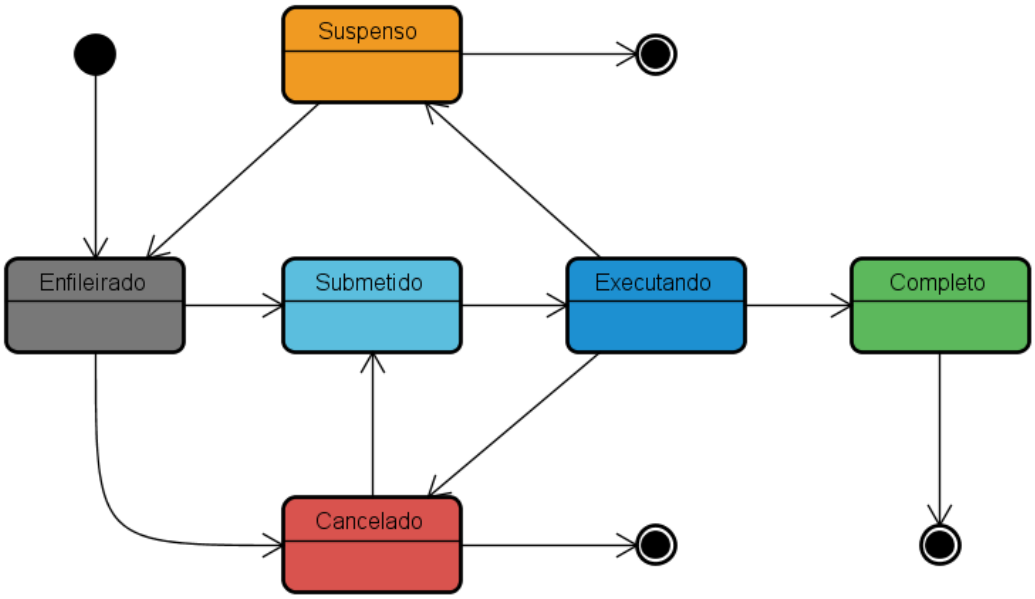
Tabela 4.1 - Descrição das colunas da interface gráfica `Dashboard` - Pesquisador.

	Coluna	Descrição
1	#	Informações referentes a cada etapa de execução do experimento.
2	Nome	Nome dado ao experimento.
3	Descrição do Experimento	Descrição sobre o experimento.
4	Aplicação	Aplicação utilizada na execução do experimento.
5	Status	Situação atual do experimento.

Fonte: Produção do autor (2015).

Durante seu ciclo de execução, cada experimento pode assumir diferentes *Status*, como mostra o diagrama de Máquina de Estados da Figura 4.18.

Figura 4.18 - Diagrama de Máquina de Estados - *Status* do Experimento.



Fonte: Produção do autor (2015).

A mudança de *Status* acontece através de uma transição. De acordo com Guedes (2011), uma transição é um evento que causa uma mudança no estado de um elemento. A descrição de cada *Status* é apresentada na Tabela 4.2.

Tabela 4.2 - Descrição dos possíveis *Status* que um experimento pode assumir.

	Status	**Descrição**
1	Enfileirado	experimento criado e aguardando ação do usuário.
2	Submetido	experimento submetido pelo usuário para execução.
3	Suspenso	experimento cancelado por apresentar alguma inconsistência.
4	Cancelado	execução do experimento interrompida pelo usuário.
5	Executando	experimento em execução.
6	Completo	execução concluída.

Fonte: Produção do autor (2015).

A Tabela 4.3 apresenta os comandos existentes no ambiente Sentinel que possibilitam ao pesquisador gerenciar seus experimentos. Estes comandos estão dispostos na coluna `Ações` da interface gráfica `Dashboard` - Pesquisador.

Tabela 4.3 - Comandos para gerenciar os experimentos.

	Ação	Descrição do comando
1		Configurar Experimento.
2		Iniciar Execução.
3		Cancelar Execução.
4		Visualizar Resultado.
5		Validar Resultado.
6		Rejeitar Resultado.
7		Compartilhar Resultado.
8		Download do Resultado.
9		Excluir Experimento.

Fonte: Produção do autor (2015).

Cada experimento após submetido para execução pelo pesquisador é processado em um ambiente isolado fazendo uso de um *container* Docker. A Figura 4.19 mostra o *container* referente ao experimento `exp_BU0JKEMD4Q` em execução.

Figura 4.19 - Execução do comando “docker ps” para listar todos os *containers* em execução.

```
sentinel@yoda:~/sentinel/app_dfa/experiments/exp_dfa_win00$ docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED
     STATUS                      PORTS               NAMES
3fd49e86926a        app_dfa             "python3 main.py 00.d"   7 seconds ago
     Exited (0) 5 seconds ago                        exp_dfa_win00
```

Fonte: Produção do autor (2017).

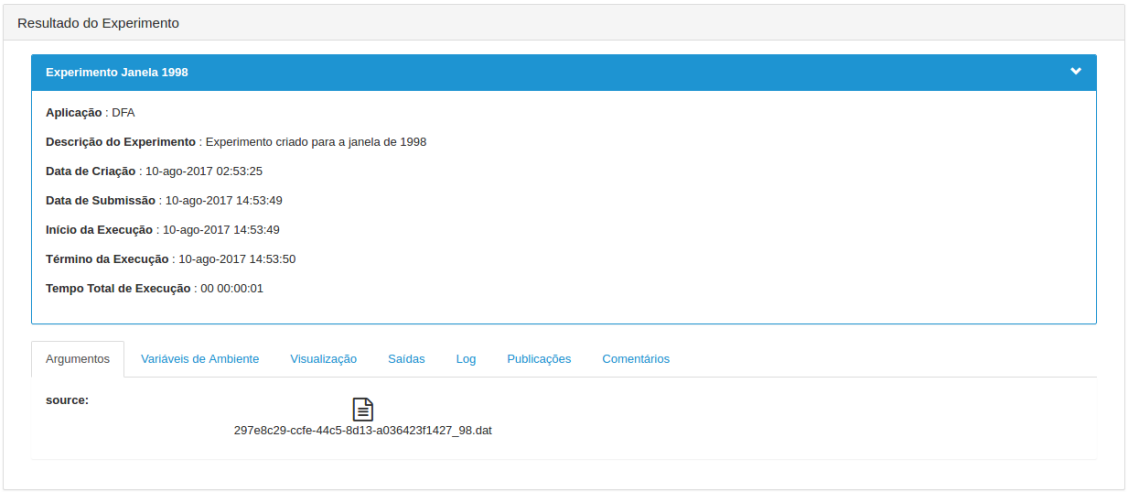
4.2.4 Etapa 4 - Visualização de Resultado

Para analisar os resultados obtidos com a execução de um experimento, o ambiente Sentinel fornece um conjunto de ferramentas aos pesquisadores. Após um experimento ser visualizado e avaliado, seu resultado pode ser aceito ou rejeitado pelo pesquisador proprietário do experimento. Os arquivos gerados por um experimento que tenha o seu resultado rejeitado serão removidos do ambiente, já os resultados aceitos ficam à disposição do pesquisador. Estas ações podem ser aplicadas a cada experimento.

Para facilitar o acesso aos dados produzidos pela execução do experimento, a interface gráfica `Resultado do Experimento` está organizada nas guias: `Argumentos`, `Variáveis de Ambiente`, `Visualização`, `Saídas`, `Log`, `Publicações` e `Comentários`.

Informações sobre o processamento do experimento podem ser encontradas no topo da interface clicando sobre o Nome do experimento, exibido na barra azul, para exibir ou ocultar esse conteúdo. Um exemplo é exibido na Figura 4.20.

Figura 4.20 - Exibição de informações sobre o experimento.



Fonte: Produção do autor (2017).

Na guia `Argumentos`, exibida na Figura 4.21, é mostrada a configuração utilizada para execução do experimento. Em outras palavras, nesta guia são exibidos os valores definidos pelo pesquisador para cada argumento de entrada para a execução do experimento.

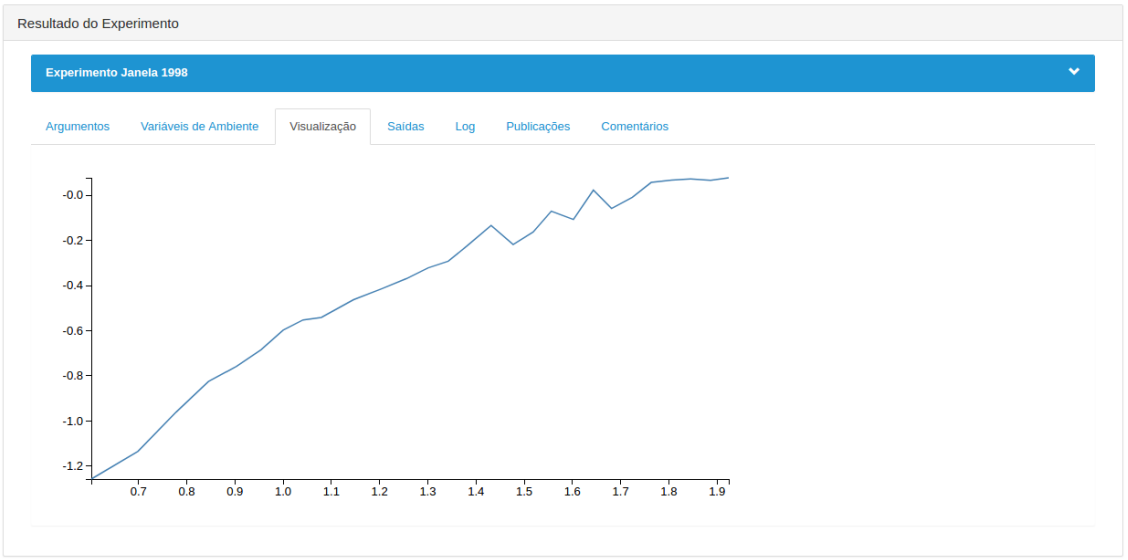
Figura 4.21 - Interface gráfica `Resultado do Experimento` - guia `Argumentos`.

Fonte: Produção do autor (2017).

Na guia `Variáveis de Ambiente` são exibidas as variáveis relacionadas a configurações do sistema operacional. No caso da execução de experimentos que utilizam a aplicação DFA não é necessária a definição de nenhuma variável de ambiente.

Na guia `Visualização`, é feita a plotagem dos dados produzidos na execução do experimento sempre que o formato do dado gerado for suportado, como exibido na Figura 4.22.

Figura 4.22 - Interface gráfica `Resultado do Experimento` - guia `Visualização`.



Fonte: Produção do autor (2017).

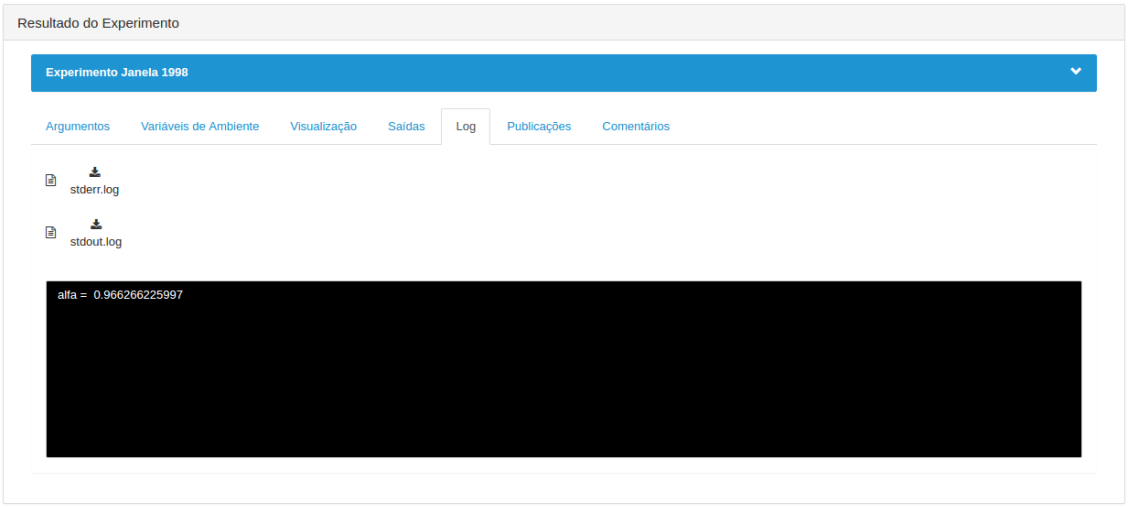
Todos os arquivos produzidos durante a execução do experimento ficam à disposição dos pesquisadores para *download* através da guia `Saídas`, como pode ser visualizado na Figura 4.23.

Figura 4.23 - Interface gráfica `Resultado do Experimento` - guia `Saídas`.

Fonte: Produção do autor (2017).

A guia `Log`, apresentada na Figura 4.24, oferece ao pesquisador detalhes sobre o processamento do experimento. Acessando essas informações o pesquisador pode realizar uma auditoria e/ou diagnosticar problemas apresentados pela aplicação durante sua execução. É possível também realizar o *download* dos arquivos de *log* da execução.

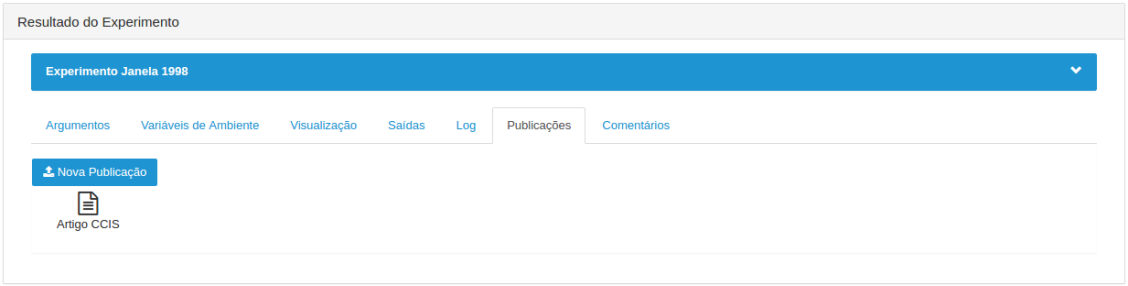
Figura 4.24 - Interface gráfica `Resultado do Experimento` - guia `Log`.



Fonte: Produção do autor (2017).

Além dos resultados, outros produtos, como artigos, apresentações e outros, gerados a partir do experimento podem ser compartilhados pelo pesquisador através da guia `Publicações`, conforme apresentado na Figura 4.25.

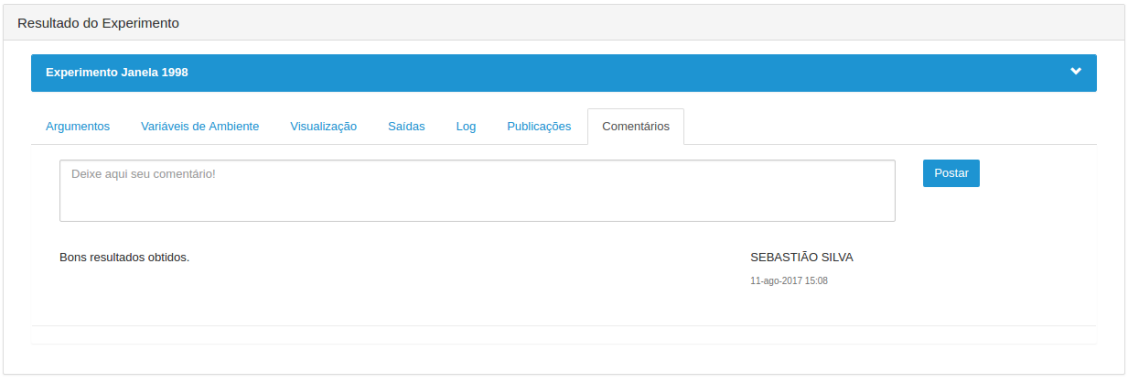
Figura 4.25 - Interface gráfica `Resultado do Experimento` - guia `Publicações`.



Fonte: Produção do autor (2017).

Os resultados obtidos com o experimento podem ser discutidos com outros pesquisadores através de comentários deixados na guia `Comentários`, como apresentado na Figura 4.26.

Figura 4.26 - Interface gráfica `Resultado do Experimento` - guia `Comentários`.



Fonte: Produção do autor (2017).

A possibilidade de discutir os resultados com outros pesquisadores pode contribuir na evolução da pesquisa. Outra finalidade do compartilhamento é poder disponibilizar os resultados como parte de uma publicação para que outros pesquisadores possam analisá-los e ainda reproduzi-los, uma vez que terão acesso a todo o processo desde a parametrização do experimento até os resultados para comparação.

4.3 Administração do Sistema

O ambiente Sentinel pode ser administrado por dois níveis de usuários, o Administrador da Organização e o Administrador Geral do Sistema. A Figura 4.27 apresenta quais são as funções que cada perfil pode requisitar.

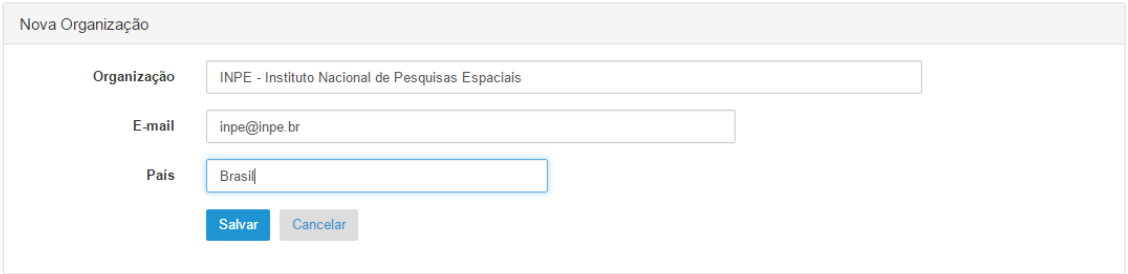
Figura 4.27 - Diagrama de Casos de Uso - Administradores.

Fonte: Produção do autor (2017).

4.3.1 Cadastro de Organização

Novas organizações podem ser incorporadas ao ambiente e desta forma, permitir a seus pesquisadores acesso a recursos oferecidos por outras organizações que colaboram com o ambiente. A requisição de cadastro de uma organização só pode ser realizada por um Administrador Geral do Sistema. A interface gráfica utilizada para realizar tal função é a `Cadastro de Organização`, mostrada na Figura 4.28.

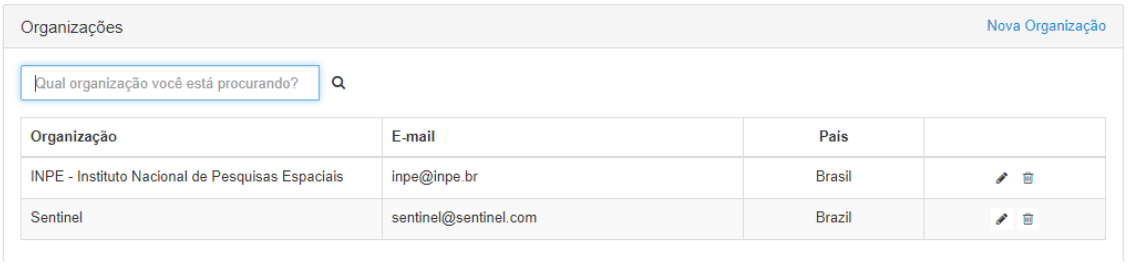
Figura 4.28 - Interface gráfica `Cadastro de Organização`.



Fonte: Produção do autor (2017).

A relação das organizações cadastradas no sistema está acessível através da interface gráfica `Organizações`. Assim como o cadastro, essa relação está disponível apenas para o Administrador Geral do Sistema. Através dessa interface, apresentada na Figura 4.29, o administrador poderá editar os dados cadastrados e excluir uma organização.

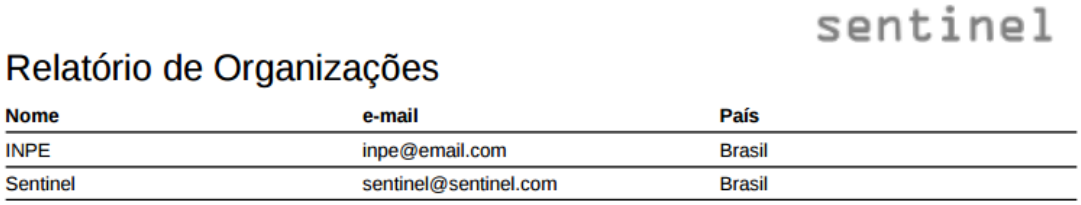
Figura 4.29 - Interface gráfica `Organizações`.



Fonte: Produção do autor (2017).

Outra forma de acessar a relação de organizações vinculadas ao ambiente é através do Relatório de Organizações, exibido na Figura 4.30.

Figura 4.30 - Exemplo de Relatório de Organizações cadastradas.



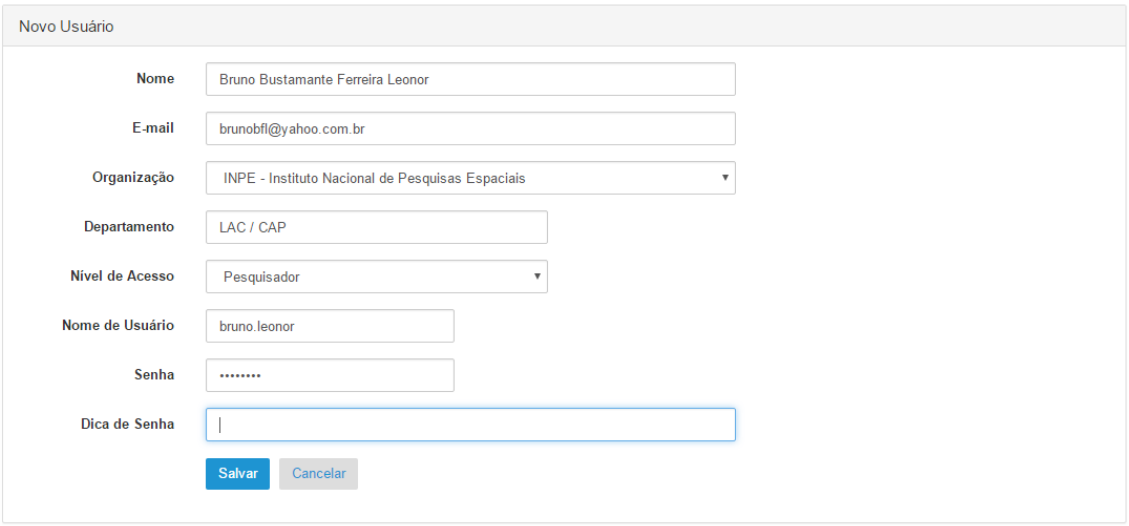
Fonte: Produção do autor (2017).

A partir do cadastro da organização é liberado o cadastro de pesquisadores para que esses possam acessar e usufruir das facilidades oferecidas pelo ambiente Sentinel para desenvolver suas pesquisas.

4.3.2 Cadastro de Usuário

O cadastro de novos usuários é realizado através da interface gráfica `Cadastro de Usuário`, apresentada na Figura 4.31.

Figura 4.31 - Interface gráfica `Cadastro de Usuário`.



Cadastro de novo usuário no ambiente Sentinel sendo realizado por um Administrador da Organização. Desta forma, o novo usuário não poderá receber o perfil de Administrador Geral do Sistema e será vinculado à mesma organização do administrador que requisitou o cadastro.

Fonte: Produção do autor (2017).

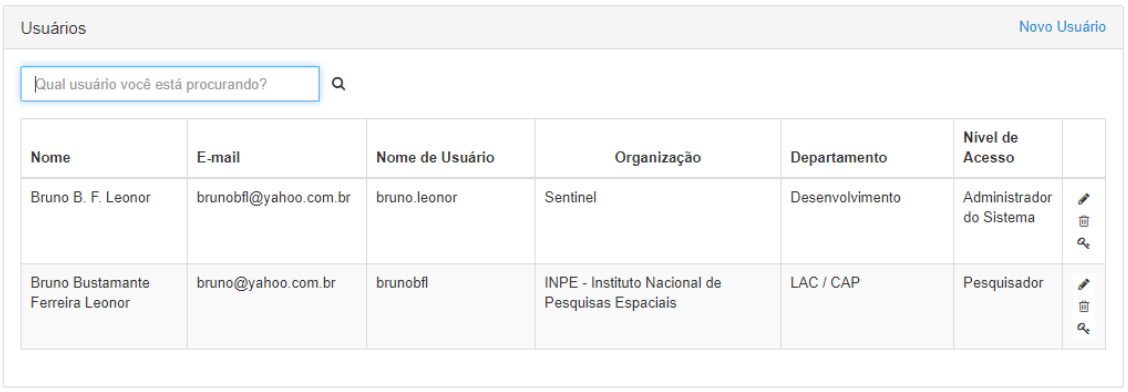
Quando é realizado o cadastro de um usuário no sistema, obrigatoriamente deve-se vincular o registro a uma organização e este usuário ser associado a um perfil que irá definir o nível de requisição as funcionalidades oferecidas pelo ambiente Sentinel, que pode ser: (1) Pesquisador, (2) Administrador da Organização ou (3) Administrador Geral do Sistema.

Se o cadastro for realizado por um Administrador da Organização, o usuário será automaticamente vinculado à mesma organização podendo este receber o perfil de Pesquisador ou Administrador da Organização. Um Administrador Geral só pode ser cadastrado por um usuário com o mesmo perfil/nível.

Para cada perfil, existe uma série de funcionalidades que podem ser requisitadas conforme é apresentado no diagrama de Casos de Uso da Figura B.1.

A relação dos usuários cadastrados no sistema está disponível através da interface gráfica `Usuários`, apresentada na Figura 4.32. Nessa interface, os administradores têm acesso à edição dos dados cadastrados, exclusão de um determinado usuário e à alteração da chave de acesso (senha) do usuário. O Administrador da Organização tem acesso apenas aos usuários que fazem parte da mesma organização a qual está vinculado.

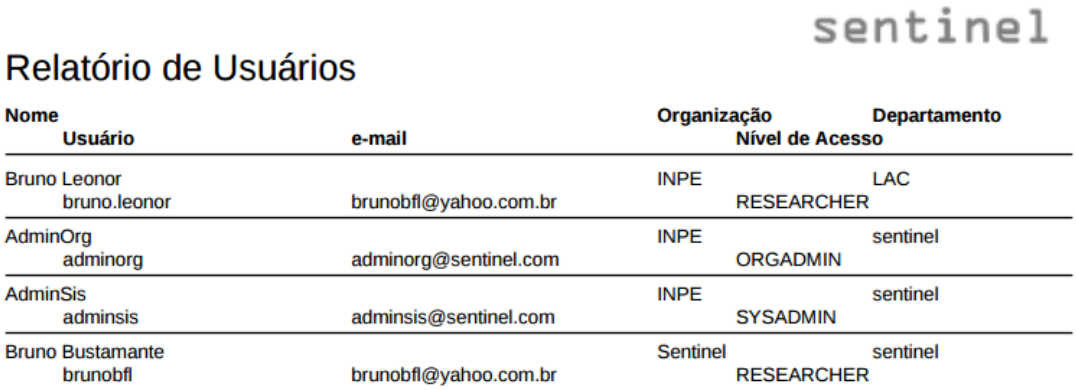
Figura 4.32 - Interface gráfica `Usuários`.



Fonte: Produção do autor (2017).

Outra forma de acessar informações sobre os usuários cadastrados é através do Relatório de Usuário, apresentado na Figura 4.33. Quando este relatório é criado por um Administrador Geral do Sistema os usuários são agrupados por Organização.

Figura 4.33 - Exemplo de Relatório de Usuários cadastrados.



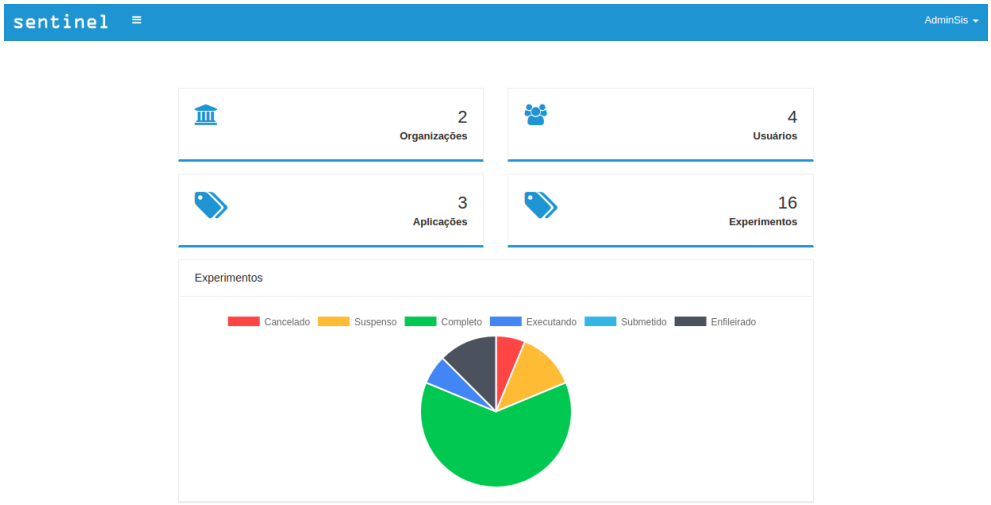
Fonte: Produção do autor (2017).

4.3.3 Dashboard - Administração do Ambiente Sentinel

Os administradores têm acesso a todas as informações do ambiente através de um *Dashboard*. Neste painel são exibidas informações personalizadas para cada perfil de administração, que podem ser sobre as organizações colaboradoras, os pesquisadores cadastrados, as aplicações cadastradas e os experimentos criados.

O Administrador Geral do Sistema tem acesso a informações sobre todos os eventos que acontecem no ambiente Sentinel independente da organização a qual está vinculado. Essas informações são resumidas e exibidas na interface gráfica `Dashboard`, apresentada na Figura 4.34.

Figura 4.34 - `Dashboard` - Administrador Geral do Sistema.



Fonte: Produção do autor (2017).

A partir do *Dashboard* de administração, podem ser gerados relatórios referentes a cada uma das informações disponíveis. O Administrador Geral do Sistema tem acesso aos relatórios de todas as organizações, usuários, aplicações e experimentos. O Administrador da Organização tem acesso somente a informações relacionadas a sua organização.

5 ESTUDO DE CASO

Este capítulo apresenta os resultados dos testes realizados durante o desenvolvimento desta tese com um estudo de caso voltado ao Clima Espacial. O programa do Clima Espacial foi adotado devido a sua relevância nas condições e nos processos que ocorrem no espaço que potencialmente afetam o ambiente próximo à Terra, seres humanos e seus ativos tecnológicos. O método DFA foi aplicado às medidas de irradiação solar (TSI - *Total Solar Irradiance*) obtidas no período de 1978 a 2012.

5.1 Clima Espacial

O Clima Espacial diz respeito à área do conhecimento dos fenômenos solares e ocorrências físicas no ambiente espacial, os quais se manifestam de forma recorrente afetando o meio entre o Sol e a Terra, e o espaço em torno da Terra.

Fenômenos que ocorrem no ambiente espacial podem ter seus efeitos sentidos e interferir em diferentes sistemas tecnológicos, em áreas como navegação e posicionamento por satélite (aeronaves, embarcações, plataformas petrolíferas), comunicação (satélites geoestacionários, aeronaves), distribuição de energia (linhas de transmissão, dutos de distribuição de gás natural e petróleo), além dos sistemas de defesa nacional.

O clima espacial foi a área de estudo adotado neste trabalho, pois o conhecimento e a previsão de diferentes fenômenos que diretamente afetam as atividades humanas, tais como atividades solares, tempestades magnéticas e tempestades ionosféricas, têm grande importância. Esses fenômenos exercem influências naturais sobre o planeta Terra às quais podem afetar sistemas tecnológicos, instalações físicas e serviços que atendem as necessidades da sociedade atual, em decorrência do estado ou modificações desse ambiente.

Através do programa de Estudo e Monitoramento Brasileiro do Clima Espacial (EMBRACE), o Instituto Nacional de Pesquisas Espaciais (INPE) busca melhorar o entendimento sobre os processos eletrodinâmicos da ionosfera equatorial e de baixas latitudes, monitorando parâmetros físicos como características do Sol, do espaço interplanetário, da magnetosfera, da ionosfera e da mesosfera (BRASIL, 2011).

O programa EMBRACE tem como objetivo: (1) proceder à coleta de dados e manutenção do sistema de observação em Clima Espacial, (2) modelar processos do sistema Sol-Terra em escala global e regional com investigação dos fenômenos, (3) disponibilizar informação em tempo real e fazer a previsão do Clima Espacial, e (4) prover diagnósticos de seus efeitos sobre diferentes sistemas tecnológicos por meio de coleta de dados de satélite, de superfície e modelagem computacional (INPE, 2014b).

5.1.1 Equipamentos

A Tabela 5.1 apresenta os equipamentos utilizados pelo EMBRACE na aquisição de dados para o estudo do Clima Espacial.

Tabela 5.1 - Equipamentos para captação dados utilizados pelo EMBRACE.

	Equipamento
1	*Brazilian Solar Spectroscope* (BSS)
2	*Brazilian Decimetric Array* (BDA)
3	*Solar Patrol Un-phased Array* (SPUA)
4	Espectrômetro Centimétrico Solar
5	Antena Adaptada de Cuiabá
6	Telescópio Cintilador de Muons (TCM)
7	Sondadores Ionosféricos
8	Sistema de Magnetometro (SMT)
9	Sistema Monitor de GIC (SMG)
10	Receptores GPS
11	Radares VHF

Fonte: Adaptado de INPE (2014b).

A Figura 5.1 apresenta dois dos equipamentos utilizados na captação de dados para o programa EMBRACE. Na Figura 5.1(a) são exibidas algumas das antenas que formam o BDA, instalado no INPE no município de Cachoeira Paulista/SP enquanto a Figura 5.1(b) apresenta o equipamento SPUA, instalado no Radio Observatório do Itapetinga na cidade de Atibaia/SP.

Figura 5.1 - Equipamentos para coleta de dados.

(a) Antena BDA (b) SPUA

Fonte: INPE (2014a)

5.2 O método DFA

O método DFA foi proposto pela primeira vez por (PENG et al., 1994) para análise de sequências de DNA (em português: ácido desoxirribonucleico). Foi basicamente projetado para revelar correlação de longo alcance em processos não estacionários (MORARIU et al., 2007). No entanto, o DFA pode também ser considerado como um método adequado para investigar a correlação de longo e curto alcance em sistemas não estacionários e estacionários.

"O comportamento do método DFA pode ser útil em uma estrutura de análise de dados operacionais, fornecendo monitoramento, alertas e previsões de explosões solares e, possivelmente, de atividade geomagnética" (VERONESE et al., 2011).

Resumidamente, o método é descrito por Goldberger et al. (2000) com as seguintes etapas: (1) Primeiro, integra-se a série temporal (com N amostras) a ser analisada. (2) Na sequência, a série temporal integrada é dividida em intervalos de comprimento igual a n, com $n = 1, 2, ..., N$. (3) Em cada um desses intervalos é calculada a tendência local da série, por uma reta de mínimos quadrados ajustada aos dados. A coordenada y dos segmentos de reta é denotada por $y_n(k)$. (4) Em seguida, destendenciamos a série integrada, $y(k)$, subtraindo a tendência local, $y_n(k)$, em cada intervalo. Para um dado intervalo de tamanho n, o tamanho característico de flutuação para a série temporal integrada e destendenciada é calculado por:

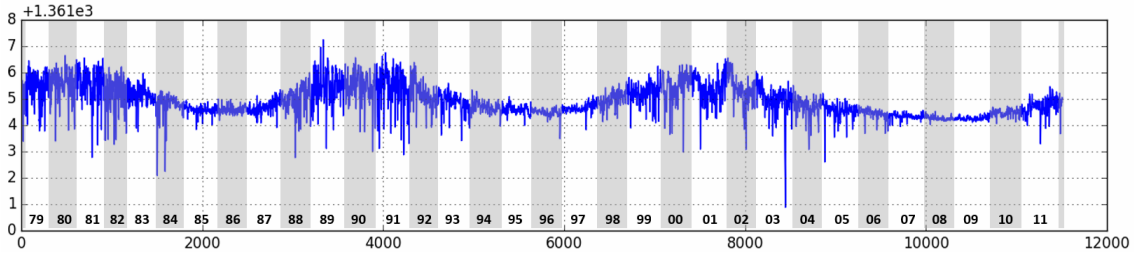
$$F(n) = \sqrt{\frac{1}{N}\sum_{k=1}^{N}[y(k) - y_n(k)]^2} \tag{5.1}$$

Este processo é repetido para todos os intervalos de tamanho n, obtendo-se uma relação entre a média das flutuações $F(n)$ e o tamanho dos intervalos n. Se essa relação for linear em um gráfico log-log, isso indica a existência de uma lei exponencial em escala, de acordo com a fórmula: $F(n) \approx n^{\alpha}$, em que α é o expoente de escala, que pode ser calculado por regressão linear em um gráfico log-log.

5.3 Fonte dos Dados

Para este estudo de caso, foram utilizados os dados da variável TSI da missão VIRGO/-SOHO (FRöHLICH; LEAN, 1998) extraído do conjunto de dados `composite_d41_62_-1204.txt` (ACRIM, 2012). Dessa série temporal principal, apresentada na Figura 5.2, foram extraídas 35 séries temporais, chamadas de janelas, que representam as observações feitas a cada ano (entre 1978 e 2012).

Figura 5.2 - Série temporal da variável TSI do experimento VIRGO/SOHO.



Fonte: Produção do autor (2017).

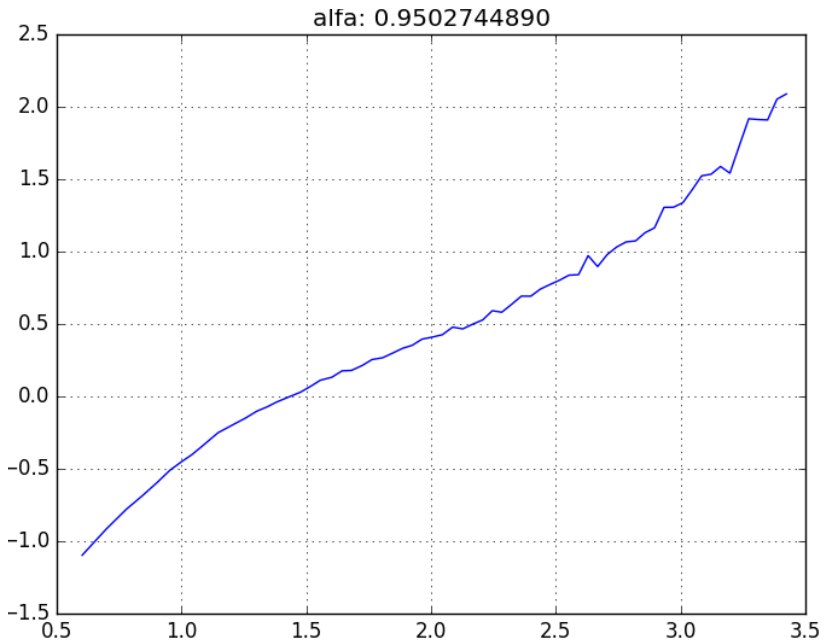
Uma série temporal é uma sequência de realizações (observações) de uma variável coletadas em intervalos regulares durante um período de tempo (WOOLDRIDGE, 2015).

