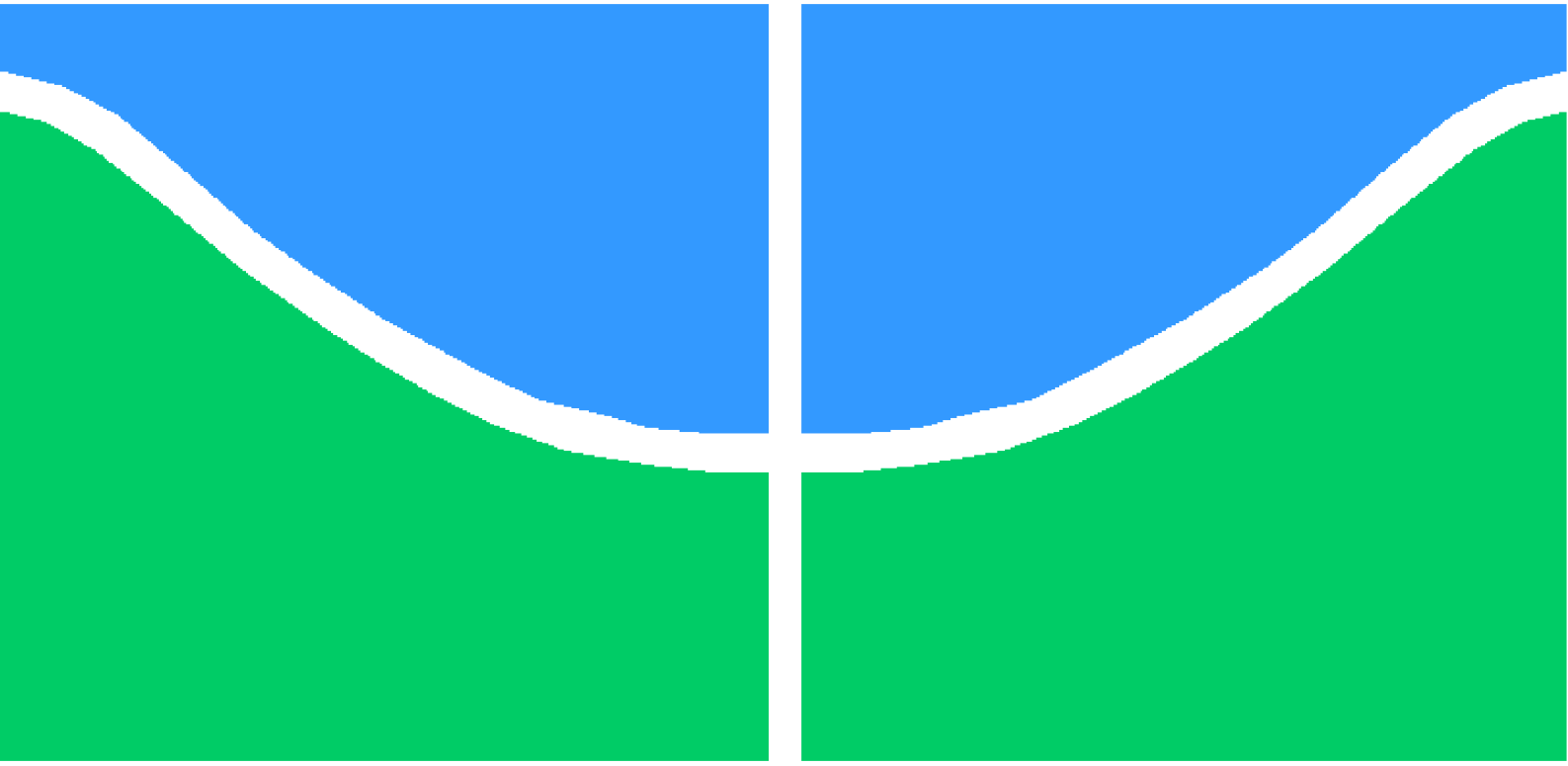

Universidade de Brasília – UnB
Faculdade UnB Gama – FGA
Engenharia de Software

Proposta de Arquitetura de Extração de Dados do Mercado de Jogos

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

Brasília, DF
2018

João Paulo Busche da Cruz

Proposta de Arquitetura de Extração de Dados do Mercado de Jogos

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
Proposta de Arquitetura de Extração de Dados do Mercado de Jogos/ João Paulo Busche da Cruz. – Brasília, DF, 2018-
46 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. Extração de Dados. I. Prof. Matheus de Sousa Faria. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Proposta de Arquitetura de Extração de Dados do Mercado de Jogos

