
# FRAMEWORK PARA SERVIÇOS DE APRENDIZADO DE MÁQUINA

PEDRO HOLLANDA BOUEKE

Projeto de Graduação apresentado ao Curso de Engenharia de Computação e Informação da Escola Politécnica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Engenheiro.

Orientador: Flávio Luis de Mello

Rio de Janeiro

Março de 2019

# FRAMEWORK PARA SERVIÇOS DE APRENDIZADO DE MÁQUINA

PEDRO HOLLANDA BOUEKE

PROJETO DE GRADUAÇÃO SUBMETIDO AO CORPO DOCENTE DO CURSO DE ENGENHARIA DE COMPUTAÇÃO E INFORMAÇÃO DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO DE COMPUTAÇÃO E INFORMAÇÃO

Autor:

PEDRO HOLLANDA BOUEKE

Orientador:

Flávio Luis de Mello

Examinador:

XXXXXXXXXXXXXXXXXXXX

Examinador:

XXXXXXXXXXXXXXXXXXXX

Rio de Janeiro

Março de 2019

Declaração de Autoria e de Direitos

Eu, *Pedro Hollanda Boueke* CPF *108.243.966.50*, autor da monografia *Framework para Serviços de Aprendizado de Máquina*, subscrevo para os devidos fins, as seguintes informações:

1. O autor declara que o trabalho apresentado na disciplina de Projeto de Graduação da Escola Politécnica da UFRJ é de sua autoria, sendo original em forma e conteúdo.

2. Excetuam-se do item 1. eventuais transcrições de texto, figuras, tabelas, conceitos e idéias, que identifiquem claramente a fonte original, explicitando as autorizações obtidas dos respectivos proprietários, quando necessárias.

3. O autor permite que a UFRJ, por um prazo indeterminado, efetue em qualquer mídia de divulgação, a publicação do trabalho acadêmico em sua totalidade, ou em parte. Essa autorização não envolve ônus de qualquer natureza à UFRJ, ou aos seus representantes.

4. O autor pode, excepcionalmente, encaminhar à Comissão de Projeto de Graduação, a não divulgação do material, por um prazo máximo de 01 (um) ano, improrrogável, a contar da data de defesa, desde que o pedido seja justificado, e solicitado antecipadamente, por escrito, à Congregação da Escola Politécnica.

5. O autor declara, ainda, ter a capacidade jurídica para a prática do presente ato, assim como ter conhecimento do teor da presente Declaração, estando ciente das sanções e punições legais, no que tange a cópia parcial, ou total, de obra intelectual, o que se configura como violação do direito autoral previsto no Código Penal Brasileiro no art.184 e art.299, bem como na Lei 9.610.

6. O autor é o único responsável pelo conteúdo apresentado nos trabalhos acadêmicos publicados, não cabendo à UFRJ, aos seus representantes, ou ao(s) orientador(es), qualquer responsabilização/ indenização nesse sentido.

7. Por ser verdade, firmo a presente declaração.

_____

Pedro Hollanda Boueke

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
Escola Politécnica - Departamento de Eletrônica e de Computação
Centro de Tecnologia, bloco H, sala H-217, Cidade Universitária
Rio de Janeiro - RJ CEP 21949-900

## AGRADECIMENTO

Dedico este trabalho ao povo brasileiro que contribuiu de forma significativa à minha formação e estada nesta Universidade. Este projeto é uma pequena forma de retribuir o investimento e confiança em mim depositados.

## RESUMO

TODO

Palavras-Chave: aprendizado de máquina, django, agendador de tarefas.

**ABSTRACT**

TODO

Key-words: machine learning, django, job sheduler.

## SIGLAS

UFRJ - Universidade Federal do Rio de Janeiro

MTV - Model Template View

# Conteúdo

# Lista de Figuras

# Lista de Tabelas

# Capítulo 1

# Introdução

## 1.1 Tema

O trabalho apresenta uma solução para uma das questões comumente negligenciadas dentro da área de tecnologias de Aprendizado de Máquina: a implantação e uso de modelos em ambientes não acadêmicos. A proposta executada se reserva à criação de um *framework* para gerência e execução de modelos de Aprendizado de Máquina de forma a automatizar os processos relacionados ao uso de tais modelos, da coleta dos dados a serem analisados, passando por sua execução até a entrega dos resultados.