5.4 Resultados obtidos

Uma série de experimentos foram produzidos no ambiente Sentinel aplicando o método DFA às janelas extraídas da série temporal de TSI e alguns dos resultados obtidos com estes experimentos serão apresentados nesta seção.

A Figura 5.3 apresenta o resultado da aplicação do método DFA na série temporal apresentada na Figura 5.2.

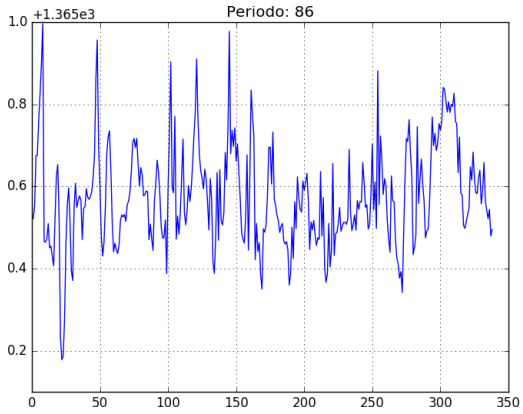
Figura 5.3 - Aplicação do método DFA na série temporal da variável TSI.



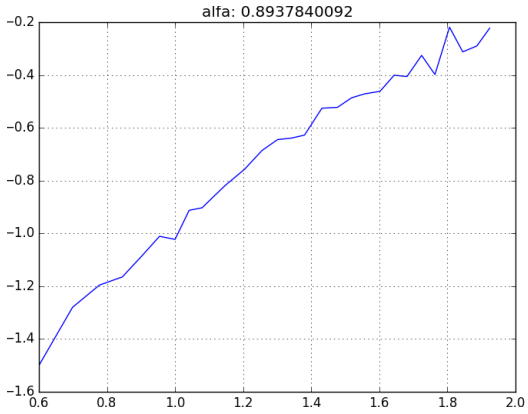
Fonte: Produção do autor (2017).

Os resultados obtidos com a execução dos experimentos no ambiente Sentinel para as janelas de 1986, 1996 e 2008 são apresentados na Figura 5.4, respectivamente 5.4(a), 5.4(c) e 5.4(e). O resultado da aplicação do método DFA para essas mesmas janelas são apresentados nas Figuras 5.4(b), 5.4(d) e 5.4(f) na mesma ordem cronológica.

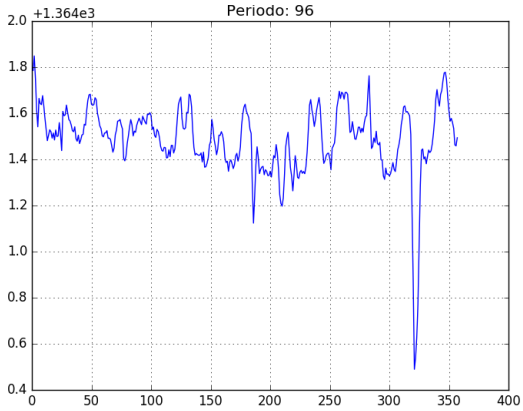
Figura 5.4 - Janelas da série temporal TSI x Resultado da aplicação do método DFA.



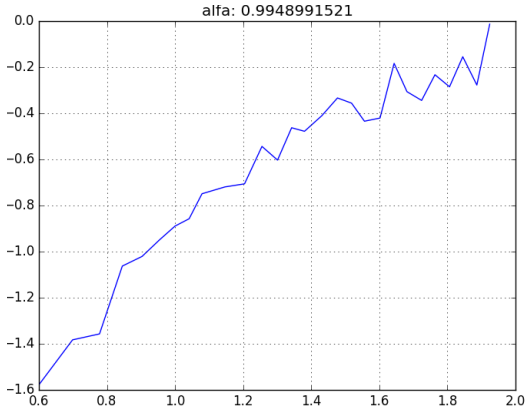
(a) Janela 1986



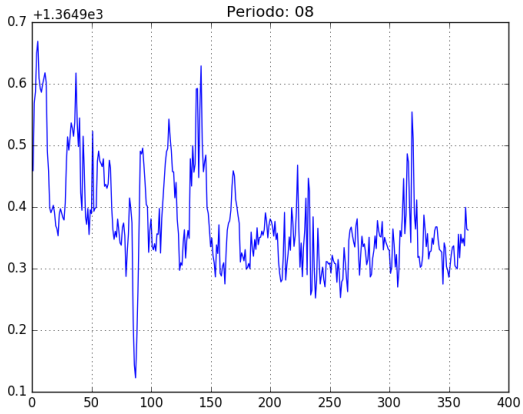
(b) DFA (1986)



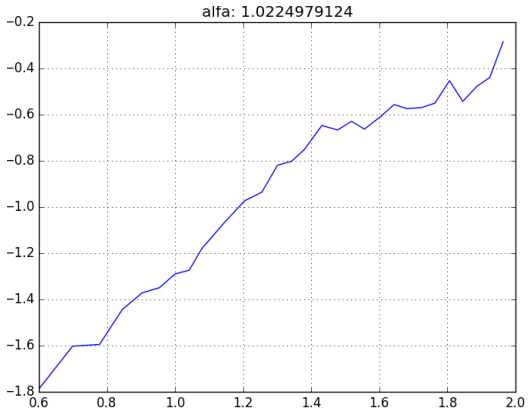
(c) Janela 1996



(d) DFA (1996)



(e) Janela 2008



(f) DFA (2008)

Fonte: Produção do autor (2017).

Os resultados exibidos na Figura 5.4 foram selecionados aleatoriamente entre os experimentos produzidos. O resultado da execução dos experimentos criados para as demais janelas pode ser encontrado no Apêndice C.

Da execução dos experimentos, foi extraído o expoente α para cada uma das janelas, apresentados na Tabela 5.2. Esses resultados não foram analisados pois não é foco da pesquisa analisar o método utilizado nos experimentos criados por este estudo de caso.

Tabela 5.2 - Expoente α de cada janela.

Janela	α	**Janela**	α
78-12	0.950274488985	95	1.06629504897
78	1.43120355455	96	0.994899152143
79	1.02221569392	97	0.944063946741
80	1.04787741919	98	0.966266225997
81	1.07067813268	99	0.960423763573
82	0.96771013622	00	1.11876844216
83	0.95327337799	01	0.958250359695
84	1.04847694358	02	0.956157571451
85	0.952884000354	03	1.15276588982
86	0.8937840092	04	1.10171427129
87	0.779929247501	05	0.977228425668
88	0.912727838497	06	1.00873551784
89	0.938213051694	07	0.960971280023
90	0.976737932127	08	1.02249791239
91	1.02867519185	09	1.01425234186
92	0.97805026371	10	1.00171721802
93	1.11578080511	11	1.03948303634
94	1.10779497669	12	1.32324887294

Fonte: Produção do Autor (2017).

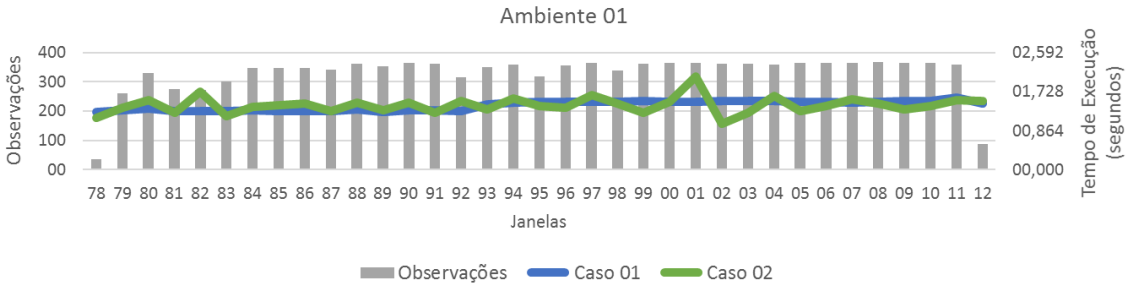
Todos os experimentos criados nesse estudo de caso foram executados em três ambientes distintos, sendo realizadas duas execuções em cada ambiente, totalizando seis execuções para cada janela, chamadas de casos. Para cada ambiente foi executado um caso local e outro conteinerizado.

No Ambiente 01, equipado com um processador AMD A10-4655M APU @ 2.0GHz com 6 GB de memória e Sistema Operacional Debian GNU/Linux 8.3 (jessie) x86_64, foram executados os casos 01 e 02. No caso 01 os experimentos foram executados diretamente no *host* e no caso 02 a execução ocorreu no *container*.

A Figura 5.5 apresenta o tempo de execução dos experimentos nos casos 01 e 02. As barras verticais indicam a quantidade de dados observados para cada janela extraída da série

temporal obtida. A linha azul representa os tempos de execução dos experimentos no caso 01 (*host*) enquanto a linha verde representa os tempos de execução dos experimentos para o caso 02 (*container*).

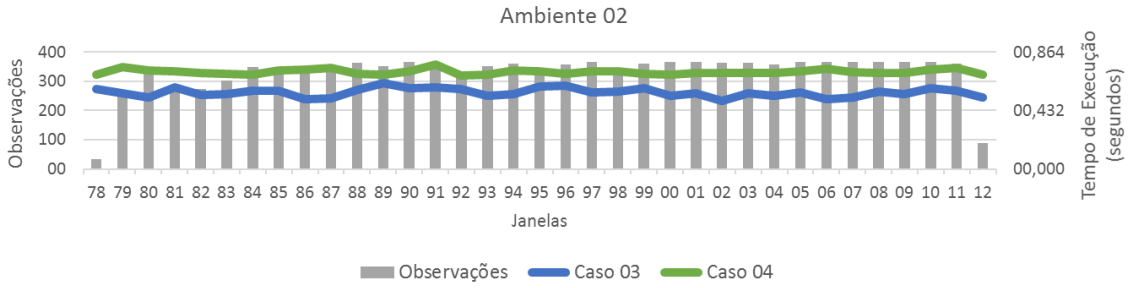
Figura 5.5 - Comparativo de tempo das execuções nos casos 1 e 2.



Fonte: Produção do autor (2017).

Os casos 03 (*host*) e 04 (*container*) foram executados no Ambiente 02, equipado com processador Intel(R) Core(TM) i7-3820 CPU @ 3.60GHz com 16 GB de memória e Sistema Operacional Ubuntu 16.04.2 LTS. Os tempos de execução dos experimentos nos casos citados são apresentados na Figura 5.6 onde as barras verticais indicam a quantidade de dados observados para cada janela, a linha azul representa os tempos de execução dos experimentos no caso 03 (*host*) e a linha verde representa os tempos de execução dos experimentos para o caso 04 (*container*).

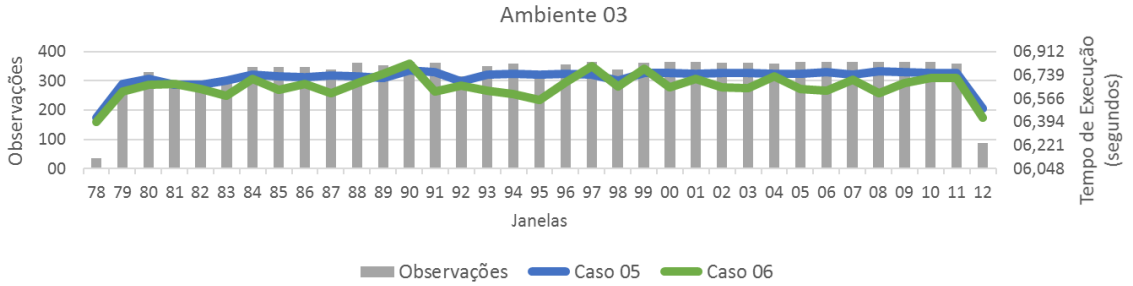
Figura 5.6 - Comparativo de tempo das execuções nos casos 3 e 4.



Fonte: Produção do autor (2017).

A Figura 5.7 apresenta os tempos de execução dos experimentos no Ambiente 03, que possui processador Quad-core 64-bit ARM Cortex A53 @ 1.2 GHz com 1 GB de memória e Sistema Operacional Raspbian. Neste ambiente, foram executados os casos 05 (*host*) e 06 (*container*). Assim como nas Figuras 5.5 e 5.6, as barras verticais indicam a quantidade de dados observados para cada janela, a linha azul representa os tempos de execução dos experimentos no *host* (caso 05) e a linha verde representa os tempos de execução dos experimentos no *container* (caso 06).

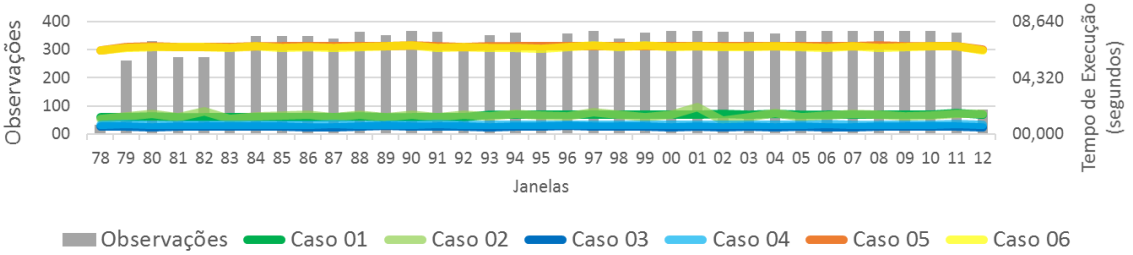
Figura 5.7 - Comparativo de tempo das execuções nos casos 5 e 6.



Fonte: Produção do autor (2017).

Os tempos obtidos com a execução dos experimentos nos três ambientes utilizados podem ser observados na Figura 5.8. As linhas em tons de verde representam as execuções dos experimentos no Ambiente 01 (casos 01 e 02), as linhas em tons de azul representam as execuções dos experimentos no Ambiente 02 (casos 03 e 04) e as linhas em tons de amarelo representam as execuções dos experimentos no Ambiente 03 (casos 05 e 06).

Figura 5.8 - Comparativo de tempo das execuções nos casos 1, 2, 3, 4, 5 e 6.



Fonte: Produção do autor (2017).

Analisando o gráfico apresentado na Figura 5.8, pode-se observar que o Sentinel apresentou-se consistente e estável durante a execução dos experimentos comparadas as execuções locais (*host*) e conteinerizadas (*container*) dentro de cada um dos ambientes utilizados.

O uso do *container* na execução dos experimentos não afetou o desempenho da aplicação uma vez que os tempos medidos ficaram próximos as execuções no *host*. A Tabela 5.3 apresenta o tempo de execução de cada um dos experimentos criados neste estudo de caso apresentados junto com o *Computing Throughput* (quantidade de dados / tempo de processamento).

As diferenças medidas nos tempos obtidos entre os ambientes se deve a configuração de cada um dos ambientes.

Tabela 5.3 - Tempo de execução e *Computing Throughput* de cada um dos casos apresentados.

Janela	**Observações**	**Tempo Decorrido**						**Computing Throughput**					
		Caso 1	Caso 2	Caso 3	Caso 4	Caso 5	Caso 6	Caso 1	Caso 2	Caso 3	Caso 4	Caso 5	Caso 6
78-12	11506	1,698s	1,160s	0,778s	0,916s	16,057s	15,705s	6776,21	9918,97	14789,2	12561,14	716,57	732,63
78	34	1,286s	1,151s	0,587s	0,697s	6,421s	6,394s	26,44	29,54	57,92	48,78	5,30	5,32
79	262	1,318s	1,377s	0,558s	0,750s	6,674s	6,615s	198,79	190,27	469,53	349,33	39,26	39,61
80	331	1,357s	1,547s	0,528s	0,726s	6,708s	6,665s	243,92	213,96	626,89	455,92	49,34	49,66
81	274	1,305s	1,257s	0,605s	0,722s	6,670s	6,676s	209,96	217,98	452,89	379,50	41,08	41,04
82	272	1,299s	1,718s	0,549s	0,709s	6,670s	6,639s	209,39	158,32	495,45	383,64	40,78	40,97
83	301	1,299s	1,188s	0,553s	0,703s	6,697s	6,585s	231,72	253,37	544,30	428,17	44,95	45,71
84	347	1,321s	1,385s	0,580s	0,699s	6,743s	6,709s	262,68	250,54	598,28	496,42	51,46	51,72
85	348	1,302s	1,430s	0,579s	0,727s	6,727s	6,628s	267,28	243,36	601,04	478,68	51,73	52,50
86	338	1,294s	1,461s	0,516s	0,735s	6,725s	6,671s	261,21	231,35	674,42	473,47	51,75	52,17
87	340	1,305s	1,292s	0,524s	0,746s	6,733s	6,606s	260,54	263,16	648,85	455,76	50,50	51,47
88	362	1,327s	1,475s	0,585s	0,700s	6,730s	6,682s	272,80	245,42	618,80	517,14	53,79	54,18
89	352	1,275s	1,319s	0,631s	0,699s	6,720s	6,747s	276,08	266,87	557,84	503,58	52,38	52,17
90	365	1,313s	1,479s	0,596s	0,719s	6,772s	6,821s	277,99	246,79	612,42	507,65	53,90	53,51
91	363	1,310s	1,265s	0,602s	0,774s	6,762s	6,615s	277,10	286,96	602,99	468,99	53,68	54,88
92	315	1,305s	1,513s	0,593s	0,687s	6,692s	6,663s	241,38	208,20	531,20	458,52	47,07	47,28
93	350	1,454s	1,324s	0,539s	0,698s	6,743s	6,623s	240,72	264,35	649,35	501,43	51,91	52,85
94	360	1,488s	1,572s	0,555s	0,730s	6,750s	6,597s	241,94	229,01	648,65	493,15	53,33	54,57
95	319	1,507s	1,410s	0,606s	0,724s	6,740s	6,556s	211,68	226,24	526,40	440,61	47,33	48,66
96	357	1,502s	1,366s	0,613s	0,705s	6,746s	6,686s	237,68	261,35	582,38	506,38	52,92	53,40
97	365	1,508s	1,656s	0,568s	0,719s	6,744s	6,803s	242,04	220,41	642,61	507,65	54,12	53,65
98	338	1,503s	1,455s	0,572s	0,724s	6,700s	6,656s	224,88	232,30	590,91	466,85	50,45	50,78
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

(Continua)

Tabela 5.3 - Continuação

Janela	Observações	Tempo Decorrido						Computing Throughput					
		Caso 1	Caso 2	Caso 3	Caso 4	Caso 5	Caso 6	Caso 1	Caso 2	Caso 3	Caso 4	Caso 5	Caso 6
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
99	361	1,512s	1,257s	0,597s	0,702s	6,755s	6,784s	238,76	287,19	604,69	514,25	53,44	53,21
00	365	1,497s	1,506s	0,541s	0,699s	6,753s	6,651s	243,82	242,36	674,68	522,17	54,05	54,88
01	365	1,508s	2,060s	0,559s	0,711s	6,751s	6,710s	242,04	177,18	652,95	513,36	54,07	54,40
02	362	1,516s	1,016s	0,503s	0,709s	6,755s	6,647s	238,79	356,30	719,68	510,58	53,59	54,46
03	363	1,512s	1,267s	0,560s	0,708s	6,757s	6,645s	240,08	286,50	648,21	512,71	53,72	54,63
04	358	1,516s	1,631s	0,542s	0,709s	6,748s	6,730s	236,15	219,50	660,52	504,94	53,05	53,19
05	365	1,511s	1,297s	0,566s	0,723s	6,749s	6,638s	241,56	281,42	644,88	504,84	54,08	54,99
06	365	1,493s	1,409s	0,515s	0,739s	6,761s	6,621s	244,47	259,05	708,74	493,91	53,99	55,13
07	365	1,488s	1,561s	0,526s	0,717s	6,745s	6,704s	245,30	233,82	693,92	509,07	54,11	54,45
08	366	1,504s	1,460s	0,570s	0,711s	6,768s	6,605s	243,35	250,68	642,11	514,77	54,08	55,41
09	365	1,513s	1,341s	0,550s	0,707s	6,759s	6,679s	241,24	272,18	663,64	516,27	54,00	54,65
10	365	1,512s	1,407s	0,596s	0,733s	6,754s	6,720s	241,40	259,42	612,42	497,95	54,04	54,32
11	360	1,602s	1,542s	0,576s	0,745s	6,753s	6,716s	224,72	233,46	625,00	483,22	53,31	53,60
12	88	1,465s	1,521s	0,529s	0,693s	6,491s	6,426s	60,07	57,86	166,35	126,98	13,56	13,69

Fonte: Produção do autor (2017).

6 CONCLUSÕES

Este trabalho apresentou um ambiente *web*, chamado Sentinel, dedicado ao gerenciamento e monitoramento de aplicações científicas. Como uma aplicação de protótipo, o ambiente Sentinel foi aplicado para o monitoramento das séries temporais de irradiação solar, utilizando o método DFA. Para o estudo de caso, foi adotado como domínio, o programa EMBRACE para o conhecimento e a previsão de diferentes fenômenos que afetam diretamente atividades humanas, como atividade solar, tempestades magnéticas e flutuações ionosféricas. Esses tópicos têm grande importância para o bem-estar da sociedade.

O ambiente Sentinel surgiu da busca por um ambiente que permitisse a pesquisadores concentrar sua pesquisa em um único local, com possibilidade de gerenciar e monitorar seus experimentos científicos.

No desenvolvimento deste trabalho, foram analisadas 314 ferramentas de *e-Science* na busca por características ausentes ou desejadas para servirem de parâmetros para o desenvolvimento do ambiente Sentinel. Dessa análise, chegou-se a um conjunto de oito características desejáveis para o ambiente, são elas: (1) ambiente multidisciplinar, (2) conjunto personalizado de aplicações, (3) controle, (4) escalonamento e orquestrações dos recursos, (5) compartilhamento dos resultados, (6) colaboração, (7) status e (8) composição de *workflow*.

A critério de comparação, a Tabela 6.1 apresenta as 15 ferramentas identificadas nessa pesquisa que possuem ao menos uma das características desejadas para o ambiente Sentinel. Essa lista é resultado do estudo realizado nas 314 ferramentas identificadas nessa pesquisa. A classificação das ferramentas na tabela é dada pela quantidade de características presentes.

Tabela 6.1 - Comparação: Ferramentas identificadas x Características esperadas.

	Ferramentas	**Etapas**						**Características**							
		A	B	C	D	E	F	1	2	3	4	5	6	7	8
1	Open Science Grid		✓					✓	✓	✓	✓				
2	VLADA		✓		✓			✓	✓	✓		✓			
3	R		✓					✓	✓	✓					
4	Google Drive		✓	✓				✓				✓	✓		
5	IPython Notebooks		✓					✓	✓	✓					
6	RTGrid		✓					✓	✓	✓					
7	SMS / ECFlow		✓					✓	✓						✓

(Continua)

Tabela 6.1 - Continuação

	Ferramentas	Etapas						Características							
		A	B	C	D	E	F	1	2	3	4	5	6	7	8
8	Zooniverse		✓					✓					✓		
9	ROpenSci	✓	✓					✓				✓			
10	Excel		✓					✓	✓						
11	SPiNMe		✓						✓	✓					
12	GridUnesp		✓					✓			✓				
13	nanoHub		✓					✓			✓				
14	SPSS		✓					✓							
15	Profrager		✓						✓						

(A) Descoberta, (B) Análise, (C) Redação, (D) Publicação, (E) Divulgação, (F) Avaliação, (1) Ambiente Multidisciplinar, (2) Conjunto Personalizado de Aplicações, (3) Controle, (4) Escalonamento e Orquestração dos Recursos, (5) Compartilhamento dos Resultados, (6) Colaboração, (7) Status e (8) Composição de *workflow*.

Fonte: Produção do autor (2016).

Espera-se que este trabalho possa contribuir com um ambiente mais amigável e mais seguro para os usuários de *e-Science*, onde podem reunir todas as suas aplicações utilizadas no desenvolvimento de suas pesquisas.

Consideradas as limitações do *hardware* adotado, o ambiente Sentinel se mostrou consistente e estável quando comparadas às execuções locais e conteinerizadas, como apresentado nos resultados obtidos com o estudo de caso realizado.

Com a proposta de conteinerização adotada neste trabalho para execução dos experimentos, buscou-se a portabilidade das aplicações e o uso eficiente dos recursos, uma vez que um *container* é padronizado e responsável por suprir todas as suas necessidades através das imagens geradas.

Além da aplicação DFA utilizada como estudo de caso nesta pesquisa, existem algumas outras aplicações que podem ter seu uso imediato no ambiente Sentinel para contribuir com os estudos do clima espacial, são exemplos: PSD (*Power Spectral Density*) (OPPENHEIM; VERGHESE, 2015), Espectro de Singularidades (BOLZAN et al., 2013), DFA2D (*Detrended Fluctuation Analysis 2D*) (GU; ZHOU, 2006) e GPA (*Gradient Pattern Analysis*) (ROSA et al., 1999).

6.1 Contribuições

Do conjunto de oito características apresentadas como desejáveis para o ambiente Sentinel, sete delas foram implementadas durante o desenvolvimento deste trabalho, podendo destacá-las como contribuições deste trabalho:

- (1) ambiente multidisciplinar - durante os testes realizados, o ambiente Sentinel apresentou-se abrangente o suficiente para suportar a colaboração entre organizações dos mais diversos domínios de aplicação.
- (2) conjunto personalizado de aplicações - o ambiente oferece recursos suficientes para que os pesquisadores possam personalizar sua área de trabalho com seu próprio conjunto de ferramentas, oferecendo suporte a diversas linguagens de programação e distribuições dos sistemas operacionais Linux.
- (3) controle - oferece ao pesquisador total liberdade para gerenciar a execução dos seus experimentos dentro ambiente.
- (4) escalonamento e orquestrações dos recursos - o emprego de *containers* na execução dos experimentos contribuiu para melhor uso dos recursos computacionais disponíveis isolando cada uma das aplicações.
- (5) compartilhar resultados - esta característica ofereceu ao ambiente um mecanismo para disponibilizar os resultados/produtos produzidos de forma aberta apoiando a reprodução da pesquisa.
- (6) colaboração - o ambiente permitiu a cooperação entres pesquisadores e organizações, contribuindo assim, com o progresso da pesquisa em questão.
- (7) status - este mecanismo facilitou o acompanhamento da execução dos experimentos informando a sua situação ao pesquisador sempre que ocorrer alguma atualização no andamento.

A falta de padrão entre as aplicações científicas dificultou o desenvolvimento de um mecanismo para gerenciar a composição de *workflows*, que corresponde a oitava característica.

Do desenvolvimento deste trabalho resultaram três publicações, sendo dois trabalhos aceitos para participação em eventos internacionais:

- *6th Intl Conference on Nonlinear Science and Complexity* - A Web Framework for Advanced and Intensive Nonlinear Time Series Analysis (Apêndice D), e

- *4th Conference of Computational Interdisciplinary Science* - A High Performance Virtual Workspace for Management and Monitoring of Scientific Applications (Apêndice E).

E um artigo completo aceito para publicação em revista:

- *Discontinuity, Nonlinearity and Complexity* - Nonlinear Time Series Analysis of Complex Systems using an e-Science Web Framework (Apêndice F).

6.2 Trabalhos Futuros

Como possibilidade para trabalhos futuros podem ser destacadas as seguintes atividades para serem desenvolvidas no ambiente Sentinel:

- Adicionar um mecanismo para permitir a composição de um *workflow*.
- Aprimorar o mecanismo de visualização de resultados para permitir que o pesquisador o adeque à sua aplicação.
- Incorporar novos mecanismos de administração do ambiente.
- Oferecer novas distribuições de Sistema Operacional para que os pesquisadores adotem o melhor para sua aplicação.
- Implantar um sistema de pontuação (gameficação) para priorizar as execuções.

REFERÊNCIAS BIBLIOGRÁFICAS

ACRIM. **composite_d41_62_1204.txt**. 2012. Disponível em: <acrim.com/TSI/composite_d41_62_1204.txt>. 71

ALECRIM, E. **O que é cloud computing (computação em nuvem)?** 2013. Disponível em: <http://www.infowester.com/cloudcomputing.php>. Acesso em: set. de 2013. 11

ALEX, B.; TAYLOR, L.; WINCH, R.; HILLERT, G. **Spring Security Reference**. [S.l.: s.n.], 2004. 34

ALISSON, E. **eScience revoluciona a forma como se faz ciência**. Agência FAPESP, Maio 2013. Disponível em: <http://agencia.fapesp.br/17279>. 1, 2, 6

AMAZON. **High Performance Computing (HPC) na AWS**. 2013. Disponível em: <http://aws.amazon.com/pt/hpc-applications/>. Acesso em: nov. de 2013. 13

AOUAR, S. **Entenda o que é o LXC (Linux Containers)**. 2017. Disponível em: <https://weloveteach.com/site/artigo/entenda-o-que-e-o-lxc-linux-containers>. Acesso em: mar. de 2017. 15

APACHE. **Velocity users guide**: the apache velocity developers. 201–. Disponível em: <http://velocity.apache.org/engine/1.7/user-guide.html>. Acesso em: set. de 2016. 35

ARMOZA, J.; BENARDOU, A.; BOREK, L.; COBLE, Z.; DAY, S.; DENBO, S.; DOMBROWSKI, Q.; MULLEN, A.; PERKINS, J.; SPIRO, L. **Digital research tools**. 2015. Disponível em: <http://dirtdirectory.org/>. Acesso em: mar. de 2017. 20

BANSAL, B.; BAWA, S. Design and development of grid portals. In: TENCON 2005, 2005, Melbourne, Qld., Australia. **Proceedings...** Melbourne: IEEE, 2005. p. 1–5. 14

BOLZAN, M. J. A.; TARDELLI, A.; PILLAT, V. G.; FAGUNDES, P. R.; ROSA, R. R. Multifractal analysis of vertical total electron content (vtec) at equatorial region and low latitude, during low solar activity. **Annales Geophysicae**, v. 31, n. 1, p. 127–133, 2013. Disponível em: <https://www.ann-geophys.net/31/127/2013/>. Acesso em: abr. de 2017. 80

BOSMAN, J.; KRAMER, B. **101 innovations in scholarly communication**. Fevereiro 2017. Disponível em: <https://innoscholcomm.silk.co/>. Acesso em: abr. de 2017. 19, 20, 21

BOSTOCK, M. **D3.js introduction**. 201–? Disponível em: <https://d3js.org/#introduction>. Acesso em: jan. de 2017. 41

BOXFUSE. **About Flyway**. 200–? Disponível em: <https://flywaydb.org/about/>. Acesso em: ago. de 2015. 36

BRAGA, R.; GANNON, D. Scientific versus business workflows. In: TAYLOR, I. J.; DEELMAN, E.; GANNON, D.; SHIELDS, M. S. (Ed.). **Workflows for e-Science**: scientific workflows for grids. Secaucus, NJ, USA: Springer Publishing Company, Incorporated, 2014. cap. 2, p. 9–16. ISBN 1849966192, 9781849966191. 22

BRAGHETTO, K. R.; CORDEIRO, D. **Introdução à modelagem e execução de workflows científicos**. jul 2014. Disponível em: <http://www.each.usp.br/dc/papers/jai-cap-workflows.pdf>. 22

BRASIL, P. **Programa de clima espacial do Inpe oferece serviço para satélites GPS**. 03 2011. Disponível em: <http://www.brasil.gov.br/meio-ambiente/2011/03/programa-de-clima-espacial-do-inpe-oferece-servico-para-satelites-gps>. Acesso em: mai. de 2014. 69

BURKE, B.; MONSON-HAEFEL, R. **Entreprise JavaBeans 3.0**: desenvolvendo componentes ejb. 5. ed. [S.l.]: Pearson Education do Brasil, 2007. 8

CARDI, M. de L. **Evolução da computação no Brasil e sua relação com fatos internacionais**. Dissertação (Mestrado) — Universidade Federal de Santa Catarina, Florianópolis, Maio 2002. 1

CARDOSO, J. **Semantic web services**: theory tools and applications. Hershey, PA: Information Science Reference, 2007. 372 p. ISBN 159904045X. 5

CASS, S. **IEEE top programming languages**: design, methods, and data sources. 2016. Disponível em: <http://spectrum.ieee.org/ns/IEEE_TPL_2016/methods.html>. Acesso em: ago. de 2016. 32

____. **The 2017 top programming languages**. Jul 2017. Disponível em: <https://spectrum.ieee.org/computing/software/the-2017-top-programming-languages>. Acesso em: jul. de 2017. 32

CASS, S.; DIAKOPOULOS, N.; ROMERO, J. J. **Interactive**: the top programming languages. Julho 2014. Disponível em: <http://spectrum.ieee.org/static/interactive-the-top-programming-languages>. Acesso em: ago. de 2014. 32

CASTRO, D. de. **Ciência aberta**: movimento defende o conhecimento científico horizontal e cidadão. Julho 2015. Disponível em: <http://www.ebc.com.br/tecnologia/2015/07/ciencia-aberta-movimento-defende-o-conhecimento-cientifico-horizontal-e Acesso em: abr. de 2017. 13

CERN. **GridCafé**. 2013. Disponível em: <http://www.gridcafe.org>. Acesso em: Set. de 2013. 11, 12

CROUZIER, T. **Digital tools for researchers**. 2016. Disponível em: <http://connectedresearchers.com/online-tools-for-researchers/>. Acesso em: mar. de 2017. 20

DAVIES, J.; FENSEL, D.; HARMELEN, F. van. **Towards the semantic web**: ontology-driven knowledge management. Wiley, 2003. ISBN 9780470858073. Disponível em: <https://books.google.com.br/books?id=kREOBAAAQBAJ>. Acesso em: jun. de 2015. 8

DEWAR, M. **Getting started with D3**. [S.l.]: O'Reilly Media, 2012. 70 p. ISBN 978-1-4493-2879-5. 41

DIGIAMPIETRI, L. A.; MEDEIROS, C. B.; SETUBAL, J. C. A framework based in web services orchestration for bioinformatics workflow management. 2004. Disponível em: <http://www.lis.ic.unicamp.br/wp-content/uploads/2014/09/2004_WOB2004.pdf>. Acesso em: mai. de 2016. 22

DOCKER. **Docker Swarm**: production scale clustering and container scheduling for docker. [S.l.: s.n.], 2016. 31

EADLINE, D. A evolução do alto desempenho. **Linux Magazine**, n. 93, p. 37–39, Agosto 2012. 13

EBERMANN, T. **The data stack**. 2017. Disponível em: <https://blog.liip.ch/archive/2017/02/13/data-stack.html>. Acesso em: mar. de 2017. 20

ELLINGWOOD, J. **O ecossistema do Docker**: uma introdução aos componentes comuns. Junho 2015. Disponível em: <https://www.digitalocean.com/community/tutorials/o-ecossistema-do-docker-uma-introducao-aos-componentes-comuns-pt>. Acesso em: mar. de 2016. 14

FAPESP, A. **e-Science é tema de seminário organizado pela FAPESP**. 2005. Disponível em: <http://www.jb.com.br/ciencia-e-tecnologia/noticias/2013/05/09/e-science-e-tema-de-seminario-organizado-pela-fapesp/>. 6

____. **Pesquisa em e-Science**. Abril 2017. Disponível em: <http://www.bv.fapesp.br/pt/402/pesquisa-em-escience/>. Acesso em: abr. de 2017. 7

FARCIC, V. **The DevOps 2.1 toolkit**: docker swarm building, testing, deploying, and monitoring services inside docker swarm clusters. [S.l.]: Leanpub, 2017. 31

FARIA, T. **Java EE 7 com JSF, PrimeFaces e CDI**. 1. ed. São Paulo: Algaworks, 2013. 27, 29

FARIA, T.; JUNIOR, N. **Java e orientação a objetos**. 1. ed. São Paulo: Algaworks, 2015. 32

____. **JPA e Hibernate**. 1. ed. São Paulo: Algaworks, 2015. 37

FAYAD, M.; SCHMIDT, D. C. Object-oriented application frameworks. **Commun. ACM**, ACM, New York, NY, USA, v. 40, n. 10, p. 32–38, out. 1997. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/262793.262798>. Acesso em: nov. de 2014. 15

FILHO, C. F. **História da computação**: o caminho do pensamento e da tecnologia. EdiPUCRS, 2007. 205 p. ISBN 978-85-7430-691-9. Disponível em: <http://www.pucrs.br/edipucrs/online/historiadacomputacao.pdf>. 1

FORCE11. **The FORCE11 resource and tool catalog**. 2017. Disponível em: <https://www.force11.org/catalog>. Acesso em: mar. de 2017. 20

FOSTER, I.; KESSELMAN, C. **The grid**: blueprint for a new computing infrastructure. 2. ed. [S.l.]: Morgan Kaufmann, 2004. 8, 11

FOUNDATION, R. P. **Raspberry Pi 3 Model B**. 201–? Disponível em: <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>. Acesso em: mar. de 2017. 42

____. **What is a Raspberry Pi?** 201–? Disponível em: <https://www.raspberrypi.org/help/faqs/>. Acesso em: mai. de 2014. 42

FRöHLICH, C.; LEAN, J. The sun's total irradiance: cycles, trends and related climate change uncertainties since 1976. **Geophysical Research Letters**, v. 25, p. 4377–4380, dec 1998. 71

GENTIL, E. **Introdução ao Spring Framework**. 2015. Disponível em: <http://www.devmedia.com.br/introducao-ao-spring-framework/26212>. Acesso em: fev. de 2016. 33

GIERKE, O.; DARIMONT, T.; STROBL, C.; PALUCH, M. **Spring Data JPA - reference documentation**. 1. ed. [S.l.: s.n.], 2017. 34

GOLDBERGER, A. L.; AMARAL, L. A. N.; GLASS, L.; HAUSDORFF, J. M.; IVANOV, P. C.; MARK, R. G.; MIETUS, J. E.; MOODY, G. B.; PENG, C.-K.; STANLEY, H. E. Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. **Circulation**, v. 101, n. 23, jun 2000. Disponível em: <http://circ.ahajournals.org/content/101/23/e215>. 71

GOMES, R. **Criando sua própria imagem docker**. 2016. Disponível em: <http://techfree.com.br/2016/03/criando-sua-propria-imagem-docker/>. Acesso em: mar. de 2016. 15

