



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-
47 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

Lista de ilustrações

Figura 1 – Arquitura Geral do Projeto	25
Figura 2 – Arquitura do Extractor	26
Figura 3 – Execução <i>Main</i> Terminal	36
Figura 4 – Exemplo Elasticsearch	37
Figura 5 – Execução <i>Schedule</i> Terminal	37

Lista de tabelas

Tabela 1	– Vantagens e Desvantagens Steam DB	21
Tabela 2	– Vantagens e Desvantagens Steam Spy	22
Tabela 3	– Vantagens e Desvantagens DFC Intelligence	22
Tabela 4	– Dados x APIs	28
Tabela 5	– Dados x Frequência	28

Lista de Algoritmos

4.1	Código do <i>Extractor</i>	29
-----	----------------------------	----

Lista de abreviaturas e siglas

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

Sumário

1	INTRODUÇÃO	15
1.1	Objetivos	15
1.2	Estrutura do Documento	16
2	FUNDAMENTAÇÃO TEÓRICA	17
2.1	<i>Application Programming Interface</i>	17
2.1.1	<i>Representational State Transfer</i>	17
2.2	<i>Business Intelligence</i>	18
2.3	<i>Game Analytics</i>	19
2.3.1	Telemetria	20
2.3.2	<i>Game Metrics</i>	20
2.4	Softwares Correlatos	21
2.4.1	Steam DB	21
2.4.2	Steam Spy	21
2.4.3	DFC Intelligence	22
3	METODOLOGIA	23
3.1	Análise de Ferramentas	23
3.1.1	Banco de Dados	23
3.1.2	Frequência na Extração dos Dados	24
3.1.3	Visualização dos Dados	24
3.1.4	Criação de Containers	24
3.2	Arquitetura do Projeto	25
3.2.1	Arquitetura Extractor	26
3.3	Extractor	26
3.3.1	Dados Temporais	28
4	RESULTADOS OBTIDOS	29
4.1	Extractor	29
	REFERÊNCIAS	39
	ANEXOS	41
	ANEXO A – CÓDIGO DOS <i>PLUGINS</i> E DO ELASTICSEARCH	43

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](#)).

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 exibe dados de jogos, apenas no ano de 2017 foram lançados 7.672 jogos na plataforma Steam¹ uma média de 21 jogos por dia. Porém, à medida que o mercado desenvolvedor cresce, aumenta-se a concorrência no meio, tornando cada vez mais difícil a aceitação do profissional pela comunidade em razão da variedade de opções na hora da compra fazendo com que algumas *features* se tornem diferenciais, como por exemplo *multiplayer*, localização, dentre 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 de um grande número de recursos humanos/tempo para extrair informações que agreguem valor ao desenvolvimento. Essas desenvolvedoras, por falta deste tipo de recursos, muitas vezes pagam para outras empresas fazerem essas análises ou desenvolvem jogos sem as análises.

Desta forma o objetivo deste trabalho é a criação de uma plataforma web onde será disponibilizada, de uma forma mais simples, diferentes métricas e agregações de dados, que permitem ao usuário moldar conforme sua necessidade. Incluindo *insights* sobre as métricas.

1.1 Objetivos

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

Os objetivos específicos são:

- Elaborar uma arquitetura responsável por fazer a extração, manipulação dos dados

¹ Plataforma que disponibiliza jogos para downloads

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 capítulo 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*, bem como serão expostos softwares similares ao 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 é 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) a própria *World Wide Web*, utilizou REST como base para o desenvolvimento do protocolo HTTP, sendo assim possível projetar um sistema de serviços web com a arquitetura REST.

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 uso dos dados. Enquanto teve uma grande evolução na maneira que esses dados eram gerenciados, sempre existiu 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 propiciaram a formação de conhecimento a respeito, dessa maneira, 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 tomada 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 tem 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 são os processos,

as tecnologias 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 análíticas e gestão de informação/conhecimento.

Business intelligence geralmente considera 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 dentro 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 é um sucesso quando todas as tarefas são 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 as 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, mas para melhorá-los.

2.3 Game Analytics

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

Analytics é o processo de descobrir e comunicar padrões em dados, solucionando problemas de negócios ou suportar decisões de gerenciamento de empresas. Esta 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 usuários jogam, e o nível de interação entre o jogador e o jogo. Vale mencionar que *game analytics* não é só GUR, já que o este é focado nos dados obtidos a partir dos usuários, e 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 é telemetria. No contexto de jogos, telemetria seria algum jogo transmitindo dados sobre a interação do usuário com o jogo.