## 1.2 Delimitação

O trabalho apresentado se destina a atender as demandas de agentes de todas as esferas do círculo de usuários de modelos de Aprendizado de Máquina. Enquanto necessárias execuções recorrentes de modelos, o *framework* desenvolvido apresenta uma solução compatível aos principais contextos em que tais modelos se apresentam. São exceções notáveis a esse conjunto de usuários parte dos desenvolvedores de modelos de Aprendizado de Máquina que, possivelmente, não estejam interessados em praticar execução de seus modelos em ambientes de produção. Também se excluem aqueles que não possuem requisitos técnicos para a execução dos componentes necessários ao sistema, como usuários do sistema operacional Windows, usuários de bancos de dados incompatíveis com a linguagem SQL e usuários impossibilidatos de

fazer uso da linguagem de programação *Python 3*.

## 1.3 Justificativa

Enquanto que algoritmos e modelos matemáticos são apresentados frequentemente pelo ambiente acadêmico como soluções aproximadas de problemas reais e complexos condizentes ao que se diz respeito da área de Aprendizado de Máquina, a sua implantação em ambientes não acadêmicos ou de produção poucas vezes é abordada. Dado esse cenário, nota-se que existe espaço e demanda para o estudo e desenvolvimento de novas metodologias e plataformas que ambicionem soluções para as dificuldades envolvidas em trazer um modelo de Aprendizado de Máquina para um ambiente não acadêmico, dentre as quais destacam-se: gerência de execuções e processamento, gerência de arquivos e modelos, visualização e acompanhamento de resultados.

## 1.4 Objetivos

Objetiva-se o desenvolvimento de um *framework* que permita o agendamento e acompanhamento de execuções de modelos de Aprendizado de Máquina. Uma execução caracteriza-se pela instanciação de um modelo, coleta dos dados a serem processados, inferência do modelo sobre os dados coletados e persistência tanto dos resultados quanto das informações relativas à execução. Por acompanhamento, refere-se à possibilidade de usuários do *framework* visualizarem os resultados das execuções, bem como de configurarem todos os aspectos da execução, como o escalonamento, o modelo e as bases de persistência.

Esse objetivo será alcançado com a implementação bem sucedida de um *framework* que permita que todos os detalhes descritos sejam executados, fazendo uso de um sistema de agendamento de tarefas para agendamento de execuções do modelo; uma plataforma de desenvolvimento de aplicações *Web* para permitir o acompanhamento e gerência das execuções; bases de dados que contenham as informações dos usuários do sistema, registros detalhando as etapas das execuções do modelo e os dados a serem processados. Dessa forma, serão objetivos parciais para se alcançar o fim

desejado:

1. Desenvolvimento de uma aplicação *Web* caracterizada por:
   (a) Permitir agendamento de execuções de programas para processamento de modelos de Aprendizado de Máquina.
   (b) Permitir a visualização de resultados e eventos gerados pelos processos agendados.
   (c) Permitir a execução de modelos de Aprendizado de Máquina, com a visualização dos resultados em tempo real.
   (d) Se comunicar com bases de dados externas para coleta e persistência de dados.
   (e) Ser reconfigurável a fim de atender ambientes diversos.
2. Desenvolvimento de uma base de dados relacionais para armazenamento de eventos do *framework*.
3. Desenvolvimento de um programa para execução de modelos de Aprendizado de Máquina.

## 1.5 Metodologia

Atendendo às expectativas de um projeto de *Software*, iniciou-se o projeto a partir de diagramas e modelos de Engenharia de Software que definem o funcionamento do sistema projetado. Foram elaborados: um diagrama de casos de uso, um diagrama de sequência, cinco diagramas de atividade e um diagrama de relacionamento de entidades.

Com o planejamento teórico concluído, foram definidas as ferramentas de desenvolvimento do sistema. Para controle de versão foi utilizada a ferramenta *Git*, com a manutenção de um repositório para todo o código produzido. Como linguagem de desenvolvimento, foi escolhida a linguagem *Python 3*, que se destaca por ser uma das principais linguagens de programação dentro da comunidade de praticantes de tecnologias de Aprendizado de Máquina [1]. Quanto ao desenvolvimento da aplicação

*Web*, foi selecionada a plataforma *Django*, que é caracterizada por depender da mesma linguagem de programação que a escolhida para o *framework* e possuir um sistema de autenticação de usuários embutido. Por fim, quanto aos bancos de dados, foram selecionados: o *SQLite3* para controle de acesso dos usuários, em uma cópia local à aplicação *Web*, o *MySQL* para armazenamento dos eventos produzidos pelo *framework* e, para o banco de coleta e persistência dos dados necessários aos modelos de Aprendizado de Máquina, um banco qualquer que possua compatibilidade com o conector genérico *ODBC*, permitindo consultados *SQL* genéricas.

