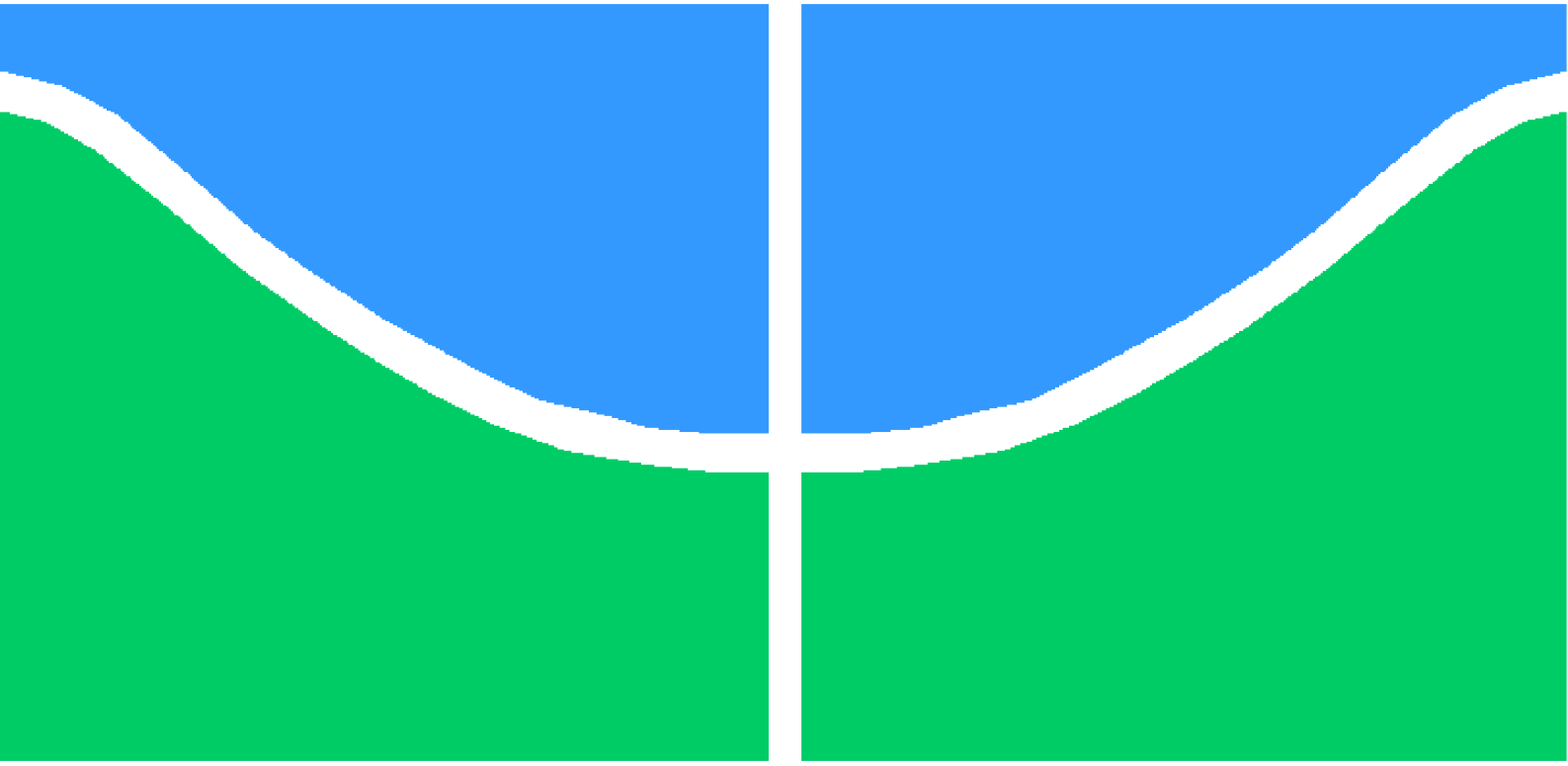

Universidade de Brasília – UnB
Faculdade UnB Gama – FGA
Engenharia de Software

Plataforma Web de Análise de Mercado de Jogos para Desenvolvedoras

Autor: João Paulo Busche da Cruz
Orientador: Prof. Matheus de Sousa Faria

Brasília, DF
2018

João Paulo Busche da Cruz

Plataforma Web de Análise de Mercado de Jogos para Desenvolvedoras

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Prof. Matheus de Sousa Faria

Coorientador: Prof. Dr. Edson Alves da Costa Júnior

Brasília, DF

2018

João Paulo Busche da Cruz
Plataforma Web de Análise de Mercado de Jogos para Desenvolvedoras/ João Paulo Busche da Cruz. – Brasília, DF, 2018-
55 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Matheus de Sousa Faria

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA , 2018.

1. Mercado de Jogos. 2. Análise de Dados. I. Prof. Matheus de Sousa Faria. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Plataforma Web de Análise de Mercado de Jogos para Desenvolvedoras

CDU 02:141:005.6

João Paulo Busche da Cruz

Plataforma Web de Análise de Mercado de Jogos para Desenvolvedoras

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Prof. Matheus de Sousa Faria
Orientador

Prof. Dr. Edson Alves da Costa Júnior
Convidado 1

Prof. Dra. Carla Silva Rocha Aguiar
Convidado 2

Brasília, DF
2018

Agradecimentos

A Deus por me dar conhecimento e oportunidades para realizar meus sonhos, e por me dar força para superar as dificuldades.

Aos meus pais por sempre me apoiarem, idenpendente do quão insano fosse meu objetivo, e por sempre estarem ao meu lado nos momentos de dificuldades.

Ao meu orientador Matheus Faria, que sempre entendeu minha dificuldades e sempre se mostrou disponível para qualquer dificuldade que eu enfrentei no decorrer do trabalho.

Ao meu coorientador Edson Alves por suas correções e incentivos.

E a todos que contribuiram diretamente ou indiretamente para a minha formação, o meu muito obrigado.

Resumo

O mercado de jogos nos dias atuais é um dos mercados mais lucrativos e também um dos que mais cresce nos decorrer dos anos. Tendo um aumento considerável no número de desenvolvedoras *indies*, houve um aumento no número de desenvolvedoras que buscam por métricas de jogos, porém a extração de informações destas métricas é um trabalho que requer muitos recursos humanos/tempo. Por isso uma plataforma web que disponibilize estas métricas e *insights* poderia beneficiar o desenvolvimento de jogos. Para o desenvolvimento desta plataforma foram feitas as seguintes atividades: definição dos requisitos e desenvolvimento dos protótipos do software de extração e da plataforma web. O protótipo do software de extração é capaz de extrair os dados de suas fontes, manipulá-los e inseri-los no Elasticsearch.

Palavras-chaves: *game analytics*, *business intelligence*, métricas de jogos.

Abstract

The gaming market is one of most profitable and one with most growing over the years. Having a considerable increase in the number of indie developers, there has been an increase in the number of developers looking for game metrics, however extracting information from these metrics is a work that requires a lot of human/time resources. For that a web platform that provides metrics and insights, could benefit the development of games. or the development of this platform the following activities were done: definition of requirements and development of prototypes of extraction software and web platform. The extraction software prototype is capable of extracting data from its sources, manipulating it, and inserting it into ElasticSearch.

Key-words: game analytics, business intelligence, game metrics.

Lista de ilustrações

Lista de tabelas

Lista de Algoritmos

Lista de abreviaturas e siglas

BI	*Business Intelligence*
GUR	*Game User Research*
API	*Application Programming Interface*
XML	*Extensible Markup Language*
HTTP	*Hypertext Transfer Protocol*
URI	*Uniform Resource Identifier*
XP	*Extreme Programming*
MVC	*Model-View-Controller*
REST	*Representational State Transfer*
TPS	Sistema Toyota de Produção
TCC	Trabalho de Conclusão de Curso

Sumário

1 Introdução

A indústria de jogos é uma das áreas de mercado mais lucrativa atualmente. No mercado brasileiro isto não seria diferente: sendo apenas o 13º maior mercado, no ano de 2017 movimentou 1,3 bilhões de dólares de acordo com o Newzoo (NEWZOO, 2017a), empresa que estuda o mercado de jogos. Um valor pequeno se comparado ao mercado chinês, o maior mercado de jogos do mundo, o qual movimentou 27,5 bilhões de dólares no ano de 2017 (NEWZOO, 2017b).