Telemetria de jogos é o termo utilizado para qualquer dado obtido 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 potencias que outras fontes de BI. *Game metrics* geralmente são definidas como um medição quantitativa de um ou mais atributos, ou 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 utilizam aquele determinado 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 online. Essas métricas geralmente são utilizadas no monitoramento dos impactos causados 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 utilizados 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 desenvolvida. 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](#)). Esta ferramenta disponibiliza *rankings* e gráficos de jogos para que o usuário tenha uma melhor visualização. Atualmente o Steam DB apresenta métricas individuais de cada jogo, não oferecendo nenhuma maneira de incremento. Na tabela 1 podemos ver as vantagens e desvantagens do Steam DB.

Vantagens	Desvantagens
<i>Open source</i>	Não possui política de contribuição
<i>Rankings</i> 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 determinado jogo ou vendas por região (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 <i>open source</i>

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, 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 das desenvolvedoras contratantes. 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 <i>open source</i>
	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 ser 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 dividido 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 se adaptar a novos fatores durante o desenvolvimento do projeto, ao invés de tentar prever o que pode ou não acontecer.

3.1 Análise de Ferramentas

Nesta seção são feitos 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 Elastic¹, 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 ferramenta 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 assíncronas baseadas na passagem de mensagens distribuídas, também oferece suporte a operações de agendamento (CELERY, 2007).

Celery possui funções que auxiliam a criação de rotinas, funcionando principalmente com Python². 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 uma 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 ferramenta é responsável por dividir as partes da arquitetura em diferentes containers.

Docker

Docker é uma ferramenta *open source* que fornece containers de *softwares* para o auxílio 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.

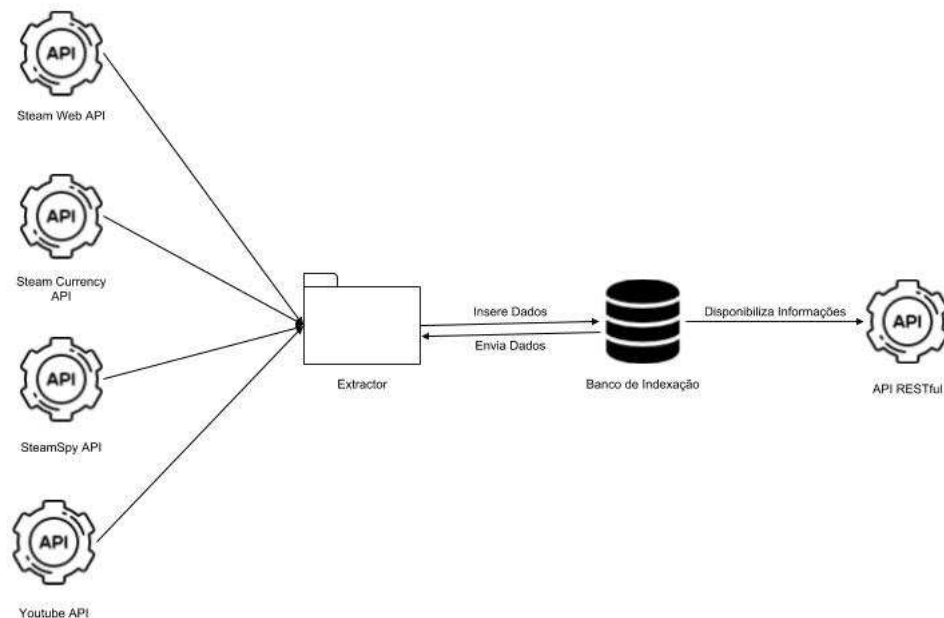


Figura 1 – Arquitetura Geral do Projeto

Fonte de Dados são locais onde se encontram os dados que serão extraídos, manipulados e inseridos no banco de indexação, estes dados poderão vir de quaisquer fontes, 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 indexação, sem que o usuário precise se conectar diretamente a ele. Como será desenvolvida um API, outros e quaisquer projetos poderão usufruir dela.