CDU 02:141:005.6

João Paulo Busche da Cruz

Proposta de Arquitetura de Extração de Dados do Mercado de Jogos

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 de Costa Júnior
Coorientador

Prof. Dr. Fábio Macedo Mendes
Convidado 1

Brasília, DF
2018

Agradecimentos

Gostaria de agradecer a Deus por ter me dado conhecimento e oportunidades para realizar meus sonhos, e por sempre me dar força para superar as dificuldades. Aos meus pais, Suely e Paulo, por me apoiarem, quaisquer que fossem meus objetivos, e por sempre estarem presentes ao meu lado. As minhas irmãs, Amanda e Mariana, por me mostrarem a felicidade, e por nunca terem desistidos de mim. A minha namorada Sarah, que por estar ao meu lado, me trouxe forças para continuar tentando, mesmo quando achei que tudo estava perdido. Ao meu orientador Matheus Faria, que tinha paciência com minhas dificuldades e mostrava o melhor caminho para o desenvolvimento deste trabalho. Ao meu coorientados Edson Alves por suas conversas que sempre me motivavam a continuar trabalhando. Aos meus amigos, que com suas brincadeiras e zoações, nunca deixaram de me incentivar e a continuar em frente. E a todos que contribuiram diretamente ou indiretamente para a minha formação, o meu muito obrigado.

Não confunda inteligência com sabedoria,
inteligência se adquire, sabedoria se conquista.

Resumo

Atualmente o mercado de jogos possui um grande número de dados, porém, estes dados não se correlacionam, o que dificulta da extração de seus conhecimento, por isso, houve a necessidade de se criar um espaço comum para todos eles. A criação de uma arquitetura de extração, que requisita esses dados de diversas fontes e insira-os num espaço comum, poderia, assim, beneficiar futuramente a extração de métricas de negócios. Será apresentado nesse trabalho a criação de uma arquitetura de extração, que vai desde sua concepção, até a sua implementação, e posterior análise dos resultados, bem como a criação de *dashboards* para sua melhor compreensão.

Palavras-chaves: arquitetura de software. extração de dados. mercados de jogos.

Abstract

Currently the game market has a large number of data, however, these data don't have any interconnection, which makes it difficult to extract knowledge of these data, so there was a need to create a common space, where all this data is connected. Creating an extraction architecture, which requests this data from several sources and inserts it into a common space, could benefit in the future the extraction of business metrics. It will be presented in this work the creation of an extraction architecture, from conception to implementation, and for the analysis of the extraction result will be created dashboards.

Key-words: software architecture. data extraction. game market.

Lista de ilustrações

Lista de tabelas

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*
JSON	*JavaScript Object Notation*
TPS	Sistema Toyota de Produção
TCC	Trabalho de Conclusão de Curso

Sumário

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 que tange o 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 Steam[1], uma plataforma que disponibiliza jogos para downloads, uma média de 21 jogos por dia. Porém, à medida que o mercado desenvolvedor cresce, se aumenta a concorrência no meio, tornando cada vez mais difícil a aceitação do profissional pela comunidade, isso se deve, em razão da variedade de opções na hora da compra, o que faz com que algumas *features* se tornem diferencias, como por exemplo *multiplayer*, localização, dentre outras.

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 e 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 arquitetura que irá extrair dados sobre o mercado de jogos e disponibiliza-los num só lugar, possibilitando a criação de ferramentas de análise dos dados.

Objetivos

O objetivo deste trabalho é o desenvolvimento de uma arquitetura de extração de variadas fontes sobre o mercado de jogos disponibilizando-as num banco de dados comum.