O desenvolvimento do projeto e artefatos de software relacionados se deu de maneira incremental ao longo de um período de pouco mais de dois meses, sendo guiado por reuniões e revisões frequentes entre os colaboradores envolvidos para discussão de detalhes técnicos. A validação dos resultados foi feita por meio de inspeção dos artefatos produzidos.

## 1.6 Descrição

No segundo capítulo será abordada a fundamentação teórica do trabalho desenvolvido, com a elaboração em cima das funções e histórico das ferramentas desenvolvidas. Serão abordados os temas: sistemas de aprendizado de máquina em ambiente de produção, ferramentas para agendamento de tarefas e desenvolvimento de software na plataforma *Django*.

A arquitetura, as ferramentas de desenvolvimento, os resultados obtidos e sua análise serão elaborados no terceiro capítulo, enquanto que o quarto capítulo se dedicará à conclusão e trabalhos futuros.

# Capítulo 2

# Fundamentção Teórica

## 2.1 Sistemas de Aprendizado de Máquina em Ambiente de Produção

Sistemas de Aprendizado de Máquina são sistemas que combinam algoritmos e modelos matemáticos da área de Aprendizado de Máquina em soluções que visam a implantação do uso de tais algoritmos e modelos em ambientes reais. Aprendizado de máquina, por sua vez, se refere a área de estudos voltata ao desenvolvimento e compreensão de modelos matemáticos caracterizados pelo aprimoramento de seus resultados por meio da ingestão de dados de treino, de forma que esses modelos possam realizar predições e decisões sem que sejam explicitamente programados para o fazerem, como aponta Christopher Bishop [2].

Com o surgimento de equipamentos de alto poder computacional a preços acessíveis, o campo de Aprendizado de Máquina foi capaz de ser remodelado a partir de princípios e modelos que anteriormente não eram praticáveis por conta das limitações técnicas da época. Com essa mudança, o campo se tornou um foco para o surgimento de novas tecnologias que se apresentam, em grande parte, como inovações da academia e de núcleos de pesquisa. O surgimento de novos paradigmas no campo não se viu acompanhado, em mesma escala, pelo surgimento de novos sistemas e metodologias de implantação das novas tecnologias em ambientes não acadêmicos.

Enquanto que modelos tomadores de decisões dominam a área, sistemas que permitiriam a implantação de tais modelos em ambientes de produção se encontram pouco difundidos. Atualmente, os principais sistemas desse tipo se encontram hospedados em soluções prontas por provedores de infra-estrutura gerenciada, como é o caso do *Amazon SageMaker* e do *Machine Learning Studio*, soluções pagas providenciadas pela *AWS* e pela *Azure*, respectivamente. Em contrapartida, soluções abertas e gratuitas ainda dependem de um grande trabalho de desenvolvimento por parte dos responsáveis pela implantação, requerendo um esforço similar ao desenvolvido no trabalho aqui exposto.

## 2.2 Agendadores de Tarefas

.

Para a implantação de um sistema de *stream* de dados para processamento constante, é necessária uma forma para a inicialização da execução de programas e tarefas correlatas ao trabalho sendo executado. Nesse sentido, considera-se úteis ferramentas agendadoras de tarefas, ou *job shedulers*. Essas são ferramentas responsáveis pela execução agendada de programas, viabilizando metas de frequencia de processamento e execução.

### 2.2.1 Slurm

Uma dessas ferramentas é o programa conhecido por *slurm* [3], um *workload manager* capaz de realizar o agendamento de tarefas. Essa é uma das principais ferramentas de código aberto utilizada em sistemas *Unix-like*, capaz de funcionar em *clusters* computacionais de forma distribuída ou baseada em apenas um instância. O Slurm não requer modificações no *kernel* do sistema operacional, sendo relativamente auto contido por conta disso.