Sendo um mercado bastante lucrativo, houve um grande aumento no número de empresas desenvolvedoras de jogos. No mercado brasileiro, houve um aumento de 600 % entre 2008 e 2016 no número de desenvolvedoras (SILVEIRA, 2016). Este aumento não se limita apenas ao mercado brasileiro: de acordo com o site Steamspy (GALYONKIN, 2018), plataforma web que exibi dados de jogos, apenas no ano de 2017 foram lançados 7.672 jogos na plataforma Steam, plataforma onde desenvolvedoras disponibilizam seus jogos, uma média de 21 jogos por dia. Porém com este grande crescimento de concorrência fica cada vez mais difícil desenvolver um jogo que seja aceito pela comunidades de jogadores, assim, os jogadores tem muitas opções na hora comprar, o que faz que com algumas *features* se tornem diferencias, como multyplayer, localização, entre outros.

Levando em conta a dificuldade de se criar um jogo nos tempos atuais, muitas desenvolvedoras *indies* buscam por métricas que auxiliem na hora do desenvolvimento. Estas métricas, apesar de estarem disponíveis no mercado, necessitam um grande numero de recursos humanos/tempo para extrair informações que agreguem valor ao desenvolvimento. Desenvolvedoras *indies*, geralmente por falta deste tipo de recursos, muitas vezes pagam para outras empresas fazerem essas análises ou desenvolvem jogos sem as análises.

Por isso o objetivo deste trabalho e a criação de uma plataforma web onde será disponibilizada, de uma forma mais simples, diferentes métricas e agregações de dados, que permite o usuário moldar a necessidade dele. Incluindo *insights* sobre as métricas.

1.1 Objetivos

O objetivo deste trabalho è desenvolver um banco de dados inicial, com os dados de diversas fontes, para que futuramente possa ser criado as métricas de negócio.

Os objetivos espécificos são:

- elaborar uma arquitetura responsável por fazer a extração, manipulação dos dados dos jogos;

- elaborar uma rotina para a extração constantes dos dados;
- desenvolver um software responsável por extrair, manipular e inserir os dados dos jogos num banco de dados.

1.2 Estrutura do Documento

Este documento será dividido em 5 capítulos, sendo o primeiro a introdução. O Capítulo 2 é responsável pelo referencial teórico do projeto e análise de concorrentes. O Capítulo 3 é responsável pela metodologia, mostrando como será feito o projeto. O Capítulo 4 é responsável pelos resultados obtidos no projeto. O Capítulo 5 é responsável pela conclusão do projeto, onde também será mostrados possíveis trabalhos futuros.

2 Fundamentação Teórica

Neste capitulo será abordada a fundamentação teórica para o entendimento do propósito da implementação do projeto. Nele são explicados os conceitos de *application programming interface*, *business intelligence* e *game analytics* e também serão abordados softwares parecidos com o que será desenvolvido.

2.1 *Application Programming Interface*

A sigla API (*Application Programming Interface*) é um conjunto de rotinas e padrões de programação para acesso a um aplicativo de software ou plataforma baseado na Web. A utilização destas rotinas servem principalmente para que softwares externos tenham acesso a funcionalidade de um produto, sem que esse tenha que envolver-se em detalhes da implementação (RETHANS, 2016).

Para a utilização de APIs Webs existem várias arquiteturas que podem ser utilizadas a mais comum e a arquitetura REST (*Representational State Transfer*), que define um conjunto de restrições e propriedades baseadas no HTTP (*Hypertext Transfer Protocol*).

2.1.1 *Representational State Transfer*

A sigla REST, originalmente, se referia a um conjunto de princípios de arquitetura, nos tempos atuais é utilizada no sentido mais amplo, descrevendo qualquer interface web que use XML (*Extensible Markup Language*) e HTTP, sem as abstrações adicionais dos protocolos baseados em padrões de trocas de mensagem. De acordo com Fielding (FIELDING, 2000) é possível projetar um sistema de serviços web com a arquitetura REST, completando que a própria *World Wide Web*, utilizou REST como base para o desenvolvimento do protocolo HTTP.

A arquitetura REST possui quatro princípios principais, sendo eles:

- **Protocolo cliente/servidor sem estado**: cada mensagem HTTP contém toda a informação necessária para compreender o pedido.
- **Operações bem definidas**: que se aplicam a todos os recursos de informações, as mais importantes são ***POST***, ***GET***, ***PUT*** e ***DELETE***.
- **Sintaxe universal**: para identificar os recursos, na arquitetura REST este recurso é direcionado pela sua URI(*Uniform Resource Identifier*).

- **Hipermídia**: para a informação da aplicação, como para as transições de estado da aplicação, normalmente são representados pelo HTTP e XML

2.2 *Business Intelligence*

No decorrer dos anos houve uma grande mudança em relação a criação, coleta e do uso de dados. Enquanto houve uma grande evolução na maneira que esses dados eram gerenciados, sempre houve um desejo de extrair valores de negócios nas grandes pilhas de dados, que são capturadas hoje em diversas fontes, sejam elas estruturas de dados ou arquivos.

Os resultados da análise destas pilhas de dados e da criação de conhecimentos sobre esta análise, criou-se uma vantagem no mercado de negócios inimagináveis. Com esse conhecimento foi possível criar políticas de escolhas de ações para negócios em diferentes cenários. A exposição e exploração destes conhecimento é apenas uma das vantagens do uso de BI (*Business Intelligence*).

Uma das maiores vantagens do uso do BI está na otimização do processo de tomadas de decisão, pois para cada processo de negócios é associado a sua performance, e num mundo perfeito cada escolha deve ser a mais otimizada, ou seja, a que têm a melhor performance (LOSHIN, 2012).

A utilização de BI no âmbito de trabalho apresenta uma evolução na performance nas seguintes dimensões de negócio:

- No valor financeiro associado ao crescimento da lucratividade, sejam elas derivadas de custos ou do aumento de receitas.
- No valor de produtividade associado a diminuição da carga de trabalho, diminuição do tempo necessário para a execução de processos ponta-a-ponta e no aumento da porcentagem de produtos de alta qualidade.
- No valor de confiança, como maior satisfação do cliente, funcionário ou fornecedor, assim como aumento na confiança de previsões, consistência operacional e relatórios gerenciais, reduções no tempo gasto com "paralisia de análise" e melhor resultados de decisões.
- No valor de risco associado com a uma melhor visibilidade da exposição do crédito, confiança no investimentos em capitais e conformidade auditável com a jurisdição e normas e regulamentos da indústria.

The Data Warehousing Institute (ECKERSON, 2002), uma instituição especializada na educação e treinamento em armazenamento de dados, define que BI é os processos, as tec-

nologias e as ferramentas necessárias para transformar dados em informação, informação em conhecimento e conhecimento em planos que dirigem rentáveis planos de negócios, ou seja BI engloba armazenamento de dados, ferramentas de análiticas e gestao de informação/conhecimento.

Business intelligence geralmente considerar as informações dos dados como ativos, por isso, pode-se valer a pena examinar o uso de informações no contexto de como o valor é criado dentro de uma organização. Para isso existem três tipos diferentes de perspectivas, sendo elas:

- **Perspectiva funcional**: Neste tipo de perspectiva, os processos focam nas tarefas relacionadas a algum tipo particular de negócio, como vendas, *marketing*, entre outros. Processos funcionais confiam nos dados que operam dentros dos padrões de atividades comerciais.
- **Perspectiva interfuncional**: Como a maioria das empresas funcionam como um aglomerado de processos funcionais, e isto reflete em informações mais complexas. Para esta perspectiva a atividade foi um sucesso quando todas as tarefas foram completadas. Pela sua própria natureza, os processos envolvidos compartilham informações em diferentes funções, e o sucesso é medido tanto em termos de conclusão bem-sucedida, bem como as características do desempenho geral
- **Perspectiva empresarial**: Num ponto de vista organizacional e observando as características de desempenho dos processos interfuncionais, pode-se informar arquitetos empresariais e analistas de negócios maneiras na qual a organização pode mudar e melhorar o jeito que as coisas são feitas. Neste ponto de vista, o dado não é mais usado apenas para executar negócios, dados são utilizados para melhorar os negócios

2.3 *Game Analytics*

O desenvolvimento de jogos hoje pode se mostrar como um grande desafio, e grande parte deste desafio se dá pelo fato do grande número de jogos publicados. Para auxiliar as desenvolvedoras a criarem jogos eficientemente foram criados várias ferramentas e técnicas, um destes métodos é o *analytics*.

Analytics é o processo de descobrir e comunicar padrões em dados, solucionando problemas de negócios ou suportar decisões de gerenciamento de empresas. Está metodologia possui seus fundamentos em mineração de dados, na matemática, estatística, programação e operações de busca, como também na visualização dos dados, de forma a comunicar padrões relevantes. Vale mencionar que o *analytics* não é apenas perguntar e relatar dados de BI, e sim análises atual daqueles dados (DAVEPORT; HARRIS, 2007).

Game analytics é uma aplicação do *analytics* para o contexto de desenvolvimento de jogos (ANDERS; EL-NASR; CANOSSA, 2013). Um dos maiores benefícios em utilizar o *game analytics* é o suporte na hora de fazer decisões em todos os níveis e áreas organizacionais. Este método é direcionado tanto como a análise de um jogo com um produto, tanto como a análise de um jogo como projeto.