Os objetivos específicos são:

- Elaborar uma arquitetura responsável por fazer a extração e a manipulação dos dados dos jogos;

[1] <https://store.steampowered.com/>

- 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.

Estrutura do Documento

Este documento é dividido em 4 capítulos. O Capítulo 1 é responsável pelo referencial teórico do projeto e análise de concorrentes. O Capítulo 2 pela metodologia, mostrando como será feito o projeto. O Capítulo 3 é responsável pelos resultados obtidos no projeto. Por fim o Capítulo 4 é responsável pela conclusão do projeto, onde também será mostrados possíveis trabalhos futuros.

1 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.

1.1 *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*).

1.1.1 *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 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 (FIELDING, 2000).

A arquitetura REST possui quatro princípios, 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

1.2 *Business Intelligence*

BI (*Business Intelligence*) é uma técnica de gerenciamento de dados, possibilitando a extração de valores de negócios das grandes pilhas de dados disponibilizadas, sejam elas de estruturas de dados ou de arquivos (LOSHIN, 2012). As análises deste dados proporcionam a criação de conhecimento, dessa maneira, originou-se vantagens inimagináveis no mercado de negócios.

Uma das maiores vantagens da utilização de BI está na otimização do processo de tomada de decisão, pois num mundo ideal, cada escolha deve ser a mais otimizada, ou seja, a que tem a melhor *performance*. Outras vantagens vão, desde a criação de póliticas de escolhas para diferentes cenários de negócios, à exposição e exploração de conhecimentos (LOSHIN, 2012).

A utilização de BI no âmbito de trabalho apresenta uma evolução na *performance* nas seguintes dimensões de negócio: valor financeiro, valor de produtividade, valor de confiança e valor de risco.

O valor financeiro é associado ao crescimento da lucratividade, sejam elas derivadas de custos ou do aumento de receitas.

O valor de produtividade é 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.

O 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.

O valor de risco é 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 análiticas e gestão de informação/conhecimento.

BI 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 tanto existem três tipos diferentes de perspectivas.

Perspectiva funcional, nela os processos focam nas tarefas relacionadas a algum tipo particular de negócio, como vendas, *marketing*, dentre outras. Processos funcionais confiam nos dados que operam dentro dos padrões de atividades comerciais.

Perspectiva interfuncional funciona como um aglomerado de processos funcionais, refletindo 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 funciona num ponto de vista organizacional, observando as características de desempenho dos processos interfuncionais, pode-se informar arquitetos empresariais e analistas de negócios as maneiras nas quais a organização pode mudar e melhorar o jeito com que as coisas são feitas. Neste ponto de vista, o dado não é mais usado apenas para executar negócios, mas para melhorá-los.

1.3 *Game Analytics*

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

Analytics é o processo de descobrir e comunicar padrões em dados, solucionando problemas de negócios ou suportando 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 atuais 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 na análise de um jogo com um produto, quanto na 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 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.

1.3.1 Telemetria

Telemetria são os dados obtidos à distância, geralmente digitais. No contexto de jogos, telemetria seria a transmissão de dados por parte do produto acerca da interação com o usuário.

Telemetria de jogos é o uso da telemetria para obtençao de dados 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).

1.3.2 *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 outras. 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 de acordo com Mellon (MELLON, 2009), sendo elas as de usuários, de *performance* e de processo. 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 onlines. 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.

1.4 Softwares Correlatos

Nesta seção são levantado os softwares correlatos, 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.

1.4.1 Steam DB

Steam DB[1] é uma ferramenta *open source* com objetivo de dar um melhor conhecimento sobre os jogos e suas atualizações disponíveis no banco de dados da Steam (STEAMDB, 2010).

Steam DB disponibiliza seus dados de maneira gratuita, porém para ter acesso a esses dados é preciso fazer login com uma conta da Steam. Estes dados estão dispostos de diversas maneiras, como *rankings* de diferentes categorias, gráficos de um jogo específico e até informações sobre o usuário. Porém, não é possível fazer contribuições a ferramenta, impossibilitando a inserção de novos dados e de novas métricas.

1.4.2 Steam Spy

Steam Spy[2] 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*.