GRADECKI, J. D.; COLE, J. **Mastering Apache Velocity**. Wiley, 2003. 372 p. (Java open source library). Disponível em: <https://books.google.com.br/books?id=wgBTAAAAMAAJ>. Acesso em: out. de 2016. 35

GU, G.-F.; ZHOU, W.-X. Detrended fluctuation analysis for fractals and multifractals in higher dimensions. **Phys. Rev. E**, American Physical Society, v. 74, p. 061104, Dec 2006. Disponível em: <https://link.aps.org/doi/10.1103/PhysRevE.74.061104>. Acesso em: abr. de 2017. 80

GUEDES, G. T. A. **UML2 uma abordagem prática**. 2. ed. São Paulo: Novatec, 2011. ISBN 978-85-7522-281-2. 57

INPE. **Instituto Nacional de Pesquisas Espaciais**. 2014. Disponível em: <www.inpe.br>. Acesso em: jun. de 2014. 70

____. **Introdução ao clima espacial**. INPE, 2014. Disponível em: <http://www2.inpe.br/climaespacial/pt/introducao>. Acesso em: mai. de 2014. 69, 70

JASPERSOFT. **The JasperReports ultimate guide**. [S.l.]: Jaspersoft Corporation, 2011. 321 p. 42

JUNIOR, N. **Primeiros passos com Spring MVC**. 1. ed. São Paulo: Algaworks, 2016. 39

JUNIOR, N.; AFONSO, A. **Produtividade no desenvolvimento de aplicações web com Spring Boot**. 1. ed. São Paulo: Algaworks, 2017. 33, 34

LAUSCHNER, T. **Especificação formal de agentes para web**. 2005. PUC-RJ. 2

LIFERAY. **O que é um portal?** 2014. Disponível em: <https://www.liferay.com/products/what-is-a-portal/web-platform>. Acesso em: jun. de 2014. 14

MAPLE, S.; SHELAJEV, O. **Java tools and technologies landscape 2016**. Boston: JRebel, Julho 2016. Disponível em: <https://zeroturnaround.com/rebellabs/java-tools-and-technologies-landscape-2016/>. Acesso em: ago. de 2016. 29, 30, 34, 35

MARTINS, M. **Relatórios em Java, JasperReports e iReport**. Novembro 2010. Disponível em: <http://www.k19.com.br/artigos/relatorios-em-java-jasperreports-e-ireport/>. Acesso em: out. de 2016. 42

MATTHIAS, K.; KANE, S. P. **Docker**: up & running shipping reliable containers in production. [S.l.]: O'Reilly Media, 2015. 232 p. ISBN 978-1-4919-1757-2. 30

MORARIU, V. V.; BUIMAGA-IARINCA, L.; VAMOS, C.; SOLTUZ, S. Detrended fluctuation analysis of autoregressive processes. **ArXiv e-prints**, jul 2007. 71

NASSI-CALò, L. **Quanto custa publicar em acesso aberto?** Setembro 2013. SciELO em Perspectiva. Disponível em: <http://blog.scielo.org/blog/2013/09/18/quanto-custa-publicar-em-acesso-aberto/>. Acesso em: abr. de 2017. 13

NICKOLOFF, J. **Docker in action**. [S.l.]: Manning, 2016. 304 p. ISBN 9781633430235. 31

O´BRIEN, T.; ZYL, J. van; FOX, B.; CASEY, J.; XU, J.; LOCHER, T. **Maven**: the complete reference. 0.3.1. ed. Mountain View, CA: Sonatype, Inc., 2013. 27

OPPENHEIM, A. V.; VERGHESE, G. C. **Signals, Systems and Inference**. 1. ed. Massachusetts: Pearson Education, 2015. 80

PALAZZI, D.; SILVA, L.; MENDES, L. F.; MATOS, W. G. anda E.; CAMPOS, F.; BRAGA, R. Uso de ontologias em projetos de e-science. **BDBComp**, 2009. 2, 5, 8

PARSONS, R.; FOWLER, M.; JANAKIRAMAN, B.; LEKE, B.; MELO, C.; DOERNENBURG, E.; BOTTCHER, E.; XU, H.; CARTWRIGHT, I.; LEWIS, J.; NORRIS, J.; LEROY, J.; MASON, M.; FORD, N.; LAYCOCK, R.; NEWMAN, S.; SHAW, S.; SRINIVASAN, S.; PALANISAMY, T. **TECHNOLOGY RADAR**. Chicago: [s.n.], January 2015. Disponível em: <thoughtworks.com/radar>. 36

PENG, C. K.; BULDYREV, S. V.; HAVLIN, S.; SIMONS, M.; STANLEY, M. H. E.; GOLDBERGER, A. L. Mosaic organization of dna nucleotides. **Phys. Rev. E**, American Physical Society, v. 49, p. 1685–1689, Feb 1994. Disponível em: <https://link.aps.org/doi/10.1103/PhysRevE.49.1685>. 71

PITANGA, M. **Computação em cluster**: o estado da arte da computação. [S.l.]: Brasport Livros e Multimídia Ltda, 2003. 13

POLIS, W. **Apresentação seminário mestrado UFSCar 2013**. Maio 2013. Disponível em: <http://planetapolis.wordpress.com/2013/05/24/apresentacao-seminarios-mestrado-ufscar-2013/>. Acesso em: Ago. de 2013. 6

RAINERE. **Publicação de software científico**. 2013. Disponível em: <https://www.cienciaaberta.net/publicacao-de-software-cientifico/>. Acesso em: ago. de 2016. 1, 3

RAMOS, V. de A. **Um sistema gerenciador de workflows científicos para a plataforma de nuvens federadas BioNimbuZ**. Brasília: [s.n.], Fev 2016. Disponível em: <http://bdm.unb.br/handle/10483/13145>. 22

ROCHA, M. **Introdução a grid computing**. 2010. 12

ROSA, R. R.; SHARMA, A. S.; VALDIVIA, J. A. Characterization of asymmetric fragmentation patterns in spatially extended systems. **International Journal of Modern**

Physics C, v. 10, n. 01, p. 147–163, 1999. Disponível em: <http://www.worldscientific.com/doi/abs/10.1142/S0129183199000103>. Acesso em: abr. de 2017. 80

SAMPAIO, C. **SOA e web services em java**. [S.l.]: BRASPORT, 2003. 8

SAUVé, J. P. **O que é um framework?** 2014. Disponível em: <http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/frame/oque.htm>. Acesso em: mar. de 2014. 16, 17

SCHILDT, H. **Java**: the complete reference. [S.l.]: McGraw-Hill, 2007. ISBN 978-0-07-163177-8. 32

SCICRUNCH. **Resources dashboard**. 2017. Disponível em: <https://scicrunch.org/browse/resourcedashboard>. Acesso em: mar. de 2017. 20

SILVA, M. S. **Bootstrap 3.3.5**: aprendendo a usar o framework bootstrap para criar layouts css complexos e responsivos. 1. ed. São Paulo: Novatec, 2015. ISBN 978-85-7522-460-1. 38

SOMMERVILLE, I. **Engenharia de software**. 9. ed. [S.l.]: Pearson Education, 2011. 9

SOROR, A. A.; MINHAS, U. F.; ABOULNAGA, A.; SALEM, K.; KOKOSIELIS, P.; KAMATH, S. Automatic virtual machine configuration for database workloads. **ACM Trans. Database Syst.**, ACM, New York, NY, USA, v. 35, n. 1, p. 7:1–7:47, fev. 2008. ISSN 0362-5915. Disponível em: <http://doi.acm.org/10.1145/1670243.1670250>. 10

SOUSA, F. R. C.; MOREIRA, L. O.; MACHADO, J. C. Computação em nuvem: Conceitos, tecnologias, aplicações e desafios. **III Escola Regional de Computação Ceará, Maranhão e Piauí (ERCEMAPI)**, v. 1, p. 150–175, 2009. 10

SPRING. **Spring Web Flow**. 20–. Disponível em: <http://projects.spring.io/spring-webflow/>. Acesso em: jun. de 2016. 33

STASH, R. **Curated collection of resources and tools for researchers**. 2017. Disponível em: <https://researchstash.com>. Acesso em: mar. de 2017. 20

STRAZDINS, P. **eScience themes**. Abril 2002. Disponível em: <http://escience.anu.edu.au/eScience.png.en.html>. Acesso em: ago. de 2013. 5

TAGLIARI, F.; CARVALHO, E. C. A. de. Grids computacionais. **Campo Dig.**, v. 1, n. 1, p. 3–10, jul/dez 2006. 11

TAHAGHOGHI, S.; WILLIAMS, H. E. **Learning MySQL**. 1. ed. Sebastopol, CA: O'Reilly Media, 2006. ISBN 978-0-596-00864-2. Disponível em: <http://www.oreilly.com/catalog/learnmysql>. 36

TAIT, T. F. C. Evolução da internet: do início secreto à explosão mundial. **Jornal PET Informática**, 2007. Disponível em: <http://www.din.uem.br/~tait/evolucao-internet.pdf>. Acesso em: jun. de 2016. 2

TATNALL, A. Portals, portals everywhere. In: TATNALL, A. (Ed.). **web portals: the new gateways to internet information and services**. Australia: Idea Group Publishing, 2005. cap. 1, p. 1–14. ISBN 1-59140-440-1. 2

THYMELEAF. **Tutorial**: using thymeleaf. 2017. Disponível em: <http://www.thymeleaf.org/doc/tutorials/3.0/usingthymeleaf.html>. Acesso em: mar. de 2017. 39, 40

TOLLE, K.; TANSLEY, S.; HEY, T. **O quarto paradigma**: descobertas científicas na era da esciente. 1. ed. [S.l.]: Oficina de Textos, 2011. 263 p. 1, 7

TOSCANI, L. V.; VELOSO, P. A. S. **Complexidade de algoritmos**. 3. ed. [S.l.: s.n.], 2012. 12

VECCHIOLA, C.; CHU, X.; BUYYA, R. **Aneka**: a software platform for .net-based cloud computing. 2009. 9

VERONESE, T.; ROSA, R.; BOLZAN, M.; FERNANDES, F. R.; SAWANT, H.; KARLICKY, M. Fluctuation analysis of solar radio bursts associated with geoeffective x-class flares. **Journal of Atmospheric and Solar-Terrestrial Physics**, v. 73, p. 1311–1316, jul 2011. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1364682610002907>. 71

VITALINO, J. F. N.; CASTRO, M. A. N. **Descomplicando o Docker**. 1. ed. Rio de janeiro: Brasport, 2016. ISBN 978-85-7452-797-0. 3, 14

VUKOTIC, A.; GOODWILL, J. **Apache Tomcat 7**. [S.l.]: Apress, 2011. ISBN 978-1-4302-3723-5. 29

WALLS, C. **Spring in action**. 4. ed. Shelter Island, NY: Manning Publications Co., 2015. 600 p. ISBN 9781617291203. 33

WEB, E. of the. **A evolução da web**. 2012. Disponível em: <http://www.evolutionoftheweb.com/>. Acesso em: mai. de 2015. 2

WEISSMANN, H. L. **Vire o jogo com Spring Framework**. 1. ed. São Paulo: Casa do Código, 2012. ISBN 978-85-66250-09-1. 33, 34

WOOLDRIDGE, J. M. **Introductory econometrics**: a modern approach. 6. ed. Boston, USA: CENGAGE Learning, 2015. ISBN 978-1305270107. 72

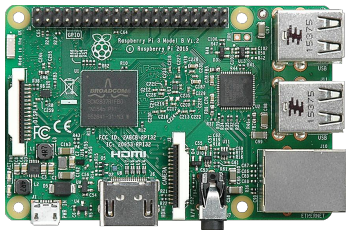
APÊNDICE A - MONTAGEM E CONFIGURAÇÃO DO *CLUSTER* RPi

O *cluster* foi construído e configurado a partir da compilação de uma série de tutoriais encontrados na *internet*. As etapas adotadas serão apresentadas nas próximas subseções.

A.1 Montagem do *cluster*

Na montagem do *cluster* foram utilizados diversos componentes, apresentados na Figura A.1 com suas respectivas quantidades.

Figura A.1 - Componentes utilizados na construção do *cluster*.



(a) 5x Raspberry Pi

(b) 1x HDD 100Gb

(c) 1x Switch 10/100Mbps 8p

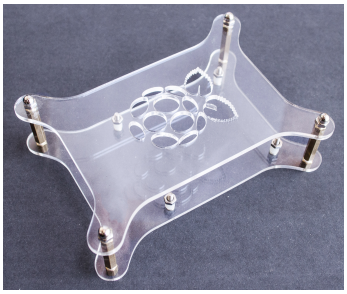
(d) 5x MicroSD 16Gb C10

(e) 5x Conector MicroUSB

(f) 6x Cabo de Rede

(g) 1x Fonte 5v 300W

(h) 3x Cooler

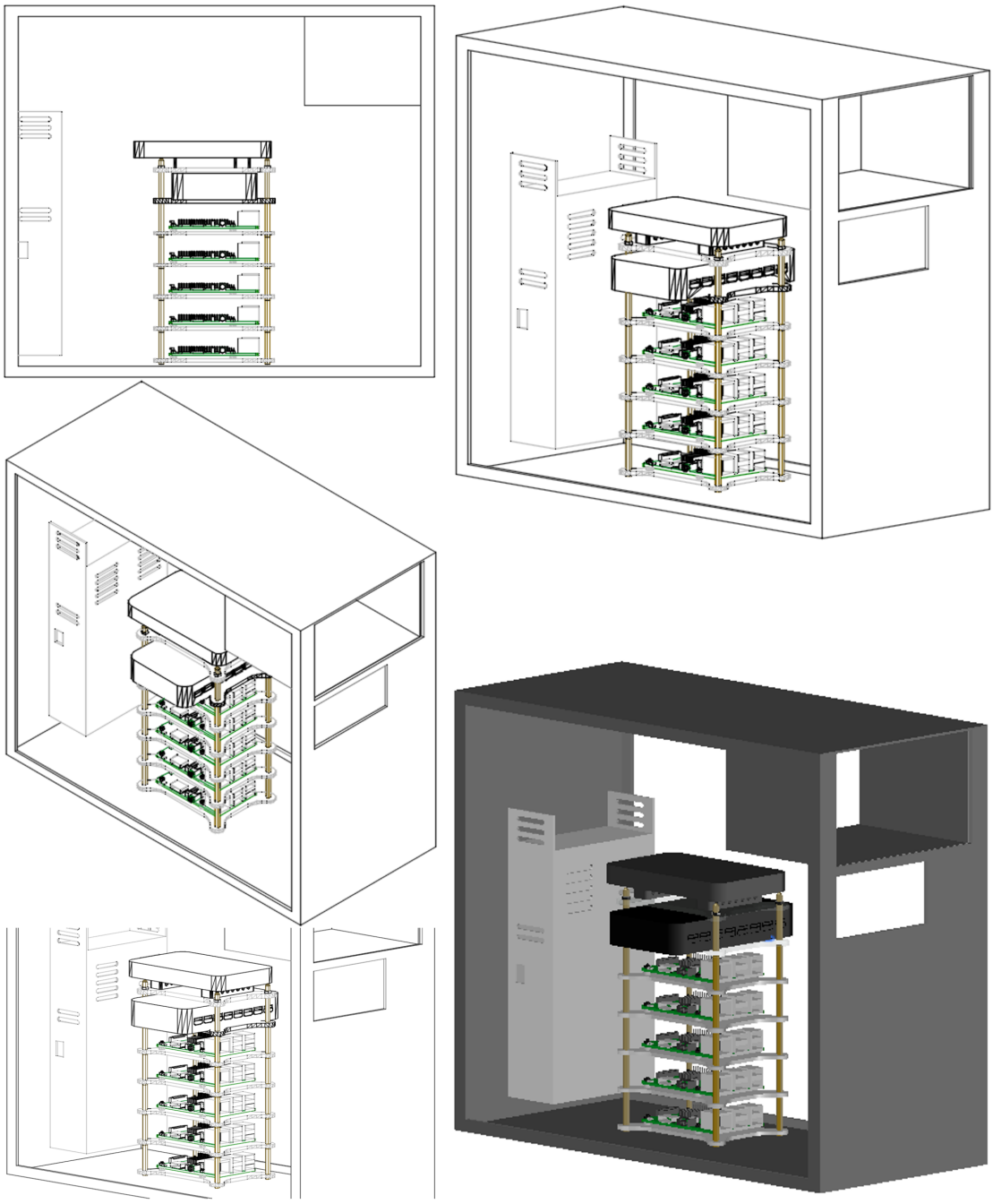


(i) 6x Case Acrílico

Fonte: Produção do Autor (2017).

Um projeto foi construído utilizando uma ferramenta CAD para auxiliar na visualização de como os componentes ficariam melhor dispostos na estrutura utilizada para acomodar o *cluster*. O projeto é apresentado na Figura A.2.

Figura A.2 - Projeto CAD do *cluster*.



Fonte: Produção do Autor (2016).

A Figura A.3 apresenta o *cluster* já finalizado e pronto para uso.

Figura A.3 - *Cluster* construído.

Fonte: Produção do Autor (2016).

A.2 Instalação do Sistema Operacional

O sistema adotado foi o Raspbian e sua instalação é bem simples. Seguindo os seguintes passos o Raspbian estará pronto para uso em alguns minutos.

1. O Raspbian será instalado em um cartão MicroSD, no computador formate-o em FAT ou FAT32. Nesta etapa foi utilizada a ferramenta SDFormatter.

2. Ainda no computador, baixe a imagem do Raspbian. O *download* é feito do *site* oficial do Raspberry Pi Foundation (`www.raspberrypi.org/downloads/raspbian/`).

3. A instalação no cartão MicroSD foi feita através da ferramenta Win32DiskImager. Localize a imagem do Raspbian que foi baixada e selecione a unidade que representa o cartão MicroSD. Na sequência, clique em `Write` e em `Yes`, logo em seguida. O tempo de gravação do Raspbian no cartão pode variar de acordo com a Classe do acessório utilizada. É recomendado o uso de um cartão Classe 10.

4. Concluída a instalação do sistema basta transferir o cartão do computador para o Raspberry Pi.

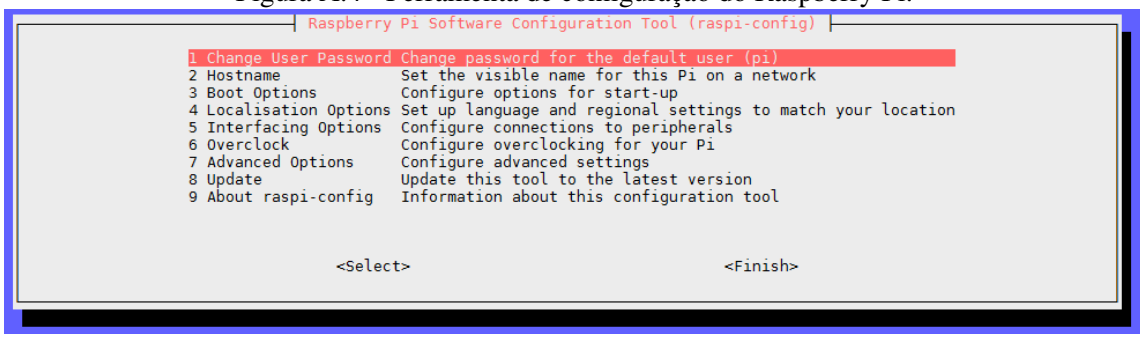
A.3 Configurando o Raspbian

O sistema vem pré-configurado e pronto para usar. Neste projeto foram feitos alguns ajustes na configuração padrão. Para acessar a ferramenta de configuração basta executar o comando

```
$ sudo raspi-config
```

A execução do comando irá abrir o menu de opções da ferramenta de configuração do Raspberry Pi, apresentado na Figura A.4.

Figura A.4 - Ferramenta de configuração do Raspberry Pi.



Através da ferramenta foram feitas duas alterações na configuração do sistema: (1) alteração do nome dos dispositivos que ficará visível na rede e (2) quantidade de memória disponibilizada para a GPU.

O *hostname* do dispositivo é alterado através da opção `2 Hostname`. Os nomes atribuídos aos dispositivos que compõe o *cluster* são: rpi01, rpi02, rpi03, rpi04 e rpi05. A quantidade de memória destinada a GPU é configurada através da opção `7 Advanced Options`, em seguida `A3 Memory Split`. A quantidade de memória foi definida para 16 em todos os dispositivos.

A.4 Configuração do endereço IP

Um IP fixo foi definido para cara um dos nós que compõe o *cluster*. A configuração é feita adicionando ao final do arquivo `/etc/dhcpcd.conf` as seguintes instruções:

```
$ sudo vi /etc/dhcpcd.conf
interface eth0

static ip_address=192.168.100.21/24
static routers=192.168.100.1
static domain_name_servers=192.168.100.1
```

Esta configuração deve ser aplicada em todos os nós variando o endereço IP. Após a modificação é necessário reiniciar a interface de rede ou simplesmente reiniciar o Raspberry Pi.

A.5 Mapeando os *hosts* conhecidos

Para facilitar o acesso aos *hosts* do *cluster* os *hostnames* foram mapeados no arquivo `/etc/hosts`. Este arquivo é utilizado pelo sistema para relacionar *hostnames* e IPs.

```
$ cat /etc/hosts
192.168.100.21 rpi01 rpi01.local rpi01.lan
192.168.100.22 rpi02 rpi02.local rpi02.lan
192.168.100.23 rpi03 rpi03.local rpi03.lan
192.168.100.24 rpi04 rpi04.local rpi04.lan
192.168.100.25 rpi05 rpi05.local rpi05.lan
```

A.6 Gerando chaves públicas SSH

Para evitar a requisição de senhas nas transferências ou mesmo nas sessões ssh, foram geradas chaves públicas. As chaves SSH de um usuário por padrão são armazeandas no diretório `~/.ssh`.

Para criar as chaves é utilizado o comando:

```
$ ssh-keygen -t rsa
```

O parâmetro `-t` especifica o tipo de chave, que nesse caso utiliza o algoritmo `rsa`. O resultado deve ser assim:

```
Generating public/private dsa key pair.
Enter file in which to save the key (/home/pi/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/pi/.ssh/id_rsa.
Your public key has been saved in /home/pi/.ssh/id_rsa.pub.
The key fingerprint is:
d7:b5:86:ec:53:19:53:49:93:1a:b1:57:94:b3:13:53
```

Primeiro confirme onde quer salvar a chave (`.ssh/id_rsa`), e então informar duas vezes por uma senha, que pode ser deixada em branco caso não queira digitar uma senha quando usar a chave.

A chave pública, cujo a extensão é .pub, deve ser enviada para os servidores. Para isso pode ser utilizado o comando:

```
$ ssh-copy-id pi@rpi02
```

Dessa vez ainda será requisitada a senha para transferência.

Uma chave pública é parecida com isso:

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCfghMQ4b0QNerxD2waDRRI
TA3NfvtZM6xycEuINhHjwnu4+VtUiig1Jb2gpb/qT3MVagFQ0oDLgy8juuF0
IiL+owLoy/znHFCYKNLhddXgd0MIwb8OvOPoVPrm3XDg8zqJHl4cZcUXe0H3
pt2IPh16abghIX6AEhQEA8R4JrfMCOZcbEn8RzDY5DPKKvvW/BoFb5oyAgPf
XRI0aL/Ix2LdTaP/i7b1sa358sz6v7bKcvS3gP97GSXTOL0nYAaiMwxTBrnu
nqF+BQwGAqMaiZ3xz56qNPMcUvJynUTMy48jf2QM3fZszcCB59yqsx/f0ZYD
Za6RX4Xd0or5dV6SHCRl pi@rpi01
```

Para verificar se a chave pública funciona corretamente basta executar o comando:

```
$ ssh pi@rpi02
```

A.7 Instalação do Docker

O instalação e configuração do Docker é bastante simples. Basta executar os seguintes comandos em cada um dos nós do *cluster*.

A instalação é feita através do comando:

```
$ curl -sSL get.docker.com | sh
```

O Docker pode ser configurado para iniciar automaticamente no momento do *boot*. Para isso basta utilizar o comando:

```
$ sudo systemctl enable docker
```

Para iniciar o Docker manualmente é utilizado o comando:

```
$ sudo systemctl start docker
```

Para executar o Docker sem a necessidade do comando `sudo` é preciso adicionar o usuário ao grupo `docker`, isso é feito com o comando:

```
$ sudo usermod -aG docker $USER
```

A.8 Montagem do *cluster* com Docker Swarm

Com o Docker devidamente instalado e configurado pode-se seguir para a criação do *cluster* com a ferramenta swarm a partir de um dos *hosts*. Neste caso o *host* escolhido foi o **rpi01**.

O comando utilizado para a inicialização do *cluster* é

```
$ docker swarm init
```

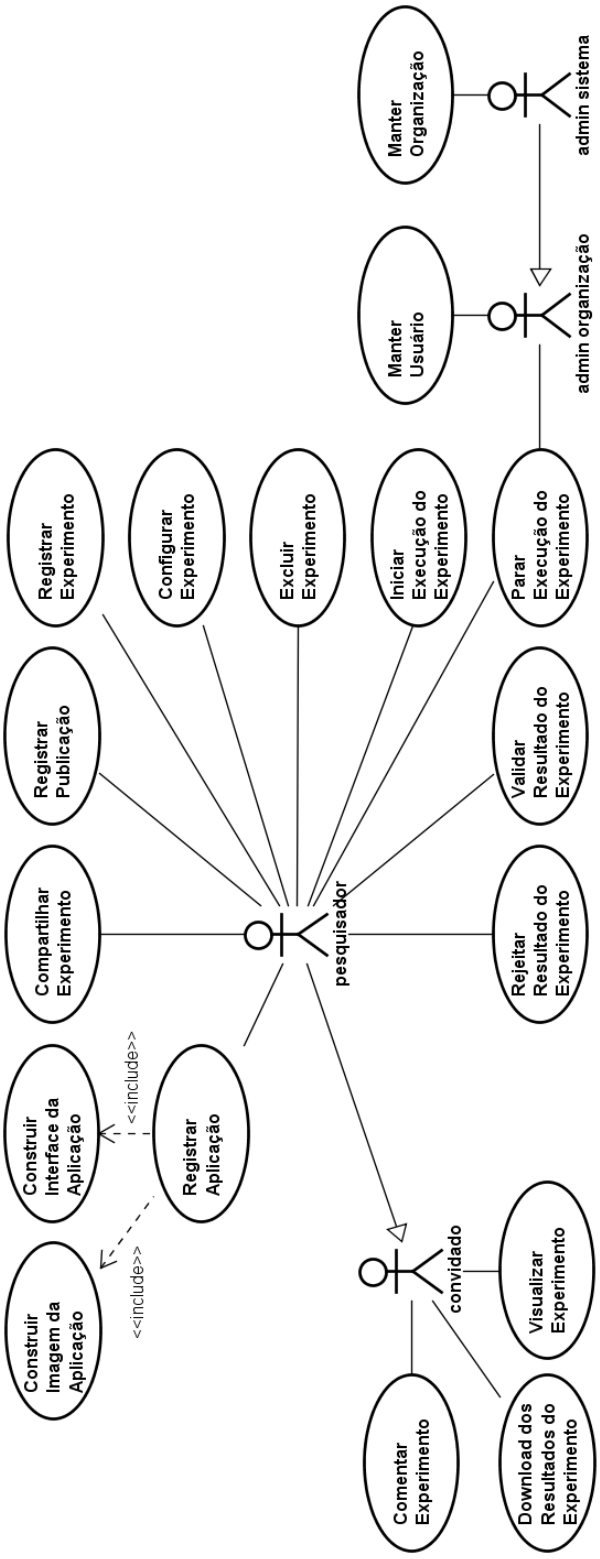
Terminada a inicialização este nó se torna o mestre, ou *leader*, e a saída do comando já informa o comando necessário para acrescenter novos nós ao *cluster*.