3.2.1 Arquitetura Extractor

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

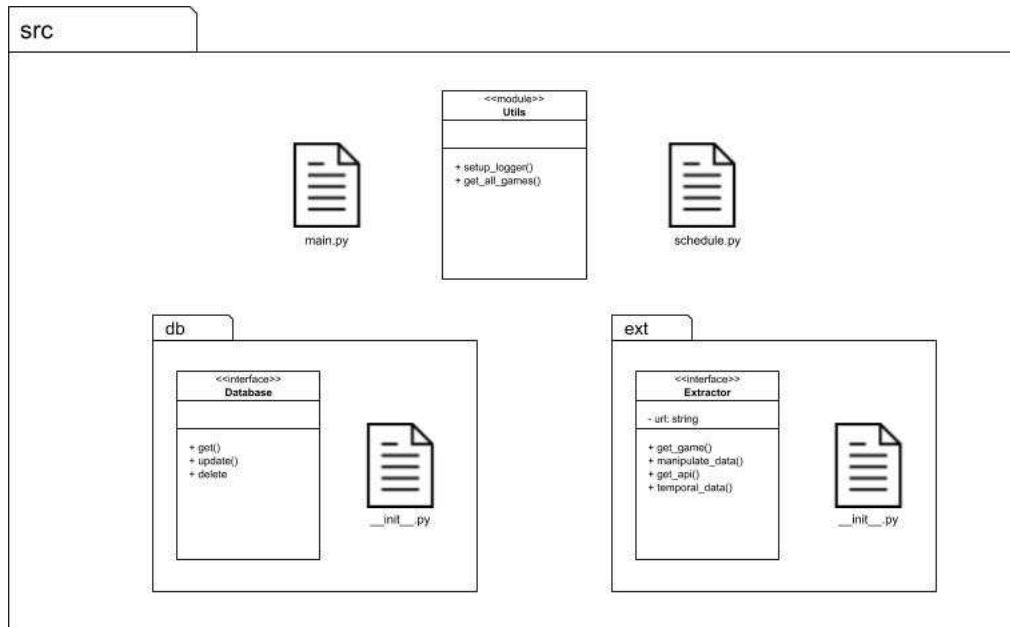


Figura 2 – Arquitetura do Extractor

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, deletar 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 responsáveis por configurar os logs e extrair todos os jogos que possuem.

Os arquivos **Main** e **Schedule** são responsáveis 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 Steamworks⁴, 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 utilizá-los, já os métodos privados é necessário uma chave de desenvolvedor cedido pela própria Steamworks. Para acessar os dados da API é preciso 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, dispõe-se de outras informações como o número de donos ou de avaliações positivas e negativas 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 Developers⁵, sendo assim a API oficial do Youtube. Ela dispõe de informações sobre cada vídeo como os números de likes, dislikes e visualizações. Para extrair os dados de cada vídeo é necessário conseguir a lista dos 20 vídeos mais relevantes de um jogo, após 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/>>

	<i>Steam WEB API</i>	<i>Steam Spy API</i>	<i>Steam Store API</i>	<i>Youtube API</i>
Nome				
Descrição				
Imagem <i>Header</i>				
Imagem <i>Background</i>				
<i>Website</i>				
Data Lançamento				
Steam Id				
Metacritic <i>Score</i>				
Avaliação Positiva				
Avaliação Negativa				
Média de Horas				
Donos				
Jogadores Online				
Visualizações				
<i>Likes</i>				
<i>Dislikes</i>				
<i>Userscore</i>				
Gêneros				
Categorias				
Linguagens				
<i>Screenshots</i>				
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 compor 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
<i>Likes</i>	1 vez por dia
<i>Dislikes</i>	1 vez por dia
<i>Userscore</i>	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 lançados 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: succeeded!')
197             game.update(steam_spy.get_game(game_id))
198             log.info('Steam SPY: succeeded!')
199             game.update(steam_currency.get_game(game_id))
200             log.info('Steam Currency: succeeded!')
201             game.update(youtube_api.get_game(game_name))
202             log.info('Youtube API: succeeded!')
203             log.info('Starting insertion in the Elasticsearch')
204             elastic.update(game_id, game)
205             log.info('Finishing insertion 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: succeeded!')
246             game.update(steam_spy.get_game(game_id))
247             log.info('Steam SPY: succeeded!')
248             game.update(steam_currency.get_game(game_id))
249             log.info('Steam Currency: succeeded!')
250             game.update(youtube_api.get_game(game_name))
251             log.info('Youtube API: succeeded!')
252             log.info('Starting insertion in the Elasticsearch')
253             elastic.update(game_id, game)
254             log.info('Finishing insertion 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 succeeded!')
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 succeeded!')
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 succeeded!')
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 succeeded!')
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: succeeded!')
344             game.update(steam_spy.get_game(game_id))
345             log.info('Steam SPY: succeeded!')
346             game.update(steam_currency.get_game(game_id))
347             log.info('Steam Currency: succeeded!')
348             game.update(youtube_api.get_game(game_name))
349             log.info('Youtube API: succeeded!')
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.conf.beat_schedule = {
361     "update-steam-api": {
362         "task": "schedule.update_steam_api",
363         "schedule": crontab(minute=0, hour=0)
364     },
365     "update-steamspy-api": {
366         "task": "schedule.update_steam_spy",
367         "schedule": crontab(minute=0, hour=0)
368     },
369     "update-currency-api": {
370         "task": "schedule.update_steam_currency",
371         "schedule": crontab(minute=0, hour=0)
372     },
373     "update-youtube-api": {
374         "task": "schedule.update_youtube_api",
375         "schedule": crontab(minute=0, hour=0)
376     },
377     "insert-new-games": {
378         "task": "schedule.insert_new_games",
379         "schedule": crontab(minute=0, hour=0, day_of_week='sunday')
380     },
381     "try-fails-id": {
382         "task": "schedule.try_fails_id",

```

```

383     "schedule": crontab(minute=0, hour=0, day_of_week='sunday')
384 }
385 }

```

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.