A aplicação padrão do *game analytics* é na hora de informar o GUR (*Game User Research*). GUR é a aplicação de várias técnicas e metodologias para avaliar a maneira na qual os jogadores jogam, e o nível de interação entre o jogador e o jogo. Vale mencionar que *game analytics* não é só GUR, já que o GUR é focado nos dados obtidos a partir dos usuários, já o *game analytics* considera todos os tipo de dados obtidos no desenvolvimento do jogo.

2.3.1 Telemetria

Telemetria são os dados obtidos à distância, geralmente digitais, porém qualquer dado transmitido à distância e telemetria. No contexto de jogos, telemetria seria algum jogo transmitindo dados sobre a interação do jogador com o jogo.

Telemetria de jogos é o termo utilizado para qualquer dado obtidos a distância que pertence durante o desenvolvimento ou evolução de um jogo, e isto inclui o monitoramento e análise de: servidores, dispositivos celulares e comportamento dos usuários. A fonte que produz mais dados por telemetrias, são os de usuário, por exemplo, interação com jogos, comportamento de compra e interações com outros jogadores ou aplicativos (BOHANNON, 2010).

2.3.2 *Game Metrics*

Em sua forma pura, os dados obtidos a partir da telemetria, não são de muito auxílio, por isso estes dados devem ser transformado em várias métricas interpretativas, como: o número de jogadores ativos por dia, bugs arrumados por semana, entre outros. Essas métricas são chamadas de game metrics. *Game metrics* possuem os mesmo potencial que outras fontes de BI. *Game metrics* geralmente são definidas como um medição quantitativa de um ou mais atributos, de um ou mais objetos que operem no contexto de um jogo.

Métricas podem ser variáveis ou agregações mais complexas, como a soma de várias variáveis, em outras palavras as métricas podem ser simples variáveis que geram uma análise básica, ou a combinação de várias variáveis para gerar uma análise mais complexa e completa. Métricas que não estão relacionadas diretamente ao jogo, são chamadas de métricas de negócios. Durante a utilização do *game analytics* é essencial a distinção entre as métricas de negócio e as *game metrics*.

As game metrics foram categorizadas em três tipos por Mellon (MELLON, 2009), ou seja, as *game metrics* podem ser definidas como:

- **Métricas de usuário**: São métricas relacionadas aos usuários que jogam aquele jogo, pela perspectiva de jogadores, ou de clientes. A perspectiva de cliente é utilizada quando as métricas são relacionadas a receita. A perspectiva de jogador é utilizada para investigar como é a interação das pessoas com o sistema do jogo e seus componentes.

- **Métricas de performance**: São métricas relacionadas a performance da tecnologia e arquitetura utilizada no jogo, muito relevantes para jogos onlines. Essas métricas geralmente são utilizadas no monitoramento dos impactos causado por alguma atualização no jogo.

- **Métricas de processo**: São métricas relacionadas ao processo de desenvolvimento de jogos. Similares as métricas de performance, são utilizadas para gerenciar e monitorar métodos que foram adotados, ou que foram adotados na hora do desenvolvimento do jogo.

2.4 Softwares Correlatos

Nesta parte de documento é levantado os softwares correlacionados, que possuem características ou objetivos parecidos com a plataforma a ser desenvolvido. Os principais são o Steam DB, uma ferramenta que disponibiliza informações sobre o banco de dados da Steam; a Steam Spy, uma plataforma web que disponibiliza informações sobre os jogos e suas vendas por região e o DFC Intelligence, uma ferramenta de pesquisa sobre o mercado de jogos.

2.4.1 Steam DB

Steam DB é uma ferramenta *open source third-party* com objetivo de dar um melhor conhecimento sobre os jogos e suas atualizações disponíveis no banco de dados da Steam (STEAMDB, 2010). Está ferramenta disponibiliza *rankings* e gráficos de jogos para que o usuário tenha uma melhor visualização destes. Atualmente o Steam DB apresenta métricas individuais de cada jogo, não oferencendo nenhuma maneira de agregação. Na tabela 1 podemos ver as vantagens e desvantagens do Steam DB.

Vantagens	**Desvantagens**
Open source	Não possui política de contribuição
Rankings e gráficos	Informações individuais de um jogo
Gratuito	Precisa de login na Steam
Informações individuais de um usuário	

Tabela 1 – Vantagens e Desvantagens Steam DB

2.4.2 Steam Spy

Steam Spy é uma plataforma Web que a partir de informações sobre os usuários da Steam, disponibiliza informações como o número de donos de algum jogo ou vendas por região de determinado jogo (GALYONKIN, 2018). Um dos objetivos especificados pelo Steam Spy é o auxílio a desenvolvedores *indies*, porém para se conseguir todos os dados disponíveis pela plataforma é preciso pagar por eles. Atualmente a Steam Spy para que o usuário possua total acesso aos dados dela, o usuário precisa pagar. Na tabela 2 podemos ver as vantagens e desvantagens do Steam Spy.

Vantagens	**Desvantagens**
Disponibiliza número de donos e vendas por região	Não é totalmente gratuito
Gráficos genéricos e confusos	Não apresenta métricas sobre os jogos
Disponibiliza API para seus dados	Não é *open source*

Tabela 2 – Vantagens e Desvantagens Steam Spy

2.4.3 DFC Intelligence

DFC Intelligence é uma ferramenta paga que auxiliam desenvolvedoras a tomar conhecimentos sobre estatísticas de seus jogos no mercado. Desenvolvedoras que utilizem essa ferramenta receberão informações sobre o número de vendas de seus jogos, picos de vendas, regiões que mais venderam, entre outras. Também será disponibilizado gráficos e métricas que informam como está o mercado de jogos (COLE, 1994). Atualmente a ferramenta DFC Intelligence é focada apenas nos jogos da desenvolvedora, que contratou seus serviços. Na tabela 3 podemos ver as vantagens e desvantagens do Steam DB.

Vantagens	**Desvantagens**
Disponibiliza informações sobre o mercado de jogos	Não é gratuita
Auxiliam desenvolvedores na hora da criação de um jogo	Não mostrar o mercado de jogos como um todo
	Não é *open source*
	Foca apenas nos jogos de uma desenvolvedora

Tabela 3 – Vantagens e Desvantagens DFC Intelligence

3 Metodologia

A metodologia escolhida para o desenvolvimento da plataforma será a metodologia ágil, a metodologia principal que vai será utilizada é o Kanban, porém, não será utilizada sua forma pura, e sim com algumas modificações, que atendem as necessidades do projeto. A escolha dessa metodologia se dá pelas seguintes características: o projeto será desenvolvido de maneira incremental, ou seja, ele poderá ser modificado no decorrer da implementação; a equipe consiste em apenas uma pessoa, o que descarta a possibilidade de utilizar o Scrum ou XP (*Extreme Programming*); o escopo do projeto será divididos em tarefas, e a utilização do Kanban facilita no descobrimento de gargalos.

A metodologia Kanban surgiu no Japão com o TPS (Sistema Toyota de Produção) (OHNO, 1997) para controlar a fabricação de automóveis e foi inserida no meio de desenvolvimento de software no ano de 2007. Kanban é um termo japonês para sinal visual e uma das grandes características dessa metodologia é evidenciar os problemas existentes no processo.

A metodologia ágil surgiu no ano de 2001, com a reunião de especialistas em processos de desenvolvimento de software para discutir maneiras de melhorar o desempenho em projetos, com isso foi criado o Manifesto Ágil (BECK et al., 2001). Uma das características das metodologias ágeis são sua capacidade de adaptar a novos fatores durante o desenvolvimento do projeto, ao invés de tentar prever o que pode acontecer e o que não pode

3.1 Análise de Ferramentas

Nesta seção são feito estudos sobre as ferramentas que serão utilizadas no desenvolvimento do projeto.

3.1.1 Banco de Dados

A ferramenta de banco de dados é responsável pelo armazenamento dos dados extraídos das fontes de dados.

Elasticsearch

Elasticsearch é uma ferramenta *open source*, desenvolvida pela Elastic[1], de análise e busca REST capaz de resolver um grande número de casos. É a parte principal de Elastic Stack, servindo como um centro de armazenamento de dados (ELASTIC, 2010a).

[1] <https://www.elastic.co/>

Elasticsearch suporta qualquer tipo de dado, além de agregar grande quantidades de dados para se ter uma visão melhor. Entre suas características as que mais se destacam são sua rapidez de busca, capacidade de detecção de falhas, múltiplos tipos de dados e suporte a múltiplas linguagens de programação.

3.1.2 Frequência na Extração dos Dados

Esta ferramente será responsável pela criação de uma rotina na hora de extrair os dados e atualizar o banco de dados.