As três principais funções do *slurm* são a alocação de recursos de maneira exclusiva e não exclusiva a usuários por um período de tempo; prover um *framework* para iniciar, executar e monitorar tarefas paralelas em um conjunto de nós computacionais; arbitrar a contenção de recursos gerenciando filas de trabalhos pendentes. Sua

arquitetura consiste de um daemon rodando em cada nó e um daemon central no nó principal. Os daemons provém comunicação hierárquica tolerante a falhas.

Os nós da rede se dividem em partições, cada uma sendo uma fila de tarefas, cada tarefa é subdividida em subtarefas, que podem ser processadas pela partição em paralelo. Uma vez alocada a um conjunto de nós, o usuário é capaz de iniciar o trabalho em paralelo no forma de subtarefas em qualquer configuração dentro da alocação de nós.

### 2.2.2 Crontab

Outra ferramenta dessa categoria, a utilizada nesse trabalho, é o *crontab* [4]. O nome é uma abreviação de *cron table*, pois se refere ao arquivo *cron file* utilizado internamente para agendamento e execução de tarefas. Esse programa consiste de um arquivo no sistema operacional que periodicamente é lido pelo *cron* daemon, fazendo uso de expressões *CRON*, e suas linhas são interpretadas como entradas de uma tabela contendo em uma coluna a expressão que define sua agenda de execução e em outra coluna o comando a ser executado como um programa de linha de comando. Seu diferencial está no fato de ser extremamente simples e possuir suporte para manipulação por meio de diversas linguagens de programação, incluindo *Python 3*, a linguagem escolhida para o desenvolvimento do sistema aqui detalhado.

## 2.3 Desenvolvimento com Django

O *Django* [5] é um *framework* de desenvolvimento de aplicações e serviços *web* disponível nas linguagens de programação *Python 2 e 3*, gratuito e de código aberto. Sua estrutura interna é baseada no paradigma MTV, particularmente caracterizada por uma forte e nativa integração com bancos de dados a partir de uma interface própria. Todos os componentes de uma aplicação *web* são reproduzidos por meio de convenções próprias ao *framework*, baseados na reusabilidade de código e orientado à integração nativa com bancos de dados diversos.

O desenvolvimento com o *framework* é marcado pela pouca quantidade de código produzido e pelas diversas interfaces de programação providas pela plataforma.

Também se destacam a a integração nativa com um sistema de administração embutido nas aplicações produzidas que fornecem, sem custos adicionais de desenvolvimento, uma aplicação e interface de autenticação de usuários também utilizável em todas as camadas de serviço contruídas paralelamente no mesmo ambiente.

O *framework* disponibiliza ao desenvolvedor interfaces próprias para o acesso a bancos de dados, templates para renderização de HTML, classes extensíveis para programação orientada a objetos, encaminhamento de requisições configurável, dentre outras facilidades.

## 2.4 Amazon AWS

A *Amazon Web Services*, ou AWS, é a maior provedora de infraestrutura em nuvem do mercado. A empresa provê serviços para websites, aplicações cliente servidor, banco de dados, análise de dados, redes, dispositivos móveis, ferramentas de gerenciamento, armazenamento de dados, aluguel de servidores entre outros [6]. Seus serviços podem ser acessados em seu portal *web* e por meio de suas *API*s e *SDK*s.

A Amazon está disponível em diversas localidades, operando *datacenters* ao redor do mundo. Cada local é composto por uma região e zonas de disponibilidade, onde uma região é uma zona geográfica totalmente independente das demais, contendo diversas zonas de disponibilidade também isoladas entre si. A AWS adota, para a maioria de seus serviços, o plano de pagamento *pay-as-you-go*, fazendo necessário apenas o pagamento pelos recusrsos consumidos.

# Capítulo 3

# Proposta de Plataforma

## 3.1 Arquitetura

O projeto desenvolvido se trata de uma aplicação *Django* que conta com três interfaces, uma para execução de um modelo preditivo em ocorrências controladas, outra para monitoramento do banco de dados e outra para o disparo escalonado de predições, chamadas, respectivamente, de *Showroom*, *Monitor* e *Disparo*. Essas três interfaces são controladas pela aplicação por meio de três classes, vistas na figura 3.1. Os componentes da aplicação podem ser visualizados na figura 3.2, onde se observa a existência de um arquivo *config.yaml*, responsável por agregar todos os parâmetros e configurações da aplicação.