```

root@69c4ecf9e2a7:/home/ExtractorAPI# python main.py
27/11/18|19:00:39 -- INFO: Initializing Elasticsearch
27/11/18|19:00:39 -- INFO: Elasticsearch connected
27/11/18|19:00:39 -- INFO: Create index Steam on Elasticsearch
27/11/18|19:00:44 -- INFO: Starting the extraction of game: 570 - Dota 2
GET http://elastic:9200/steam/_search [status:404 request:0.003s]
27/11/18|19:00:45 -- INFO: Steam API: succeeded!
GET http://elastic:9200/steam/_search [status:404 request:0.003s]
GET http://elastic:9200/steam/_search [status:404 request:0.002s]
GET http://elastic:9200/steam/_search [status:404 request:0.003s]
27/11/18|19:00:45 -- INFO: Steam SPY: succeeded!
GET http://elastic:9200/steam/_search [status:404 request:0.003s]
27/11/18|19:00:46 -- INFO: Steam Currency: succeeded!
27/11/18|19:01:11 -- INFO: Youtube API: succeeded!
27/11/18|19:01:11 -- INFO: Starting insertion in the Elasticsearch
27/11/18|19:01:12 -- INFO: Finishing insertion in the Elasticsearch
27/11/18|19:01:12 -- INFO: Starting the extraction of game: 578090 - PLAYERUNKNOWN'S BATTLEGROUNDS
27/11/18|19:01:12 -- INFO: Steam API: succeeded!
27/11/18|19:01:13 -- INFO: Steam SPY: succeeded!
27/11/18|19:01:14 -- INFO: Steam Currency: succeeded!
27/11/18|19:01:39 -- INFO: Youtube API: succeeded!
27/11/18|19:01:39 -- INFO: Starting insertion in the Elasticsearch
27/11/18|19:01:40 -- INFO: Finishing insertion in the Elasticsearch
27/11/18|19:01:40 -- INFO: Starting the extraction of game: 730 - Counter-Strike: Global Offensive
27/11/18|19:01:41 -- INFO: Steam API: succeeded!
27/11/18|19:01:41 -- INFO: Steam SPY: succeeded!
27/11/18|19:01:41 -- INFO: Steam Currency: succeeded!
27/11/18|19:02:07 -- INFO: Youtube API: succeeded!
27/11/18|19:02:07 -- INFO: Starting insertion in the Elasticsearch
27/11/18|19:02:07 -- INFO: Finishing insertion in the Elasticsearch
27/11/18|19:02:07 -- INFO: Starting the extraction of game: 440 - Team Fortress 2
27/11/18|19:02:08 -- INFO: Steam API: succeeded!
27/11/18|19:02:09 -- INFO: Steam SPY: succeeded!
27/11/18|19:02:11 -- INFO: Steam Currency: succeeded!
27/11/18|19:02:37 -- INFO: Youtube API: succeeded!
27/11/18|19:02:37 -- INFO: Starting insertion in the Elasticsearch
27/11/18|19:02:37 -- INFO: Finishing insertion in the Elasticsearch
27/11/18|19:02:37 -- INFO: Starting the extraction of game: 304930 - Unturned
27/11/18|19:02:37 -- INFO: Steam API: succeeded!
27/11/18|19:02:38 -- INFO: Steam SPY: succeeded!
27/11/18|19:02:38 -- INFO: Steam Currency: succeeded!
27/11/18|19:02:38 -- INFO: Youtube API: succeeded!
27/11/18|19:03:04 -- INFO: Starting insertion in the Elasticsearch
27/11/18|19:03:04 -- INFO: Finishing insertion in the Elasticsearch
27/11/18|19:03:04 -- INFO: Starting the extraction of game: 230410 - Warframe
27/11/18|19:03:05 -- INFO: Steam API: succeeded!
27/11/18|19:03:05 -- INFO: Steam SPY: succeeded!
27/11/18|19:03:06 -- INFO: Steam Currency: succeeded!
27/11/18|19:03:32 -- INFO: Youtube API: succeeded!
27/11/18|19:03:32 -- INFO: Starting insertion in the Elasticsearch
27/11/18|19:03:32 -- INFO: Finishing insertion in the Elasticsearch
27/11/18|19:03:32 -- INFO: Starting the extraction of game: 550 - Left 4 Dead 2
27/11/18|19:03:32 -- INFO: Steam API: succeeded!
27/11/18|19:03:33 -- INFO: Steam SPY: succeeded!
27/11/18|19:03:33 -- INFO: Steam Currency: succeeded!
27/11/18|19:03:59 -- INFO: Youtube API: succeeded!
27/11/18|19:03:59 -- INFO: Starting insertion in the Elasticsearch