Celery

Celery é uma ferramenta *open source* focada em operações em tempo real numa fila de tarefas assincronas baseadas na passagem de mensagens distribuidas, tambem oferece suporte a operações de agendamento (CELERY, 2007).

Celery possui funções que auxiliam a criação de rotinas, funcionando principalmente com Python[2]. Como Celery trabalha com a utilização de tarefas, podemos agendar seus funcionamentos utilizando *cronjobs*[3] para suas frequências.

3.1.3 Visualização dos Dados

Esta ferramenta será responsável por demonstrar os dados do banco de uma maneira mais intuitiva.

Kibana

Kibana é um ferramenta *open source*, desenvolvida pela Elastic, que permite a visualização dos dados guardados no Elasticsearch, possuindo diversas maneiras diferentes de disponibilização visual dos dados. É a parte de visualização do Elastic Stack, servindo como um centro de monitoramento (ELASTIC, 2010b).

3.1.4 Criação de Containers

Esta ferramente é responsável por dividir as partes da arquitetura em diferentes containers.

Docker

Docker é uma ferramenta *open source* que proveem containers de *softwares* para o auxilios de desenvolvedores. Outras funcionalidades que são disponibilizadas são a integra-

[2] <https://www.python.org/>
[3] <https://cron-job.org/en/>

ção com o desenvolvimento, suporte a multiplataformas e acesso a uma extensa biblioteca de servidores (DOCKER, 2014).

3.2 Arquitetura do Projeto

A arquitetura do projeto é dividida em quatro partes: as fontes de dados, o extractor, o banco de indexação e a API RESTful. A ligação entre as partes e sua posição na arquitetura ficam evidenciado na figura 1.

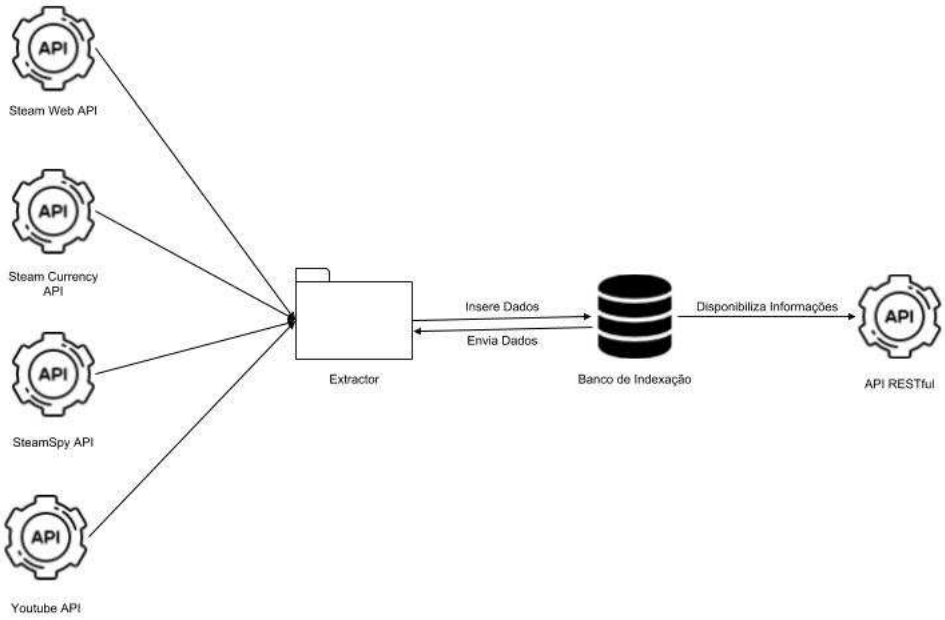

Figura 1 – Arquitura Geral do Projeto

Fonte de Dados são locais onde se encontrar os dados que serão extraidos, manipulados e inseridos no banco de indexação, estes dados poderão vir de quaisquer fonte, sejam elas um banco de dados, de *crawlers*, de APIs ou de arquivos locais.

Banco de Indexação é um banco de dados responsável por guardar os dados manipulados pelo Extractor. Como será utilizado o banco de jogos da Steam, o banco de indexação precisa suportar um grande número de dados e rapidez na atualização dos dados e na suas buscas.

API RESTful é a parte responsável por disponibilizar os dados do banco de indexaçõa, sem que o usuário precisa conectar diretamente a ele. Como será desenvolvida um API, outros e quaisquers projetos poderão usufruir dela.

3.2.1 Arquitetura Extractor

A arquitetura do Extractor e composta de três classes principais e de dois arquivos. Esta arquitetura está representada na figura 2.

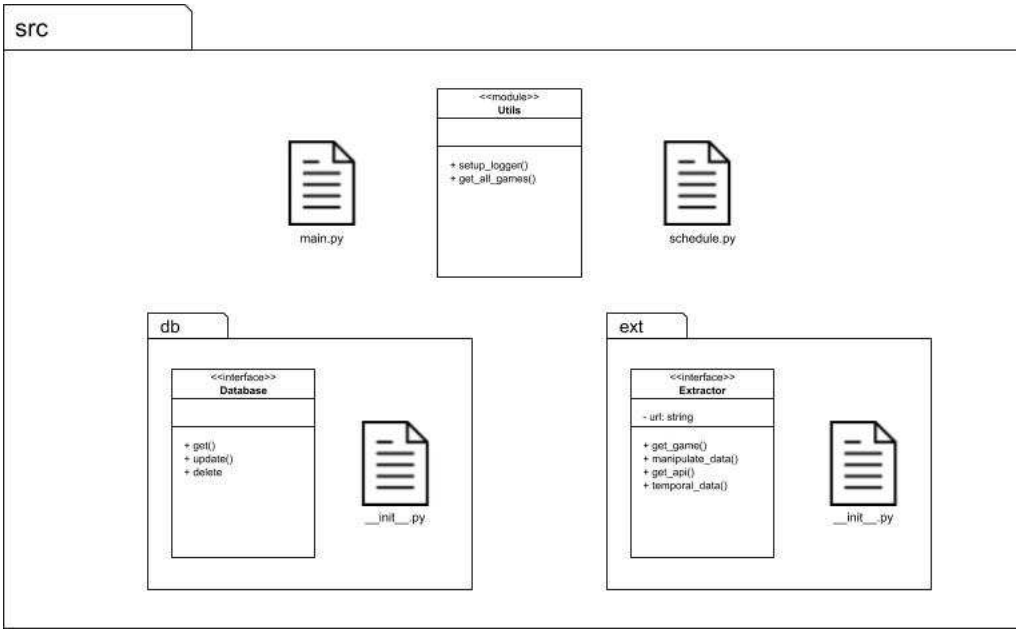

Figura 2 – Arquitura do Extractor

A interface ***Extractor*** deverá ser herdada por todos os plugins. Esta interface possui funções responsáveis por fazer requisições na API, extrair dados desta API e manipular estes dados, sejam eles estáticos ou temporais.

A interface ***Database*** deverá ser herdada por todos os bancos de dados que possam a ser utilizado. Esta interface possui funções responsáveis por inserir/atualizar, delete e mostrar os dados guardados pelo banco.

O módulo ***Utils*** é responsável por implementar funções que não se encaixam em nenhuma classe do software. A princípio este módulo possui funções responsaveis por configurar os logs e extrair todos os jogos que possuem.

Os arquivos ***Main*** e ***Schedule*** são resposáveis por, respectivamente, fazer a primeira inserção no banco de dados e por manter um rotina de atualizações nos dados dos jogos.

3.3 Extractor

O Extractor é responsável pela extração, manipulação das informações das fontes de dados e também por inserir e atualizar o banco de dados. Como o Extractor lidará

com dados que se modificam com o tempo, será preciso criar rotinas que lidarão com esses dados temporais. No escopo inicial serão utilizados quatro APIs.

A API ***Steam WEB API*** é fornecida pela Steamworks[4], sendo assim é uma API oficial da Steam (STEAMWORKS, 2018). Ela possui métodos públicos e métodos privados. Os métodos públicos são abertos para qualquer usuário utiliza-los, já os métodos privados é necessário uma chave de desenvolvedor cedido pela própria Steamworks. Para acessar quaisquer dados da API é necessário um id, seja de usuário ou de jogo.

A API ***Steam Store API*** é fornecida pela Steam. Ela disponibiliza informações sobre os jogos guardados no banco de dados da Steam. Para acessar estes dados é necessário o id do jogo na Steam, porém também é possível passar filtros para pesquisas mais específicas.