```
...
To add a worker to this swarm, run the following command:
    docker swarm join \
    --token SWMTKN-1-«HASH» \
    192.168.100.21:2377
...
```

APÊNDICE B - DETALHAMENTO DA MODELAGEM DO SISTEMA

B.1 Diagrama de Casos de Uso

Figura B.1 - Diagrama de Casos de Uso.



B.1.1 Descrição dos Casos de Uso

B.1.1.1 Caso de Uso Registrar Aplicação

Tabela B.1 - Descrição do Caso de Uso Registrar Aplicação

Caso de Uso:	**Registrar Aplicação**
Resumo:	Permite o cadastro de nova aplicação na área de trabalho do pesquisador.
Ator:	pesquisador
Pré-Condições:	
Pós-Condições:	(1) Acionar Caso de Uso “Construir Interface da Aplicação”. (2) Acionar Caso de Uso “Construir Imagem da Aplicação”.
Fluxo Principal:	Pesquisador deve preencher formulário de cadastro de aplicação.
Fluxo Alternativo:	
Fluxo de Exceção:	

Notas:	visibilidade {PRIVATE / PUBLIC},
	argumentos {
	nome,
	label,
	tipo {TEXT/NUMBER/FILE},
	valores,
	descrição,
	valor padrão
	}

B.1.1.2 Caso de Uso Registrar Experimento

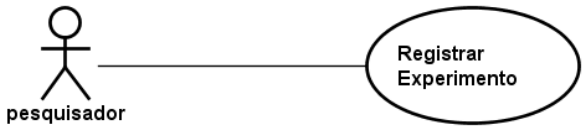


Tabela B.2 - Descrição do Caso de Uso Registrar Experimento

Caso de Uso:	**Registrar Experimento**
Resumo:	Permite a criação de novo experimento.
Ator:	pesquisador
Pré-Condições:	Aplicação validada pelo sistema.
Pós-Condições:	Exibir no *dashboard* do usuário
Fluxo Principal:	(1) Selecionar aplicação. (2) Preencher formulário de registro de experimento.
Fluxo Alternativo:	
Fluxo de Exceção:	
Notas:	

B.1.1.3 Caso de Uso Manter Usuário

Tabela B.3 - Descrição do Caso de Uso Manter Usuário

Caso de Uso:	**Manter Usuário**
Resumo:	Permite ao administrador gerenciar o cadastro dos usuário do sistema.
Ator:	admin sistema e/ou admin organização
Pré-Condições:	Deve existir no mínimo uma organização cadastrada.
Pós-Condições:	
Fluxo Principal:	(1) Informar dados do usuário.
Fluxo Alternativo:	
Fluxo de Exceção:	
Notas:	Admin só pode gerenciar usuários com mesmo nível ou com nível inferior.

B.1.1.4 Caso de Uso Manter Organização

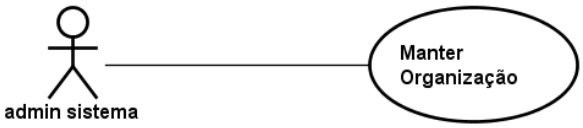


Tabela B.4 - Descrição do Caso de Uso Manter Organização

Caso de Uso:	**Manter Organização**
Resumo:	Permite o gerenciamento das organizações cadastradas no sistema.
Ator:	admin sistema
Pré-Condições:	
Pós-Condições:	
Fluxo Principal:	(1) Informar dados da organização.
Fluxo Alternativo:	
Fluxo de Exceção:	
Notas:	

B.1.1.5 Caso de Uso Configurar Experimento

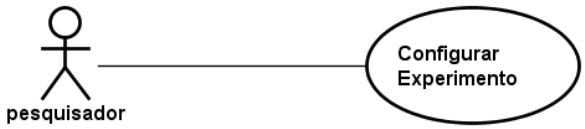


Tabela B.5 - Descrição do Caso de Uso Configurar Experimento

Caso de Uso:	**Configurar Experimento**
Resumo:	Permitir ao pesquisador configurar seu experimento para execução.
Ator:	pesquisador
Pré-Condições:	
Pós-Condições:	
Fluxo Principal:	(1) Carregar interface do experimento. (2) Usuário preenche os argumentos com os valores iniciais para execução.
Fluxo Alternativo:	
Fluxo de Exceção:	
Notas:	

B.1.1.6 Caso de Uso Iniciar Execução do Experimento

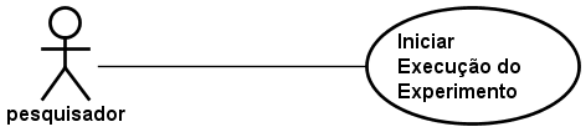


Tabela B.6 - Descrição do Caso de Uso Iniciar Execução do Experimento

Caso de Uso:	**Iniciar Execução do Experimento**
Resumo:	Permite que o usuário inicie a execução de um experimento criado.
Ator:	pesquisador
Pré-Condições:	Experimento cadastrado.
Pós-Condições:	Enviar email ao responsável pelo experimento com atualização de status.
Fluxo Principal:	(1) Recuperar dados de entrada e argumentos registrados. (2) Iniciar *container* a partir da imagem da aplicação. (3) Disparar execução da aplicação. (4) Transferir resultados. (5) Finalizar *container*.
Fluxo Alternativo:	
Fluxo de Exceção:	
Notas:	Arquivos de *log* da execução ficam disponíveis para verificação.

B.1.1.7 Caso de Uso Parar Execução do Experimento

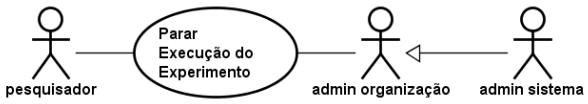


Tabela B.7 - Descrição do Caso de Uso Parar Execução do Experimento

Caso de Uso:	**Parar Execução do Experimento**
Resumo:	Permitir ao usuário parar a execução de um experimento se necessário.
Ator:	pesquisador, admin organização e admin sistema
Pré-Condições:	Experimento em execução.
Pós-Condições:	
Fluxo Principal:	(1) Identificar *container* de execução. (2) Parar a execução da aplicação. (3) Finalizar *container*.
Fluxo Alternativo:	
Fluxo de Exceção:	
Notas:	

B.1.1.8 Caso de Uso Visualizar Experimento

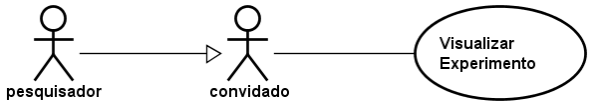


Tabela B.8 - Descrição do Caso de Uso Visualizar Experimento

Caso de Uso:	**Visualizar Experimento**
Resumo:	Permitir ao usuário visualizar os resultados obtidos na execução do experimento.
Ator:	pesquisador, convidado
Pré-Condições:	Execução do experimento estar finalizada (*status* COMPLETO).
Pós-Condições:	
Fluxo Principal:	(1) Exibir resultados*
Fluxo Alternativo:	
Fluxo de Exceção:	
Notas:	*Visualização só é possível se formato dos dados for suportado pela aplicação.

B.1.1.9 Caso de Uso Validar Resultado do Experimento

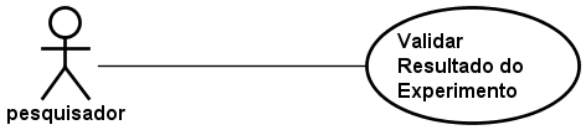


Tabela B.9 - Descrição do Caso de Uso Validar Resultado do Experimento

Caso de Uso:	**Validar Resultado do Experimento**
Resumo:	Permitir ao usuário aceitar o resultado obtido com o experimento.
Ator:	pesquisador
Pré-Condições:	Execução do experimento estar finalizada (*status* COMPLETO).
Pós-Condições:	
Fluxo Principal:	(1) Selecionar experimento. (2) Validar resultados.
Fluxo Alternativo:	
Fluxo de Exceção:	
Notas:	

B.1.1.10 Caso de Uso Rejeitar Resultado do Experimento

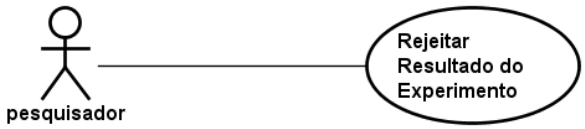


Tabela B.10 - Descrição do Caso de Uso Rejeitar Resultado do Experimento

Caso de Uso:	**Rejeitar Resultado do Experimento**
Resumo:	Permitir ao usuário rejeitar o resultado obtido com o experimento excluindo seus resultados mas mantém experimento e configuração de execução visíveis no *dashboard*.
Ator:	pesquisador
Pré-Condições:	Execução do experimento estar finalizada (*status* COMPLETO).
Pós-Condições:	
Fluxo Principal:	(1) Selecionar experimento. (2) Rejeitar resultados. (3) Excluir resultados.
Fluxo Alternativo:	
Fluxo de Exceção:	
Notas:	

B.1.1.11 Caso de Uso Compartilhar Experimento

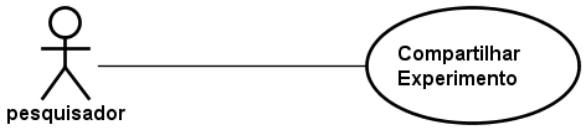


Tabela B.11 - Descrição do Caso de Uso Compartilhar Resultado

Caso de Uso:	**Compartilhar Resultado**
Resumo:	Possibilita o compartilhamento do resultado com outros pesquisadores.
Ator:	pesquisador
Pré-Condições:	Execução do experimento estar finalizada (*status* COMPLETO)
Pós-Condições:	
Fluxo Principal:	(1) Selecionar experimento. (2) Indicar convidados. (3) Enviar *e-mail* aos convidados com *link* de acesso.
Fluxo Alternativo:	
Fluxo de Exceção:	
Notas:	

B.1.1.12 Caso de Uso Download Resultados do Experimento

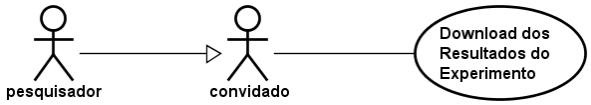


Tabela B.12 - Descrição do Caso de Uso Download Resultados do Experimento

Caso de Uso:	**Download Resultados do Experimento**
Resumo:	Realizar o *download* dos resultados obtidos para que possam ser analisados com outras ferramentas.
Ator:	pesquisador e convidado
Pré-Condições:	Execução do experimento estar finalizada (*status* COMPLETO).
Pós-Condições:	
Fluxo Principal:	(1) Selecionar experimento. (2) *Download* do resultado.
Fluxo Alternativo:	
Fluxo de Exceção:	
Notas:	

B.1.1.13 Caso de Uso Excluir Experimento

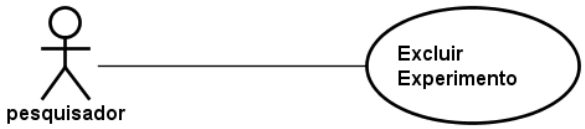


Tabela B.13 - Descrição do Caso de Uso Excluir Experimento

Caso de Uso:	**Excluir Experimento**
Resumo:	Permitir a remoção do experimento do *dashboard*.
Ator:	pesquisador
Pré-Condições:	Experimento cadastrado.
Pós-Condições:	
Fluxo Principal:	(1) Selecionar experimento. (2) Excluir experimento. (2.1) Excluir arquivos de entrada. (2.2) Excluir arquivos de saída. (2.3) Excluir do *dashboard*.
Fluxo Alternativo:	
Fluxo de Exceção:	
Notas:	

B.1.1.14 Caso de Uso Comentar Experimento

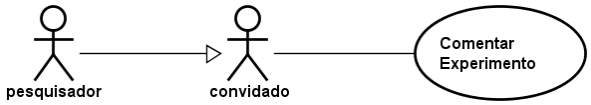


Tabela B.14 - Descrição do Caso de Uso Comentar Experimento

Caso de Uso:	**Comentar Experimento**
Resumo:	Permitir que outros usuários contribuam com a pesquisa.
Ator:	pesquisador e convidado
Pré-Condições:	
Pós-Condições:	
Fluxo Principal:	(1) Selecionar experimento. (2) Enviar comentário.
Fluxo Alternativo:	
Fluxo de Exceção:	
Notas:	

B.1.1.15 Caso de Uso Registrar Publicação

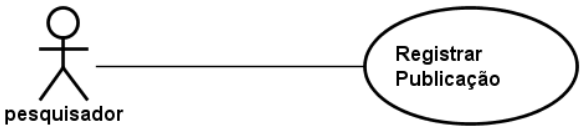
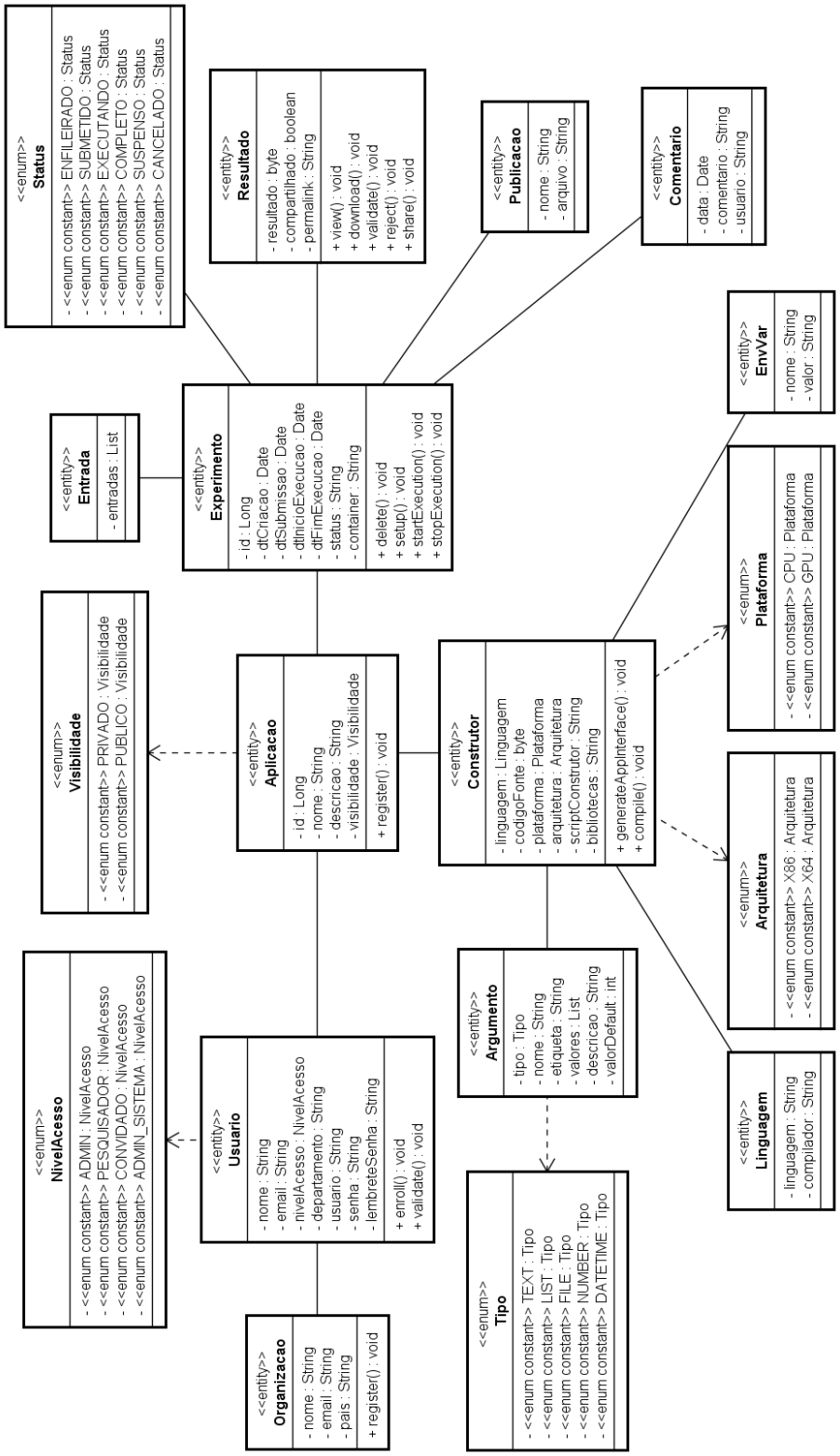


Tabela B.15 - Descrição do Caso de Uso Registrar Publicação

Caso de Uso:	**Registrar Publicação**
Resumo:	Permitir ao usuário registrar publicação gerada a partir do resultado do experimento para que outros possam ter acesso.
Ator:	pesquisador
Pré-Condições:	
Pós-Condições:	
Fluxo Principal:	(1) Selecionar experimento. (2) Selecionar arquivo. (3) Registrar publicação.
Fluxo Alternativo:	
Fluxo de Exceção:	
Notas:	Publicações podem ser artigos, apresentações, ou qualquer outro produto gerado a partir do experimento.

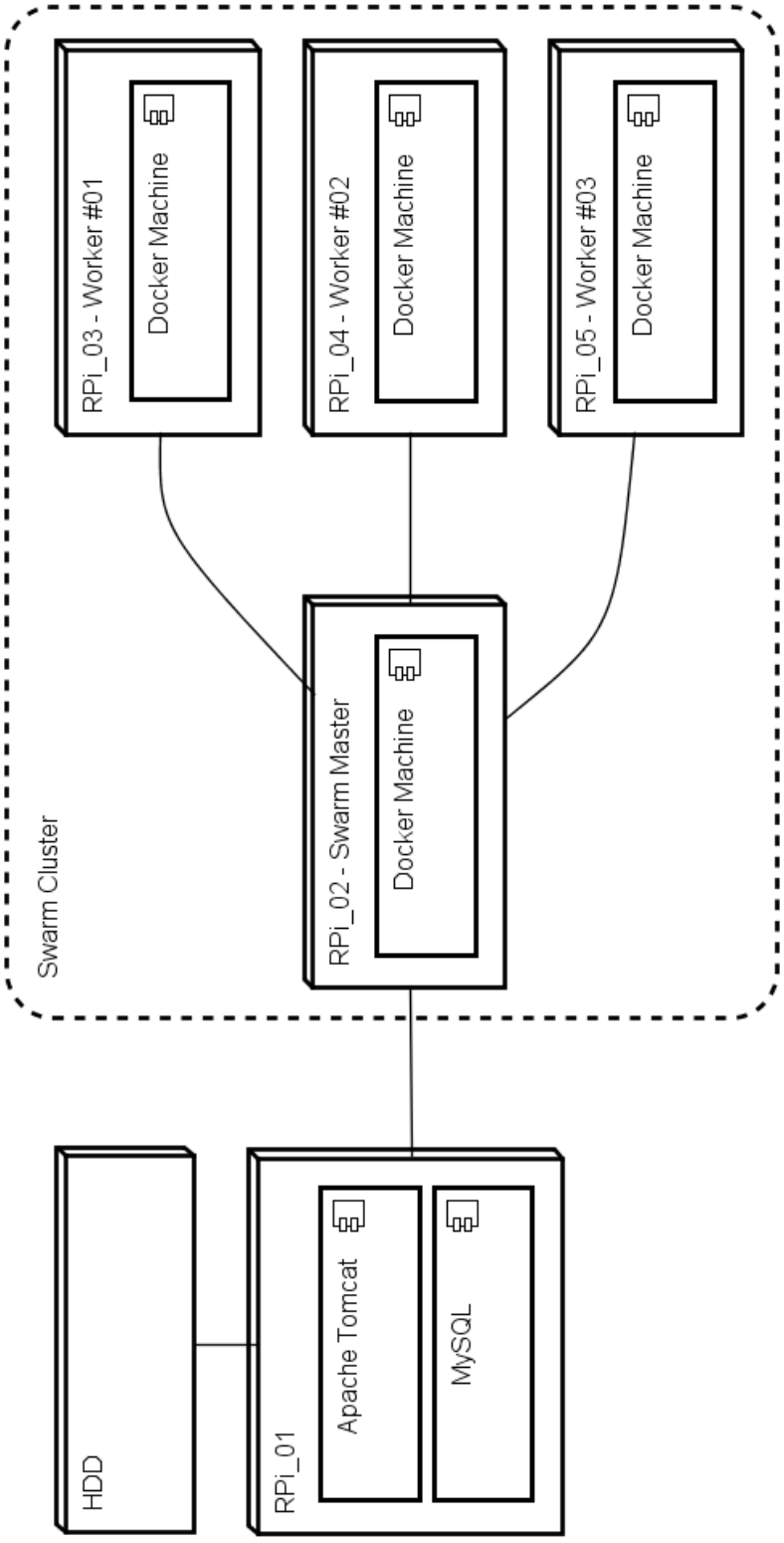
B.2 Diagrama de Classes

Figura B.2 - Diagrama de Classes.



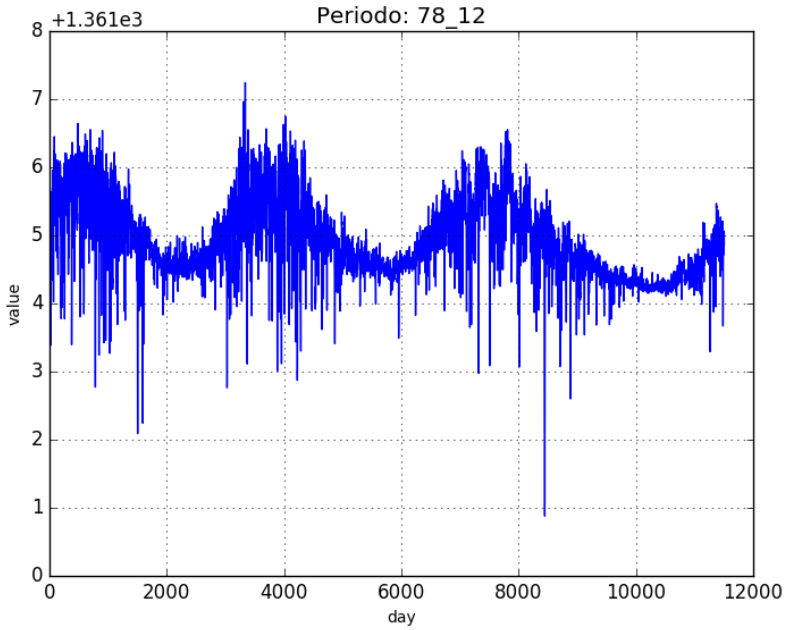
B.3 Diagrama de Implantação

Figura B.3 - Diagrama de Implantação.

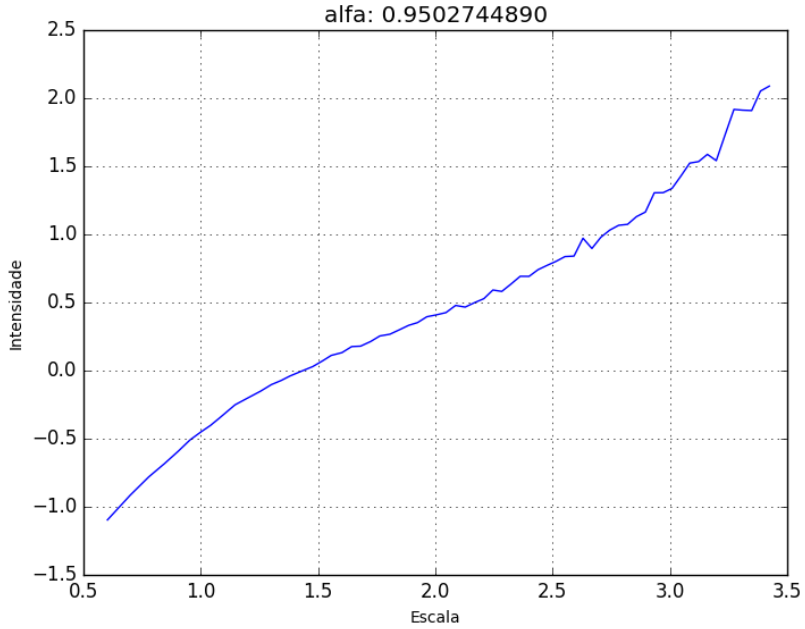


APÊNDICE C - RESULTADOS COMPLEMENTARES DFA

Figura C.1 - Série Temporal Total Solar Irradiance (TSI).

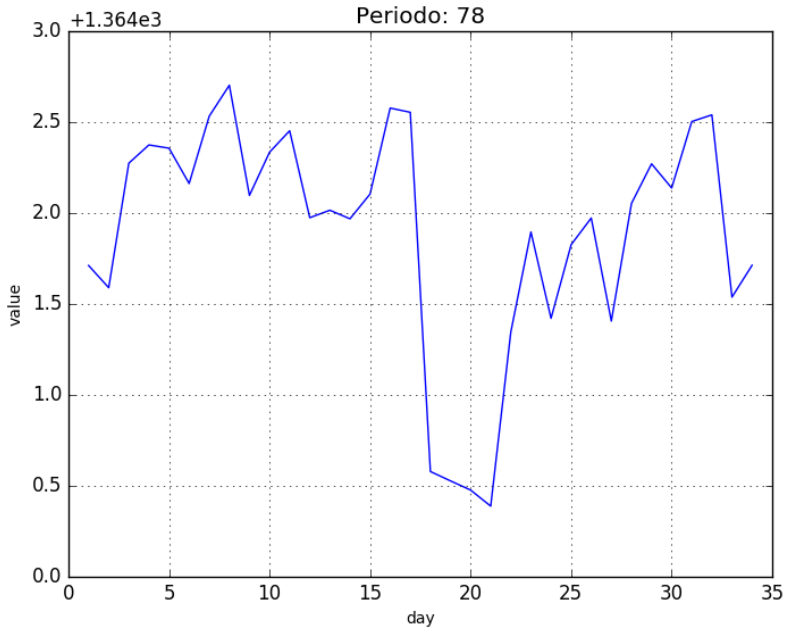


(a) Série Completa

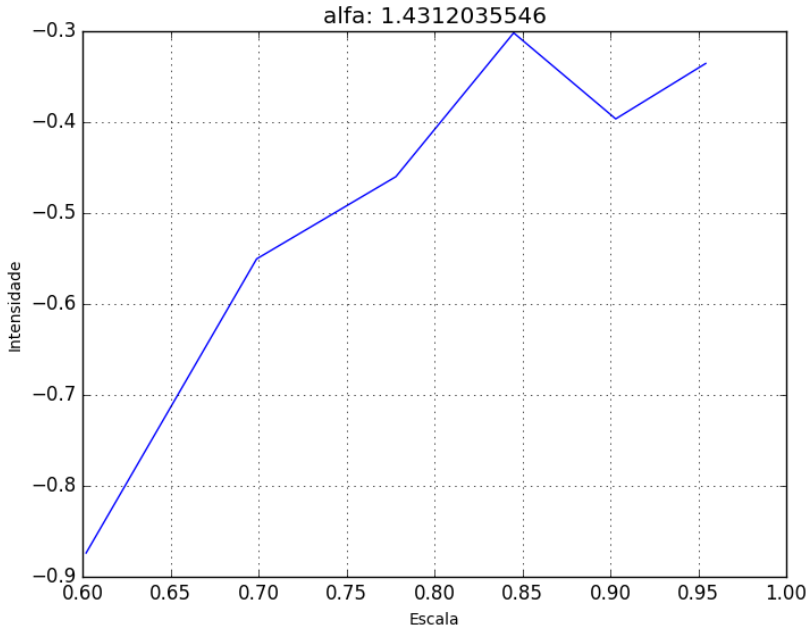


(b) DFA

Figura C.2 - Série Temporal TSI - Janela 1978.

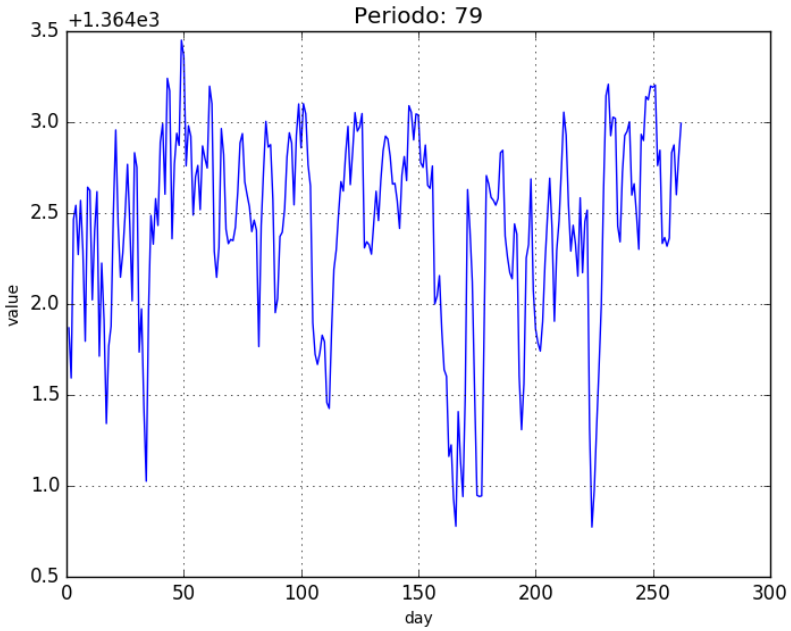


(a) Série Temporal 1978

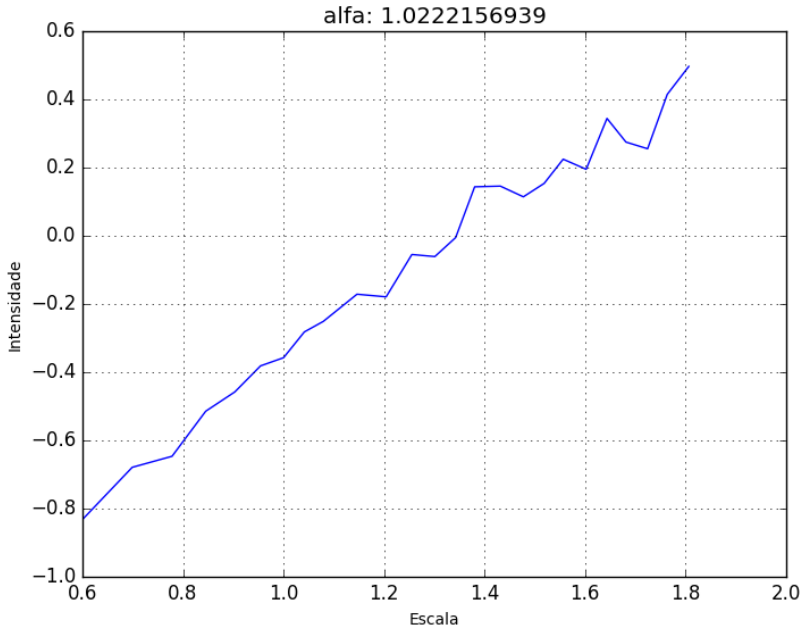


(b) DFA 1978

Figura C.3 - Série Temporal TSI - Janela 1979.

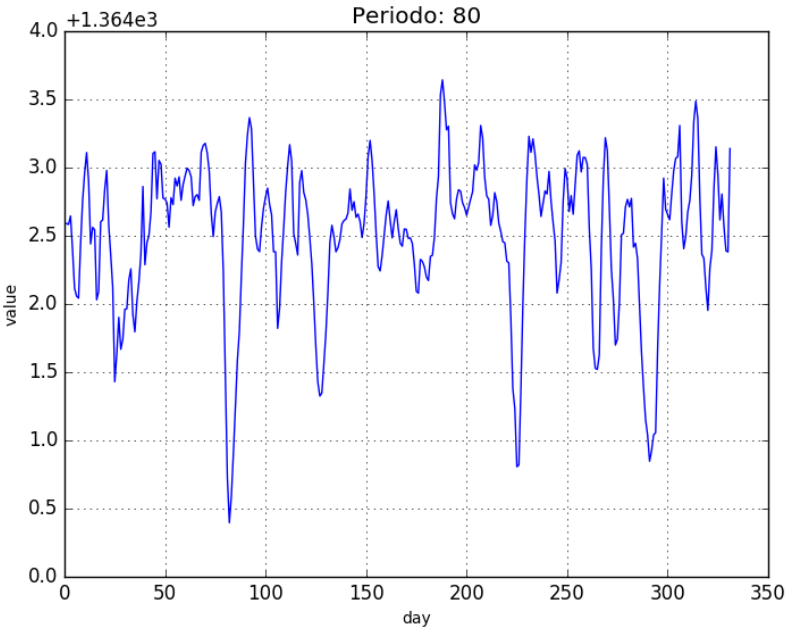


(a) Série Temporal 1979

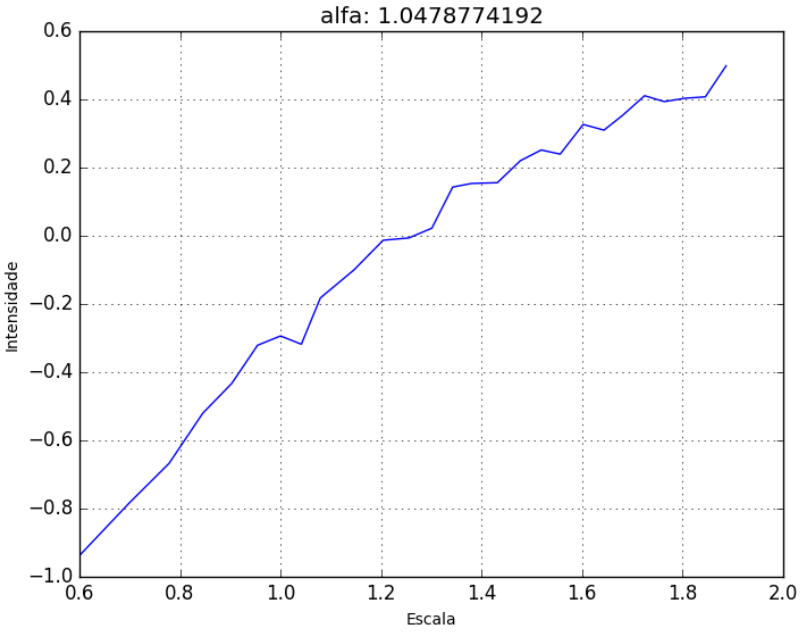


(b) DFA 1979

Figura C.4 - Série Temporal TSI - Janela 1980.

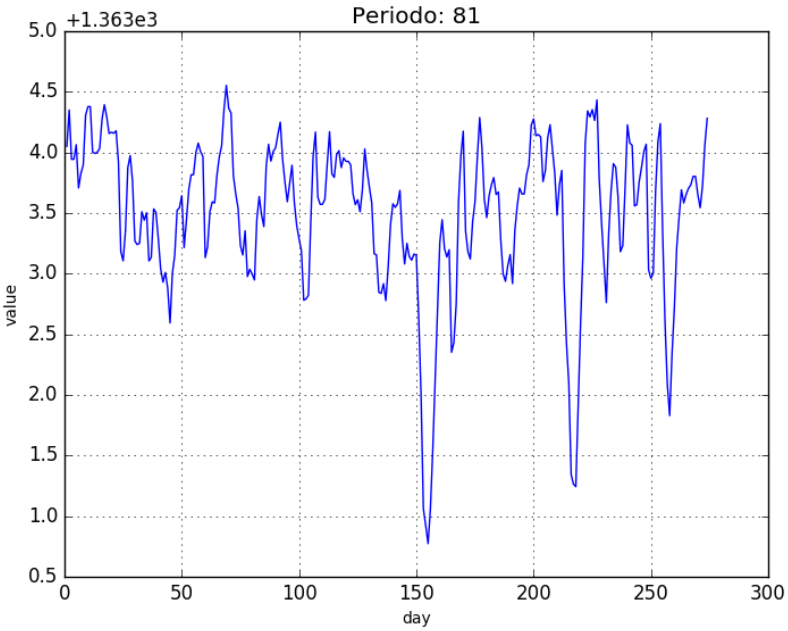


(a) Série Temporal 1980

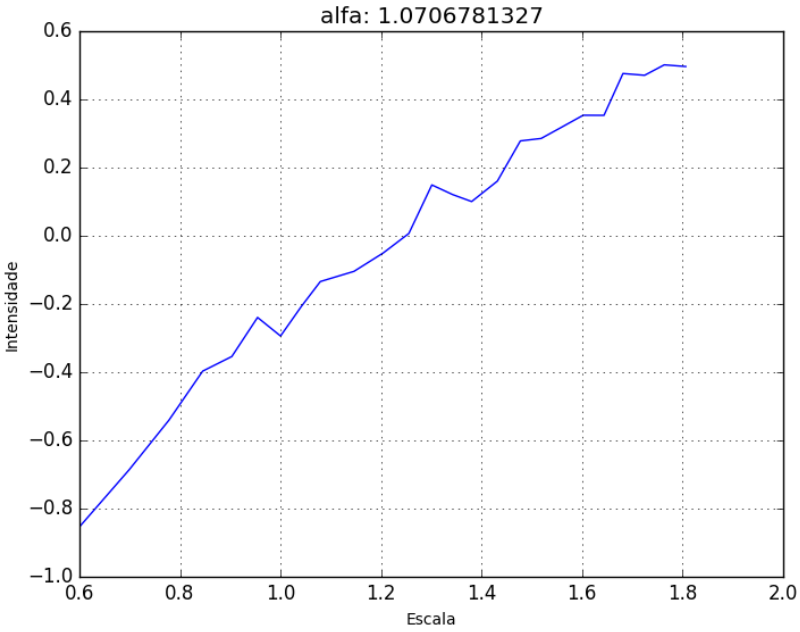


(b) DFA 1980

Figura C.5 - Série Temporal TSI - Janela 1981.

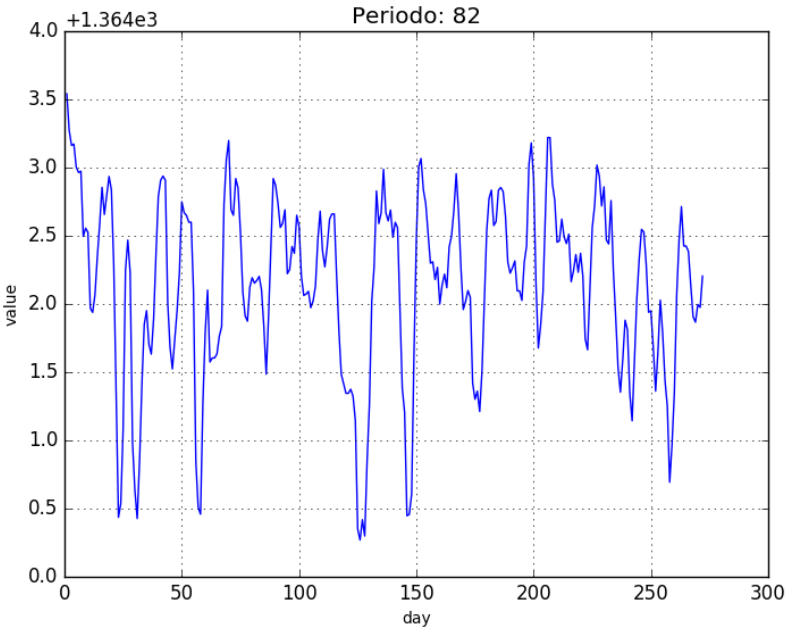


(a) Série Temporal 1981

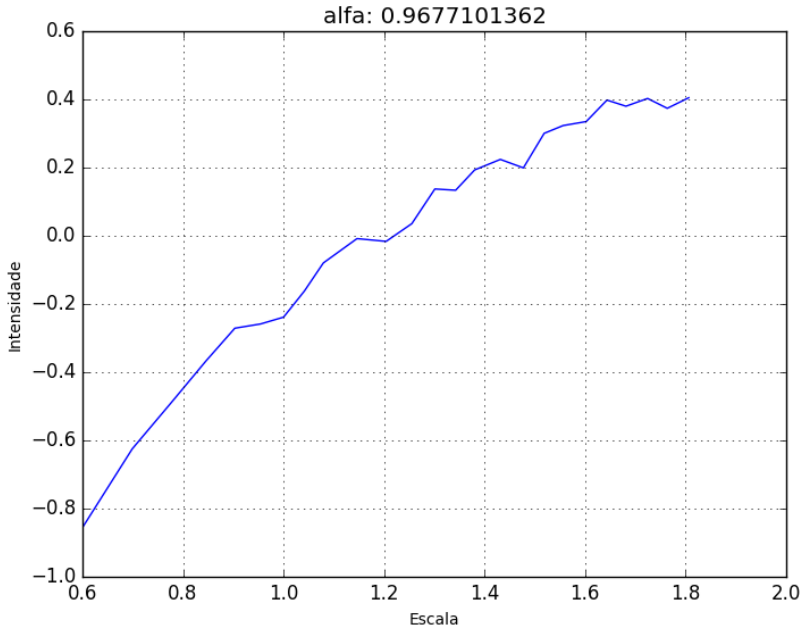


(b) DFA 1981

Figura C.6 - Série Temporal TSI - Janela 1982.

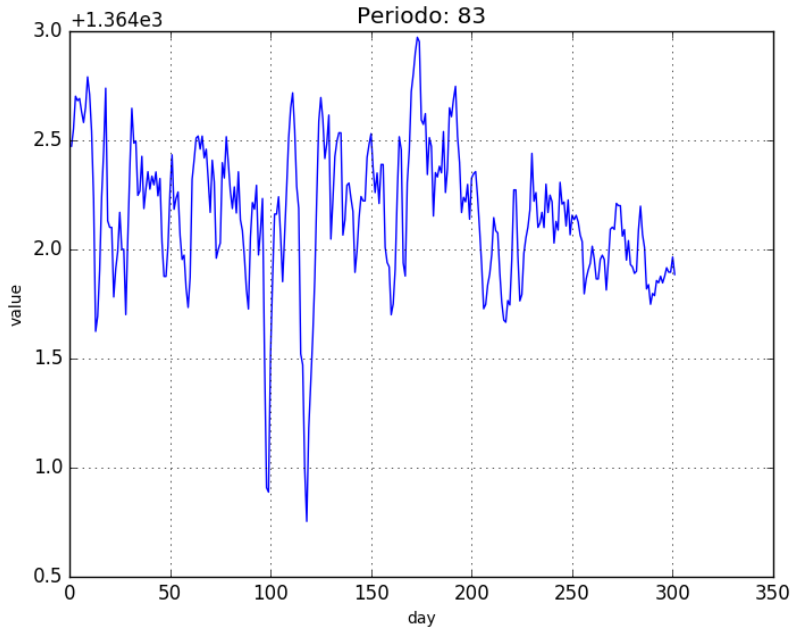


(a) Série Temporal 1982

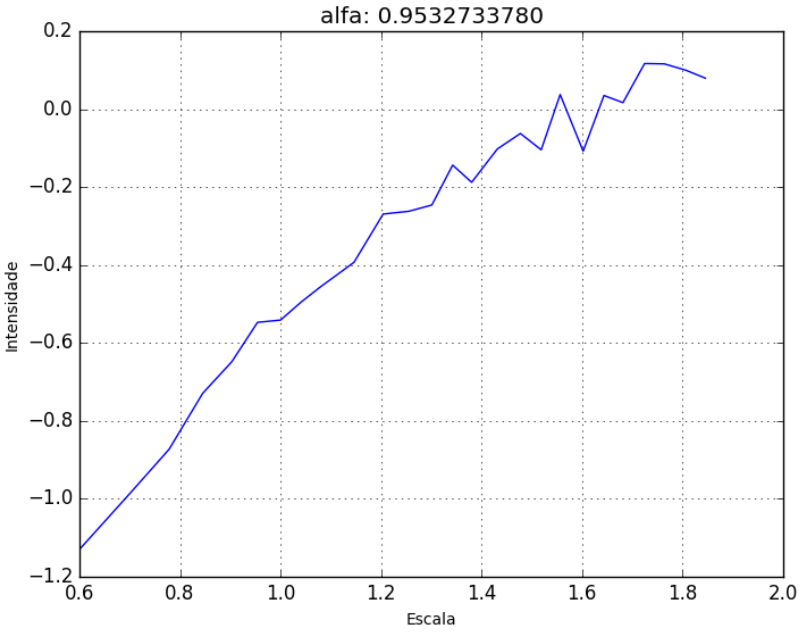


(b) DFA 1982

Figura C.7 - Série Temporal TSI - Janela 1983.

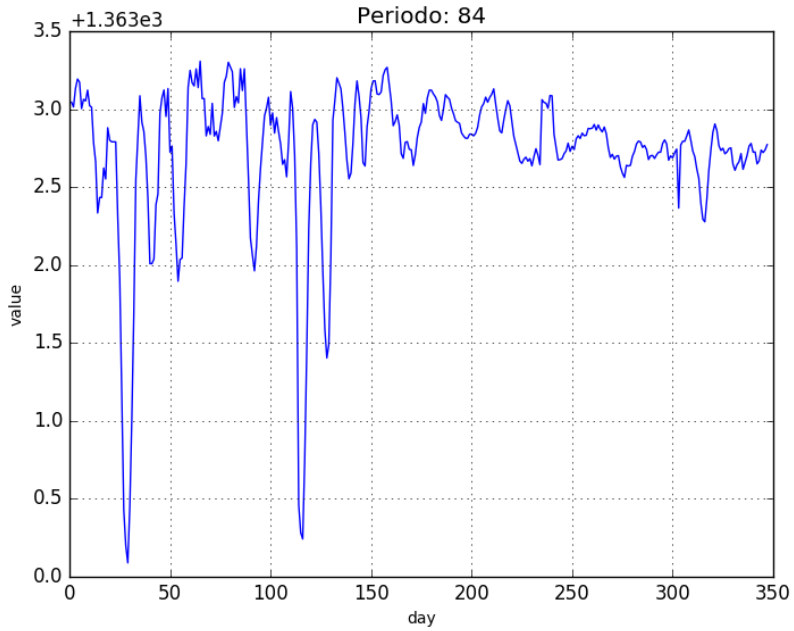


(a) Série Temporal 1983

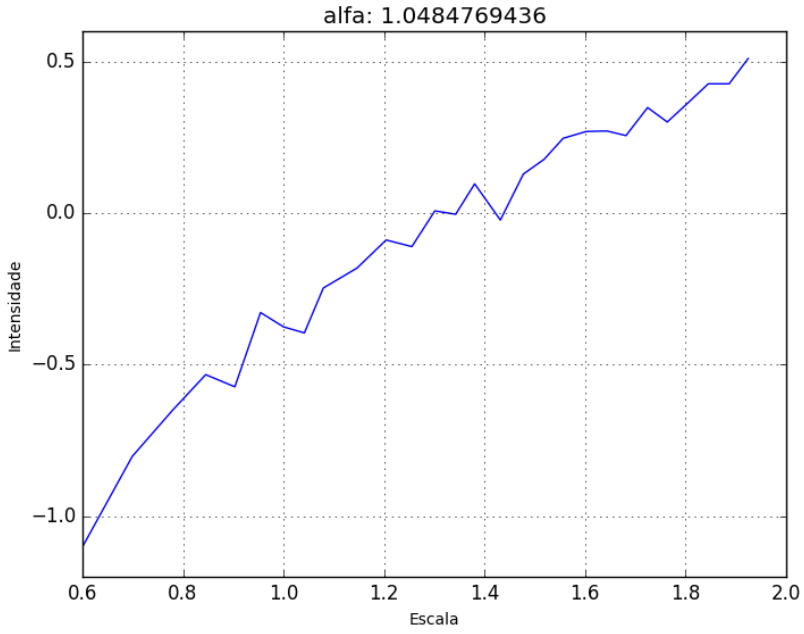


(b) DFA 1983

Figura C.8 - Série Temporal TSI - Janela 1984.

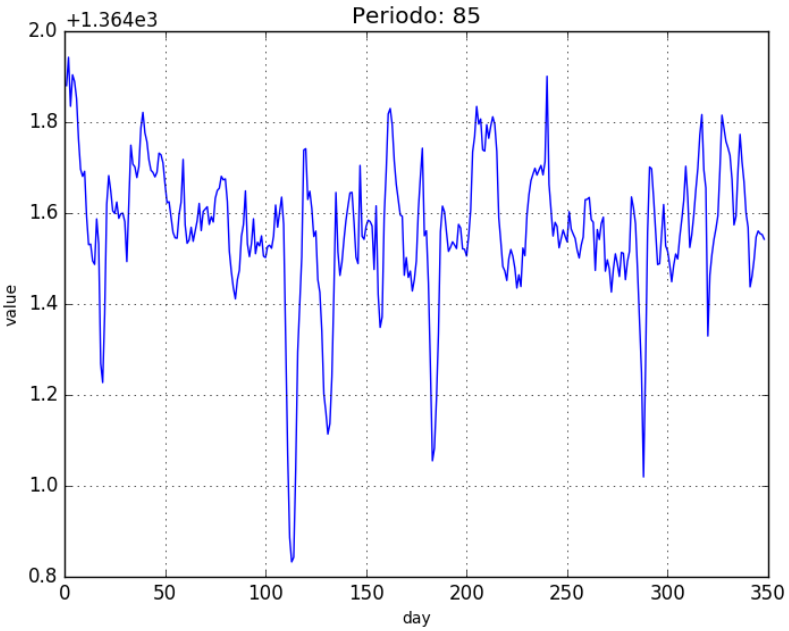


(a) Série Temporal 1984

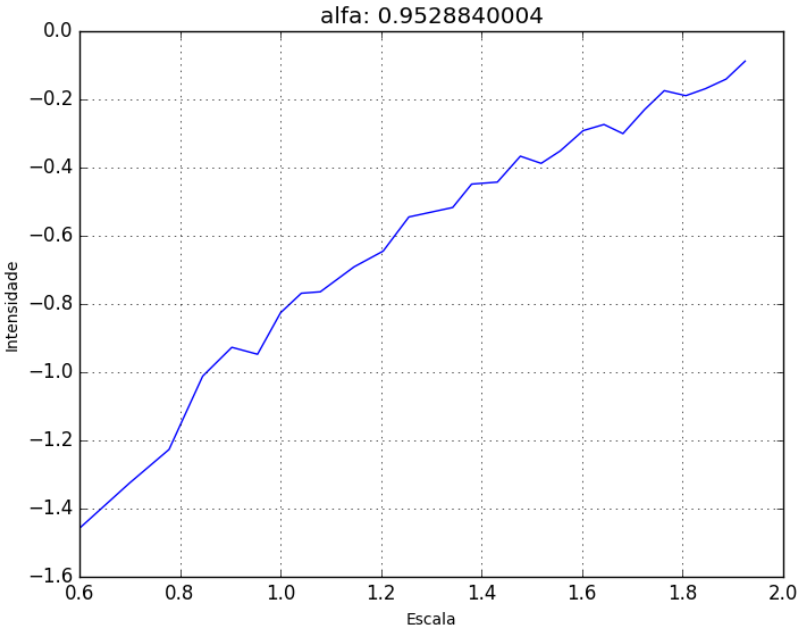


(b) DFA 1984

Figura C.9 - Série Temporal TSI - Janela 1985.

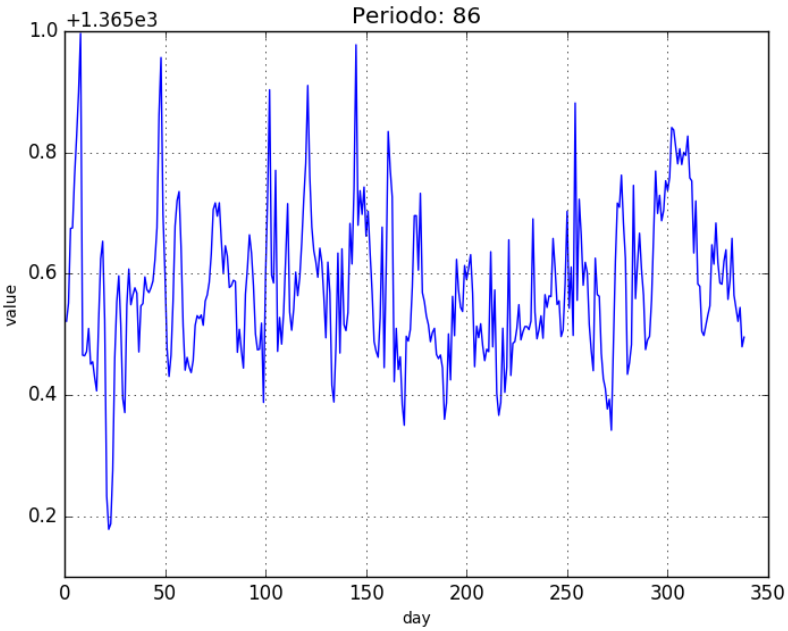


(a) Série Temporal 1985

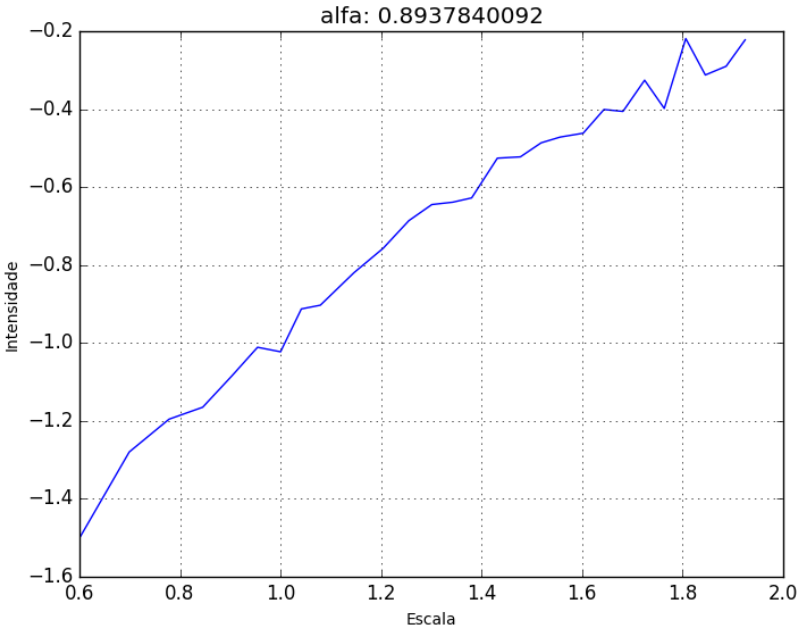


(b) DFA 1985

Figura C.10 - Série Temporal TSI - Janela 1986.

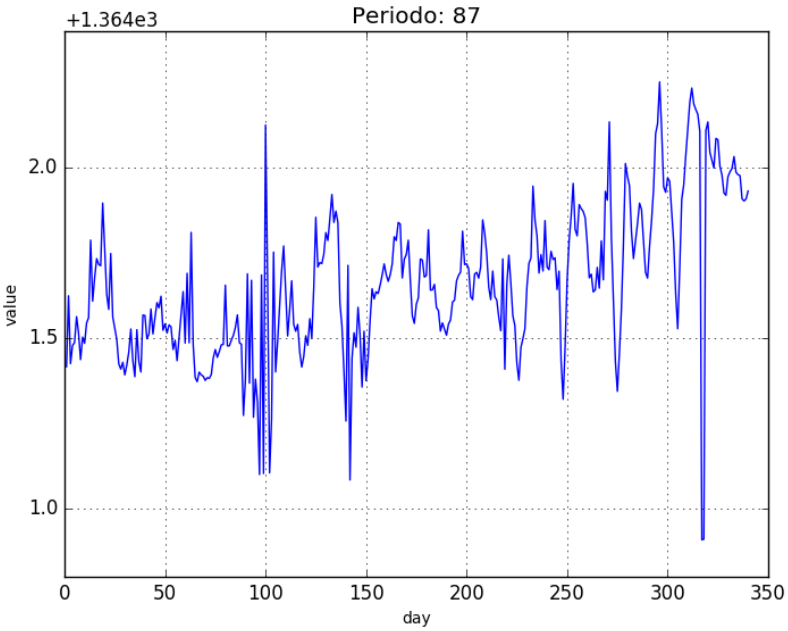


(a) Série Temporal 1986

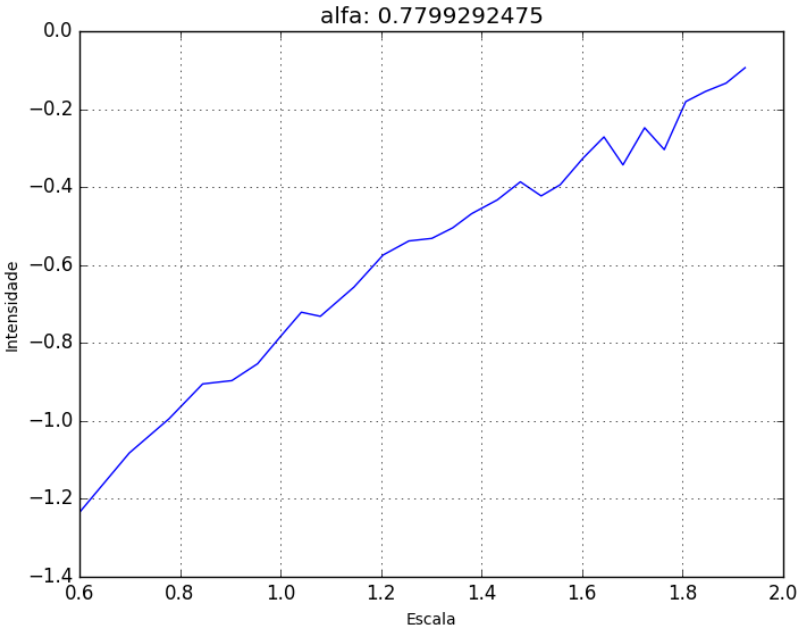


(b) DFA 1986

Figura C.11 - Série Temporal TSI - Janela 1987.

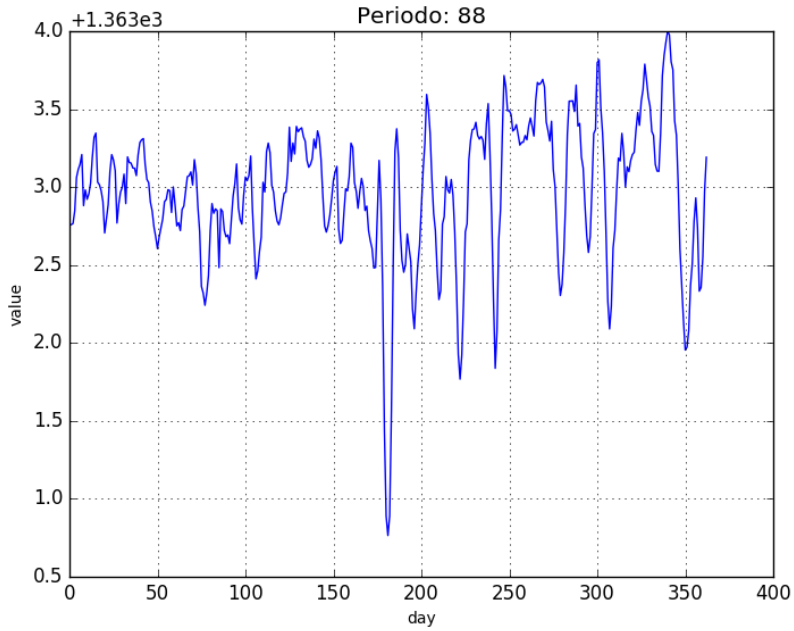


(a) Série Temporal 1987

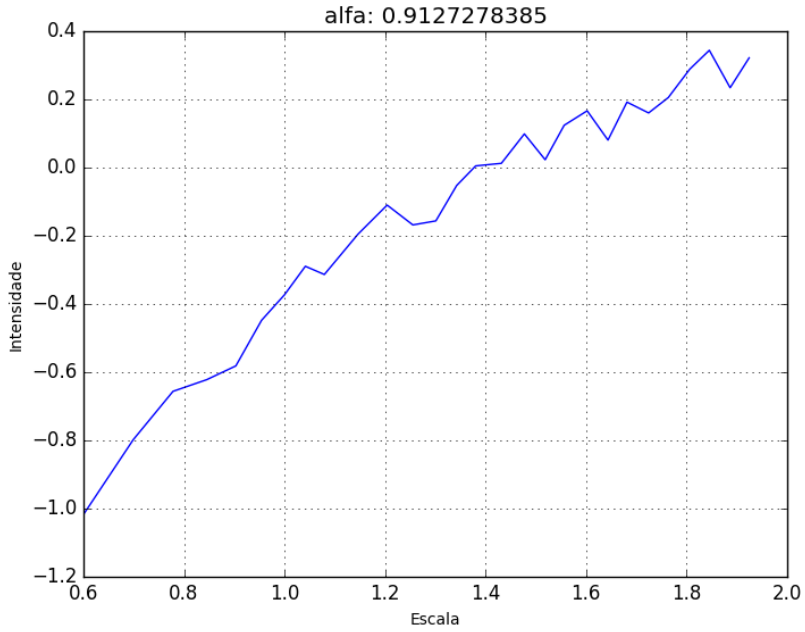


(b) DFA 1987

Figura C.12 - Série Temporal TSI - Janela 1988.

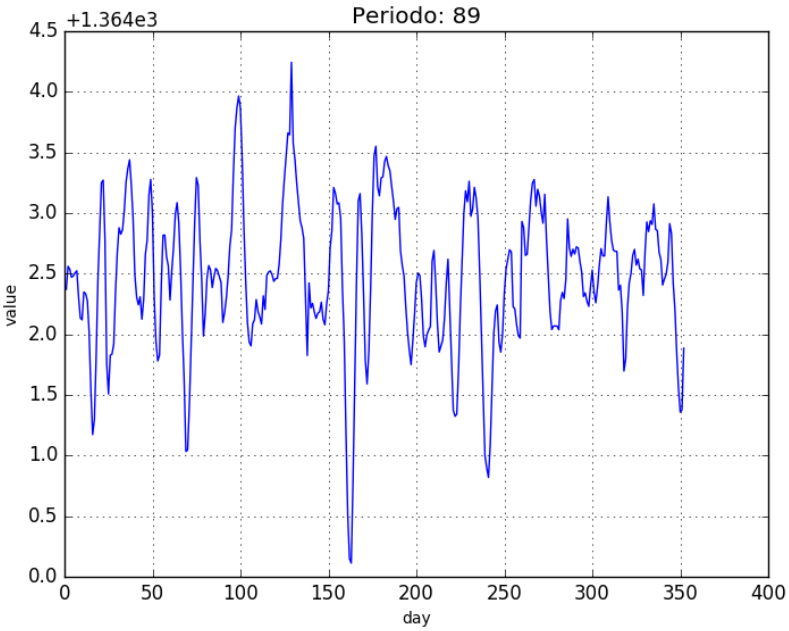


(a) Série Temporal 1988

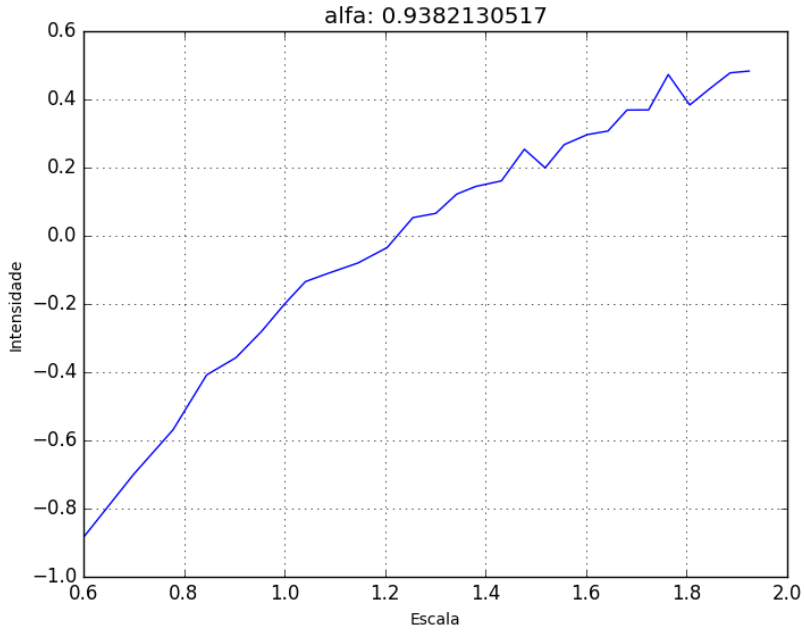


(b) DFA 1988

Figura C.13 - Série Temporal TSI - Janela 1989.

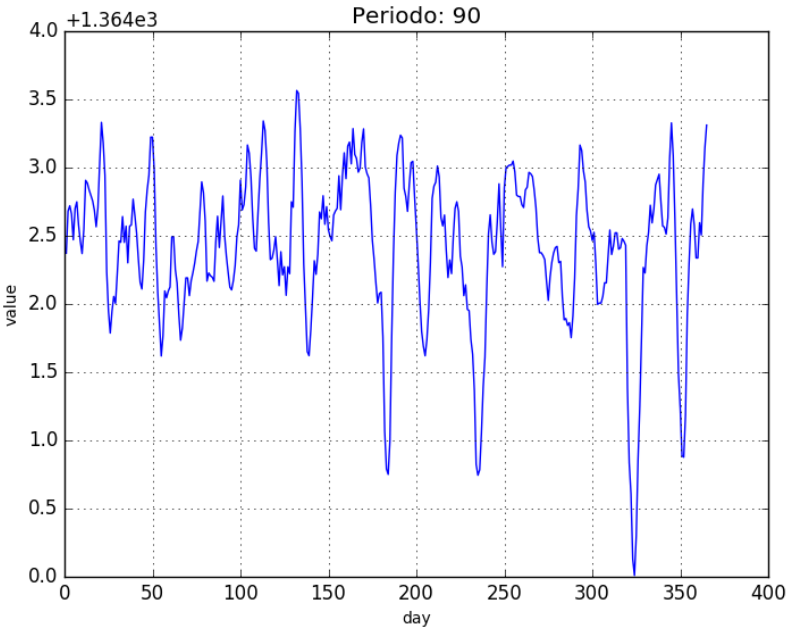


(a) Série Temporal 1989

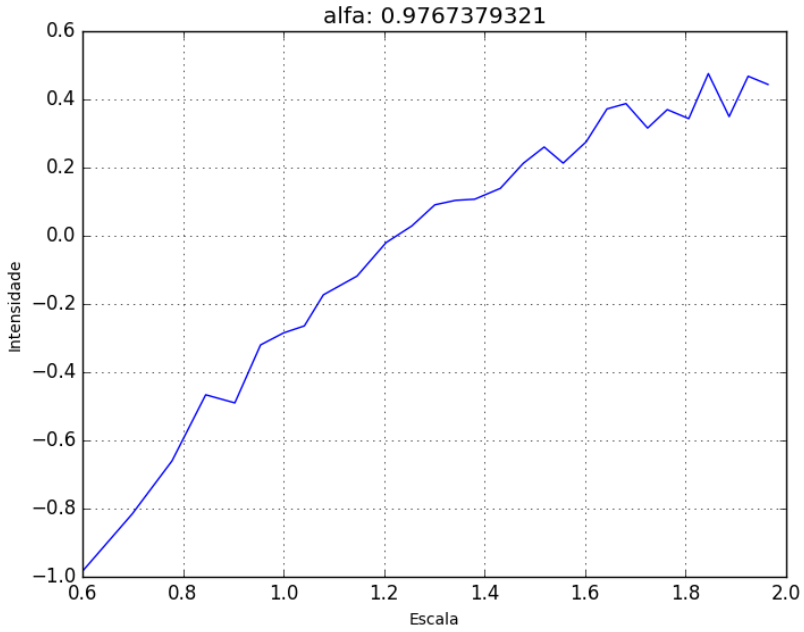


(b) DFA 1989

Figura C.14 - Série Temporal TSI - Janela 1990.

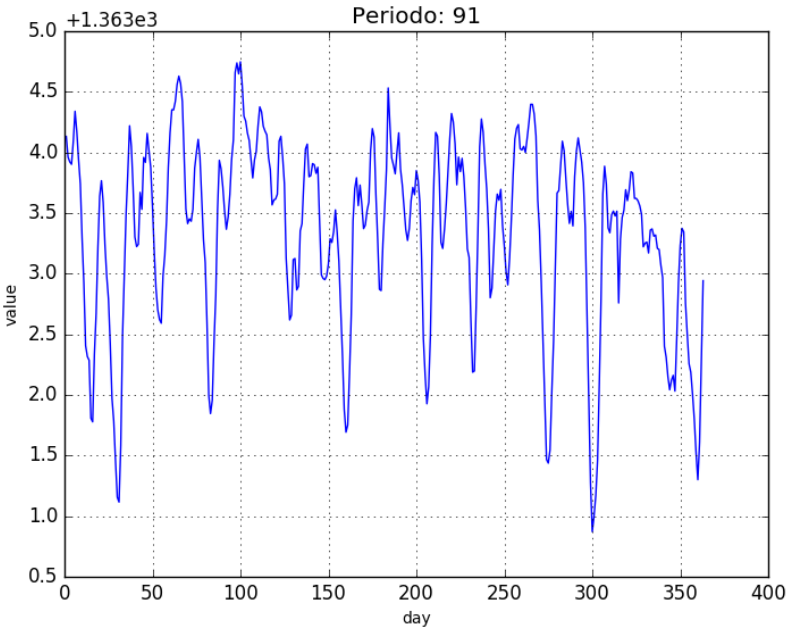


(a) Série Temporal 1990

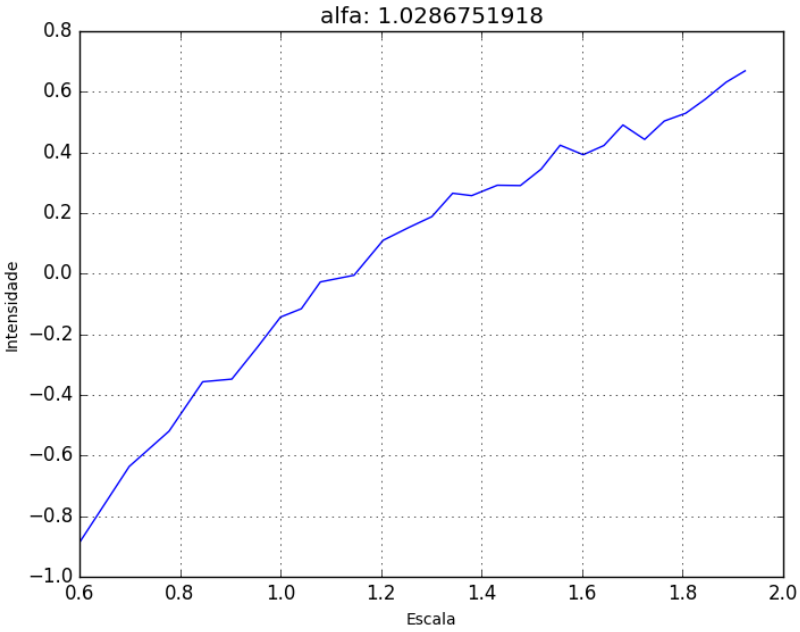


(b) DFA 1990

Figura C.15 - Série Temporal TSI - Janela 1991.

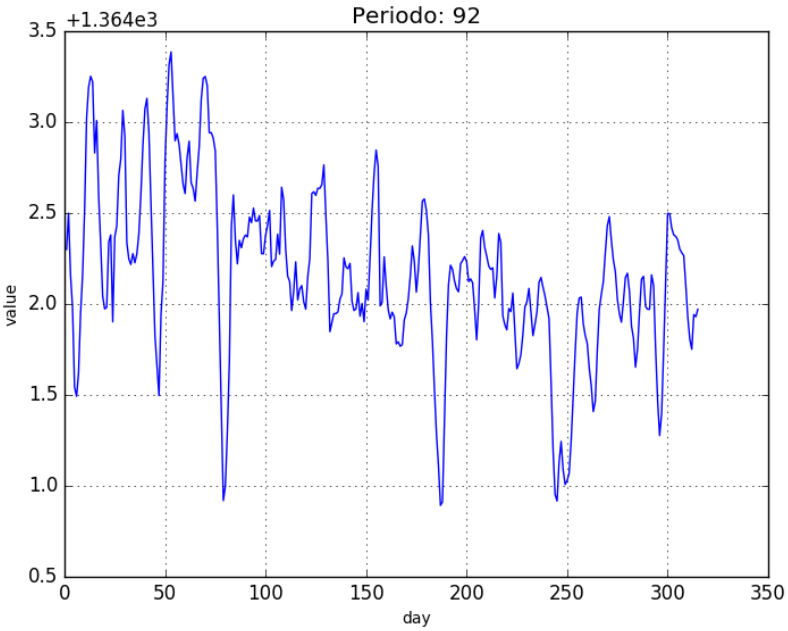


(a) Série Temporal 1991

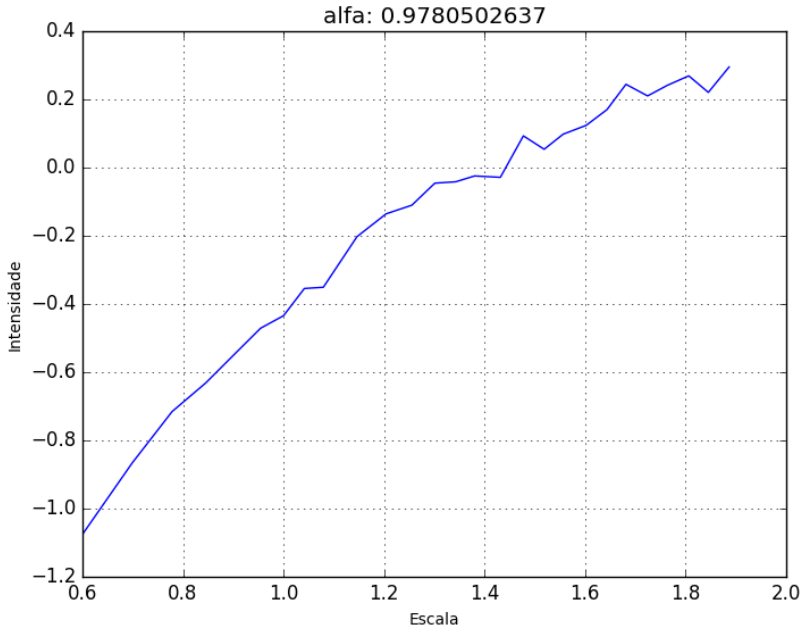


(b) DFA 1991

Figura C.16 - Série Temporal TSI - Janela 1992.

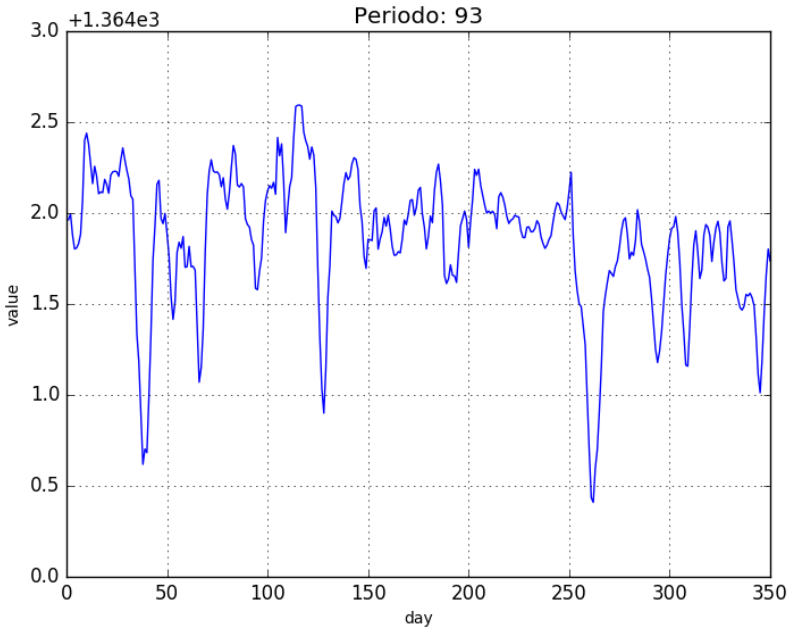


(a) Série Temporal 1992

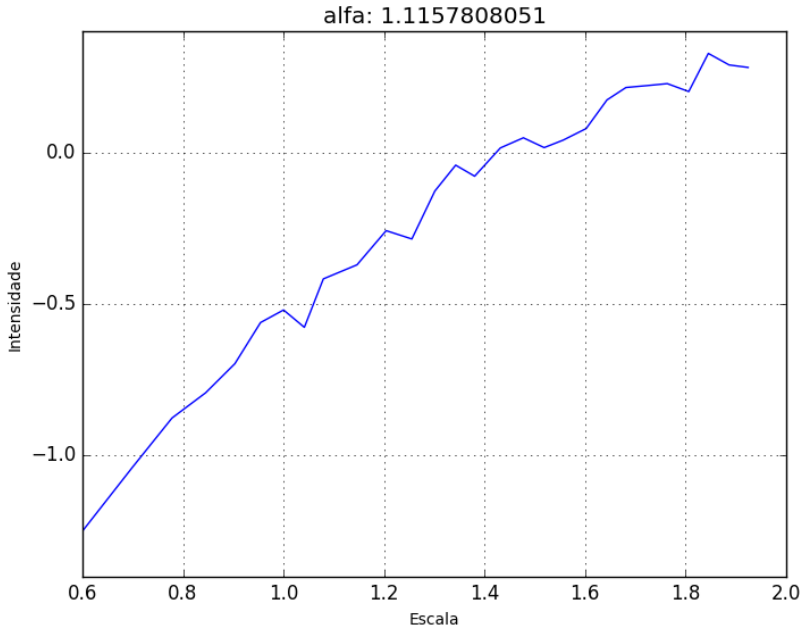


(b) DFA 1992

Figura C.17 - Série Temporal TSI - Janela 1993.

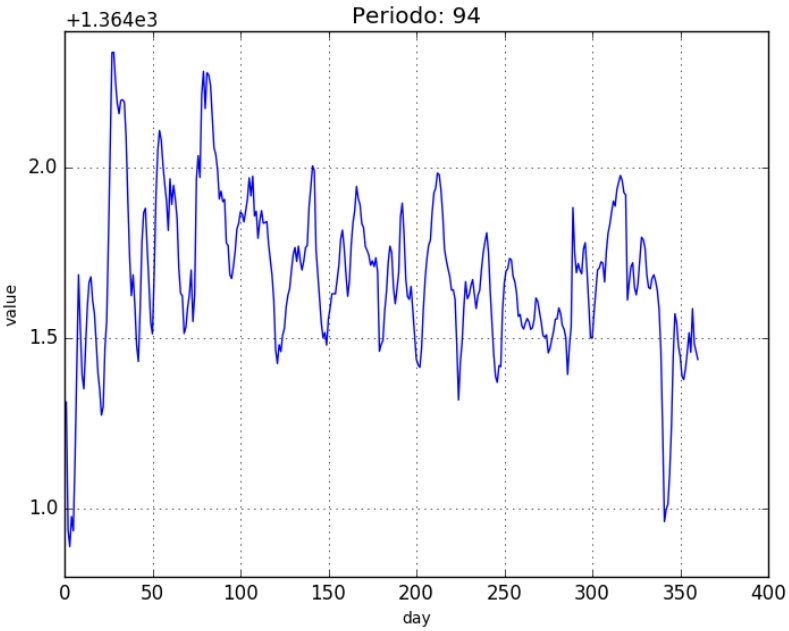


(a) Série Temporal 1993

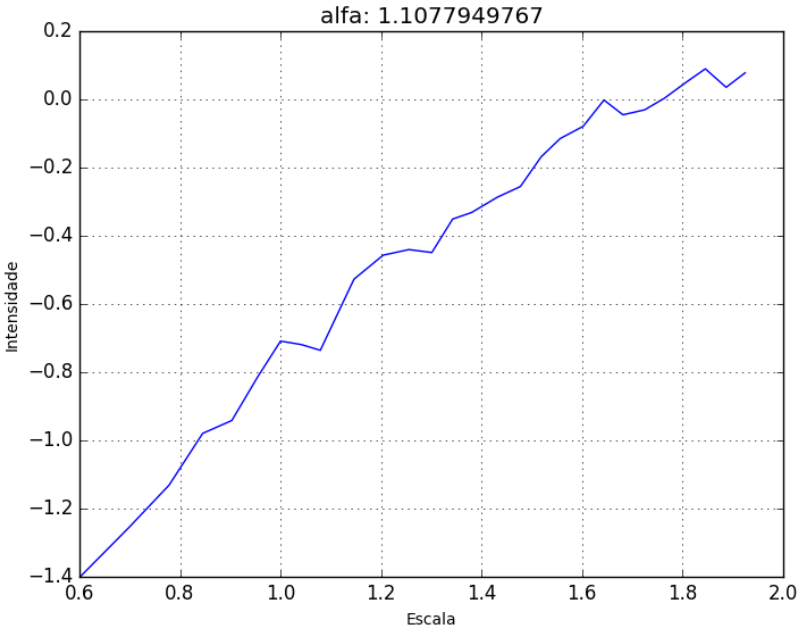


(b) DFA 1993

Figura C.18 - Série Temporal TSI - Janela 1994.

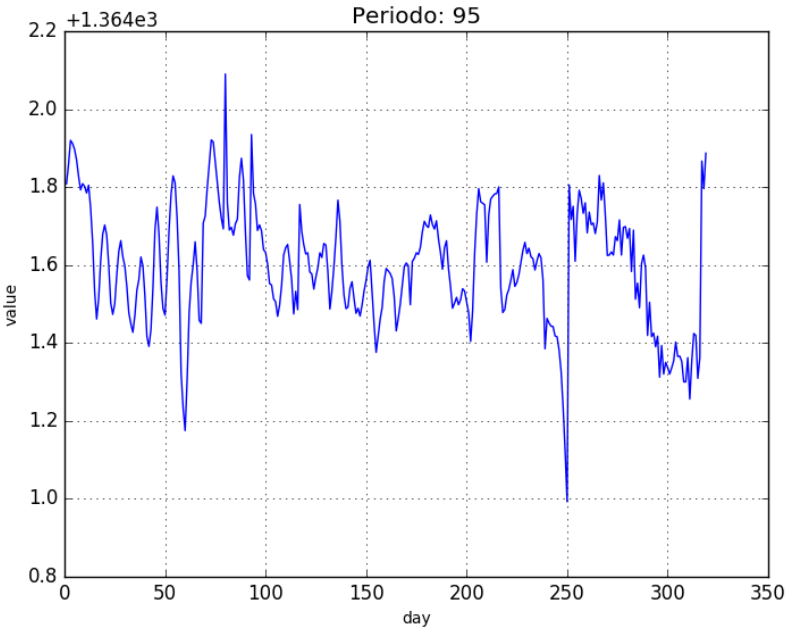


(a) Série Temporal 1994

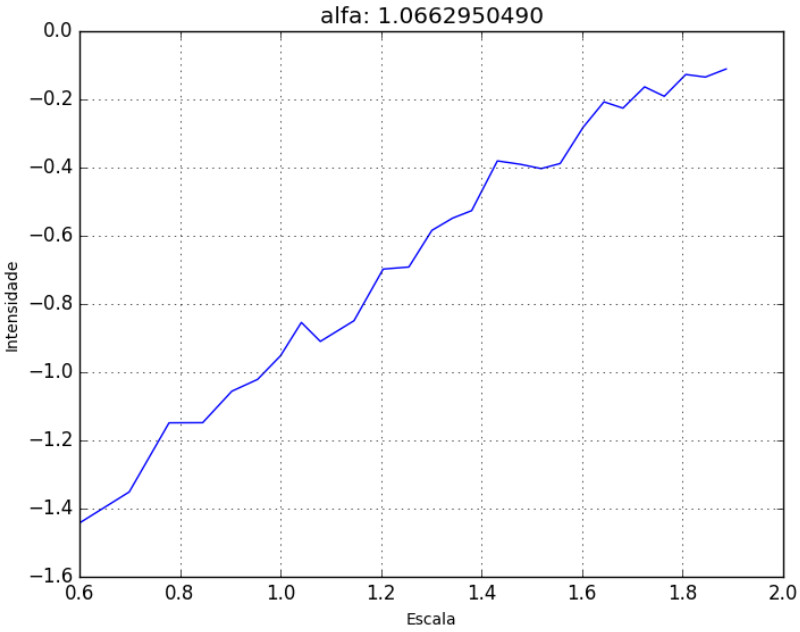


(b) DFA 1994

Figura C.19 - Série Temporal TSI - Janela 1995.

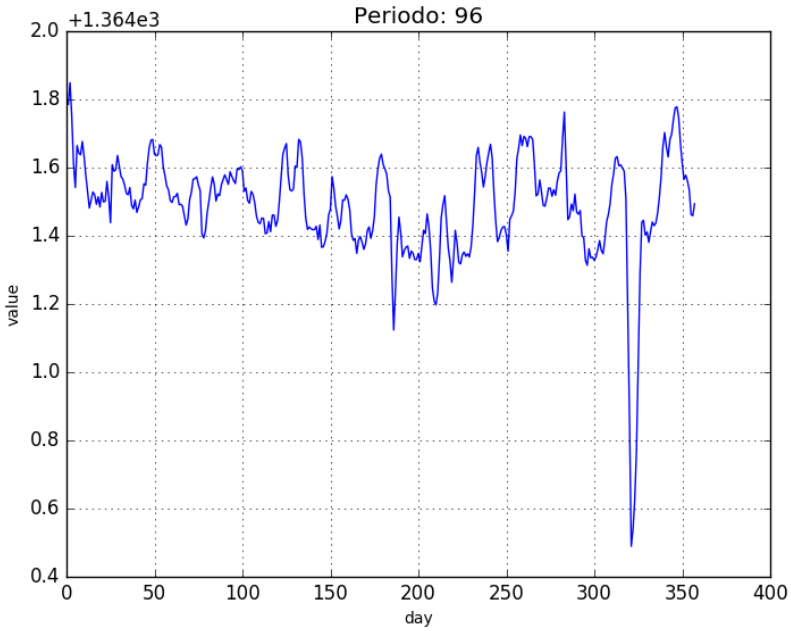


(a) Série Temporal 1995

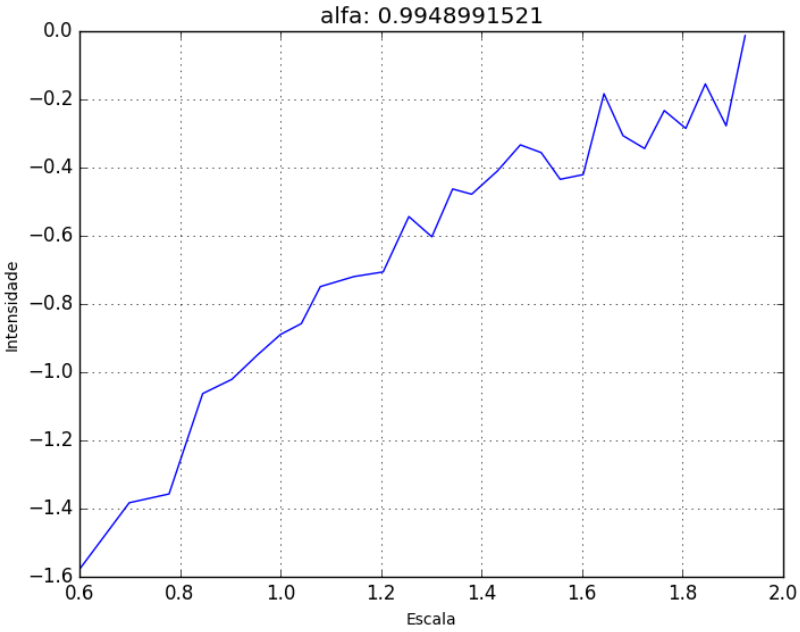


(b) DFA 1995

Figura C.20 - Série Temporal TSI - Janela 1996.

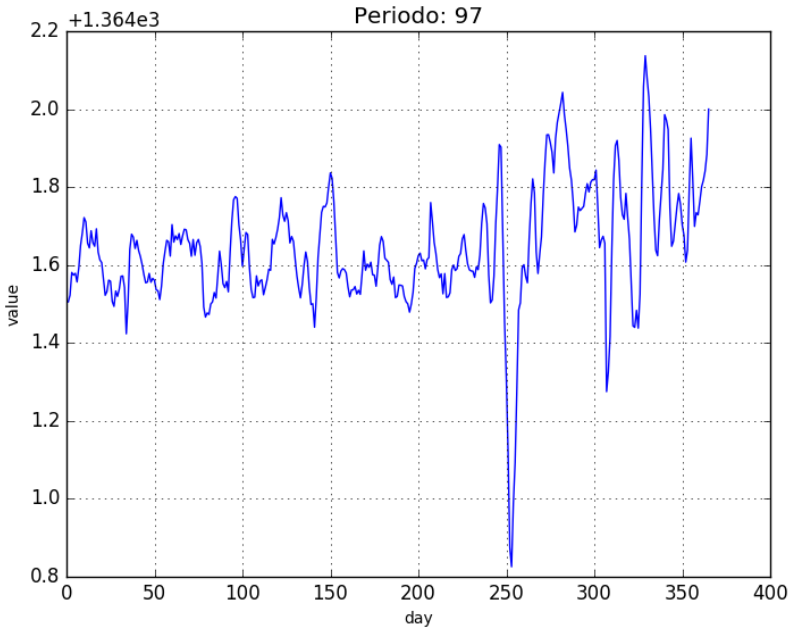


(a) Série Temporal 1996

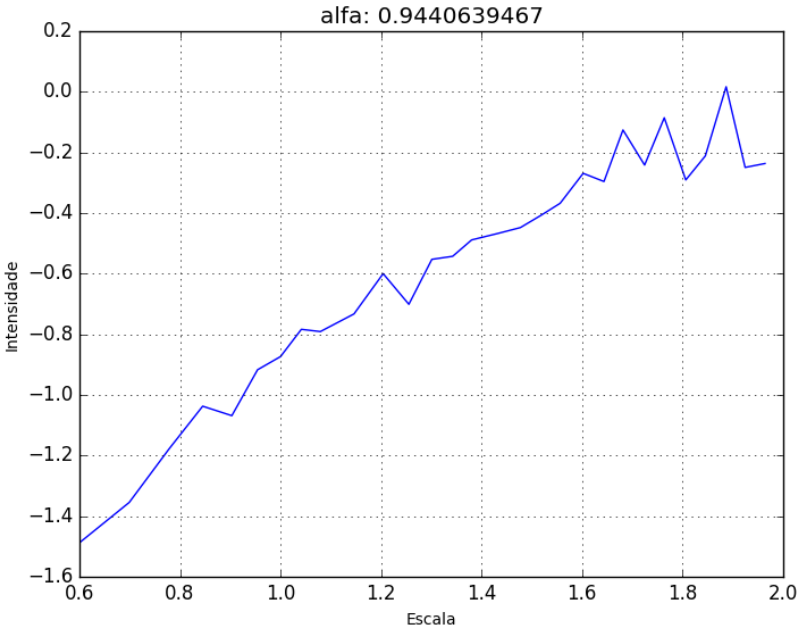


(b) DFA 1996

Figura C.21 - Série Temporal TSI - Janela 1997.

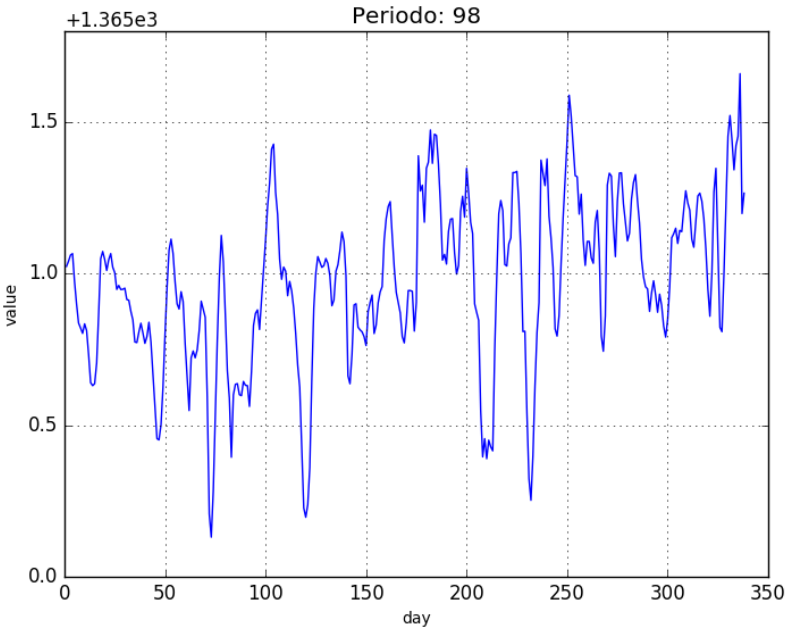


(a) Série Temporal 1997

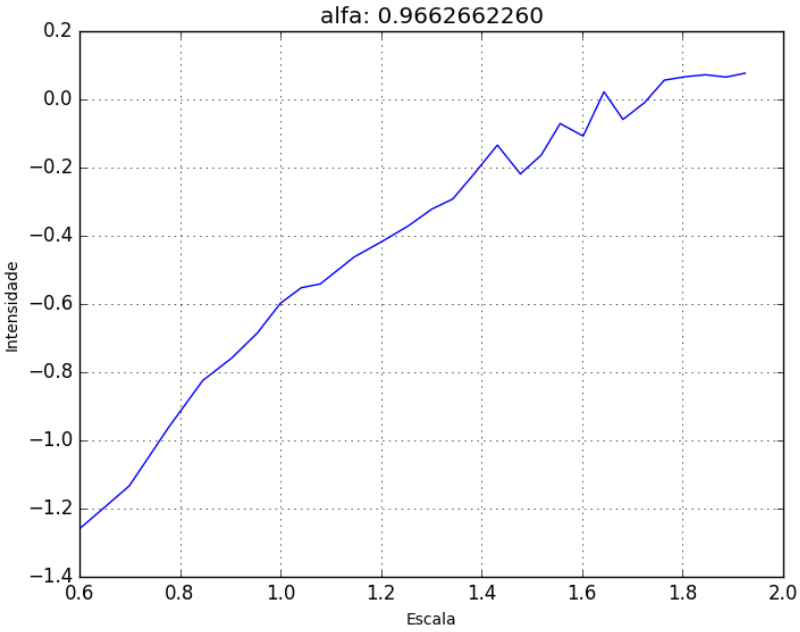


(b) DFA 1997

Figura C.22 - Série Temporal TSI - Janela 1998.

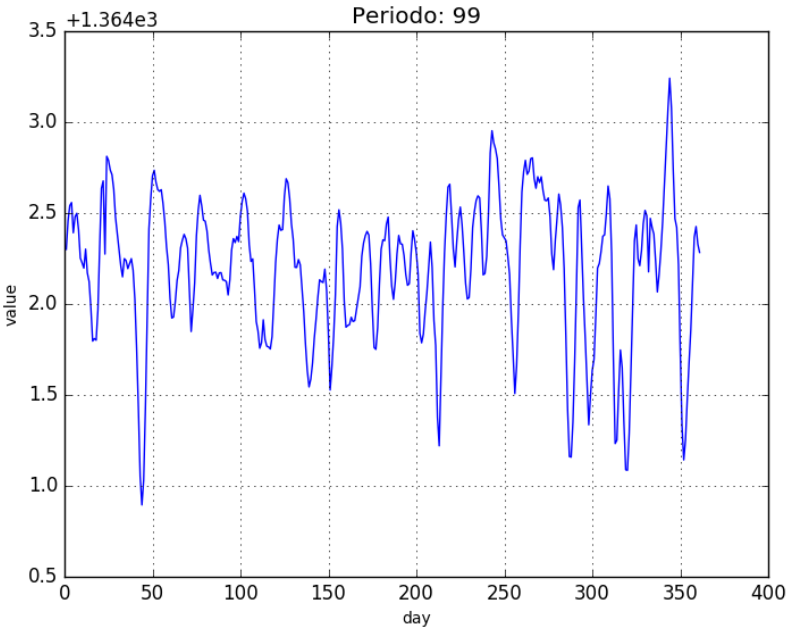


(a) Série Temporal 1998

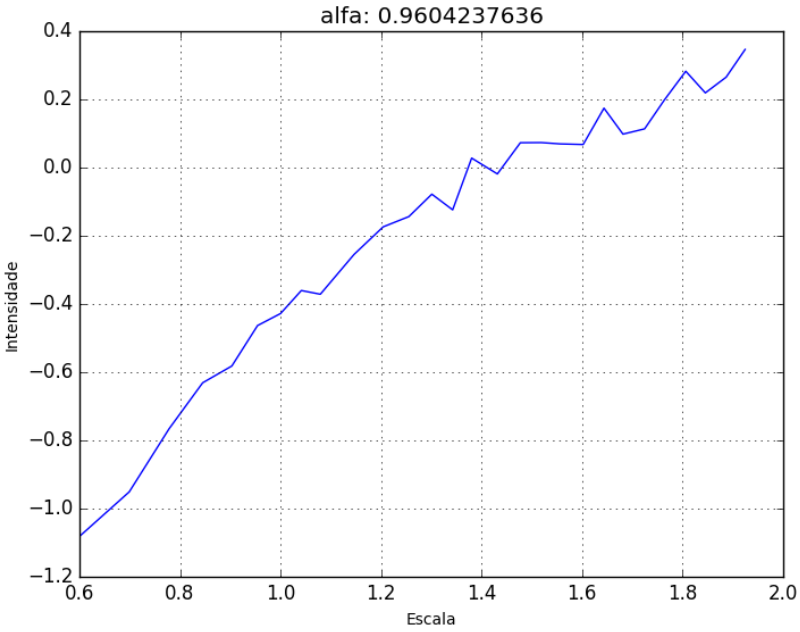


(b) DFA 1998

Figura C.23 - Série Temporal TSI - Janela 1999.

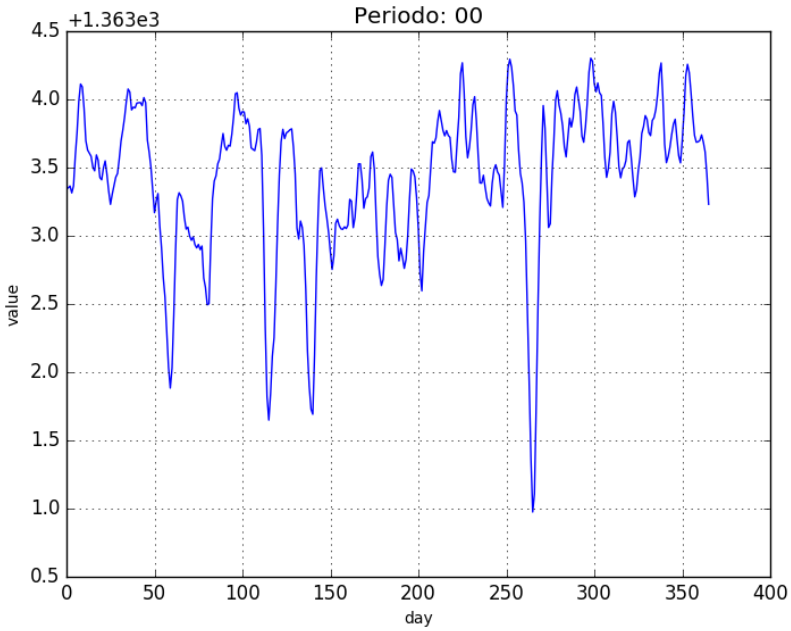


(a) Série Temporal 1999