27/11/18|19:03:59 -- INFO: Finishing insertion in the Elasticsearch
27/11/18|19:03:59 -- INFO: Starting the extraction of game: 444090 - Paladins
27/11/18|19:04:00 -- INFO: Steam API: succeeded!
27/11/18|19:04:00 -- INFO: Steam SPY: succeeded!
27/11/18|19:04:01 -- INFO: Steam Currency: succeeded!

```

Figura 3 – Execução *Main* Terminal

JSON	Raw Data	Headers
Save	Copy	
index:	"steam"	
type:	"game"	
id:	"578080"	
version:	1	
found:	true	
source:		
genres:		
0:	"Action"	
1:	"Adventure"	
2:	"Massively Multiplayer"	
publishers:		
0:	"PUBG Corporation"	
developers:		
0:	"PUBG Corporation"	
platforms:		
0:	"Windows"	
1:	"Mac"	
2:	"Linux"	
screenshots:		
0:	"https://steampowered.akamaihd.net/steam/apps/578080/ss_23af2e50855a833c22d0c11ca23a719f54a554ff.1920x1080.jpg?h=1539308944"	
1:	"https://steampowered.akamaihd.net/steam/apps/578080/ss_2c79b3b590b186b10bf082d37674621f204a3497.1920x1080.jpg?h=1539308944"	
2:	"https://steampowered.akamaihd.net/steam/apps/578080/ss_e7e7847eff0933de92192bb628bc7068d611da.1920x1080.jpg?h=1539308944"	
3:	"https://steampowered.akamaihd.net/steam/apps/578080/ss_a3f3f6894f4a4eb4d17d7e41c8e1f195f37ba896.1920x1080.jpg?h=1539308944"	
4:	"https://steampowered.akamaihd.net/steam/apps/578080/ss_108e2981889423b057b778cd07ae25ac18406cf1.1920x1080.jpg?h=1539308944"	
5:	"https://steampowered.akamaihd.net/steam/apps/578080/ss_2a1de6f978ae9eb6debf8cf661395677c0460ab.1920x1080.jpg?h=1539308944"	
6:	"https://steampowered.akamaihd.net/steam/apps/578080/ss_a403a1f4071d36d42bea7a505080956570b2569.1920x1080.jpg?h=1539308944"	
7:	"https://steampowered.akamaihd.net/steam/apps/578080/ss_10947072cf05f5b301e3d3acdf3009718b6451c4.1920x1080.jpg?h=1539308944"	
8:	"https://steampowered.akamaihd.net/steam/apps/578080/ss_4bbcaec1ef97d962c68c1a5e4675cd081de564.1920x1080.jpg?h=1539308944"	
9:	"https://steampowered.akamaihd.net/steam/apps/578080/ss_8112cd376568d9470c2edde841980fcd46f1529.1920x1080.jpg?h=1539308944"	
10:	"https://steampowered.akamaihd.net/steam/apps/578080/ss_c4ac13f9038cc049353605848302f9090ee73106.1920x1080.jpg?h=1539308944"	
11:	"https://steampowered.akamaihd.net/steam/apps/578080/ss_08a8df0756673179e190c2e16d090de63cfb2e.1920x1080.jpg?h=1539308944"	
12:	"https://steampowered.akamaihd.net/steam/apps/578080/ss_f98ff7b48c80541cfe27399285013159707c85c.1920x1080.jpg?h=1539308944"	
13:	"https://steampowered.akamaihd.net/steam/apps/578080/ss_29d0709711f3204e84b6f9d69f3be163ebe12486.1920x1080.jpg?h=1539308944"	
name:	"PLAYERUNKNOWN'S BATTLEGROUNDS"	
steam_id:	578080	
price:		
0:		
value:	55.99	
date:	"2018-11-27"	
is_free:	false	
description:	"strong>PLAYERUNKNOWN'S BATTLEGROUNDS is a battle royale shooter that pits 100 players against each other in a struggle for survival. Gather supplies and outwit your opponents to become the last person standing, pioneer of the battle royale genre and the creator of the battle royale game modes in the ARMA series and H1Z1: King of the Kill. At PUBG Corp., Greene is working with a veteran team of developers to make PUBG into the wor	
header_image:	"https://steampowered.akamaihd.net/steam/apps/578080/header.jpg?h=1539308944"	
background_image:	"https://steampowered.akamaihd.net/steam/apps/578080/page_bg_generated_v6b.jpg?h=1539308944"	
website:	"http://www.playbattlegrounds.com"	
release_date:	"21 Dec, 2017"	
metacritic_score:	86	
languages:		

Figura 4 – Exemplo Elasticsearch

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