A API ***Steam Spy API*** é fornecida pelo Steam Spy. Ela também disponibiliza informações sobre os jogos, porém ela também dispõe de outras informações como o número de donos ou número de avaliações positivas e negaticas de um determinado jogo. Para acessar estes dados é necessário o id do jogo na Steam, porém também é possível passar filtros para pesquisas mais específicas

A API ***Youtube API*** é fornecida pelo Google Developers[5], sendo assim a API oficial do Youtube. Ela dispõe de informações sobre cada vídeo como seus números de likes, dislikes e visualizações. Para extrair os dados de cada vídeo são necessário conseguir a lista dos 20 vídeos mais relevantes de um jogo, ápos isso com o id dos vídeos extrair suas informações.

A relação entre os dados que serão extraídos e de qual API será utilizado esta representado na tabela 4.

[4] <https://partner.steamgames.com>
[5] <https://developers.google.com/>

	Steam WEB API	***Steam Spy API***	***Steam Store API***	***Youtube API***
Nome				
Descrição				
Imagem *Header*				
Imagem *Background*				
Website				
Data Lançamento				
Steam Id				
Metacritic *Score*				
Avaliação Positiva				
Avaliação Negatica				
Média de Horas				
Donos				
Jogadores Online				
Visualizações				
Likes				
Dislikes				
Userscore				
Gêneros				
Categorias				
Linguagens				
Screenshots				
Desenvolvedoras				
Publicadoras				
Plataformas				
Preço				
Legendas				
Dados Estáticos				
Dados Temporais				

Tabela 4 – Dados x APIs

3.3.1 Dados Temporais

Dados temporais são aqueles dados que serão atualizados de acordo com alguma frequência. Como serão guardados seus valores no decorrer do tempo, será possível comporar o mesmo dado de um jogo num determinado espaço de tempo, podendo assim extrair mais informações e ocasionalmente mais métricas. A frequência que estes dados serão atualizado estão representado na tabela 5.

Tipo do Dado	**Frequência**
Média de Horas	1 vez por semana
Donos	1 vez por semana
Jogadores Online	1 vez por dia
Visualizações	1 vez por dia
Likes	1 vez por dia
Dislikes	1 vez por dia
Userscore	1 vez por semana
Preço	1 vez por semana

Tabela 5 – Dados x Frequência

Outro função que será chamanda numa determinada frequência será a inserção de novos jogos no banco de dados, pois novos jogos são lançandos a cada dia. Para não causar uma sobrecarga no Celery, a inserção de novos jogos ocorrerão uma vez por semana.

4 Resultados Obtidos

Neste capítulo serão exibidos os resultados obtidos no decorrer do projeto.

4.1 Extractor

Para o Extractor foram criados as interface ***Extractor*** e ***Database***, representados nos códigos 4.1 e 4.2, respectivamente, o módulo ***Utils***, representado pelo código 4.3 e os arquivos ***Main*** e ***Schedule*** representados pelos códigos 4.4 e 4.5, respectivamente. Também foram criados classes para cada *plugin* e para o banco de dados do Elasticsearch, que podem ser encontrados no anexo A.

```
1 from abc import ABC, abstractmethod
2 from db.elastic import Elastic
3 import datetime
4 import requests
5 import json
6
7 class Extractor(ABC):
8
9     url = "http://exampleofurl.com/api={}"
10
11     @abstractmethod
12     def get_game(self, identifier):
13         pass
14
15     def manipulate_data(self, data):
16         pass
17
18     def get_api(self, identifier):
19         response = requests.get(self.url.format(identifier))
20         if response.status_code == requests.codes.ok:
21             response = json.loads(response.text)
22             return response
23         else:
24             raise PageNotFound("Page not found!!!")
25
26     def temporal_data(self, identifier, value, data_name):
27         elastic = Elastic('elastic:9200', 'steam')
28         array = []
29         try:
30             game = elastic.get(identifier)
31             array = game[data_name]
32         except:
33             array = []
34         current_date = datetime.datetime.now()
35         data = {}
36         data['value'] = value
37         data['date'] = current_date.strftime("%Y-%m-%d")
38         array.append(data)
39         return array
```

```
40
41
42 class PageNotFound(Exception):
43     pass
44
45 class GameNotFound(Exception):
46     pass
```

Algoritmo 4.1 – Código do ***Extractor***

```
47 from abc import ABC, abstractmethod
48
49 class Database(ABC):
50
51     @abstractmethod
52     def update(self, identifier, data):
53         pass
54
55     @abstractmethod
56     def get(self, identifier):
57         pass
58
59     @abstractmethod
60     def delete(self, identifier):
61         pass
62
63 class DataNotFound(Exception):
64     pass
```

Algoritmo 4.2 – Código do ***Database***

```
65 import colorlog
66 import requests
67 import json
68 from ext.extractor import PageNotFound
69
70 def setup_logger():
71     formatter = colorlog.ColoredFormatter(
72         "%(asctime)s - %(log_color)s%(levelname)s: %(message)s",
73         datefmt = '%d/%m/%y|%H:%M:%S',
74         reset = True,
75         log_colors = {
76             'DEBUG': 'cyan',
77             'INFO': 'green',
78             'WARNING': 'purple',
79             'ERROR': 'red',
80             'CRITICAL': 'bold_red'
81         }
82     )
83     handler = colorlog.StreamHandler()
84     handler.setFormatter(formatter)
85     logger = colorlog.getLogger('Extractor')
86     logger.addHandler(handler)
87     logger.setLevel('DEBUG')
88     return logger
89
90 def get_all_games():
```

```
91      games = []
92      response = requests.get('http://steamspy.com/api.php?request=all')
93      if response.status_code == requests.codes.ok:
94          response = json.loads(response.content)
95          for game in response:
96              pair = (response[game]['appid'], response[game]['name'])
97              games.append(pair)
98          return games
99      else:
100         raise PageNotFound("Page␣not␣found!!
```

Algoritmo 4.3 – Código do ***Utils***

```
101 from ext.currency import Currency
102 from ext.steam_api import SteamAPI
103 from ext.steam_spy import SteamSpy
104 from ext.youtube_api import YoutubeAPI
105 from ext.extractor import GameNotFound
106 from db.elastic import Elastic
107 from utils import setup_logger, get_all_games
108
109 steam_api = SteamAPI()
110 steam_spy = SteamSpy()
111 steam_currency = Currency()
112 youtube_api = YoutubeAPI()
113
114 log = setup_logger()
115 log.info('Initializing␣Elasticsearch')
116 fail_id = open("ids_fails.txt", "a")
117
118 index_body = {
119     "mapping": {
120         "game": {
121             "properties": {
122                 "name": { "type": "text" },
123                 "description": { "type": "text" },
124                 "header_image": { "type": "text" },
125                 "background_image": { "type": "text" },
126                 "website": { "type": "text" },
127                 "release_date": { "type": "text" },
128                 "steam_id": { "type": "long" },
129                 "metacritic_score": { "type": "long" },
130                 "positive_avaliantion": { "type": "long" },
131                 "negative_avaliantion": { "type": "long" },
132                 "median_hours_played": { "type": "nested",
133                     "properties": {
134                         "value": { "type": "double" },
135                         "date": {"type": "date"}
136                     } },
137                 "owners": { "type": "nested",
138                     "properties": {
139                         "value": { "type": "double" },
140                         "date": {"type": "date"}
141                     } },
142                 "currency": { "type": "nested",
143                     "properties": {
144                         "value": { "type": "double" },
145                         "date": {"type": "date"}
146                     } },
```

```
147                 "view_count": { "type": "nested",
148                     "properties": {
149                         "value": { "type": "double" },
150                         "date": {"type": "date"}
151                     } },
152                 "like_count": { "type": "nested",
153                     "properties": {
154                         "value": { "type": "double" },
155                         "date": {"type": "date"}
156                     } },
157                 "dislike_count": { "type": "nested",
158                     "properties": {
159                         "value": { "type": "double" },
160                         "date": {"type": "date"}
161                     } },
162                 "userscore": { "type": "nested",
163                     "properties": {
164                         "value": { "type": "double" },
165                         "date": {"type": "date"}
166                     } },
167                 "genres": { "type": "text", "store": "true" },
168                 "categories": { "type": "text", "store": "true" },
169                 "languages": { "type": "text", "store": "true" },
170                 "screenshots": { "type": "text", "store": "true" },
171                 "developers": { "type": "text", "store": "true" },
172                 "publishers": { "type": "text", "store": "true" },
173                 "platforms": { "type": "text", "store": "true" },
174                 "is_free": { "type": "boolean" },
175                 "price": { "type": "nested",
176                     "properties": {
177                         "value": { "type": "double" },
178                         "date": {"type": "date"}
179                     }}
180             }
181         }
182     }
183 }
184
185 try:
186     elastic = Elastic('elastic:9200', 'steam')
187     log.info('Elasticsearch connected')
188     log.info('Create index Steam on Elasticsearch')
189     elastic.create_index(index_body)
190     games = get_all_games()
191     for game in games:
192         game_id, game_name = int(game[0]), str(game[1])
193         log.info('Starting the extraction of game: %s - %s', game_id, game_name)
194         try:
195             game = steam_api.get_game(game_id)
196             log.info('Steam API: successed!')
197             game.update(steam_spy.get_game(game_id))
198             log.info('Steam SPY: successed!')
199             game.update(steam_currency.get_game(game_id))
200             log.info('Steam Currency: successed!')
201             game.update(youtube_api.get_game(game_name))
202             log.info('Youtube API: successed!')
203             log.info('Starting insersion in the Elasticsearch')
204             elastic.update(game_id, game)
205             log.info('Finishing insersion in the Elasticsearch')
206         except Exception as error:
```

```
207             if type(error) == GameNotFound:
208                 log.warning(error)
209             else:
210                 log.error(error)
211                 fail_id.write(str(game_id) + " || " + str(game_name) + "\n")
212 except Exception as error:
213     log.error(error)
```