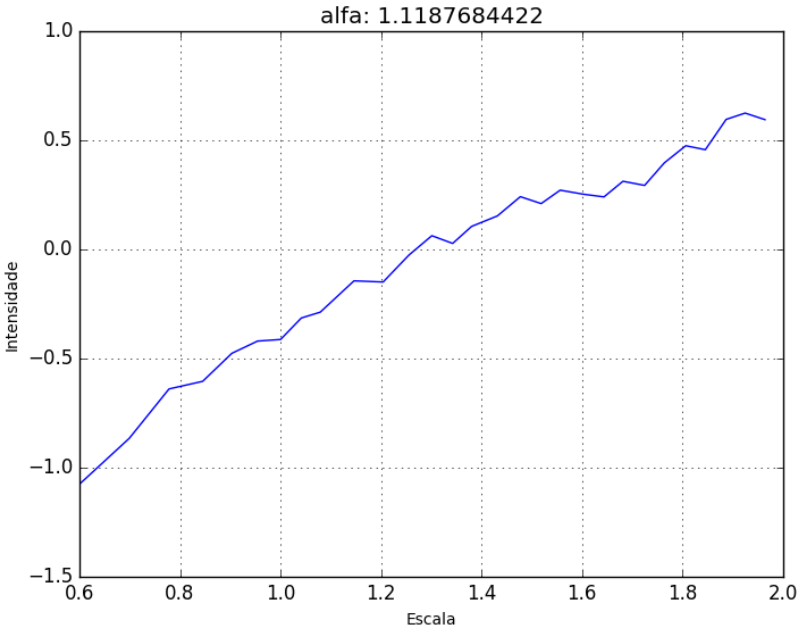


(b) DFA 1999

Figura C.24 - Série Temporal TSI - Janela 2000.

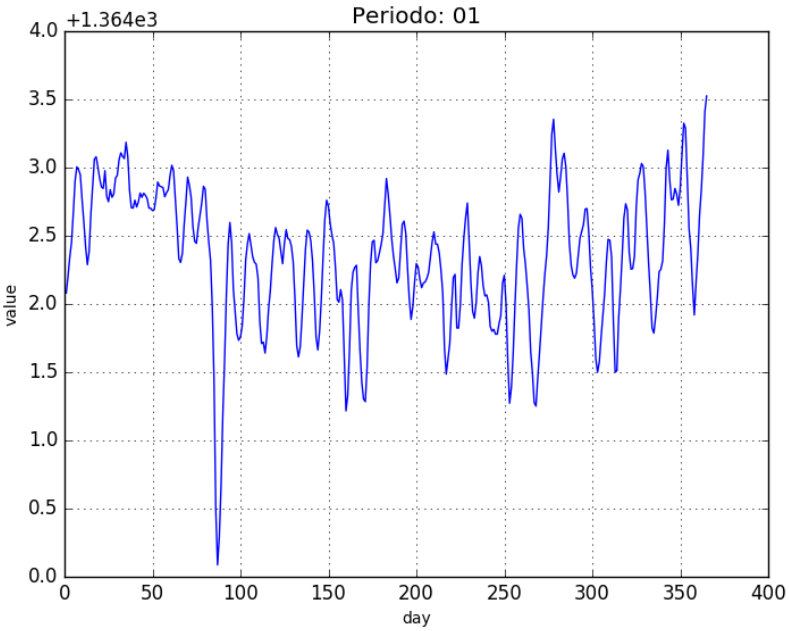


(a) Série Temporal 2000

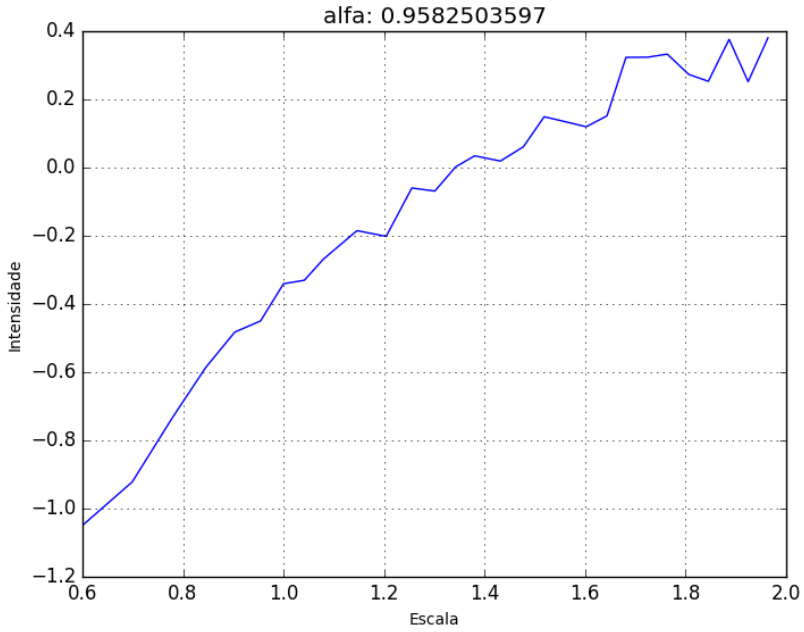


(b) DFA 2000

Figura C.25 - Série Temporal TSI - Janela 2001.

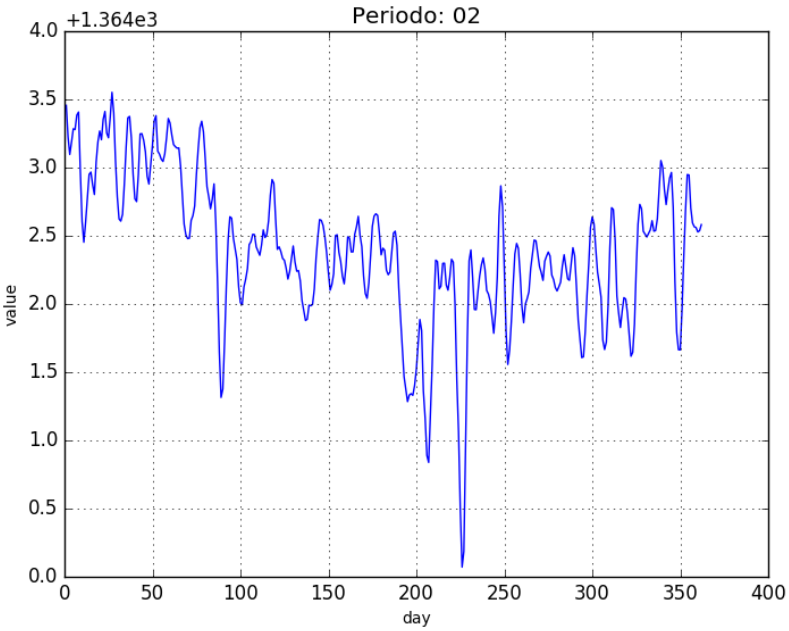


(a) Série Temporal 2001

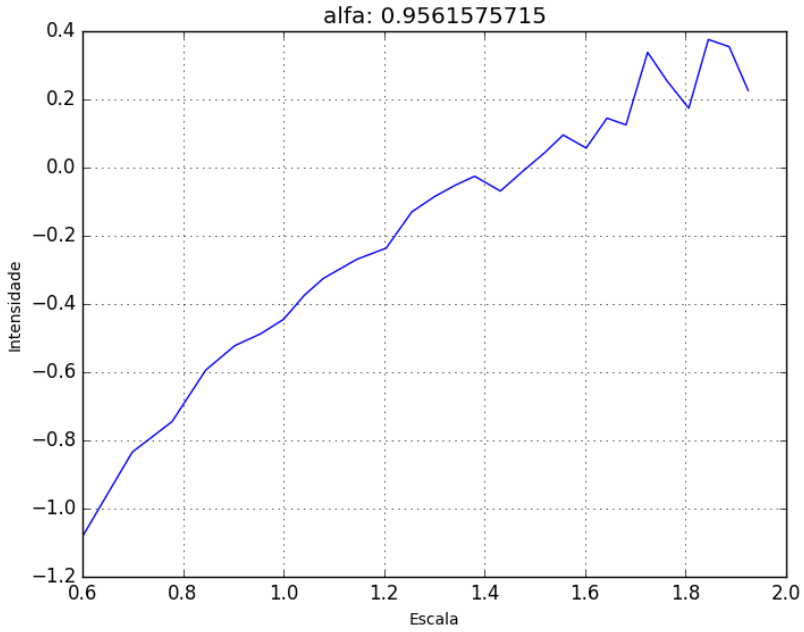


(b) DFA 2001

Figura C.26 - Série Temporal TSI - Janela 2002.

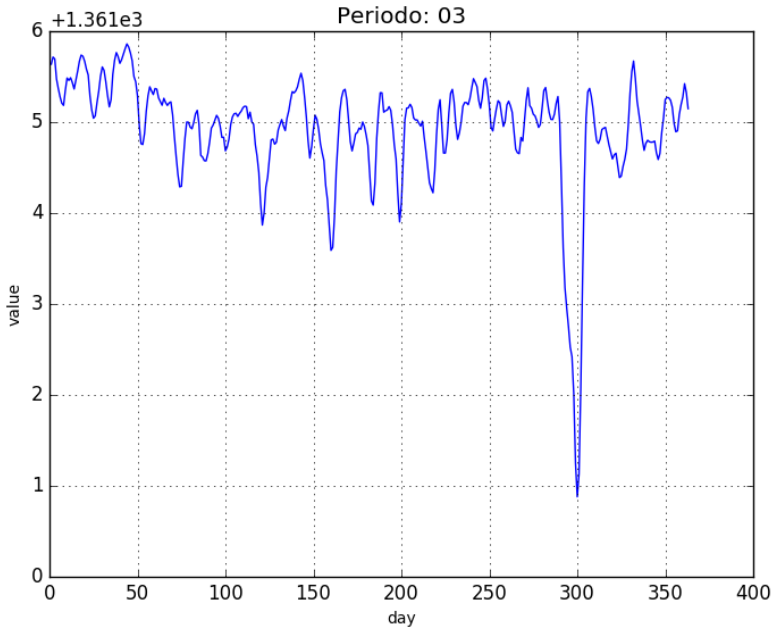


(a) Série Temporal 2002

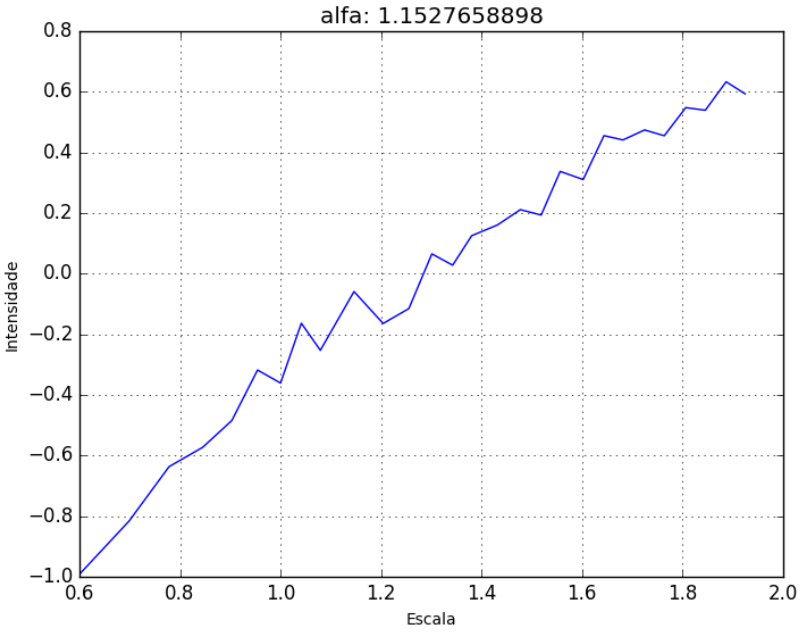


(b) DFA 2002

Figura C.27 - Série Temporal TSI - Janela 2003.

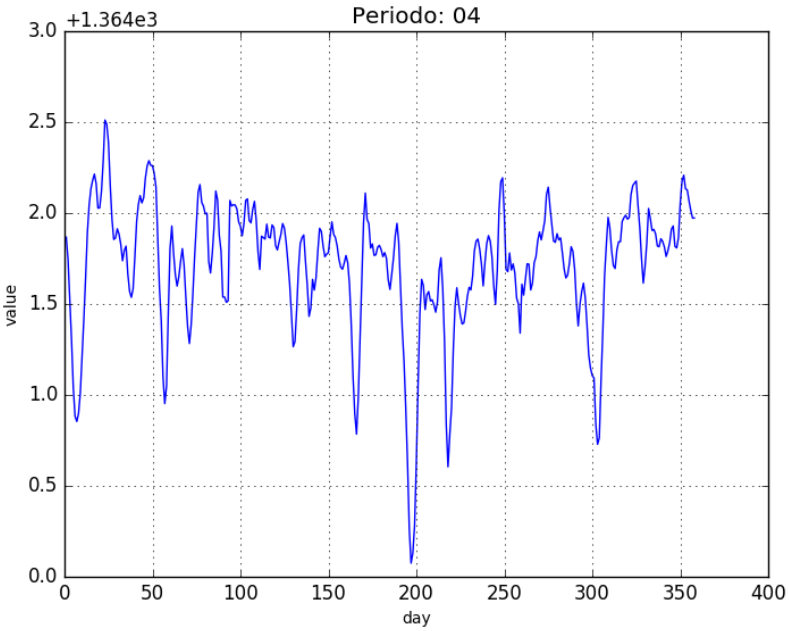


(a) Série Temporal 2003

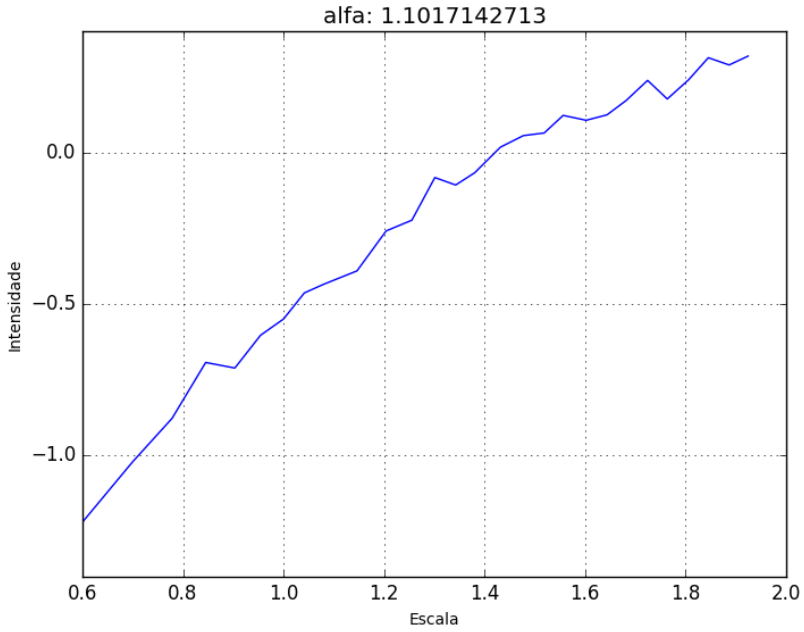


(b) DFA 2003

Figura C.28 - Série Temporal TSI - Janela 2004.

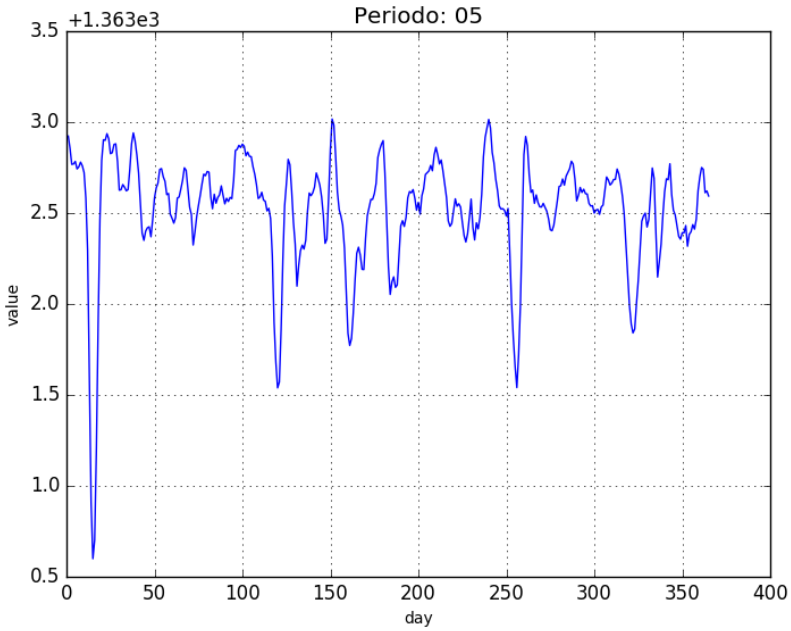


(a) Série Temporal 2004

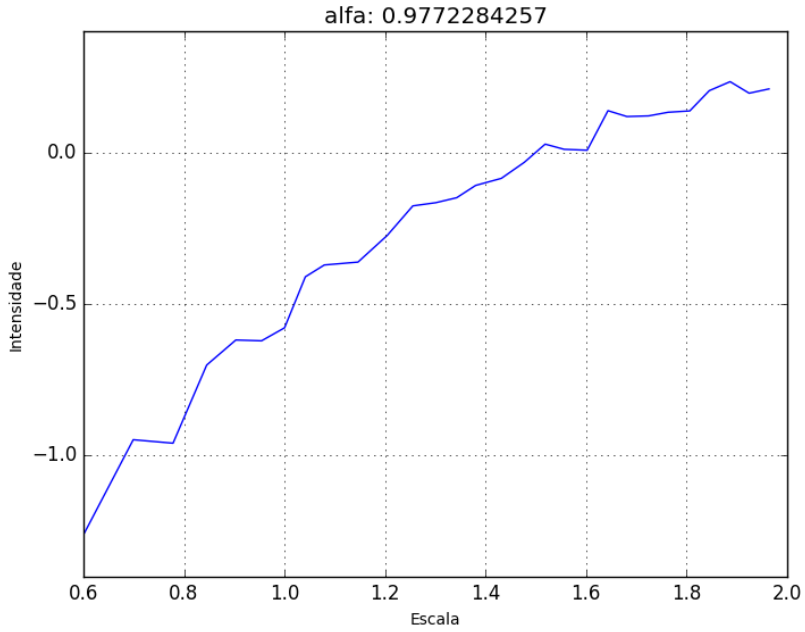


(b) DFA 2004

Figura C.29 - Série Temporal TSI - Janela 2005.

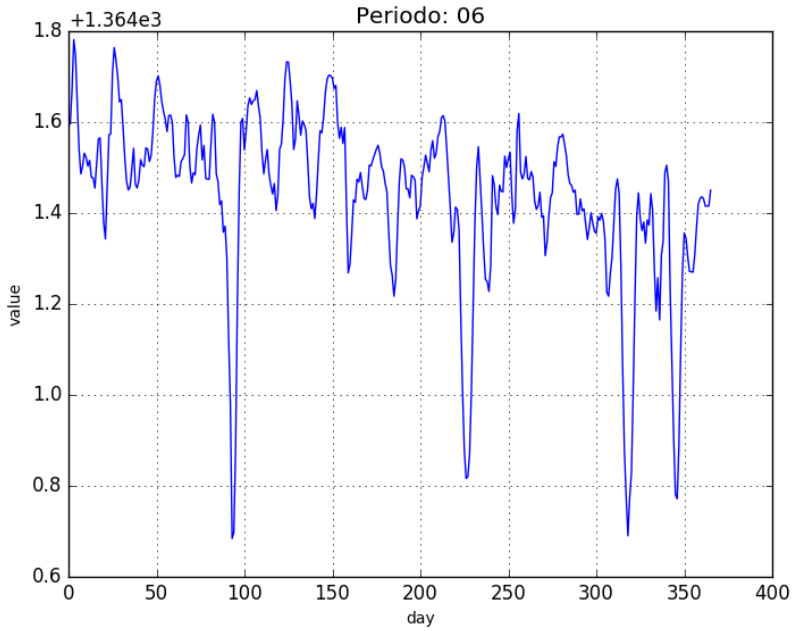


(a) Série Temporal 2005

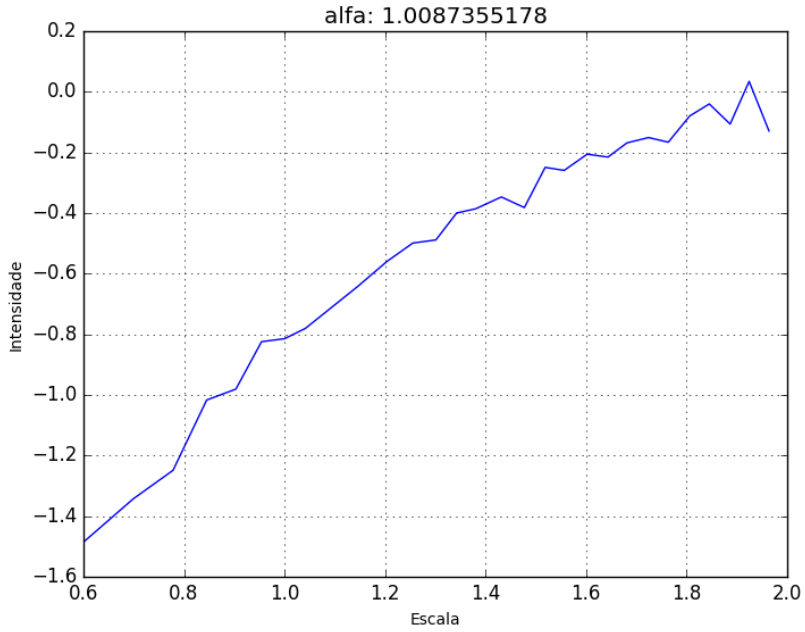


(b) DFA 2005

Figura C.30 - Série Temporal TSI - Janela 2006.

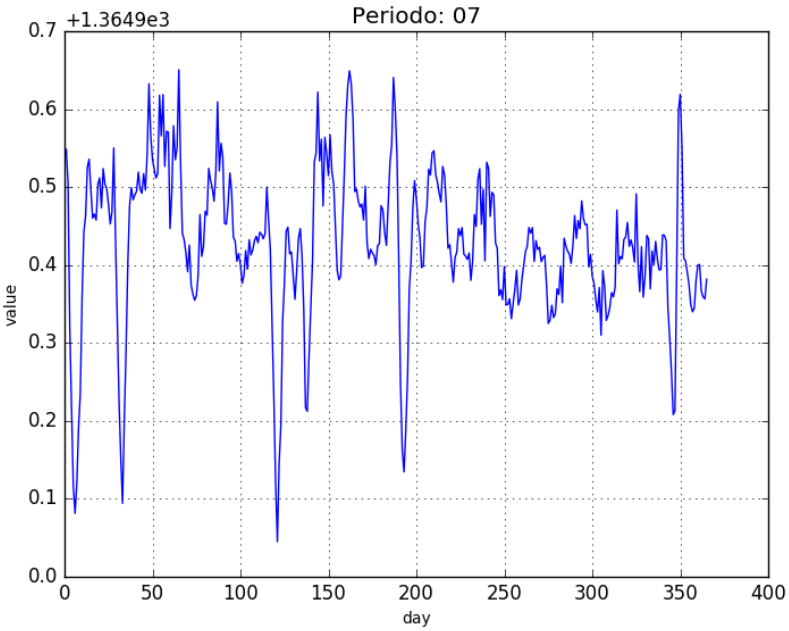


(a) Série Temporal 2006

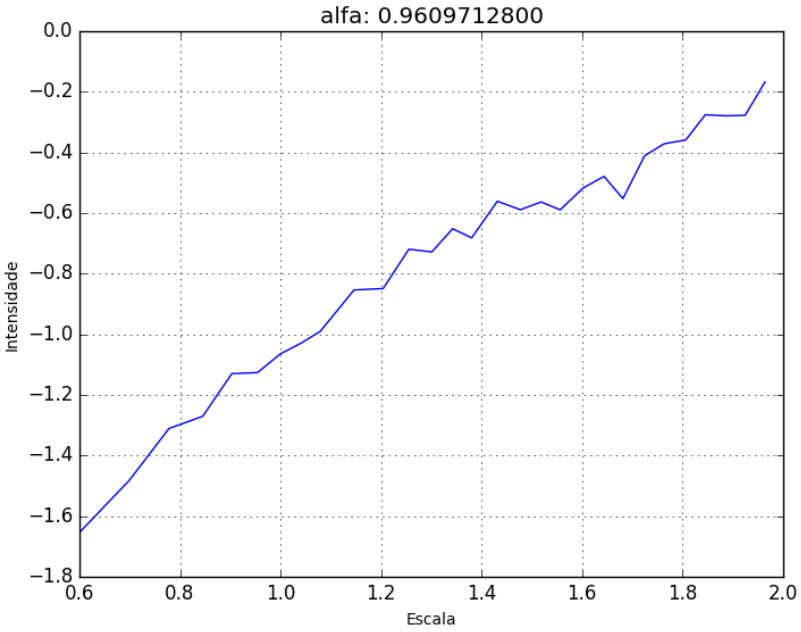


(b) DFA 2006

Figura C.31 - Série Temporal TSI - Janela 2007.

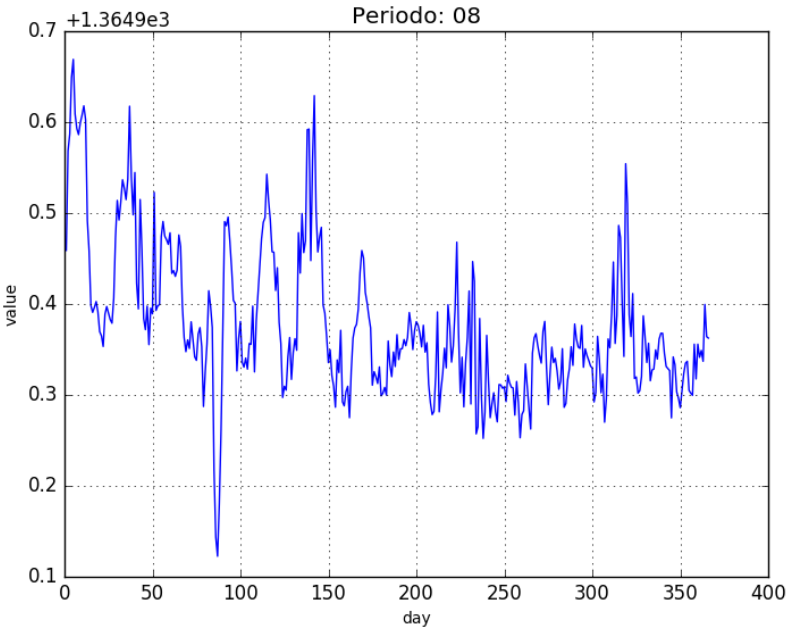


(a) Série Temporal 2007

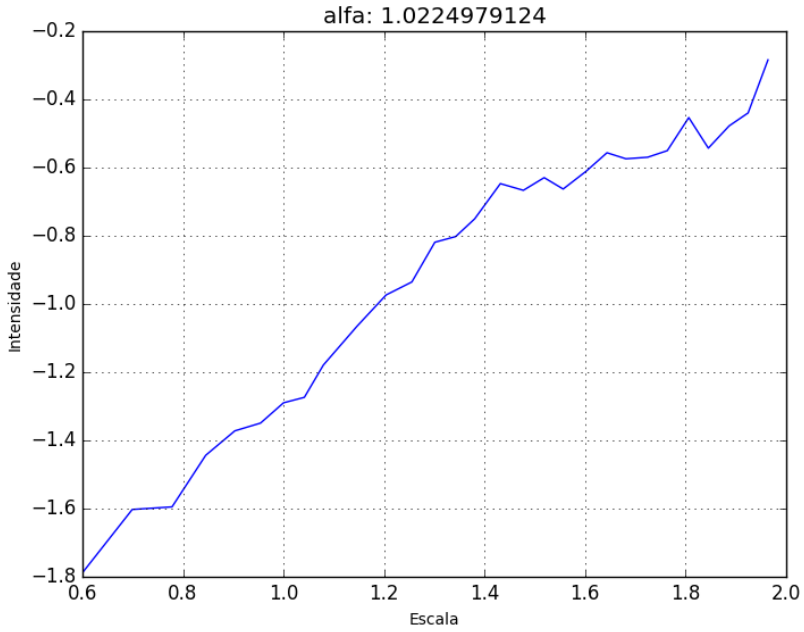


(b) DFA 2007

Figura C.32 - Série Temporal TSI - Janela 2008.

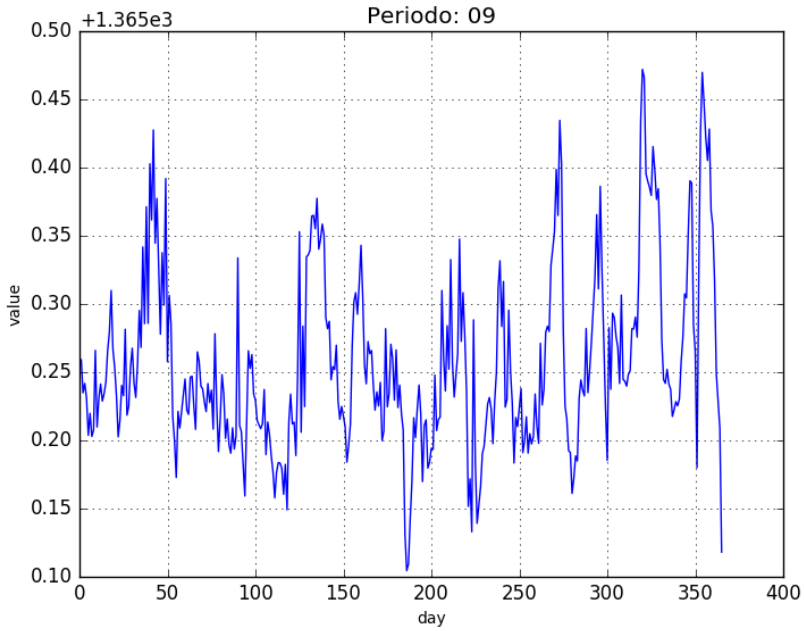


(a) Série Temporal 2008

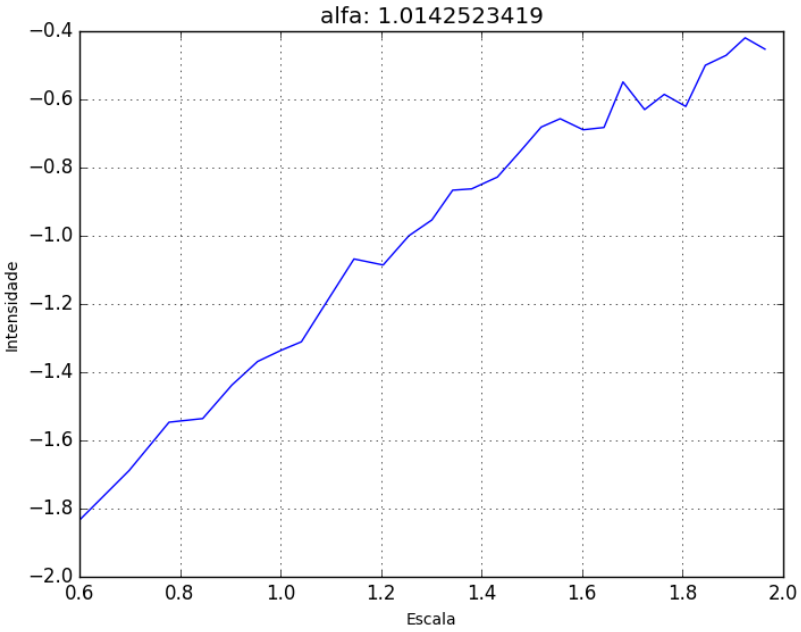


(b) DFA 2008

Figura C.33 - Série Temporal TSI - Janela 2009.

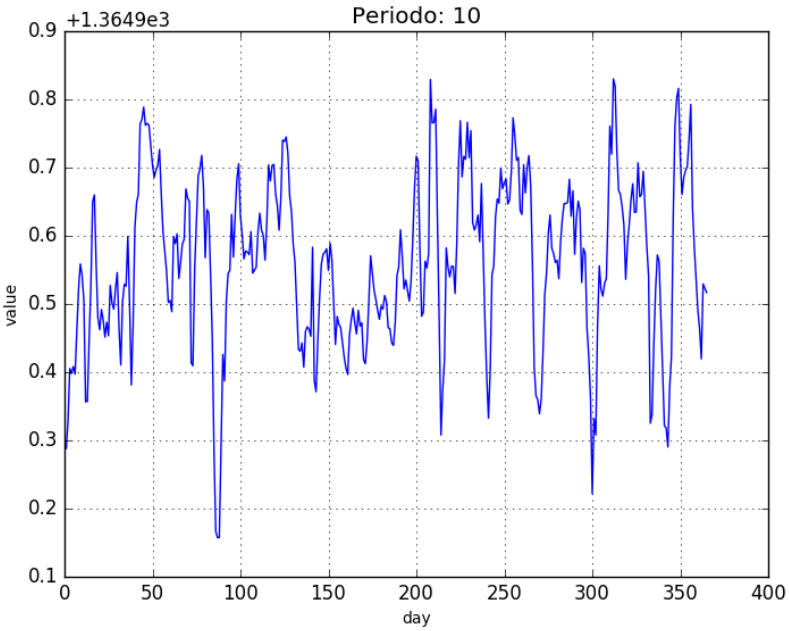


(a) Série Temporal 2009

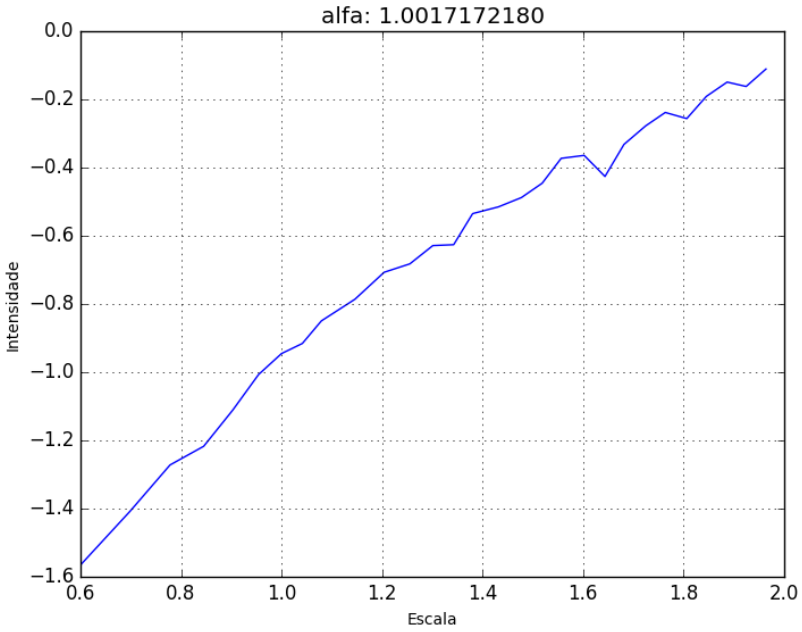


(b) DFA 2009

Figura C.34 - Série Temporal TSI - Janela 2010.

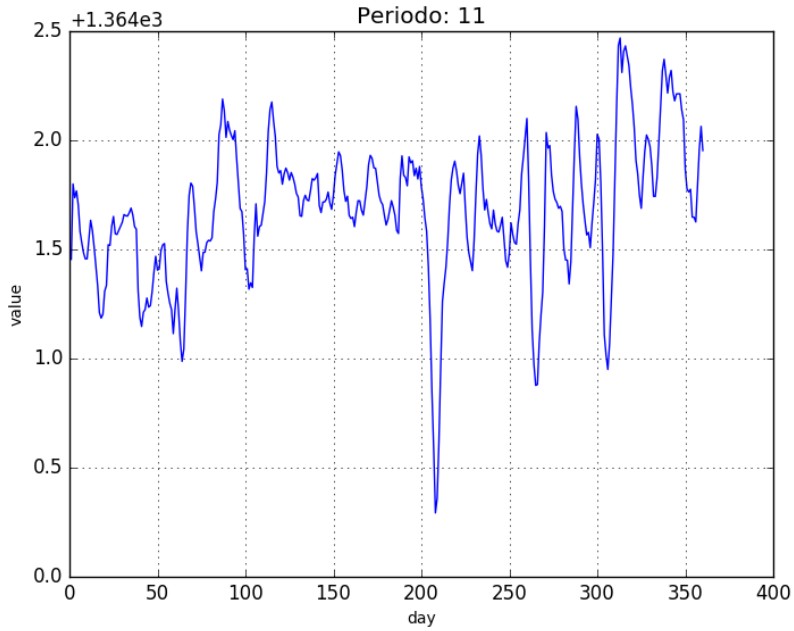


(a) Série Temporal 2010

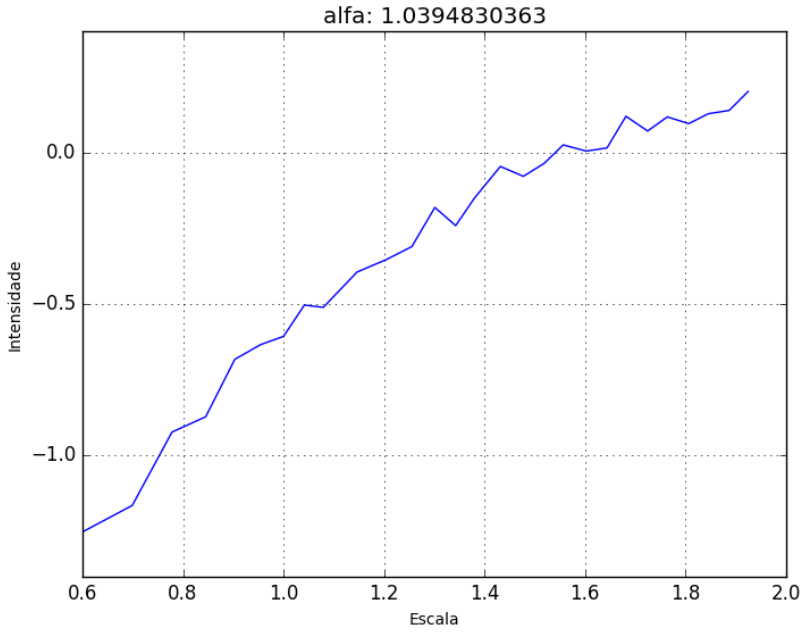


(b) DFA 2010

Figura C.35 - Série Temporal TSI - Janela 2011.

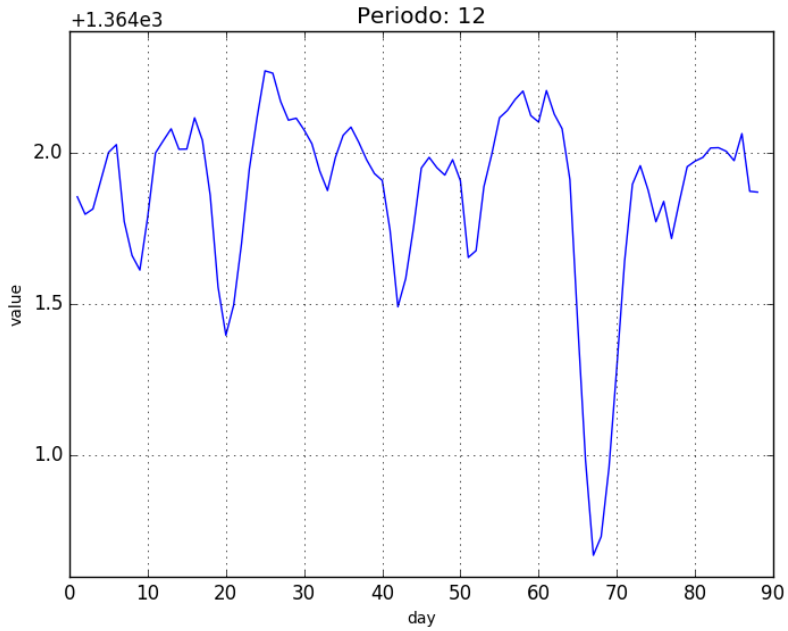


(a) Série Temporal 2011

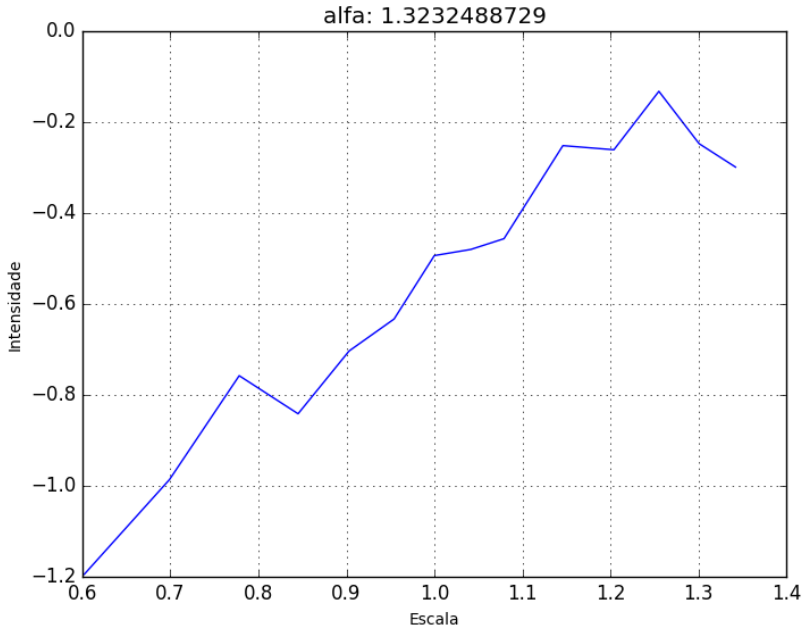


(b) DFA 2011

Figura C.36 - Série Temporal TSI - Janela 2012.



(a) Série Temporal 2012



(b) DFA 2012

APÊNDICE D - ARTIGO NSC

Este trabalho, desenvolvido durante a pesquisa de Doutorado em Computação Aplicada, foi apresentado durante o evento ***6th International Conference on Nonlinear Science and Complexity*** que ocorreu no período de 16-20 de maio de 2016, em São José dos Campos - SP, no Instituto Nacional de Pesquisas Espaciais - INPE, na Sessão de Pôster II (P02-05).

doi:10.20906/CPS/NSC2016-0023

https://ssl4799.websiteseguro.com/swge5/PROCEEDINGS/

INPE – National Institute for Space Research
São José dos Campos – SP – Brazil – May 16-20, 2016

A WEB FRAMEWORK FOR ADVANCED AND INTENSIVE NONLINEAR TIME SERIES ANALYSIS

Bruno B. F. Leonor[1], *Walter A. dos Santos*[2], *Asiel B. Jr*[3], *Reinaldo R. Rosa*[4]

[1]National Institute for Space Research - INPE-LAC-CAP, São José dos Campos, Brazil, brunobfl@yahoo.com.br
[2]National Institute for Space Research - INPE-DEA-ETE, São José dos Campos, Brazil, walter.abrahao@inpe.br
[3]National Institute for Space Research - INPE-CPTEC, São José dos Campos, Brazil, asiel.bomfin@cptec.inpe.br
[4]National Institute for Space Research - INPE-LAC, São José dos Campos, Brazil, rrrosa.inpe@gmail.com

Abstract: Using the e-Science paradigm, computers play a key role in supporting scientific areas such as those using nonlinear time series analysis to understand seemingly unpredictable behavior. This work presents a web framework supporting researchers in this class of analysis in a cloud environment where they can access, parameterize, initialize and monitor their applications transparently. Moreover, new applications can be added and managed by researchers on the portal easily. As a case study, techniques such as detrended fluctuation analysis (DFA) and gradient pattern analysis (GPA) will be added and used for intensive time series analysis in the Brazilian Space Weather Program.

keywords: Time Series Analysis, Nonlinear Dynamics and Complex Systems, Chaos and Global Nonlinear Dynamics, Web Framework, High-Performance Computing.

1. INTRODUCTION

The web has been causing a growing impact on scientific research, technological development and society as a whole with its growth over the past few years and has brought a different style of computing [1].

Tony Hey, vice president of Microsoft Research during the Latin American e-Science Workshop in 2013, said: ”Some technologies used in computer science will help solve scientific problems. In contrast, the use of these tools to solve scientific problems also make possible the very development of computer science” [2].

This relationship with computing various scientific areas are being benefited with the use of computing environments increasingly modern. This has contributed to the creation of a powerful compound ecosystem of humans and machines that together are able to solve problems that could not be solved alone.

This paper presents an e-Science environment named Sentinel to support scientists in their research in a cloud environment where they can add, parameterize, initialize and monitor their applications transparently using any enabled device to access the web.

As a case study, techniques such as detrended fluctuation analysis (DFA) and gradient pattern analysis (GPA) will be added and used for intensive time series analysis in the Brazilian Space Weather Program (EMBRACE).

2. E-SCIENCE

The term e-Science is commonly used to describe the development of software services infrastructure [3]. This enables access to remote facilities, distributed computing resources, information storage in dedicated databases, dissemination and sharing of data, results and knowledge.

Research in e-Science requires joint efforts and multidisciplinary, where computer scientists assist researchers from other fields of knowledge to develop research more quickly and efficiently and that this partnership, the computer is also made of innovative and transformative way [4].

The research in e-Science can be described by the fol-

A Web Framework for Advanced and Intensive Nonlinear Time Series Analysis

Bruno B. F. Leonor, Walter A. dos Santos, Asiel B. Jr., Reinaldo R. Rosa

National Institute for Space Research, S. J. dos Campos, Brazil
brunobfl@gmail.com, walter.abrahao@inpe.br, asiel.bomfin@cptec.inpe.br, rrrosa.inpe@gmail.com

Abstract

Using the e-Science paradigm, computers play a key role in supporting scientific areas such as those using nonlinear time series analysis to understand seemingly unpredictable behavior. This work presents a web framework supporting researchers in this class of analysis in a cloud environment where they can access, parameterize, initialize and monitor their applications for complex systems analysis. Moreover, new applications, taking into account intensive data analysis from big data files, can be added and managed by researchers on the portal easily. As a case study, techniques such as detrended fluctuation analysis (DFA) and gradient pattern analysis (GPA) will be added and used for intensive time series analysis in the Brazilian Space Weather Program.

Introduction

The web has been causing a growing impact on scientific research, technological development and society as a whole with its growth over the past few years and has brought a different style of computing mainly into the big data paradigm [1, 2].

This paper presents an e-Science environment named **Sentinel** to support scientists in their research in a cloud environment using any enabled device to access the web. As a case study, techniques such as detrended fluctuation analysis (DFA) and gradient pattern analysis (GPA) will be added and used for intensive time series analysis in the Brazilian Space Weather Program (EMBRACE) [3] (see Figure 1).

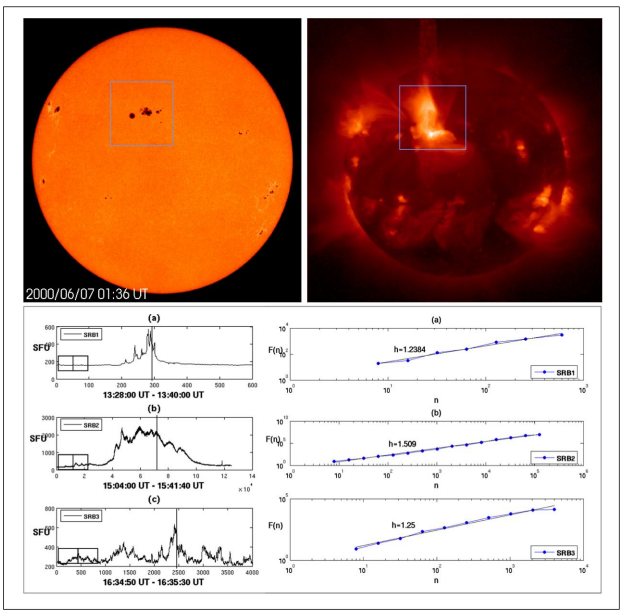


Figure 1: An example of a typical solar flare observed in multiple frequences (H-alpha (Hα), Soft X-ray and Radio(3 GHz)) due to a nonlinear plasma process in the solar atmosphere. The evolution of this complex dynamics can be characterized by means of the scaling exponents from DFA analysis.

e-Science

The term e-Science is commonly used to describe the development of software services infrastructure [4]. This enables access to remote facilities, distributed computing resources, information storage in dedicated databases, dissemination and sharing of data, results and knowledge.

Research in e-Science requires joint efforts and multidisciplinary and can be described by the following characteristics [5]: (1) involves collaboration of researchers computing with researchers from other fields of knowledge, and (2) requires the creation of sophisticated computational methods to handle large volumes of data (Big Data) and/or to perform simulations and programs that require advanced computational solutions for complex systems analysis.

Sentinel

The environment proposed in this paper is named **Sentinel** and it is an e-Science environment that aims to provide high performance computing resources to scientists to develop their research. Scientists affiliated to the portal can register their applications and manage them, more specifically, they can access, parameterize, initialize, monitor, analyze and share the results obtained through any device that allows access to the web.

There are two ways to access the portal, as a researcher or guest. The main features for a researcher are shown in Figure 2. Guests can only access the portal to view the results of the experiment and download it when shared by the researcher owner.

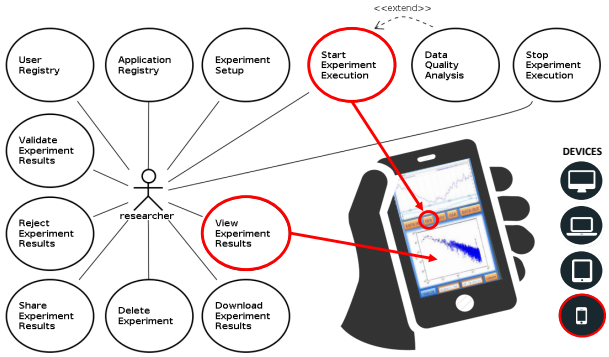


Figure 2: Use case diagram showing the functions provided to the researchers by the Sentinel Portal highlighting the functions Start Experiment Execution and View Experiment Results that can be used through any device that allows access to the web, showing the use through a smartphone.

The researcher has the option to check your input data before starting an experiment through a Data Quality Analysis tool and each application can have its own configuration for analysis.

Through the Dashboard Interface (Figure 3), researchers will have access a set of informations about each experiment, which are: (1) PID - experiment identifier, (2) Application - chosen application, (3) ComputingElement - which node is running, (4) CreationDate - creation date of the experiment, (5) SubmissionDate - request date for execution, (6) StartExectution - initial date of execution, (7) EndExecution - end of the execution date, (8) JobStatus - current situation of the experiment and (9) Actions - actions to manage it.

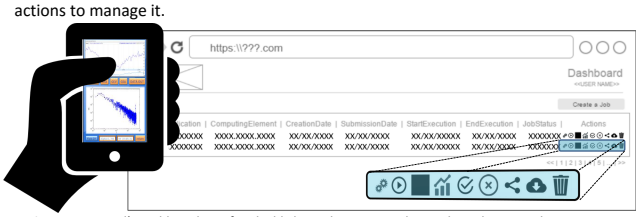


Figure 3: Sentinel's Dashboard interface highlighting the Actions column where the researcher can Setup, Start, Stop, View, Accept, Reject, Share, Download and Delete an experiment and its results.

Sentinel portal has too a tool named "Application Registry" that allows the inclusion of new applications, this way new techniques and algorithms may be incorporated into the environment by researchers to complement their studies as well as new visualization tools. To register a new application the researcher should provide some information about it, which enable the portal to create the mechanisms to allow the researcher to manage new experiments.

Case Study

As a case study, using the Application Registry tool, techniques such as DFA [6] and GPA [7] will be incorporated into the Sentinel portal and used for analysis of time series obtained the Brazilian Space Weather Program, named "EMBRACE".

The EMBRACE program was chosen as a case study for the knowledge and prediction of different phenomena that directly affect human activities, such as solar activity, magnetic storms and ionospheric storms, they have great importance for the well being of society.

Conclusion

This work introduces a technical conception for an innovative web framework named **Sentinel** dedicated and suitable for managing e-Science applications.

It is expected that this work will contribute to a more environmentally friendly and safe for e-Science users, where they can gather all the applications used in the development of their research.

The framework prototype has been applied for time series analysis using two techniques, DFA and GPA. Future work will incorporate more challenging experiments and algorithms.

References

[1] N. Shadbolt, W. Hall, J. A. Hendler, W. H. Dutton,"Web science: a new frontier", The Royal Society, vol. 371, n. 1987, 2013.

[2] J. P. Collins, "Sailing on an Ocean of 0s and 1s", Science Magazine, vol. 327, n. 5971, 2010.

[3] INPE, "O Programa Embrace", available at: http://www2.inpe.br/climaespacial/portal/oprograma-embrace/, accessed January 2016.