```

root@69c4ecf9e2a7:/home/ExtractorAPI# celery -A schedule worker -B
27/11/18|19:06:38 -- INFO: Updating data from Steam API!
27/11/18|19:06:38 -- INFO: Starting the extraction of game: 578080 - PLAYERUNKNOWN'S BATTLEGROUNDS
27/11/18|19:06:38 -- INFO: Extraction succeeded!
27/11/18|19:06:38 -- INFO: Starting update in the Elasticsearch
27/11/18|19:06:38 -- INFO: Finishing update in the Elasticsearch
27/11/18|19:06:38 -- INFO: Starting the extraction of game: 304930 - Unturned
27/11/18|19:06:39 -- INFO: Extraction succeeded!
27/11/18|19:06:39 -- INFO: Starting update in the Elasticsearch
27/11/18|19:06:39 -- INFO: Finishing update in the Elasticsearch
27/11/18|19:06:39 -- INFO: Starting the extraction of game: 444090 - Paladins®
27/11/18|19:06:40 -- INFO: Extraction succeeded!
27/11/18|19:06:40 -- INFO: Starting update in the Elasticsearch
27/11/18|19:06:40 -- INFO: Finishing update in the Elasticsearch
27/11/18|19:06:40 -- INFO: Starting the extraction of game: 340 - Half-Life 2: Lost Coast
27/11/18|19:06:40 -- INFO: Extraction succeeded!
27/11/18|19:06:40 -- INFO: Starting update in the Elasticsearch
27/11/18|19:06:41 -- INFO: Finishing update in the Elasticsearch
27/11/18|19:06:41 -- INFO: Starting the extraction of game: 730 - Counter-Strike: Global Offensive
27/11/18|19:06:42 -- INFO: Extraction succeeded!
27/11/18|19:06:42 -- INFO: Starting update in the Elasticsearch
27/11/18|19:06:42 -- INFO: Finishing update in the Elasticsearch
27/11/18|19:06:42 -- INFO: Starting the extraction of game: 227940 - Heroes & Generals
27/11/18|19:06:43 -- INFO: Extraction succeeded!
27/11/18|19:06:43 -- INFO: Starting update in the Elasticsearch
27/11/18|19:06:43 -- INFO: Finishing update in the Elasticsearch
27/11/18|19:06:43 -- INFO: Starting the extraction of game: 570 - Dota 2
27/11/18|19:06:43 -- INFO: Extraction succeeded!
27/11/18|19:06:43 -- INFO: Starting update in the Elasticsearch
27/11/18|19:06:43 -- INFO: Finishing update in the Elasticsearch
27/11/18|19:06:43 -- INFO: Starting the extraction of game: 230410 - Warframe
27/11/18|19:06:44 -- INFO: Extraction succeeded!
27/11/18|19:06:44 -- INFO: Starting update in the Elasticsearch
27/11/18|19:06:44 -- INFO: Finishing update in the Elasticsearch
27/11/18|19:06:44 -- INFO: Starting the extraction of game: 350 - Left 4 Dead 2
27/11/18|19:06:45 -- INFO: Extraction succeeded!
27/11/18|19:06:45 -- INFO: Starting update in the Elasticsearch
27/11/18|19:06:45 -- INFO: Finishing update in the Elasticsearch
27/11/18|19:06:45 -- INFO: Starting the extraction of game: 440 - Team Fortress 2
27/11/18|19:06:45 -- INFO: Extraction succeeded!
27/11/18|19:06:45 -- INFO: Starting update in the Elasticsearch
27/11/18|19:06:45 -- INFO: Finishing update in the Elasticsearch

```

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 19.
- BECK, K. et al. *Manifesto Ágil*. 2001. Disponível em: <<http://agilemanifesto.org/iso/ptbr/manifesto.html>>. Citado na página 23.
- BOHANNON, J. Game-miners grapple with massive data. 2010. Citado na página 20.
- CELERY. *Celery*. 2007. Disponível em: <<http://www.celeryproject.org/>>. Citado na página 24.
- COLE, D. *DFC Intelligence*. 1994. Disponível em: <<https://www.dfciint.com>>. Citado na página 22.
- DAVEPORT, T. H.; HARRIS, J. G. Competing on analytics: The new science of winning. Boston, 2007. Citado na página 19.
- DOCKER. *Docker*. 2014. Disponível em: <<https://www.docker.com/>>. Citado na página 25.
- ECKERSON, W. The rise of analytic applications: Build or buy. 2002. Citado na página 18.
- ELASTIC. *Elasticsearch*. 2010. Disponível em: <<https://www.elastic.co/products/elasticsearch>>. Citado na página 23.
- ELASTIC. *Kibana*. 2010. Disponível em: <<https://www.elastic.co/products/kibana>>. Citado na página 24.
- FIELDING, R. T. Architectural styles and the design of network-based software architectures. 2000. Citado na página 17.
- GALYONKIN, S. *Steam Spy*. 2018. Disponível em: <<http://steamspy.com>>. Citado 2 vezes nas páginas 15 e 22.
- LOSHIN, D. Business intelligence: The savvy manager's guide. 2012. Citado na página 18.
- MELLON, L. Apply metrics driven development to mmo costs and risks. República Tcheca, 2009. Citado na página 20.
- NEWZOO. *The Brazilian Gamer 2017*. 2017. Disponível em: <<https://newzoo.com/insights/infographics/the-brazilian-gamer-2017>>. Citado na página 15.
- NEWZOO. *The Chinese Gamer 2017*. 2017. Disponível em: <<https://newzoo.com/insights/infographics/chinese-gamer-2017>>. Citado na página 15.
- OHNO, T. O sistema toyota de produção. Porto Alegre, 1997. Citado na página 23.

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 17.

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 15.

STEAMDB. *SteamDB*. 2010. Disponível em: <<https://steamdb.info>>. Citado na página 21.

STEAMWORKS. *Steam Web API*. 2018. Disponível em: <https://partner.steamgames.com/doc/webapi_overview>. Citado na página 27.

Anexos

ANEXO A – Código dos *plugins* e do Elasticsearch