Algoritmo 4.4 – Código da ***Main***

```
214 from celery import Celery
215 from celery.schedules import crontab
216 from db.elastic import Elastic
217 from utils import get_all_games, setup_logger
218 from ext.currency import Currency
219 from ext.steam_api import SteamAPI
220 from ext.steam_spy import SteamSpy
221 from ext.youtube_api import YoutubeAPI
222 from ext.extractor import GameNotFound
223 import os
224
225 app = Celery('schedule')
226 steam_api = SteamAPI()
227 steam_spy = SteamSpy()
228 steam_currency = Currency()
229 youtube_api = YoutubeAPI()
230 app.conf.broker_url = 'redis://redis:6379/0'
231 log = setup_logger()
232 elastic = Elastic('elastic:9200', 'steam')
233
234 @app.task
235 def insert_new_games():
236     fail_id = open("ids_fails.txt", "a")
237     lst1 = elastic.get_all()
238     lst2 = get_all_games()
239     games = [game for game in lst2 if game not in lst1]
240     for game in games:
241         game_id, game_name = int(game[0]), str(game[1])
242         log.info('Starting the extraction of game: %s - %s', game_id, game_name)
243         try:
244             game = steam_api.get_game(game_id)
245             log.info('Steam API: successed!')
246             game.update(steam_spy.get_game(game_id))
247             log.info('Steam SPY: successed!')
248             game.update(steam_currency.get_game(game_id))
249             log.info('Steam Currency: successed!')
250             game.update(youtube_api.get_game(game_name))
251             log.info('Youtube API: successed!')
252             log.info('Starting insersion in the Elasticsearch')
253             elastic.update(game_id, game)
254             log.info('Finishing insersion in the Elasticsearch')
255         except Exception as error:
256             if type(error) == GameNotFound:
257                 log.warning(error)
258             else:
259                 log.error(error)
260             fail_id.write(str(game_id) + " || " + str(game_name) + "\n")
261
262 @app.task
```

```
263 def update_steam_api():
264     games = elastic.get_all()
265     for game in games:
266         log.info('Starting the extraction of game: %s - %s', game[0], game[1])
267         try:
268             gm = steam_api.get_game(int(game[0]))
269             log.info('Extraction successed!')
270             log.info('Starting update in the Elasticsearch')
271             elastic.update(int(game[0]), gm)
272             log.info('Finishing update in the Elasticsearch')
273         except Exception as error:
274             if type(error) == GameNotFound:
275                 log.warning(error)
276             else:
277                 log.error(error)
278             fail_id.write(str(game[0]) + " || " + str(game[1]) + "\n")
279
280 @app.task
281 def update_steam_spy():
282     games = elastic.get_all()
283     for game in games:
284         log.info('Starting the extraction of game: %s - %s', game[0], game[1])
285         try:
286             gm = steam_spy.get_game(int(game[0]))
287             log.info('Extraction successed!')
288             log.info('Starting update in the Elasticsearch')
289             elastic.update(int(game[0]), gm)
290             log.info('Finishing update in the Elasticsearch')
291         except Exception as error:
292             if type(error) == GameNotFound:
293                 log.warning(error)
294             else:
295                 log.error(error)
296             fail_id.write(str(game[0]) + " || " + str(game[1]) + "\n")
297
298 @app.task
299 def update_steam_currency():
300     games = elastic.get_all()
301     for game in games:
302         log.info('Starting the extraction of game: %s - %s', game[0], game[1])
303         try:
304             gm = steam_currency.get_game(int(game[0]))
305             log.info('Extraction successed!')
306             log.info('Starting update in the Elasticsearch')
307             elastic.update(int(game[0]), gm)
308             log.info('Finishing update in the Elasticsearch')
309         except Exception as error:
310             if type(error) == GameNotFound:
311                 log.warning(error)
312             else:
313                 log.error(error)
314             fail_id.write(str(game[0]) + " || " + str(game[1]) + "\n")
315
316 @app.task
317 def update_youtube_api():
318     games = elastic.get_all()
319     for game in games:
320         log.info('Starting the extraction of game: %s - %s', game[0], game[1])
321         try:
322             gm = youtube_api.get_game(str(game[1]))
```

```
323             log.info('Extraction successed!')
324             log.info('Starting update in the Elasticsearch')
325             elastic.update(int(game[0]), gm)
326             log.info('Finishing update in the Elasticsearch')
327         except Exception as error:
328             if type(error) == GameNotFound:
329                 log.warning(error)
330             else:
331                 log.error(error)
332             fail_id.write(str(game[0]) + " || " + str(game[1]) + "\n")
333
334 @app.task
335 def try_fails_id():
336     games = open("ids_fails.txt", "r")
337     for game in games:
338         game_id, game_name = game.split(" || ")
339         game_id = int(game_id)
340         log.info('Starting the extraction of game: %s - %s', game_id, game_name)
341         try:
342             game = steam_api.get_game(game_id)
343             log.info('Steam API: successed!')
344             game.update(steam_spy.get_game(game_id))
345             log.info('Steam SPY: successed!')
346             game.update(steam_currency.get_game(game_id))
347             log.info('Steam Currency: successed!')
348             game.update(youtube_api.get_game(game_name))
349             log.info('Youtube API: successed!')
350             log.info('Starting insersion in the Elasticsearch')
351             elastic.update(game_id, game)
352             log.info('Finishing insersion in the Elasticsearch')
353         except Exception as error:
354             if type(error) == GameNotFound:
355                 log.warning(error)
356             else:
357                 log.error(error)
358     os.remove("ids_fails.txt")
359
360 @app.on_after_configure.connect
361 def setup_periodic_tasks(sender, **kwargs):
362     log.info('Updating data from Steam API!')
363     sender.add_periodic_task(crontab(minute=0, hour=0), update_steam_api())
364     log.info('Updating data from Steamspy API!')
365     sender.add_periodic_task(crontab(minute=0, hour=0), update_steam_spy())
366     log.info('Updating data from Steam Currency!')
367     sender.add_periodic_task(crontab(minute=0, hour=0), update_steam_currency())
368     log.info('Updating data from Youtube API!')
369     sender.add_periodic_task(crontab(minute=0, hour=0), update_youtube_api())
370     log.info('Inserting new games!')
371     sender.add_periodic_task(crontab(minute=0, hour=0, day_of_week='sunday'),
        insert_new_games())
372     log.info('Trying again the fails inserts!')
373     sender.add_periodic_task(crontab(minute=0, hour=0, day_of_week='sunday'),
        try_fails_id())
```

Algoritmo 4.5 – Código do ***Schedule***

O resultado da execução do arquivo ***Main*** pode ser encontrado na figura 3, e um exemplo de como o arquivo fica guardado no Elasticsearch na figura 4.