## 3.2 Implementação

No modelo seguido pelo *framework Django*, o modelo *MTV*, todas as páginas e interfaces da aplicação são renderizadas a partir de templates HTML preenchidos pelo controlador da aplicação, que no caso se trata do arquivo *views.py*. Os templates HTML se encontram dentro do diretório *templates*. Os modelos se encontram no arquivo models.py, fazendo uso direto das classes de modelo do *framework*. As componentes *monitor.py*, *showroom.py* e *dispatch.py* se encontram no diretório *scripts*.

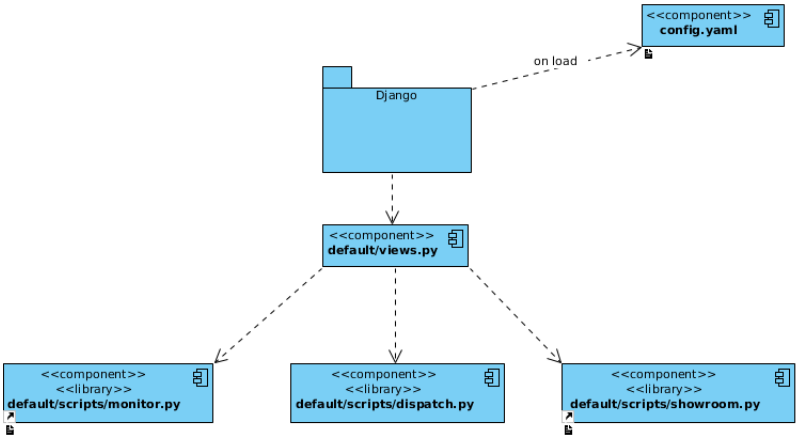


Figura 3.1: Diagrama de componentes da aplicação.

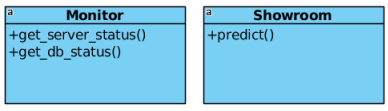


Figura 3.2: Diagrama de classes da aplicação.

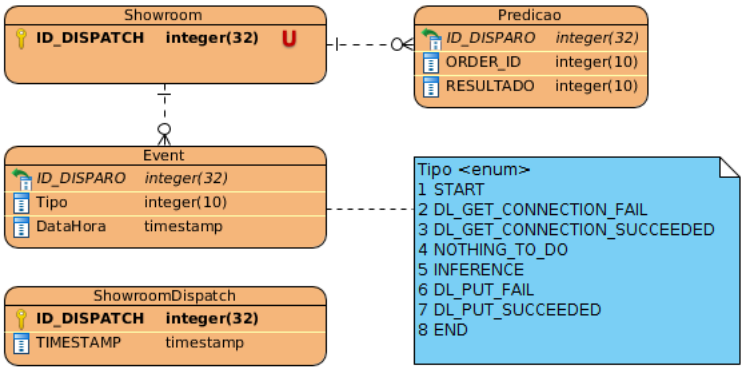


Figura 3.3: Diagrama Entidade-Relacionamento do banco.

### 3.2.1 Configuração

Toda a configuração da aplicação é feita a partir de dois arquivos, o *settings.py* e o *config.yaml*. No primeiro são configuradas as propriedades básicas de uma aplicação *Django*, se tratando de seu arquivo de configuração padrão. No segundo, estão as configurações específicas à aplicação. O caminho ao arquivo *config.yaml* é escrito como a variável *CONFIG_FILE*, dentro do arquivo *settings.py*, para que as configurações possam ser carregadas na aplicação durante sua inicialização.

O arquivo *config.yaml* contém todos os parâmetros necessários para o funcionamento da aplicação. DETALHAR TODOS NUMA TABELA?

### 3.2.2 Showroom

A *view Showroom* é caracterizada pela existência de casos sobre os quais serão inferidas classificações a partir do modelo de aprendizado de máquina carregado na aplicação. Toda execução gerada a partir dessa *view* segue os passos vistos no diagrama de sequência apresentado na figura 3.4. Os dados para a predição são coletados, o evento é registrado na tabela de log de eventos, o resultado é computado e a *view* é atualizada com os resultados da predição. A tabela de log de enventos, por sua vez, é populada por eventos relacionados a cada execução, representados pelas colunas da entidade *ShowroomDispatch* na figura 3.3.