Steam Spy dispõe seus dados de duas maneiras, uma delas é pela plataforma, como gráficos e *rankings* por categoria. A outra forma e por meio de sua API, disponibilizada para desenvolvedores criarem suas próprias ferramentas. Porém, para ter acesso total ao seus gráficos é preciso pagar por eles.

[1] <https://steamdb.info/>
[2] <http://steamspy.com/>

1.4.3 DFC Intelligence

DFC Intelligence[3] é uma ferramenta que auxiliam desenvolvedoras a tomar conhecimentos sobre estatísticas de seus jogos no mercado.

Para as desenvolvedoras que contratem o serviço da DFC Intelligence receberão métricas sobre seus jogos, e também sobre informações sobre como o mercado está agindo para jogos semelhantes. Porém, para que uma desenvolvedora possa usufruir de seus dados é preciso pagar uma taxa de consulta, além do pagamento pela ferramenta.

[3] <https://www.dfcint.com/>

2 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 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.

2.1 Arquitetura do Projeto

A arquitetura do projeto é dividida em quatro partes: as fontes de dados, o software de extração e o banco de indexação. A ligação entre as partes e sua posição na arquitetura ficam evidenciado na Figura 1.

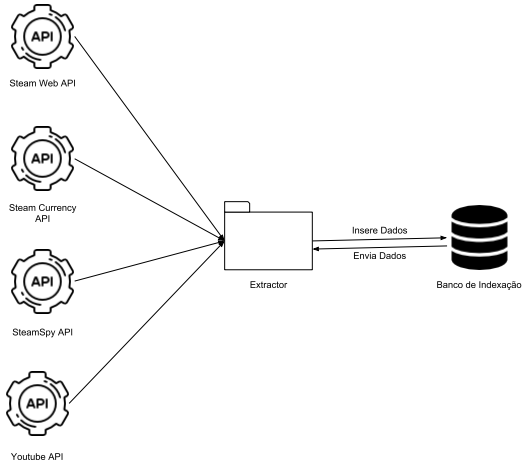

Figura 1 – Arquitura Geral do Projeto

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 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.

2.1.1 Arquitetura de Extração

A arquitetura do Extractor é composta de três classes principais e de dois arquivos de controle de inserção, como visto na Figura 2.

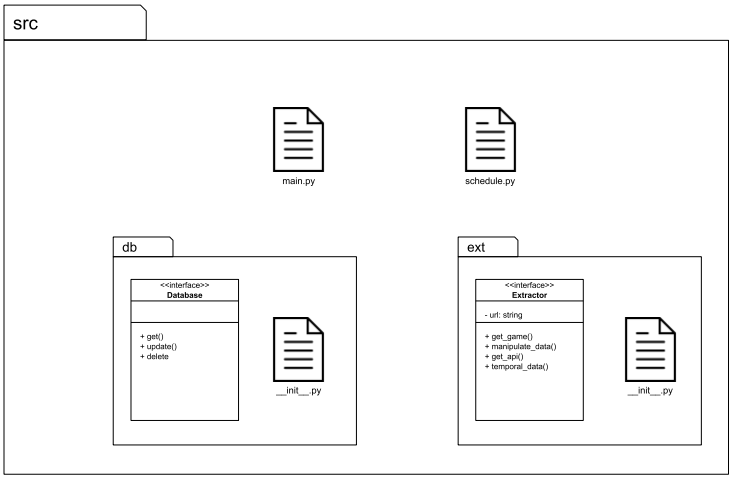

Figura 2 – Arquitura de Extração

A interface ***Extractor*** deverá ser herdada por todos as fontes de dados. Esta interface possui funções responsáveis por fazer requisições na HTTP, extrair dados desta requisição e manipular estes dados.

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

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

2.2 Análise de Ferramentas

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

2.2.1 Banco de Indexação

A ferramenta de banco de indexação é responsável pelo armazenamento dos dados extraídos das fontes de dados. A escolhida para o projeto foi o Elasticsearch. Esta 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).

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.

2.2.2 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. A escolhida para o projeto foi o Celery. Esta 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 a linguagem de programção Python[2]. Como Celery trabalha com a utilização de tarefas, podemos agendar seus funcionamentos utilizando *cronjobs*[3] para suas frequências.