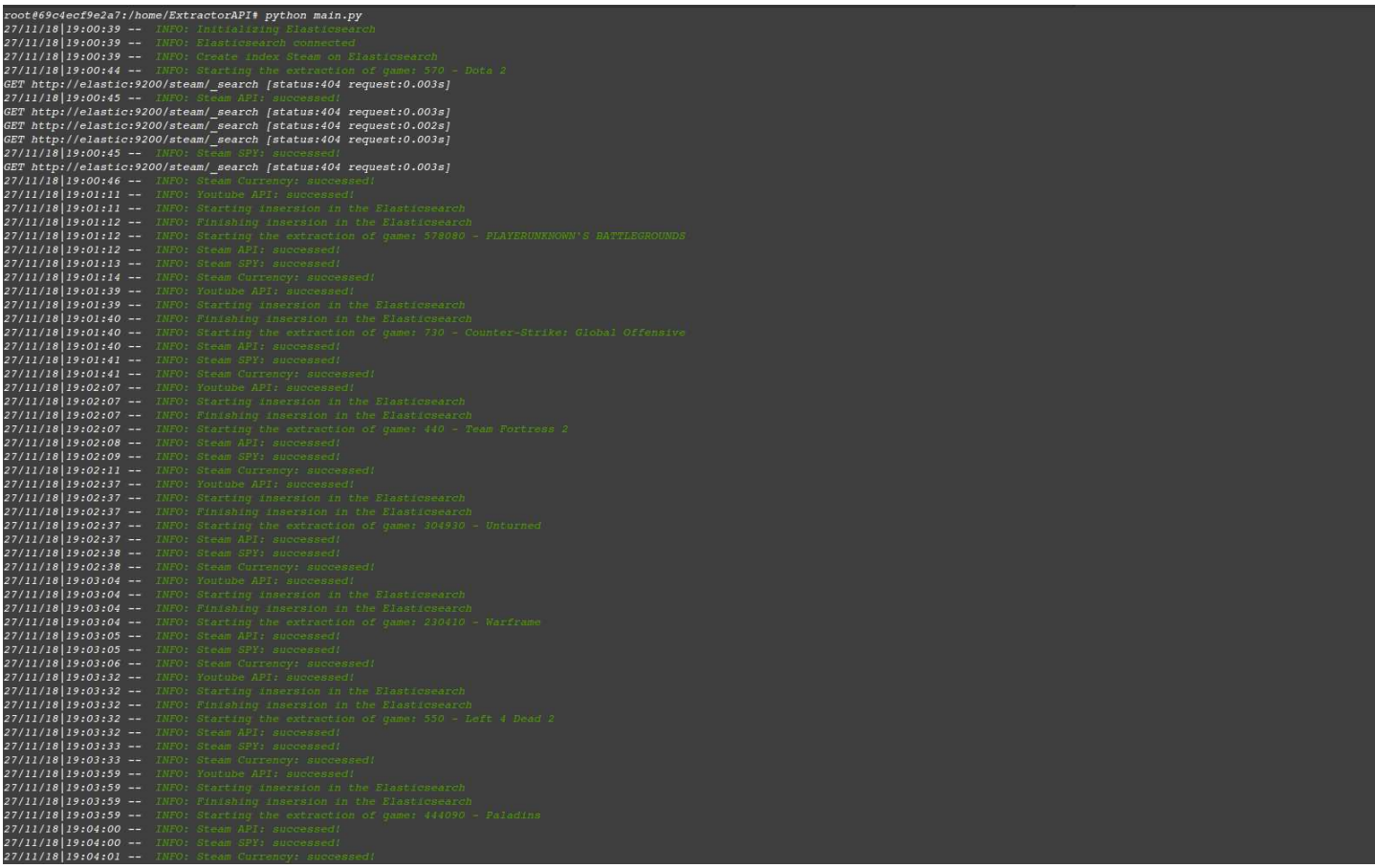

Figura 3 – Execução ***Main*** Terminal

Figura 4 – Exemplo Elasticsearch

O resultado da execução do arquivo ***Schedule*** pode ser encontrado na figura 5.

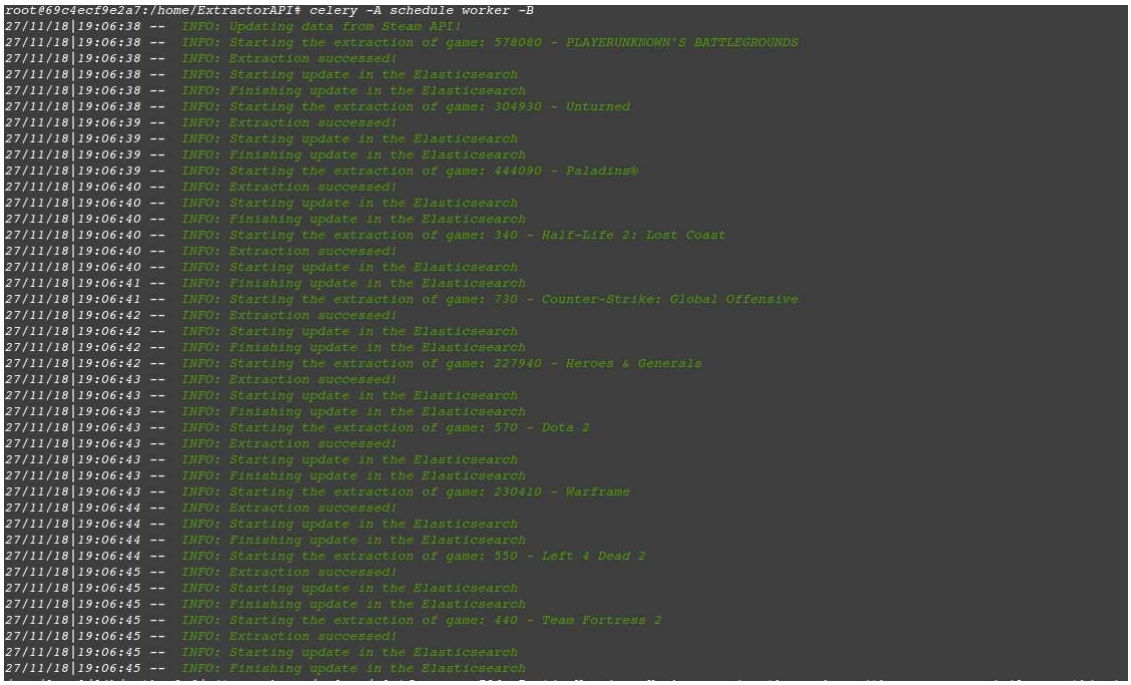

Figura 5 – Execução ***Schedule*** Terminal

Referências

ANDERS, D.; EL-NASR, M. S.; CANOSSA, A. Game analytics: Maximizing the value of player data. Londres, 2013. Citado na página 26.

BECK, K. et al. *Manifesto Ágil*. 2001. Disponível em: <http://agilemanifesto.org/iso/ptbr/manifesto.html>. Citado na página 31.

BOHANNON, J. Game-miners grapple with massive data. 2010. Citado na página 26.

CELERY. *Celery*. 2007. Disponível em: <http://www.celeryproject.org/>. Citado na página 32.

COLE, D. *DFC Intelligence*. 1994. Disponível em: <https://www.dfcint.com>. Citado na página 28.

DAVEPORT, T. H.; HARRIS, J. G. Competing on analytics: The new science of winning. Boston, 2007. Citado na página 25.

DOCKER. *Docker*. 2014. Disponível em: <https://www.docker.com/>. Citado na página 33.

ECKERSON, W. The rise of analytic applications: Build or buy. 2002. Citado na página 24.

ELASTIC. *Elasticsearch*. 2010. Disponível em: <https://www.elastic.co/products/elasticsearch>. Citado na página 31.

ELASTIC. *Kibana*. 2010. Disponível em: <https://www.elastic.co/products/kibana>. Citado na página 32.

FIELDING, R. T. Architectural styles and the design of network-based software architectures. 2000. Citado na página 23.

GALYONKIN, S. *Steam Spy*. 2018. Disponível em: <http://steamspy.com>. Citado 2 vezes nas páginas 21 e 28.

LOSHIN, D. Business intelligence: The savvy manager's guide. 2012. Citado na página 24.

MELLON, L. Apply metrics driven development to mmo costs and risks. República Tcheca, 2009. Citado na página 27.

NEWZOO. *The Brasilian Gamer 2017*. 2017. Disponível em: <https://newzoo.com/insights/infographics/the-brazilian-gamer-2017>. Citado na página 21.

NEWZOO. *The Chinese Gamer 2017*. 2017. Disponível em: <https://newzoo.com/insights/infographics/chinese-gamer-2017>. Citado na página 21.

OHNO, T. O sistema toyota de produção. Porto Alegre, 1997. Citado na página 31.

RETHANS, J. *APIs: Leverage For Digital Transformation*. 2016. Disponível em: <https://www.forbes.com/sites/forbestechcouncil/2017/05/08/apis-leverage-for-digital-transformation/#70b07fa17140>. Citado na página 23.

SILVEIRA, D. *Crescimento Desenvolvedoras*. 2016. Disponível em: <https://g1.globo.com/economia/negocios/noticia/numero-de-desenvolvedores-de-games-cresce-600-em-8-anos-diz-associacao.ghtml>. Citado na página 21.

STEAMDB. *SteamDB*. 2010. Disponível em: <https://steamdb.info>. Citado na página 27.

STEAMWORKS. *Steam Web API*. 2018. Disponível em: <https://partner.steamgames.com/doc/webapi_overview>. Citado na página 35.