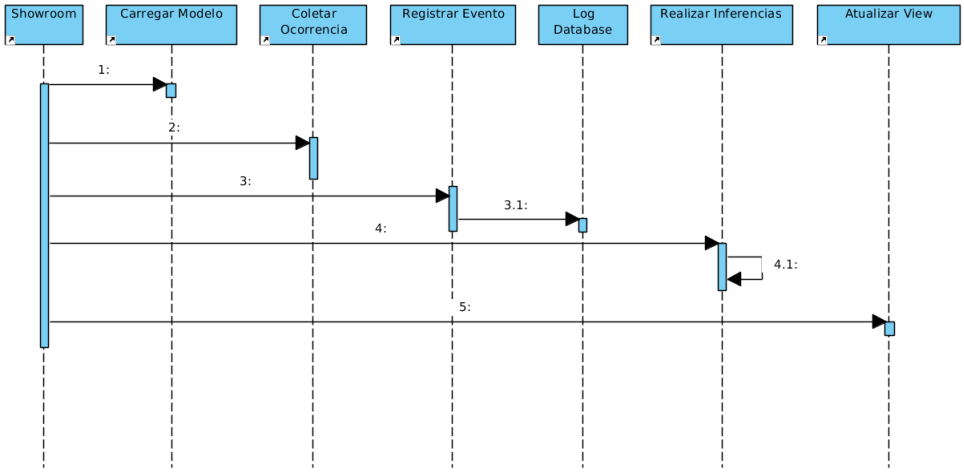


Figura 3.4: Diagrama de sequência da *view Showroom*.

### 3.2.3 Disparo

A *view Disparo* é responsável por permitir a visualização dos disparos executados, onde cada disparo é dado pela execução do script de inferência.

### 3.2.4 Monitor

A *view Monitor* é responsável por monitorar: o estado de um *host*, o estado do banco de dados, as datas dos último e próximo disparos e o estado do processo a ser executado. Em intervalos periódicos de tempo a *view* é atualizada trazendo informações recentes para a interface.

Para a verificação do estado do *host*, uma conexão socket é criada em uma porta aberta qualquer do host. A conexão é tida como *offline* se ocorrer um timeout, sendo 10 segundos o intervalo de tempo para que isso ocorra. O host informado pode ser tanto como um endereço IPV4 quanto um endereço IPV6.

A verificação do estado do banco de dados é feita por meio de uma conexão, usando a cadeia de conexão adequada ao banco a ser conectado. Após a conexão ser estabelecida, é realizada uma consulta simples em uma tabela qualquer pertencente ao banco, sem que nada seja retornado, com o intuito de que todos os passos da


Figura 3.5: Diagrama de implantação

conexão sejam testados. Caso ocorra algum problema em algum momento durante a tentativa de conexão, esse problema é reportado.

O último disparo e o estado do próximo disparo são monitorados fazendo uso da biblioteca *crontab* para *Python 3*. Tanto a agenda de execuções quanto o comando a ser executado pelo *daemon cron* são obtidos por meio do uso dessa biblioteca.Nessa *view*, o estado da execução do comando pode ser alterado por meio de um botão que remove o comando do arquivo *crontab* e outro que o insere.

## 3.3 Implantação

A implantação do sistema depende de um servidor de aplicação e um servidor para o banco de dados, como visto na figura 3.5. A aplicação *Django* será hospedada no servidor de aplicação enquanto que o banco de dados será hospedado em uma máquina que permita o acesso da máquina da aplicação ao serviço de banco de dados hospedado.

# Capítulo 4

# Conclusoes

## 4.1 TODO

# Bibliografia


[1] THE STACK OVERFLOW NETWORK, "Stack Overflow Annual Developer Survey", https://insights.stackoverflow.com/survey, 2018, [Data File].

[2] BISHOP, C. M., "Pattern Recognition and Machine Learning", *Springer*, v. ISBN 978-0-387-31073-2, 2006.

[3] "Slurm Workload Manager", https://slurm.schedmd.com/, 2019, [Software].

[4] "crontab - schedule periodic background work - Commands and Utilities Reference", https://pubs.opengroup.org/onlinepubs/9699919799/utilities/crontab.html, 2019, [Software].

[5] "Django", https://www.djangoproject.com/, 2018, [Software].

[6] "Amazon Web Services", https://aws.amazon.com/pt/products/, 2019.