2.2.3 Visualização dos Dados

Esta ferramenta será responsável por demonstrar os dados do banco de uma maneira mais intuitiva. A escolhida para o projeto foi o Kibana, esta 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).

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

Apesar da ferramente escolhida ser o Kibana, o ideal para o projeto seria a criação de uma plataforma web voltada especificamente para o contexto de jogos. Porém, pelo escopo do projeto, esta plataforma não será implementada.

2.3 Fontes de Dados

Nesta seção serão citadas as fontes de dados escolhidas para o escopo inicial. Serão utilizadas três APIs específicas do mercado de jogos, e mais uma genérica para testar a flexibilidade da arquitetura proposta.

Steam WEB API

Esta 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 utilizá-los, já os métodos privados necessitam de uma chave de desenvolvedor cedido pela própria Steamworks. Para acessar os dados da API é preciso um id, seja de usuário ou de jogo.

Ao fazer uma requisição a ***Steam WEB*** irá retornar um JSON[5] (*JavaScript Object Notation*), com as informações pedidas pela requisição. Esta é uma API dividida em interfaces, onde cada interface fornece determinados tipos de serviços, para cada interface é necessário inserir qual método irá utilizar e qual a versão do mesmo, além das informações necessárias para o funcionamento do método. Alguns exemplos de requisições podem ser vista na Figura 3.

Requisição de número de jogadores online de um jogo

```
https://api.steampowered.com/ISteamUserStats/GetNumberOfCurrentPlayers/v1/?appid=730
```

Requisição de todos aplicativos da Steam

```
https://api.steampowered.com/ISteamApps/GetAppList/v2/
```

Figura 3 – Requisições na ***Steam WEB***

Steam Store API

Esta 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 utilizar filtros para pesquisas mais específicas.

Ao fazer uma requisição a ***Steam Store*** irá retornar um JSON, com as informações pedidas pela requisição. Como os dados da Steam divergem ao serem requisitados em

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

[5] É um formato compacto, de padrão aberto independente.

diferentes regiões, ao fazer a requisição também e possível inserir paramêtros de moeda corrente e linguagem desejada. Alguns exemplos de requisições podem ser vista na Figura 4.

Requisição de detalhe de um jogo

```
https://store.steampowered.com/api/appdetails/?appids=853900
```

Requisição de detalhe de um jogo - Moeda Corrente

```
https://store.steampowered.com/api/appdetails/?appids=853900&cc=brl
```

Requisição de detalhe de um jogo - Linguagem

```
https://store.steampowered.com/api/appdetails/?appids=853900&ln=pt-br
```

Figura 4 – Requisições na ***Steam Store***

Ao se tratar do número de requisições, a ***Steam Store*** possui limitações. Após um número variável de requisições a API começa a retonar arquivos JSONs vazios, isso dificulta a inserção de dados, porém passada meia hora as requisições retornam ao normal.

Steam Spy API

Esta 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 acessá-la é necessário o id do jogo na Steam.

Ao fazer uma requisição a ***Steam Spy*** irá retornar um JSON, com as informações pedidas pela requisição. Além das informações gerais de cada jogo, esta API permite filtros, para que a busca seja mais detalhada, ou que mostre o mesmo dado de maneira diferente. Alguns exemplos de requisições podem ser vista na Figura 5.

Requisição de detalhes um jogo

```
http://steamspy.com/api.php?request=appdetails&appid=742120
```

Requisição de todos os jogos da Steam Spy

```
http://steamspy.com/api.php?request=all
```

Figura 5 – Requisições na ***Steam Spy***

Youtube API

Esta API é fornecida pelo Google Developers[6]. Ela dispõe de informações sobre cada vídeo como os números de *likes*, *dislikes* e visualizações. Vale ressaltar que a API do Youtube é a única que não é específica para jogos.

[6] <https://developers.google.com/>

Ao fazer uma requisição a ***Youtube API*** irá retornar um JSON, com as informações pedidas pela requisição. Para extrair os dados de cada vídeo é necessário conseguir uma lista de vídeos mais relevantes, após isso com o id dos vídeos pode-se extrair suas informações. Como a Steam possui muitos jogos, e alguns dele são menos populares, serão extraídas informações apenas dos 5 vídeos mais relevantes daquele jogo. Alguns exemplos de requisições podem ser vista na Figura 6.