```

386 from ext.extractor import Extractor, GameNotFound
387 import json
388
389 class Currency(Extractor):
390
391     url = 'https://api.steampowered.com/ISteamUserStats/GetNumberOfCurrentPlayers/v1
        /?appid={}'
392
393     def get_game(self, identifier):
394         response = self.get_api(identifier)
395         data = response['response']
396         if data['result'] == 1:
397             result = {}
398             result['currency'] = self.temporal_data(identifier, data['player_count'],
        'currency')
399             return result
400         else:
401             raise GameNotFound("Game not found!!!")

```

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

```

402 from ext.extractor import Extractor, GameNotFound
403
404 class SteamAPI(Extractor):
405
406     url = 'https://store.steampowered.com/api/appdetails/?appids={}'
407
408     def get_game(self, identifier):
409         response = self.get_api(identifier)
410         data = response[str(identifier)]
411         if data['success'] and data is not None:
412             result = self.manipulate_data(data['data'], identifier)
413             return result
414         else:
415             raise GameNotFound("Game not found!!!")
416
417     def manipulate_data(self, data, identifier):
418         result = {}
419         result['genres'] = []
420         result['publishers'] = []
421         result['developers'] = []
422         result['platforms'] = []
423         result['screenshots'] = []
424         result['name'] = data['name']
425         result['steam_id'] = data['steam_appid']
426         if data['is_free'] or not 'price_overview' in data:
427             result['price'] = self.temporal_data(identifier, 0.0, 'price')
428             result['is_free'] = True
429         else:

```

```

430         result['price'] = self.temporal_data(identifier, data['price_overview']['initial'] / 100, 'price')
431         result['is_free'] = False
432         result['description'] = data['about_the_game']
433         result['header_image'] = data['header_image']
434         result['background_image'] = data['background']
435         if data['website'] is not None:
436             result['website'] = data['website']
437         else:
438             result['website'] = ""
439         if 'genres' in data:
440             for gnr in data['genres']:
441                 result['genres'].append(gnr['description'])
442         for pbl in data['publishers']:
443             result['publishers'].append(pbl)
444         if 'developers' in data:
445             for dvp in data['developers']:
446                 result['developers'].append(dvp)
447         for plt in data['platforms']:
448             if plt: result['platforms'].append(str(plt).capitalize())
449         if data['release_date']['date'] != "":
450             result['release_date'] = data['release_date']['date']
451         else:
452             result['release_date'] = '12_Set, 2003'
453         if 'metacritic' in data:
454             result['metacritic_score'] = data['metacritic']['score']
455         else:
456             result['metacritic_score'] = 0
457         for screen in data['screenshots']:
458             result['screenshots'].append(screen['path_full'])

```

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

```

459 from ext.extractor import Extractor, GameNotFound
460
461 class SteamSpy(Extractor):
462
463     url = 'http://steamspy.com/api.php?request=appdetails&appid={}'
464
465     def get_game(self, identifier):
466         response = self.get_api(identifier)
467         if response['name'] is not None:
468             result = self.manipulate_data(response)
469             return result
470         else:
471             raise GameNotFound("Game not found!!!")
472
473     def manipulate_data(self, data):
474         result = {}
475         result['languages'] = []
476         result['categories'] = []
477         result['positive_avaliantion'] = data['positive']
478         result['negative_avaliantion'] = data['negative']
479         total = data['positive'] + data['negative']
480         score = data['positive'] / total * 100
481         result['userscore'] = self.temporal_data(identifier, round(score, 2), 'userscore')
482         result['median_hours_played'] = self.temporal_data(identifier, data['median_forever'] // 60, 'median_hours_played')

```

```

483         numbers = data['owners'].split('...')
484         own = (int(numbers[0].replace(',','')) + int(numbers[1].replace(',','')))
// 2
485         result['owners'] = self.temporal_data(identifier, own, 'owners')
486         for lng in data['languages'].split(','):
487             result['languages'].append(lng)
488         for ctg in data['tags']:
489             result['categories'].append(ctg)
490         return result

```

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