[4] D. Palazzi, L. Silva, L. F. Mendes, W. Gaspar, E. Matos, F. Campos, R. Braga, "Uso de ontologias em projetos de e-science", BDBComp, 2009.

[5] Fapesp, "Programa FAPESP de Pesquisa em eScience", available at: http://www.fapesp.br/publicacoes/2015/folder_escience.pdf, accessed December 2015.

[6] V. V. Morariu, L. Buimaga-Iarinca, C. Vamos, S. Soltuz, "Detrended Fluctuation Analysis of Autoregressive Processes", Cornell University Library, 2007.

[7] R. R. Rosa, M. Karlický, T. B. Veronese, N. L. Vijaykumar, H. S. Sawant, A. I. Borgazzi, M. S. Dantas, E. B. M. Barbosa, R. A. Sych, O. Mendes, "Gradient pattern analysis of short solar radio bursts", Advances in Space Research, vol. 42, 2008.

APÊNDICE E - ARTIGO CCIS

Este trabalho, desenvolvido durante a pesquisa de Doutorado em Computação Aplicada, foi apresentado durante o evento ***4th Conference of Computational Interdisciplinary Science*** que ocorreu no período de 07-10 de novembro de 2016, em São José dos Campos - SP, no Instituto Nacional de Pesquisas Espaciais - INPE, na Sessão "TS3b - *Data Science and e-Science*" (p32).

A High Performance Virtual Workspace for Management and Monitoring of Scientific Applications

Bruno B. F. Leonor[a 1], Walter A. dos Santos [a] and Reinaldo R. Rosa[a]
[a]National Institute for Space Research, São José dos Campos, SP, Brazil

Abstract

Several scientific areas benefits from the research methods empowered by computers in an e-Science paradigm. One of the major issues in this context regards to the underlying infrastructure interoperability. This work provides an e-Science environment named Sentinel to support scientists in their research in a cloud environment using any enabled device to access the web. The web portal allows parameterization, initialization and monitoring of experiments transparently from net infrastructure. As an illustration, techniques as such as Detrended Fluctuation Analysis (DFA) and Gradient Pattern Analysis (GPA) will be discussed in the context of a time series analysis application to understand seemingly unpredictable behaviors as extreme events in space physics. The EMBRACE (Brazilian Space Weather Program) program was chosen as a DFA case study for the knowledge and prediction of different phenomena that directly affect human activities, such as solar activity, magnetic storms and ionospheric storms, they have great importance for the well being of society. Furthermore, new applications can be uploaded to Sentinel easily empowering researchers to manage their scientific workflows. It is hoped that this work may contribute for a friendlier environment for e-Science users.

Keywords: Time Series Analysis, Web Portal, High-Performance Computing.

1. Introduction

The web has been causing a growing impact on scientific research, technological development and society as a whole with its growth over the past few years and has brought a different style of computing involving new concepts as: *cloud computing*, *e-Science*, *big data* and *data science* [1, 2].

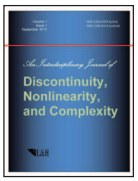
Many important dynamical phenomena in science can only be investigated using methods for analyzing the correspondent complex underlying process. Usually, such process are represented as high resolution data collection so that meaningful statistics and other major data characteristics. Due to the huge amount of heterogenoues scientific datasets produced daily, the sinergy among those areas are being benefited with the use of new advanced scientific computing environments [3].

[1]E-mail Corresponding Author: brunobfl@gmail.com

APÊNDICE F - ARTIGO REVISTA DNC

Este trabalho, desenvolvido durante a pesquisa de Doutorado em Computação Aplicada, foi aceito para publicação na revista ***Discontinuity, Nonlinearity and Complexity***.

Discontinuity, Nonlinearity, and Complexity 3(1) (2014) 301–320



Discontinuity, Nonlinearity, and Complexity

https://lhscientificpublishing.com/Journals/DNC-Default.aspx

Nonlinear Time Series Analysis of Complex Systems Using an e-Science Web Framework

Bruno B. F. Leonor†, Walter A. dos Santos, Asiel Bomfin Jr., Reinaldo R. Rosa

National Institute for Space Research, São José dos Campos, Brazil

Submission Info

Communicated by Referees
Received DAY MON YEAR
Accepted DAY MON YEAR
Available online DAY MON YEAR

Keywords

Time Series Analysis
Nonlinear Dynamics and Complex Systems
Chaos and Global Nonlinear Dynamics
Web Framework
High-Performance Computing

Abstract

The analysis of time series in the era of Big Data has become a major challenge for computational framework research. Furthemore, in the areas of space science which deals with a large variety of data, the practical consistence between workload, workflow and cloud computing is crucial. Here, such consistence is provided by an innovative e-Science framework project named Sentinel which is based on a NoSQL data base (MongoDB) and a containerization platform (Docker). This web framework supports researchers for time series analysis in a cloud environment where they can easily access, parameterize, initialize and monitor their applications. As a case study in the Brazilian Space Weather Program, we consider the intensive analysis of time series from a complex information system for solar activity monitoring and forecasting. As a prototype for implementing the framework, the DFA (detrended fluctuation analysis) technique was used as a nonlinear spectrum analyzer applied to the solar irradiance measurements from 1978 to 2012. Moreover, new applications can be added and managed by researchers on the portal easily to complement their data analysis purposes.

©2012 L&H Scientific Publishing, LLC. All rights reserved.

1 Introduction

Many important dynamical phenomena in science can only be investigated using methods for analyzing the correspondent complex underlying process. Usually, such process are represented as high resolution data collection so that meaningful statistics and other major data characteristics can be available to researchers using Data Warehousing Platforms. Into this context, the main objective of this work is to present an innovative e-Science framework project named Sentinel which is based on a NoSQL database (MongoDB) and a containerization platform (Docker). This a natural tendency as the web has been causing a growing impact on scientific research, technological development and society as a whole with its growth over the past few years and has brought a different style of computing involving new concepts as: *Cloud Computing*, *e-Science*, *Big Data* and *Data Science* [1, 2]. Due to the huge amount of heterogenoues scientific datasets produced daily, the sinergy among those areas are being benefited with the use of new advanced scientific computing environments [3]. This has

†Corresponding author.
Email address: brunobfl@gmail.com

ISSN 2164 – 6376, eISSN 2164 – 6414/$-see front materials © 2014 L&H Scientific Publishing, LLC. All rights reserved.
DOI : 10.5890/DNC.2013.02. ∗ ∗∗

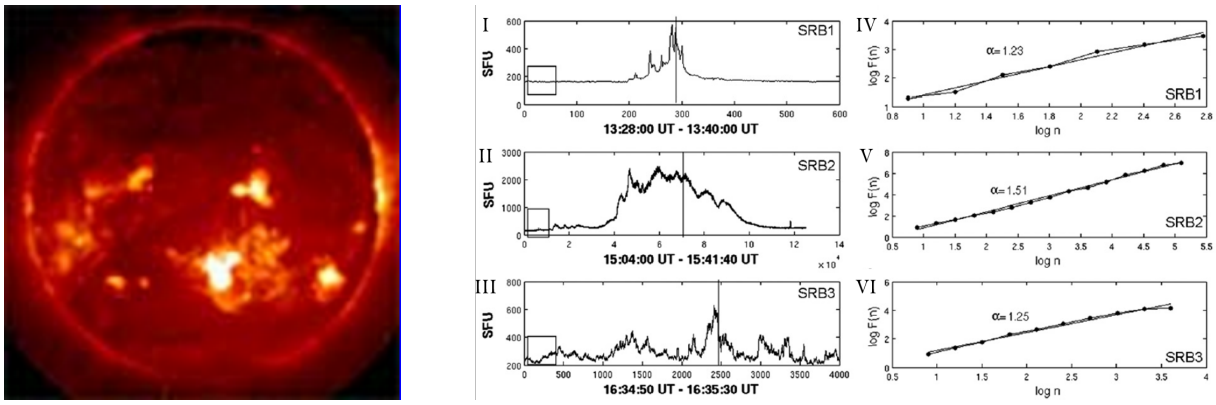
contributed to the creation of a powerful compound ecosystem of humans and machines that together are able to solve problems that could not be solved alone before.

Among the main basic solutions for analysis of Big Scientific Data offered by a framework is its efficiency in the compatible use of cloud computing management and alternative nonrelational database as NoSQL. Therefore, this paper applies these concepts to support intensive time series analysis in a cloud environment where scientists can add, parameterize, initialize and monitor their applications transparently using any enabled device to access the web. In their scientific research, each researcher will dispose of a set of tools created by your own as well as take advantage of tools shared by other researchers in your workspace.

The time variability of many natural and social phenomena is not well described by standard methods of data analysis. However, nonlinear time series analysis comprises a wide variety of measures that allows one to extract different features of a dynamical system underlying some measured signal [4]. The results can be applied to real data from physics, biology, medicine, and engineering. The Sentinel is based on the detailed improvement of the platform called VLADA [5]. As a case study, the prototyped technique namely detrended fluctuation analysis (DFA) is addressed for intensive time series analysis in the Brazilian Space Weather Program, named "Embrace"(Brazilian Studies and Monitoring of Space Weather) and created in 2007 [a].

Embrace's mission is to carry out the monitoring, modeling and dissemination of space weather information regarding to phenomena research and forecasting of significant effects on the space and on the surface of the Brazilian territory, including impacts on space and ground technological systems [7]. EMBRACE is mainly concerned with the development and service area related to solar phenomena and physical occurrences in the space environment [7], which are recurrently manifested.

An illustrative example is the solar activity of June 6th 2000 [8, 9] when solar filament eruptions were accompanied by intense solar flares including a full-halo coronal mass ejection (CME) in association with this event (Solar Flare NOAA Report) [8,9]. This observation is depicted in Figure 1 where (a) shows an observation example of Soft X-ray and (b) shows the phenomena observed as solar radio bursts events (SRB - SRB1 (I), SRB2 (II) and SRB3 (III)) by means of the 3 GHz radiometer and the respective DFA scaling exponents (IV, V and VI) [10]. This DFA analysis can be useful for operational data analysis providing solar flares monitoring, alerts and forecasts and, possibly, of geomagnetic activity.



(a) (b)

Fig. 1 Example of a typical solar flare observed in multiple frequencies: (a) Soft X-ray and (b) Radio (3 GHz). Due to a nonlinear plasma process in the solar atmosphere the evolution of this complex dynamics can be characterized by means of the evolution of the scaling exponent calculated recursively from the DFA spectrum [10].

[a]Embrace was conceived within international efforts to monitor space weather which includes a number of international initiatives with similar purposes [6].

Many solar phenomena and physical occurrences can be expressed by a time series which is basically a series of data points listed or graphically displayed in time order that are usually used as source of information for grasping phenomena understanding and scientific research in its various aspects. Time series analysis assumes that the set of observations is being generated by a nonlinear process and as such, it has a probabilistic framework that can be characterized and described [11]. For analysis of the time series obtained by Embrace program, this work has selected only one well accepted technique as a case study, namely DFA which has been useful for space weather forecast [10].

The DFA method was first proposed by [12] to analysis of DNA sequences. DFA was basically designed to reveal long range correlation in non stationary processes [13]. However, DFA can also be regarded as a suitable method to investigate both long-range and short range correlation in non-stationary and stationary systems.

The space weather was chosen as a complex phenomena case study for the knowledge, monitoring and prediction of different solar geoeffective process that directly involve a huge variety of data. The term space weather refers to conditions on the Sun and in the solar wind, magnetosphere, ionosphere, and thermosphere that can influence the performance and reliability of space-borne and ground-based technological systems and that can affect human life and health [14]. Observation of space weather is done both for scientific research and for practical applications [15]. Scientific observation has evolved with the state of knowledge, while application-related observation expanded with the ability to exploit the analysis of such big real-time data and the variety of services potentially offered to the space weather area.

The major impacts of a severe space weather event can roughly be divided into two areas [16]: (1) impacts on technology and threats to equipment and (2) health in space and high altitude as below: (a) Power grid outages, (b) Disruption to Global Positioning Satellites (GPS), (c) High Frequency (HF) radio communications outages, (d) Satellite damage and (e) Increased radiation threat at high altitude.

In Section 2, we briefly describe the case study data base which has been the main motivation to the Sentinel framework implementation. Section 3 contains a detailed description of the Sentinel framework. The practical results from the case study are shown in Section 4. Section 5 contains the main concluding remarks.