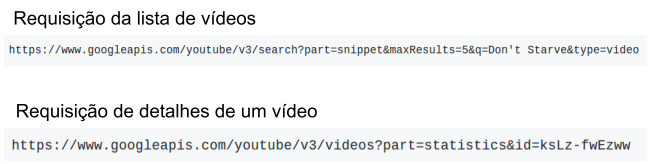

Figura 6 – Requisições na ***Youtube API***

A ***Youtube API*** possui um número máximo de requisições por dia, que são explicadas mais detalhadamente em sua documentação[7]. Esse tipo de limitação atrapalha a inserção dos dados, o que obriga que as rotinas de atualização sejam modificadas para atender a essa limitação.

A relação entre os dados que serão extraídos e de qual API será utilizado esta representado na Tabela 1.

Tabela 1 – Dados extraídos de cada API

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

[7] <https://developers.google.com/youtube/v3/getting-started>

2.3.1 Tipos de Dados

Nessa seção serão estudados os tipos de dados que serão extraídos. Como o software de extração lidará com dados que se modificam com o tempo, será preciso criar rotinas que lidarão com esses dados temporais.

2.3.1.1 Dados Temporais

Dados temporais são aqueles dados que serão atualizados com determinada periodicidade. 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 2.

Tabela 2 – Frequência de Extração de Dados

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

Os dados que pertencem a ***Youtube API***, visualizações, *likes* e *dislike*, são extraídos uma vez por semana, justamente por causa das limitações de sua API. Como sua API possui um número máximo de requisições por dia, os dados serão retirados ao decorrer da semana.

Outra função que será chamada 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.

2.3.1.2 Dados Estáticos

Dados estáticos são aqueles dados que após a primeira inserção, não são mais modificados. Dados que possuem uma taxa de modificação baixa ou inexistente, como suas modificações raramente acontecem, o acompanhamento destas mudanças não agregam valor a arquitetura. Alguns exemplos de dados estáticos são: nome, id, data de lançamento, dentre outras.

3 Resultados Obtidos

Neste capítulo são exibidos os resultados obtidos desta arquitetura de extração.

3.1 Arquitetura de Extração

Para documentar o funcionamento da arquitetura de extração foi criado um diagrama de fluxo de dados, este representado na Figura 7.