Anexos

ANEXO A – Código dos *plugins* e do Elasticsearch

```
374 from ext.extractor import Extractor, GameNotFound
375 import json
376
377 class Currency(Extractor):
378
379     url = 'https://api.steampowered.com/ISteamUserStats/GetNumberOfCurrentPlayers/v1
      /?appid={}'
380
381     def get_game(self, identifier):
382         response = self.get_api(identifier)
383         data = response['response']
384         if data['result'] == 1:
385             result = {}
386             result['currency'] = self.temporal_data(identifier, data['player_count'],
       'currency')
387             return result
388         else:
389             raise GameNotFound("Game not found!!!")
```

Algoritmo A.1 – Código do *plugin* ***Steam Currency***

```
390 from ext.extractor import Extractor, GameNotFound
391
392 class SteamAPI(Extractor):
393
394     url = 'https://store.steampowered.com/api/appdetails/?appids={}'
395
396     def get_game(self, identifier):
397         response = self.get_api(identifier)
398         data = response[str(identifier)]
399         if data['success'] and data is not None:
400             result = self.manipulate_data(data['data'], identifier)
401             return result
402         else:
403             raise GameNotFound("Game not found!!!")
404
405     def manipulate_data(self, data, identifier):
406         result = {}
407         result['genres'] = []
408         result['publishers'] = []
409         result['developers'] = []
410         result['platforms'] = []
411         result['screenshots'] = []
412         result['name'] = data['name']
413         result['steam_id'] = data['steam_appid']
414         if data['is_free'] or not 'price_overview' in data:
415             result['price'] = self.temporal_data(identifier, 0.0, 'price')
416             result['is_free'] = True
417         else:
```

```
418             result['price'] = self.temporal_data(identifier, data['price_overview']['
        initial'] / 100, 'price')
419             result['is_free'] = False
420         result['description'] = data['about_the_game']
421         result['header_image'] = data['header_image']
422         result['background_image'] = data['background']
423         if data['website'] is not None:
424             result['website'] = data['website']
425         else:
426             result['website'] = ""
427         if 'genres' in data:
428             for gnr in data['genres']:
429                 result['genres'].append(gnr['description'])
430         for pbl in data['publishers']:
431             result['publishers'].append(pbl)
432         if 'developers' in data:
433             for dvp in data['developers']:
434                 result['developers'].append(dvp)
435         for plt in data['platforms']:
436             if plt: result['platforms'].append(str(plt).capitalize())
437         if data['release_date']['date'] != "":
438             result['release_date'] = data['release_date']['date']
439         else:
440             result['release_date'] = '12 Set, 2003'
441         if 'metacritic' in data:
442             result['metacritic_score'] = data['metacritic']['score']
443         else:
444             result['metacritic_score'] = 0
445         for screen in data['screenshots']:
446             result['screenshots'].append(screen['path_full']
```

Algoritmo A.2 – Código do *plugin* ***Steam Store API***

```
447 from ext.extractor import Extractor, GameNotFound
448
449 class SteamSpy(Extractor):
450
451     url = 'http://steamspy.com/api.php?request=appdetails&appid={}'
452
453     def get_game(self, identifier):
454         response = self.get_api(identifier)
455         if response['name'] is not None:
456             result = self.manipulate_data(response)
457             return result
458         else:
459             raise GameNotFound("Game not found!!!")
460
461     def manipulate_data(self, data):
462         result = {}
463         result['languages'] = []
464         result['categories'] = []
465         result['positive_avaliantion'] = data['positive']
466         result['negative_avaliantion'] = data['negative']
467         total = data['positive'] + data['negative']
468         score = data['positive'] / total * 100
469         result['userscore'] = self.temporal_data(identifier, round(score, 2), '
        userscore')
470         result['median_hours_played'] = self.temporal_data(identifier, data['
        median_forever'] // 60, 'median_hours_played')
```

```
471         numbers = data['owners'].split(' .. ')
472         own = (int(numbers[0].replace(',', '')) + int(numbers[1].replace(',', ''))) // 2
473         result['owners'] = self.temporal_data(identifier, own, 'owners')
474         for lng in data['languages'].split(', '):
475             result['languages'].append(lng)
476         for ctg in data['tags']:
477             result['categories'].append(ctg)
478         return result
```

Algoritmo A.3 – Código do *plugin* ***Steam Spy API***

```
479 from ext.extractor import Extractor, GameNotFound, PageNotFound
480 import requests
481 import json
482
483 class YoutubeAPI(Extractor):
484
485     API_KEY = 'AIzaSyD7uEK8S8NE79iHUx70rTvwXBCEHp5BosQ'
486
487     def get_game(self, identifier):
488         videos = self.get_videos(identifier)
489         if len(videos) != 0:
490             result = self.manipulate_data(videos)
491             return result
492         else:
493             raise GameNotFound("Game not found!!!")
494
495
496     def get_videos(self, identifier):
497         videos_id = []
498         url = 'https://www.googleapis.com/youtube/v3/search?part=snippet&maxResults=20&q={}&type=video&key={}&order=relevance'
499         response = self.get_api(url, identifier)
500         for vid in response['items']:
501             videos_id.append(vid['id']['videoId'])
502         return videos_id
503
504     def get_api(self, url, identifier):
505         response = requests.get(url.format(identifier, self.API_KEY))
506         if response.status_code == requests.codes.ok:
507             response = json.loads(response.text)
508             return response
509         else:
510             raise PageNotFound("Page not found!!!")
511
512     def manipulate_data(self, data):
513         result = {}
514         url = 'https://www.googleapis.com/youtube/v3/videos?part=statistics&id={}&key={}'
515         sum_view = 0
516         sum_like = 0
517         sum_dislike = 0
518         for vid in data:
519             response = self.get_api(url, vid)['items'][0]['statistics']
520             if 'viewCount' in response: sum_view += int(response['viewCount'])
521             if 'likeCount' in response: sum_like += int(response['likeCount'])
522             if 'dislikeCount' in response: sum_dislike += int(response['dislikeCount'])
```

```
523         result['view_count'] = self.temporal_data(identifier, sum_view, 'view_count')
524         result['like_count'] = self.temporal_data(identifier, sum_like, 'like_count')
525         result['dislike_count'] = self.temporal_data(identifier, sum_dislike, '
        dislike_count')
526         return result
```

Algoritmo A.4 – Código do *plugin* ***Youtube API***

```
527 import requests
528 import json
529 from elasticsearch import Elasticsearch
530 from db.database import Database, DataNotFound
531
532 class Elastic(Database):
533
534     def __init__(self, host, index):
535         self.hostname = host
536         self.index_name = index
537         try:
538             self.elastic = self.connect_elasticsearch()
539         except ElasticNotConnected as enc:
540             raise ElasticNotConnected('Elasticsearch couldn\'t be connected!!!')
541
542     def get(self, identifier):
543         res = self.elastic.search(index=self.index_name, body={"query": {"match": {"
        steam_id": int(identifier)}}})
544         if res['hits']['total'] == 1:
545             data = res['hits']['hits'][0]['_source']
546             return data
547         else:
548             raise DataNotFound('Data has not found in the Database!!!')
549
550     def update(self, identifier, data):
551         if self.has_data(identifier):
552             body = {}
553             body['doc'] = data
554             self.elastic.update(index=self.index_name, doc_type='game', id=identifier
        , body=body)
555         else:
556             self.elastic.index(index=self.index_name, doc_type='game', id=identifier,
         body=data)
557
558     def delete(self, identifier):
559         url = 'http://{}/{}/game/{}'
560         response = requests.delete(url.format(self.hostname, self.index_name,
        identifier))
561         if response.status_code != requests.codes.ok:
562             raise DataNotFound('Data has not found in the Database!!!')
563
564     def has_data(self, identifier):
565         url = 'http://{}/{}/game/{}'
566         response = requests.get(url.format(self.hostname, self.index_name, identifier
        ))
567         if response.status_code == requests.codes.ok:
568             return True
569         else:
570             return False
571
572     def connect_elasticsearch(self):
```

```
573         elastic = Elasticsearch([self.hostname])
574         if elastic.ping():
575             return elastic
576         else:
577             raise ElasticNotConnected('Elasticsearch couldn\'t be connected!!!')
578 
579     def delete_index(self):
580         self.elastic.indices.delete(index=self.index_name, ignore=[400, 404])
581 
582     def get_all(self):
583         res = self.elastic.search(index=self.index_name, body={"query": {"match_all"
        :{}}})
584         size = res['hits']['total']
585         res = self.elastic.search(index=self.index_name, body={"size": size, "query":
         {"match_all":{}}})
586         games = []
587         for game in res['hits']['hits']:
588             pair = (int(game['_id']), game['_source']['name'])
589             games.append(pair)
590         return games
591 
592     def create_index(self, index_body):
593         if self.elastic.indices.exists(self.index_name): self.delete_index()
594         self.elastic.indices.create(index=self.index_name, body=index_body, ignore
        =400)
595 
596 
597 class ElasticNotConnected(Exception):
598     pass
```

Algoritmo A.5 – Código do ***Elasticsearch***