# Apêndice A

# O que é um apêndice

Elemento que consiste em um texto ou documento elaborado pelo autor, com o intuito de complementar sua argumentação, sem prejuízo do trabalho. São identificados por letras maiúsculas consecutivas e pelos respectivos títulos.

# Apêndice B

# Encadernação do Projeto de Graduação

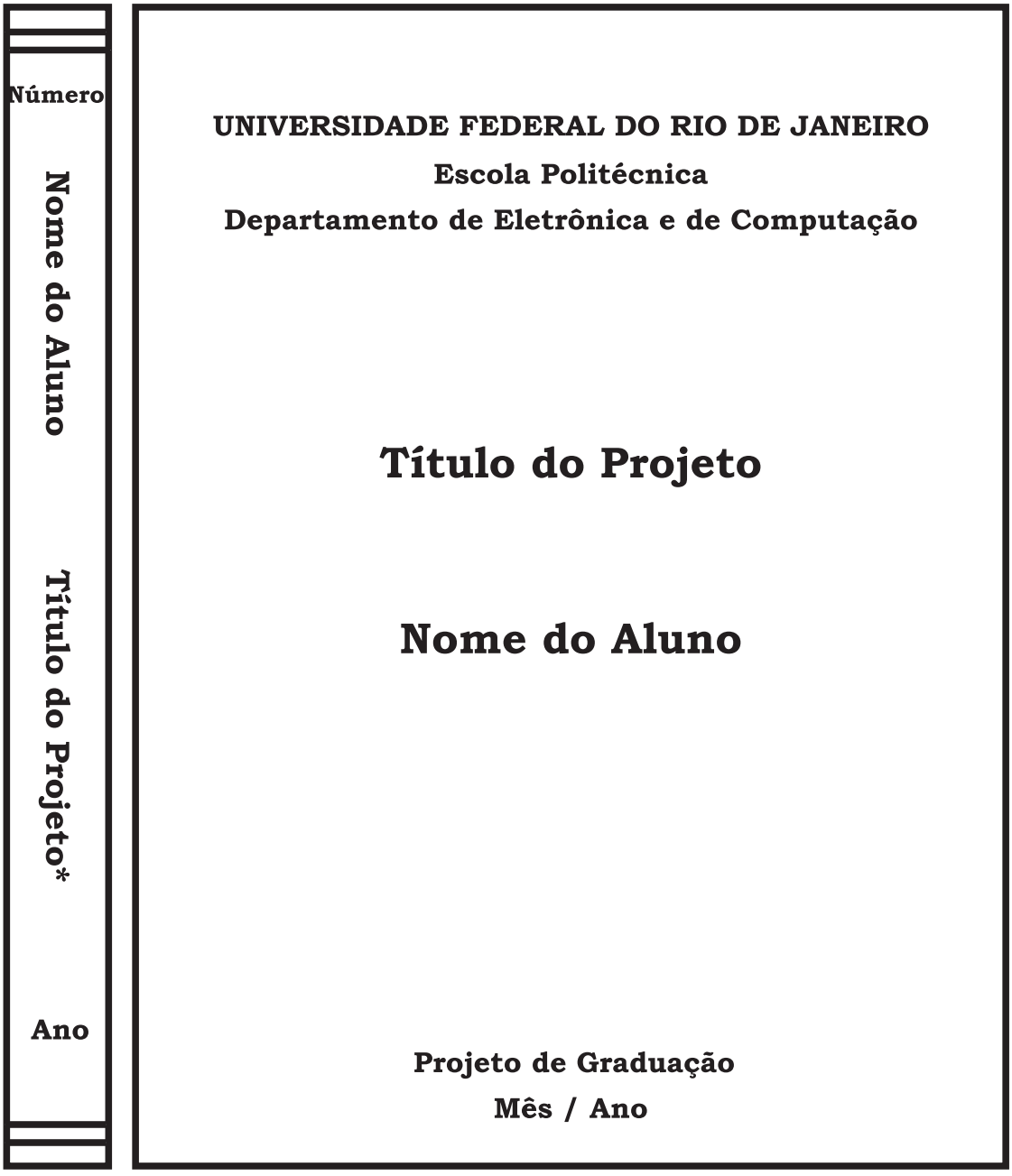


Figura B.1: Encadernação do projeto de graduação.

# Apêndice C

# O que é um anexo

Documentação não elaborada pelo autor, ou elaborada pelo autor mas constituindo parte de outro projeto.