2 Data Base: Workload and Workflow

The data base from space weather observations was chosen as a complex data warehousing for the knowledge, monitoring and prediction of different solar geoeffective process that directly involve a huge variety of data (time series and digital images). The term space weather refers to conditions on the Sun and in the solar wind, magnetosphere, ionosphere, and thermosphere that can influence the performance and reliability of space-borne and ground-based technological systems and that can affect human life and health [14]. Observation of space weather is done both for scientific research and for practical applications [15]. Scientific observation has evolved with the state of knowledge, while application-related observation expanded with the ability to exploit the analysis of such big real-time Data and the variety of services potentially offered to the space weather area. An example of the compound observation on space weather condictions during solar activity can be seen in Figure 2. This event, known as The Great Halloween Solar Storm, was observed by satellites GOES-11 and GOES-12 which are part of the US National Oceanic and Atmospheric Administration's Geostationary Operational Environmental Satellite system. The typical average workload of such data base is 6 GB/day. Assuming a space weather data networking involving at least 100 missions (set of different real time instruments) we project a real time workload $>$ 600 GB/day.

The volume of data must be processed several times by different users on different platforms. The workflow of usage involving different applications (analytical tools, such as DFA) should be made optimal use of the cloud and the algorithm (which may distribute threads among the cores of a hybrid High-Performance Computing architecture). Once defined the workload of the virtual tools (tw), data (dw), and the user device (speedup-s) will be possible to calculate the workflow as a function $f(tw, dw, s)$. The workload of the virtual tools will depends on the clouding computing solution. In the present prototype, the computational infrastructure used

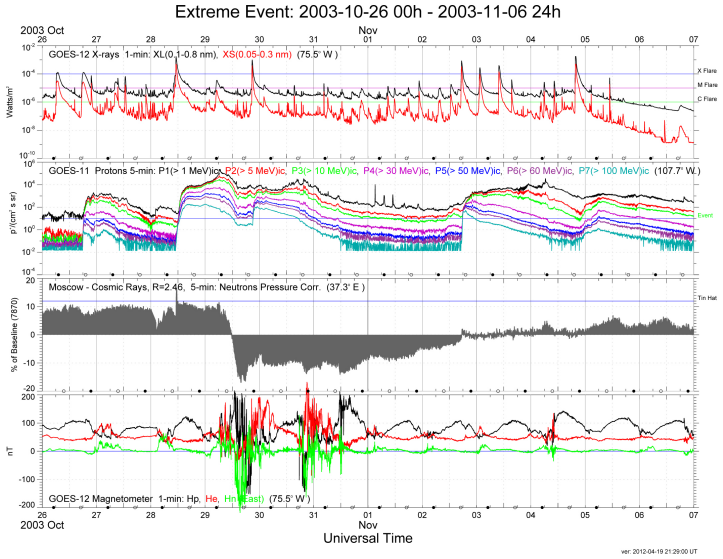


Fig. 2 Space weather condictions monitored by GOES-11 and GOES-12 during the Oct. 2003 solar activity [18].

for storage and execution of experiments was the Amazon Web Services (AWS). AWS is a collection of cloud computing services or web services that form a computing platform in the cloud [17].

3 The Web Framework for Time Series Analysis

As mentioned, the main purpose of this paper is to introduce an e-Science environment (based on the DFA prototype) named Sentinel dedicated and suitable for managing scientific applications. Scientists affiliated to this portal can register their applications and transparently manage them, more specifically, they can access, parameterize, initialize, monitor, analyze and share the results obtained through any device that allows access to the web.

3.1 e-Science Frameworks

e-Science mainstream are commonly used to describe the development of software services infrastructure to support scientific workflows in which the Sentinel web framework falls [19]. This enables access to remote facilities, distributed computing resources, information storage in dedicated databases, dissemination and sharing of data, results and knowledge.

Research in e-Science requires joint and multidisciplinary efforts where computer scientists assist researchers from other fields of knowledge to develop their research more quickly and efficiently and, in this partnership, the computer solutions tend to be made innovative and transformatively [20].

Basically, the e-Science research can be described by the following characteristics [20]: (1) involves collaboration of researchers computing with researchers from other fields of knowledge, and (2) requires the creation of sophisticated computational methods to handle large volumes of data (Big Data) and/or to perform simulations and programs that require complex computational systems.

3.2 The Sentinel e-Science Framework

Sentinel e-Science environment described in this paper allows each researcher to develop their scientific research through any web-enabled device. In their scientific research each researcher will have available a set of tools created by him/her as well as take advantage of tools shared by other researchers in your desktop.

There are two ways to access the portal, either as a researcher or guest. Figure 3 shows the main features for a ”researcher” profile.

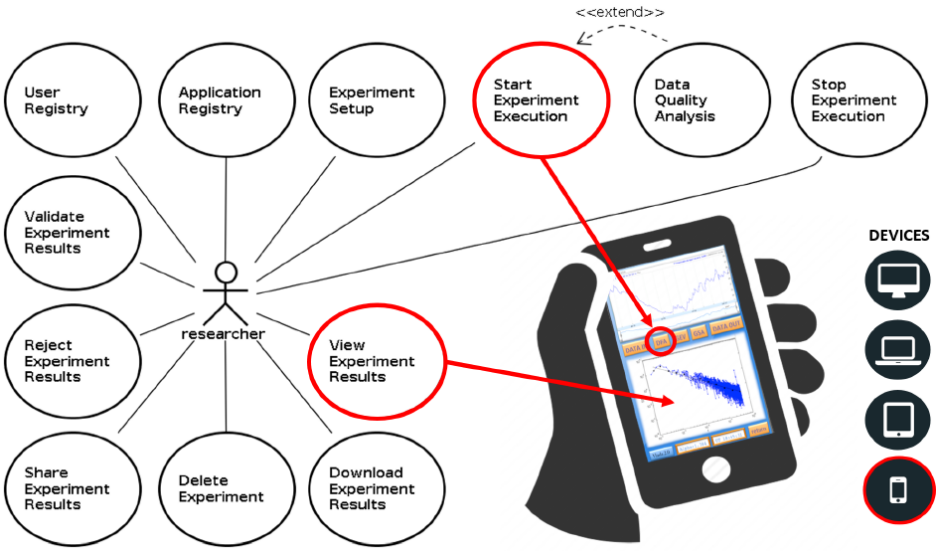


Fig. 3 Use case diagram showing the functions provided to the researchers by the Sentinel Portal where the functions Start Experiment Execution and View Experiment Results are highlighted and can be used through any web-enabled device, for example a smartphone as shown here.

Most of these features are available for each of the experiments via the Dashboard interface in an ”Actions” column. Each researcher will have his/her own desktop with your set of tools that will be built from the application registry interface. A ”guest” can only view and download shared results.

The Sentinel portal allows researchers to concentrate all necessary applications in order to develop their research in a single environment without the need round up his/her own computing infrastructure.

The Sentinel development is currently being done using the Java language since it is the programming language most commonly used by developers around the world according to the rankings: Tiobe [21], RedMonk [22] and Spectrum [23]. The authors believe that this may facilitate future maintenance and improvements in the environment due to the large community of developers.

3.2.1 Application Registration

In order to have its own set of tools available in your desktop, a researcher must first incorporate them into the environment so he/she can perform his/her experiments. Therefore, in Application Registry interface, shown in Figure 4, the researcher will add all the information necessary for the implementation of application, ranging from the programming language used in the application setting up parameters needed for its execution. This informations are grouped into the tabs `File`, `Variables`, `Lib` and `Compiler`.

Each registered application may have their visibility set by the researcher to be either public or private. If set as public other researchers will have access to use it.

In the `File` tab, as shown in Figure 4, is intended for uploading the application source code. In this case it shows the DFA application registry.

In the `Variables` tab the input variables are define for the application parameterization. If necessary, for each field, it will be assigned a type (text, number, date, file or options), a default value and description.

If necessary, the inclusion of some special library may be added to the environment by `Lib` tab.

The last tab, `Compiler`, is used to define the application’s build parameters within the Sentinel portal environment where the researcher can enter a script to compile and run the application.

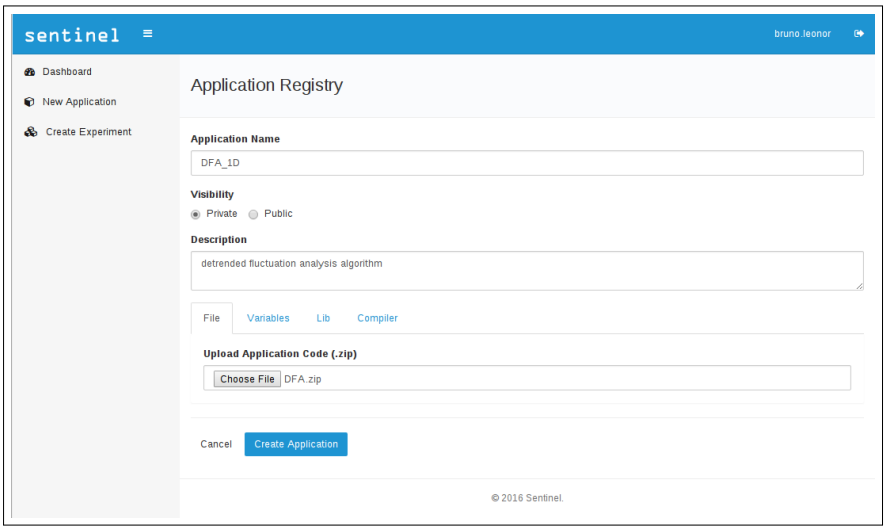


Fig. 4 The Sentinel Application Registry Interface - `File` tab highlighted.

3.2.2 Experiment Setup

After registering by the researcher in the environment, the application is ready to run. The ”Experiment Setup” interface, shown in Figure 5, is created automatically by the framework based on the information entered in the application registry interface - `Variables` tab. The user needs to set the values that will serve as input for the execution of the application. In this case the input variable needed is a file containing a time series that will be analized by DFA.

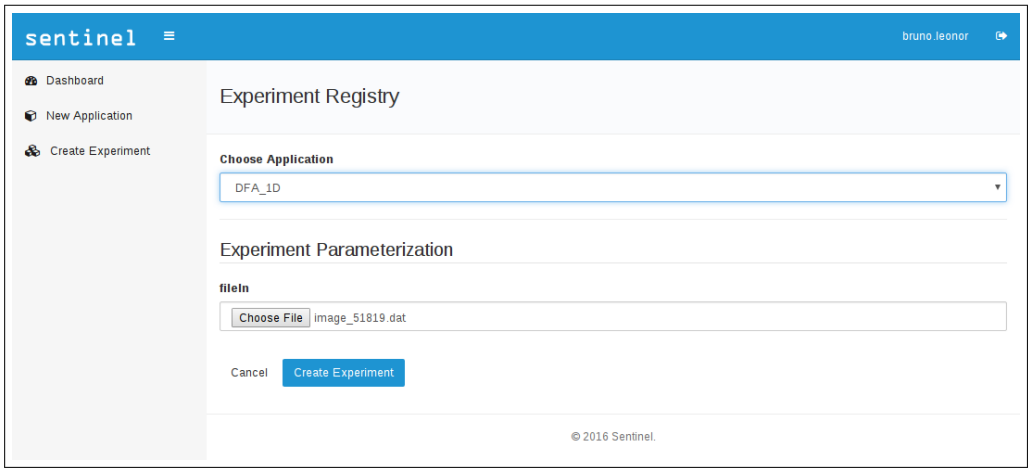


Fig. 5 The Sentinel Experiment Setup Interface.

All the settings created to perform this step are generically called an experiment.

Input and output data of each experiment are storaged at the MongoDB. Instead of using tables and rows as in relational databases, MongoDB (a NoSQL database) is built on an architecture of collections and documents. A document is a data structure composed of field and value pairs. MongoDB documents are similar to JSON (JavaScript Object Notation) objects. The values of fields may include other documents, arrays, and arrays of documents. [24]

A case study from Solar Physics is hereafter presented as an example of using the system. To this study it was

used the total solar irradiance (TSI) data from VIRGO/SOHO [25] extracted from composite_d41_62_1204.txt [26] file. Figure 6 shows the plot of the complete dataset.

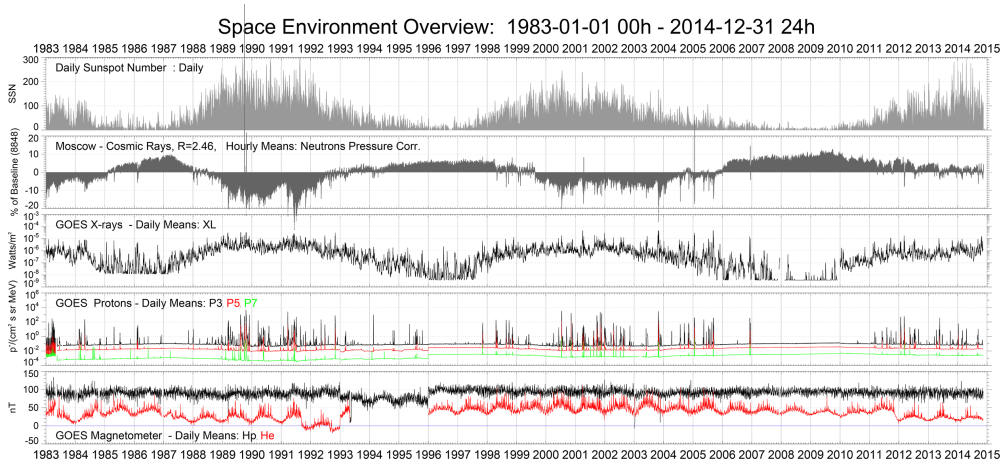


Fig. 6 Total Solar Irradiance (TSI) dataset from VIRGO/SOHO for Space Environment Overview [27].

3.2.3 Experiment Management

The Sentinel Dashboard interface, shown in Figure 7, is where the user can track all experiments.

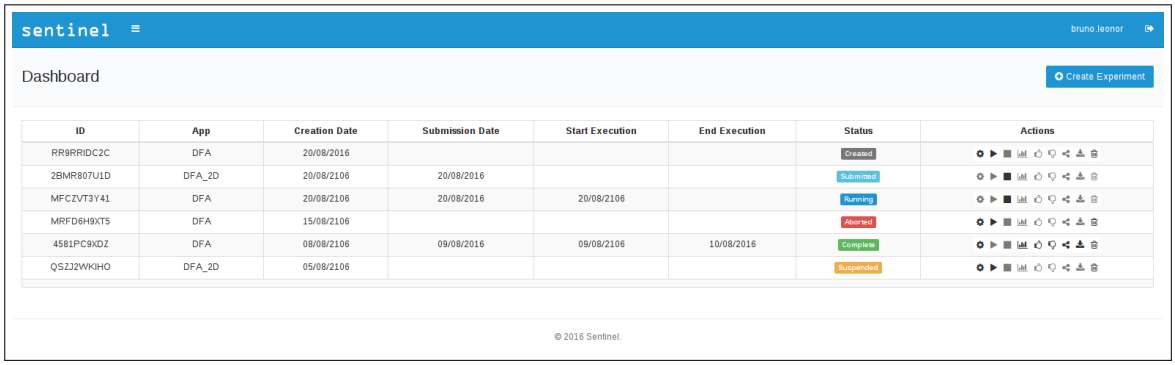


Fig. 7 The Sentinel Dashboard Interface.

Through this interface, researchers will have access a set of information regarding each experiment, namely:

ID - experiment identifier.

App - chosen application.

Creation Date - creation date of the experiment.

Submission Date - request date for execution.

Start Execution - initial date of execution.

End Execution - end of the execution date.

Status - current situation of the experiment.

Actions - actions to be taken by the researcher: Experiment Setup (⚙), Start Execution (▶), Stop Execution (■), View Results (📊), Validate Results (👍), Reject Results (👎), Share Results (🔗), Download Results (📥) and Delete an Experiment (🗑) and its results.

Each experiment, after starting, is performed in an isolated and secure environment through Docker containers. Docker containers wrap a piece of software in a complete filesystem that contains everything needed to run: code, runtime, system tools, system libraries - anything that can be installed on a server. This guarantees that the software will always run the same, regardless of its environment. [28]

By using containers, resources can be isolated, services restricted, and processes provisioned to have an almost completely private view of the operating system with their own process ID space, file system structure, and network interfaces [29].

3.2.4 Viewing Results

In order to view the experiment results, Sentinel portal provides to researchers a set of visualization tools with charts, 2D and 3D plot, statistics, histogram, among others. After the experiment is visualized and evaluated, the results may be accepted or rejected. Rejected results will be removed from the platform; accepted results will be available to the scientists for a time not set yet. These actions can be applied to any experiment.

To facilitate access to the data generated in the experiment processing the View interface, as well as Application Registry interface, is also organized into the tabs: `View`, `Variables`, `Outputs` and `Log`.

`View` tab is where the experiment results are plotted, in graphs for example. The possible display modes depend on the format of the generated application result. Figure 8 shows a line chart as result of an experiment using DFA technique.

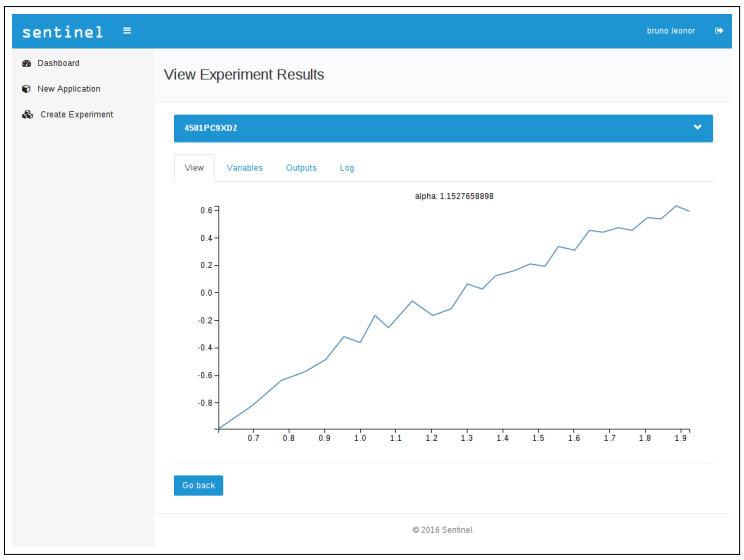


Fig. 8 The Sentinel View Interface - `View` tab highlighted.

Information about dates of experiment processing can be found on top of the interface. For this purpose, just a click up the experiment ID displayed in the blue bar to show or hidden this content.

In the `Variables` tab is shown the experiment configuration. In other words in this tab is shown the values defined by the researcher for each needed input variable to process the experiment.

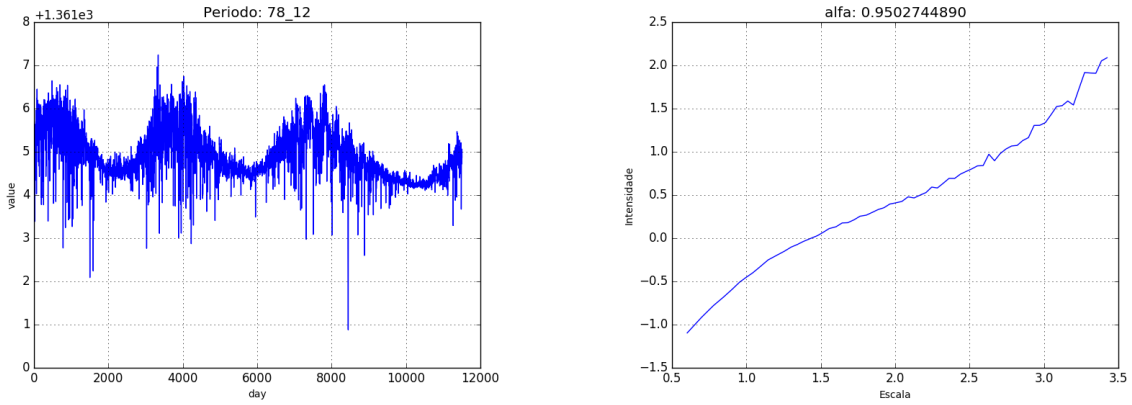
All files produced by experiment processing can be downloaded individually through the `Outputs` tab.

If details are needed about experiment processing this information is available on the `Log` tab.

4 Output Results from Sentinel

Several experiments on the Total Solar Irradiance (TSI) dataset from VIRGO/SOHO were conducted in Sentinel applying the DFA technique in the time series presented in section 3.2.2 and some of these results obtained in these experiments will be presented in this section.

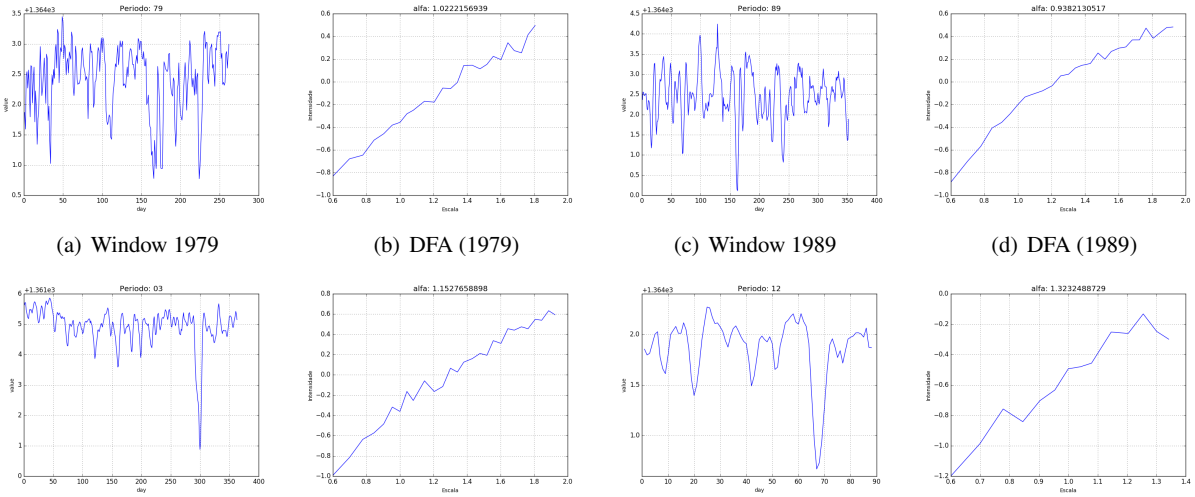
The Figure 9(a) presents the complete time series for the variable TSI and the result of DFA application to this series is shown in Figure 9(b).



(a) TSI variable (b) DFA result

Fig. 9 Complete time series for TSI variable (1978 - 2012).