```

491 from ext.extractor import Extractor, GameNotFound, PageNotFound
492 import requests
493 import json
494
495 class YoutubeAPI(Extractor):
496
497     def get_game(self, identifier):
498         videos = self.get_videos(identifier)
499         if len(videos) != 0:
500             result = self.manipulate_data(videos)
501             return result
502         else:
503             raise GameNotFound("Game not found!!!")
504
505     def get_videos(self, identifier):
506         videos_id = []
507         url = 'https://www.googleapis.com/youtube/v3/search?part=snippet&maxResults
=20&q={}&type=video&key={}&order=relevance'
508         response = self.get_api(url, identifier)
509         for vid in response['items']:
510             videos_id.append(vid['id']['videoId'])
511         return videos_id
512
513     def get_api(self, url, identifier):
514         response = requests.get(url.format(identifier, self.API_KEY))
515         if response.status_code == requests.codes.ok:
516             response = json.loads(response.text)
517             return response
518         else:
519             raise PageNotFound("Page not found!!!")
520
521     def manipulate_data(self, data):
522         result = {}
523         url = 'https://www.googleapis.com/youtube/v3/videos?part=statistics&id={}&key
={}'
524         sum_view = 0
525         sum_like = 0
526         sum_dislike = 0
527         for vid in data:
528             response = self.get_api(url, vid)['items'][0]['statistics']
529             if 'viewCount' in response: sum_view += int(response['viewCount'])
530             if 'likeCount' in response: sum_like += int(response['likeCount'])
531             if 'dislikeCount' in response: sum_dislike += int(response['dislikeCount']
532
533         result['view_count'] = self.temporal_data(identifier, sum_view, 'view_count')
534         result['like_count'] = self.temporal_data(identifier, sum_like, 'like_count')

```

```

535         result['dislike_count'] = self.temporal_data(identifier, sum_dislike, '
dislike_count')
536         return result

```

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

```

537 import requests
538 import json
539 from elasticsearch import Elasticsearch
540 from db.database import Database, DataNotFound
541
542 class Elastic(Database):
543
544     def __init__(self, host, index):
545         self.hostname = host
546         self.index_name = index
547         try:
548             self.elastic = self.connect_elasticsearch()
549         except ElasticNotConnected as enc:
550             raise ElasticNotConnected('Elasticsearch couldn\'t be connected!!!')
551
552     def get(self, identifier):
553         res = self.elastic.search(index=self.index_name, body={"query": {"match": {"steam_id": int(identifier)}}})
554         if res['hits']['total'] == 1:
555             data = res['hits']['hits'][0]['_source']
556             return data
557         else:
558             raise DataNotFound('Data has not found in the Database!!!')
559
560     def update(self, identifier, data):
561         if self.has_data(identifier):
562             body = {}
563             body['doc'] = data
564             self.elastic.update(index=self.index_name, doc_type='game', id=identifier, body=body)
565         else:
566             self.elastic.index(index=self.index_name, doc_type='game', id=identifier, body=data)
567
568     def delete(self, identifier):
569         url = 'http://{}/{}/game/{}'.format(self.hostname, self.index_name, identifier)
570         response = requests.delete(url)
571         if response.status_code != requests.codes.ok:
572             raise DataNotFound('Data has not found in the Database!!!')
573
574     def has_data(self, identifier):
575         url = 'http://{}/{}/game/{}'.format(self.hostname, self.index_name, identifier)
576         response = requests.get(url)
577         if response.status_code == requests.codes.ok:
578             return True
579         else:
580             return False
581
582     def connect_elasticsearch(self):
583         elastic = Elasticsearch([self.hostname])
584         if elastic.ping():

```

```

585         return elastic
586     else:
587         raise ElasticNotConnected('Elasticsearch couldn\'t be connected!!!')
588
589     def delete_index(self):
590         self.elastic.indices.delete(index=self.index_name, ignore=[400, 404])
591
592     def get_all(self):
593         res = self.elastic.search(index=self.index_name, body={"query": {"match_all":
: {}}})
594         size = res['hits']['total']
595         res = self.elastic.search(index=self.index_name, body={"size": size, "query":
{"match_all": {}}})
596         games = []
597         for game in res['hits']['hits']:
598             pair = (int(game['_id']), game['_source']['name'])
599             games.append(pair)
600         return games
601
602     def create_index(self, index_body):
603         if self.elastic.indices.exists(self.index_name): self.delete_index()
604         self.elastic.indices.create(index=self.index_name, body=index_body, ignore
=400)
605
606
607 class ElasticNotConnected(Exception):
608     pass

```

Algoritmo A.5 – Código do *Elasticsearch*