The complete time series was cut in various windows representing the observations made for each year and exponent α values were calculated for each window. Some results are presented in Figure 10 for the years 1979 (Figure 10(a)), 1989 (Figure 10(c)), 2003 (Figure 10(e)) and 2012 (Figure 10(g)). The results of the DFA technique application for these windows are presented in the same chronological order in Figure 10(b), Figure 10(d), Figure 10(f) and Figure 10(h).



(a) Window 1979 (b) DFA (1979) (c) Window 1989 (d) DFA (1989)

(e) Window 2003 (f) DFA (2003) (g) Window 2012 (h) DFA (2012)

Fig. 10 Some windows of the time series for TSI variable and their results applying DFA technique.

To help monitoring the environment behavior the elapsed time [b] in each experiment execution was calculated. These results are presented in Table 1.

Table 1: α value and elapsed time of execution for each experiment (window)

Experiment ID	**Application**	**Window**	**Data Points**	α	**Elapsed Time**	**Computing Throughput** (points / second)
KVV7PTRLN4	DFA	1978-2012	11506	0.950274488985	0m1.130s	10182,30
15G1A2RLJ2	DFA	1978	34	1.43120355455	0m0.776s	43,81
ZMW2JDLFL8	DFA	1979	262	1.02221569392	0m0.788s	332,49
58AJWU6P11	DFA	1980	331	1.04787741919	0m0.868s	381,34
683DIIHVOR	DFA	1981	274	1.07067813268	0m0.802s	341,64
RMRR37DI2L	DFA	1982	272	0.96771013622	0m0.809s	336,22
MZ97XUIPWV	DFA	1983	301	0.95327337799	0m0.800s	376,25
3WCIY48UJY	DFA	1984	347	1.04847694358	0m0.882s	393,42
QSUVQ396QD	DFA	1985	348	0.952884000354	0m0.834s	417,27
F4QBJOEOA3	DFA	1986	338	0.8937840092	0m0.837s	403,82
RA1BW6HM3F	DFA	1987	340	0.779929247501	0m0.841s	404,28
0VSBK6ETAR	DFA	1988	362	0.912727838497	0m0.839s	431,47
KFWWQD16HN	DFA	1989	352	0.938213051694	0m0.871s	404,13
MG98JFBX8U	DFA	1990	365	0.976737932127	0m0.810s	450,62
K16182LNUG	DFA	1991	363	1.02867519185	0m0.910s	398,90
RJGDHX5KAL	DFA	1992	315	0.97805026371	0m0.804s	391,79
6H8ZDRYCXH	DFA	1993	350	1.11578080511	0m0.836s	418,66
4JMBJ0F1RQ	DFA	1994	360	1.10779497669	0m0.811s	443,90
XB142MK6ZE	DFA	1995	319	1.06629504897	0m0.962s	331,60
PB7Z5Q8BV5	DFA	1996	357	0.994899152143	0m0.841s	424,49
TWTB32LUPB	DFA	1997	365	0.944063946741	0m0.779s	468,55
QH1ICIFUKK	DFA	1998	338	0.966266225997	0m0.823s	410,69
17DLS1570O	DFA	1999	361	0.960423763573	0m0.791s	456,38
NJ7RTKXONW	DFA	2000	365	1.11876844216	0m0.901s	405,10
BU5XLTSSV2	DFA	2001	365	0.958250359695	0m0.772s	472,80
HXBQDEN6M9	DFA	2002	362	0.956157571451	0m0.809s	447,47
4581PC9XDZ	DFA	2003	363	1.15276588982	0m0.851s	426,57
O1EK4GUY65	DFA	2004	358	1.10171427129	0m0.810s	441,97
UGBRIKHS8R	DFA	2005	365	0.977228425668	0m0.822s	444,04
VRKUJ1DPW3	DFA	2006	365	1.00873551784	0m0.815s	447,85
J85NCGR0H3	DFA	2007	365	0.960971280023	0m0.863s	422,94
0FWLTPD0HE	DFA	2008	366	1.02249791239	0m0.813s	450,18
48L69ATUMQ	DFA	2009	365	1.01425234186	0m0.810s	450,62
UBZL9AWQZQ	DFA	2010	365	1.00171721802	0m0.844s	432,46
RIVAQFPRRE	DFA	2011	360	1.03948303634	0m0.942s	382,16
FGQJU7DJEY	DFA	2012	88	1.32324887294	0m0.790s	111,39

To measure the performance the computing throughput was calculated for each experiment. Figure 11 shows

[b] Based on Intel(R) Core(TM) i7-3820 CPU @ 3.60GHz processor and 16 GB of memory.

the amount of observed data for each year and their computing throughput.

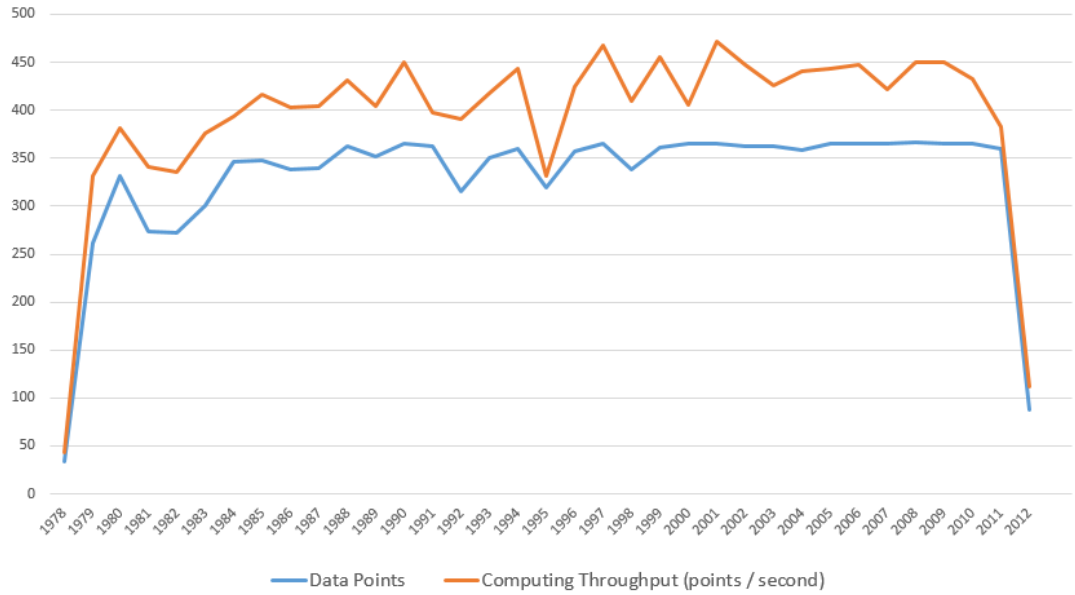


Fig. 11 Amount of data points for year X Computing Throughput (data points processed by seconds).

Finally, the evolution of the α for each window is shown in Figure 12. The behavior from DFA analysis can be useful in a framework of operational data analysis providing monitoring, alerts and forecasts of solar flares and, possibly, of geomagnetic activity [10].

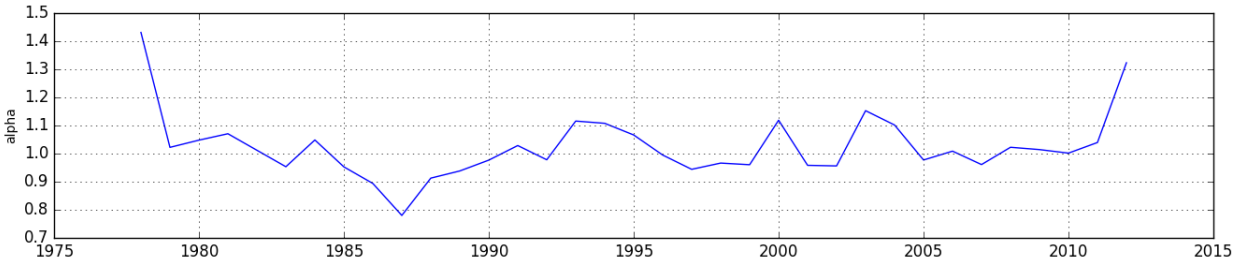


Fig. 12 Evolution of calculated DFA α parameter for each year of observation.

5 Concluding Remarks

This work presented a web framework named Sentinel dedicated and suitable for managing e-Science applications for Space Weather purposes. In particular, the Embrace space weather program was chosen as a domain for the case study for the knowledge and prediction of different phenomena that directly affect human activities, such as solar activity, magnetic storms and ionospheric fluctuations. These topics have great importance for the well-being of society. As a prototype application, the Sentinel framework has been applied for the solar irradiance time series monitoring using DFA. Based on the output results of the present application both the workload and workflow of the framework were discussed.

Future work will incorporate more challenging experiments and algorithms, taking into account intensive

data analysis from big data files including digital images. Other applications that can be incorporated for mutidimensional time series analysis are: PSD (Power Spectral Density) [30], Singularity Spectrum [31], DFA2D (Detrended Fluctuation Analysis 2D) [32], GPA (Gradient Pattern Analysis) [33], among others.

Since the framework is general and open source, new techniques and algorithms, as any area of research, can be incorporated into the Sentinel portal by scientists and scientific teams. It is important to mention that a module of paramount importance in the system concerns the data quality. This module will be incorporated and discussed in a future paper. The authors expect that this work may contribute to a friendlier and safer environment to e-Science users, where they can gather all their applications used in the development of their research and information technology services.

Acknowledgments

The authors would like to thank Vanessa C. Oliveira de Souza for providing the DFA code used in the case study presented. The authors are greatful to the PMOD/WRC, Davos, Switzerland for the dataset provided (composite_d41_62_1204.txt) from the VIRGO Experiment on the cooperative ESA/NASA Mission SOHO. RRR acknowledge financial support from FAPESP through the grant No 2014/11156.

References

[1] N. Shadbolt, W. Hall, J. A. Hendler, W. H. Dutton,"Web science: a new frontier", *The Royal Society*, **371**(1987), 2013.
[2] E. Alisson, "eScience revoluciona a forma como se faz ciência.", available at: http://agencia.fapesp.br/17279, accessed January 2016.
[3] K. Tolle, S. Tansley, T. Hey, "O Quarto Paradigma: Descobertas Científicas na era da eSciente", *Oficina de Textos*, 2011.
[4] H. Kantz, T. Schreiber, "Nonlinear Time Series Analysis", *Cambridge University Press*, 2004.
[5] M. S. Dantas, R. R. Rosa, N. SantAnna, M. G. Cereja Jr, T. B. Veronese, S. Bianchi, J. C. Rosa, K. M. Alexiev, J. D. S. da Silva, "The VLADA white paper: building an active Virtual Lab for Advanced Data Analysis", *J. Comp. Int. Sci.*, **2**(1), 47-56, 2011.
[6] F. J. G. Pinheiro, M. T. Barata, J. M. Fernandes, "Comparison of Space Weather Services: Information Systems, Activity and Forecasts", *J. Comp. Int. Sci.*, **7**(2), 3-24, 2016.
[7] INPE, "O Programa Embrace", available at: http://www2.inpe.br/climaespacial/portal/, accessed January 2016.
[8] H. Zell, "What is solar activity?" , available at: https://www.nasa.gov/content/goddard/what-is-solar-activity, accessed August 2016.
[9] C. A. Young, "Solar Activity" , available at: http://www.thesuntoday.org/the-sun/solar-activity/, accessed July 2016.
[10] T. B. Veronese, R. R. Rosa, M. J. A. Bolzan, F. C. Rocha Fernandes, H.S . Sawant, M. Karlicky, "Fluctuation analysis of solar radio bursts associated with geoeffective X-class flares", *Journal of Atmospheric and Solar-Terrestrial Physics*, **73**(11-12), 1311-1316, 2011, available at: http://www.sciencedirect.com/science/article/pii/S1364682610002907.
[11] R. M. Kirchner, R. C. Souza, F. A. Ziegelmann, "Identificação de estruturas não-lineares de séries temporais através de regressão linear local e modelos aditivos", Pesquisa Operacional, available at: http://dx.doi.org/10.1590/S0101-74382008000100003 **28**, 2008.
[12] C. K. Peng, S. V. Buldyrev, S. Havlin, M. Simons, M. H. E. Stanley, A. L. Goldberger, "Mosaic organization of DNA nucleotides", *Phys. Rev. E*, **49**(2), 1685-1689, 1994.
[13] V. V. Morariu, L. Buimaga-Iarinca, C. Vamos, S. Soltuz, "Detrended Fluctuation Analysis of Autoregressive Processes", *Cornell University Library*, 2007.
[14] R. Schwenn, "Space Weather: The Solar Perspective", *Living Reviews in Solar Physics*, **3**(2), 2006, available at: http://www.livingreviews.org/lrsp-2006-2.
[15] W. B. Cade III, C. Chan-Park, "The Origin of Space Weather", *Space Weather*, **13**(2), 99-103, 2015.
[16] MetOffice, "Space Weather Impacts" , available at: http://www.metoffice.gov.uk/publicsector/emergencies/space-weather/impacts, accessed July 2016.
[17] AWS, "Overview of Amazon Web Services", available at: https://d0.awsstatic.com/whitepapers/aws-overview.pdf, accessed August 2016.
[18] D. Wilkinson, "Extreme Space Weather Events", *National Geophysical Data Center*, 2012, available at: https://commons.wikimedia.org/wiki/File:ExtremeEvent_20031026-00h_20031106-24h.jpg.
[19] D. Palazzi, L. Silva, L. F. Mendes, W. Gaspar, E. Matos, F. Campos, R. Braga, "Uso de ontologias em projetos de

e-science", BDBComp, 2009.
[20] Fapesp, "Programa FAPESP de Pesquisa em eScience", available at: http://www.fapesp.br/publicacoes/2015/folder_escience.pdf, accessed December 2015.
[21] Tiobe, "TIOBE Index for February 2016", available at: http://www.tiobe.com/tiobe_index, accessed February 2016.
[22] RedMonk, "The RedMonk Programming Language Rankings: January 2016", available at: https://redmonk.com/sogrady/category/programming-languages/, accessed February 2016.
[23] Spectrum, "The 2015 Top Ten Programming Languages", available at: http://spectrum.ieee.org/computing/software/the-2015-top-ten-programming-languages, accessed February 2016.
[24] MongoDB, "Introduction to MongoDB", available at: available at: https://docs.mongodb.com/manual/introduction/, accessed August 2016.
[25] C. Frohlich, J. Lean, "The Sun's total irradiance: Cycles, trends and related climate change uncertainties since 1976", *Geophysical Research Letters*, **25**(23), 4377-4380, 1998, available at: http://dx.doi.org/10.1029/1998GL900157.
[26] ACRIM, "composite_d41_62_1204.txt", 2012, available at: acrim.com/TSI/composite_d41_62_1204.txt.
[27] NOOA, "Space Environment Overview", 2014, available at: http://satdat.ngdc.noaa.gov/sem/goes/data/new_plots/special/Overview_19830101-00h_20141231-24h.png.
[28] Docker, "What is Docker?", available at: https://www.docker.com/what-docker, accessed August 2016.
[29] Cirrus Tech Ltd, "Docker: Reorganizing the Data Center", available at: http://www.cirrushosting.com/cloud-hosting-blog, accessed August 2016.
[30] A. V. Oppenheim, G. C. Verghese, "Signals, Systems and Inference", *Prentice-Hall signal processing series*, Pearson Education, 2015.
[31] M. J. A. Bolzan, A. Tardelli, V. G. Pillat, P. R. Fagundes, R. R. Rosa, "Multifractal analysis of vertical total electron content (VTEC) at equatorial region and low latitude, during low solar activity", *Annales Geophysicae*, **31**(1), 127-133, 2013.
[32] G.F. Gu, W.X. Zhou, "Detrended fluctuation analysis for fractals and multifractals in higher dimensions", *Phys. Rev. E*, **74**(6), American Physical Society, 2006.
[33] R. R. Rosa, A. S. Sharma, J. A. Valdivia, "Characterization of Asymmetric Fragmentation Patterns in Spatially Extended Systems", *International Journal of Modern Physics C*, **10**(01), 147-163, World Scientific Publishing Company, 1999.

PUBLICAÇÕES TÉCNICO-CIENTÍFICAS EDITADAS PELO INPE

Teses e Dissertações (TDI)

Teses e Dissertações apresentadas nos Cursos de Pós-Graduação do INPE.

Manuais Técnicos (MAN)

São publicações de caráter técnico que incluem normas, procedimentos, instruções e orientações.

Notas Técnico-Científicas (NTC)

Incluem resultados preliminares de pesquisa, descrição de equipamentos, descrição e ou documentação de programas de computador, descrição de sistemas e experimentos, apresentação de testes, dados, atlas, e documentação de projetos de engenharia.

Relatórios de Pesquisa (RPQ)

Reportam resultados ou progressos de pesquisas tanto de natureza técnica quanto científica, cujo nível seja compatível com o de uma publicação em periódico nacional ou internacional.

Propostas e Relatórios de Projetos (PRP)

São propostas de projetos técnico-científicos e relatórios de acompanhamento de projetos, atividades e convênios.

Publicações Didáticas (PUD)

Incluem apostilas, notas de aula e manuais didáticos.

Publicações Seriadas

São os seriados técnico-científicos: boletins, periódicos, anuários e anais de eventos (simpósios e congressos). Constam destas publicações o Internacional Standard Serial Number (ISSN), que é um código único e definitivo para identificação de títulos de seriados.

Programas de Computador (PDC)

São a seqüência de instruções ou códigos, expressos em uma linguagem de programação compilada ou interpretada, a ser executada por um computador para alcançar um determinado objetivo. Aceitam-se tanto programas fonte quanto os executáveis.

Pré-publicações (PRE)

Todos os artigos publicados em periódicos, anais e como capítulos de livros.