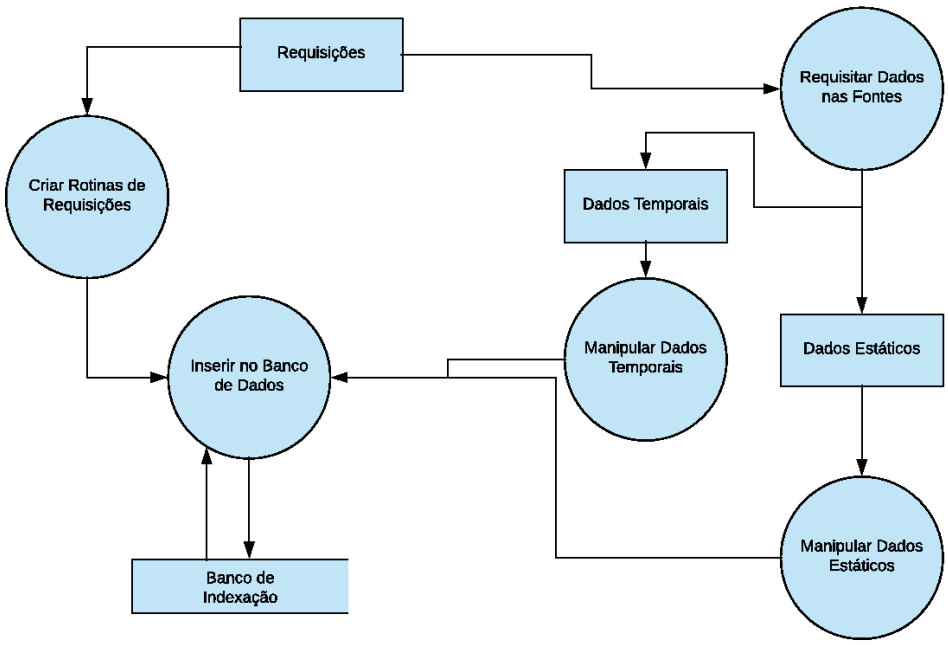

Figura 7 – Diagrama de Fluxo de Dados

Este diagrama possui três entidades, cinco processo e um armazenamento. As entidades representam as requisições que a arquitetura faz, os dados estáticos e temporais que serão manipulados. O armazenamento representa o banco de indexação onde estes dados serão inseridos.

O processo de **Requisitar Fontes de Dados** representa a funcionalidade da arquitetura de conseguir extrair dados de diversas fontes do mercado de jogos, para futuramente serem manipulados e inseridos no armazenamento.

O processo de **Criar Rotinas de Requisições** representa a funcionalidade da arquitetura de elaborar rotinas onde os dados temporais do armazenamento serão atualizados.

O processo de **Manipular Dados Estáticos** representa a funcionalidade da arquitetura de receber os dados da requisições, separar os que são estáticos, manipulá-los e prepará-los para a inserção no armazenamento.

O processo de **Manipular Dados Temporais** representa a funcionalidade da arquitetura de receber os dados da requisições, separar os que são temporais, manipulá-los e prepará-los para a inserção no armazenamento.

O processo de **Inserir no Banco de Dados** representa a funcionalidade da arquitetura de pegar os dados que foram manipulados e inseri-los no armazenamento.

3.2 Software de Extração

Seguindo a arquitetura proposta para a extração dos dados, foi verificado que a extração ocorria da maneira prevista, onde apenas nas primeiras 8 horas, foram recolhidos 8341 jogos. Porém, após as 8 horas passadas, foram identificados os gargalos da arquitetura.

E não era de conhecimento, que a API do Youtube possuía limitações de requisições. Este foi o gargalo da arquitetura, pois passadas 8 horas, nenhum jogo mais era adicionado ao banco. Após isso foi feita uma mudança significativa no software de extração, há separação entre dois tipos, assim a primeira inserção, seria feita apenas com os dados estáticos, e as atualizações com os dados temporais.

Com o problema da primeira inserção resolvido, foi pensando em como seria contornado as limitações da API do Youtube, para isso foi pensado em duas soluções. Há primeira, a qual foi implementada, foi diminuir a frequência que estes dados era requisitados, e com isso tem-se mais tempo para o tratamento dos erros que ocorreram.

A segunda seria a aquisição de diversas chaves da API, assim, seria feito uma alternância entre essas chaves para contornar as limitações da API, no momento da criação do projeto existiam 26.586 jogos no banco de dados da Steam, como cada chave insere aproximadamente 8 mil jogos, eram necessárias quatro chaves para fazer a inserção completa, porém, este tipo de abordagem possui seus problemas, pois para cada chave, era preciso ter um conta de desenvolvedor na Google Developers, por conseguinte quanto mais jogos houvesem, mais chaves seriam necessárias.

No tocante das limitações de requisições das outras APIs, como essas não atrapalhavam diretamente a inserção dos dados no Elasticsearch, a solução implementada é mais simples. Quando ocorre algum erro de alguma destas requisições, o id do jogo é guardado num arquivo, e futuramente, e realizada nova inserção dos jogos guardados no arquivo, caso ocorra erro novamente, o jogo é deletado.

Após feita a divisão dos dados, foi possível notar que, para fazer a primeira inserção

dos dados estáticos, foram inseridos 12.813 jogos, em 7 horas. Nota-se então que houve uma melhora na *performace* do algoritmo implementado.

3.3 Visualização dos Dados

Nesta seção são mostrados alguns *dashboards* criados com a ferramenta Kibana, possibilitando uma melhor visualização dos dados.

Dashboard Total

Neste *dashboard* é mostrado as relações entre todos os jogos no banco de dados da Steam. Um exemplo deste *dashboard* pode ser visto na Figura 8.

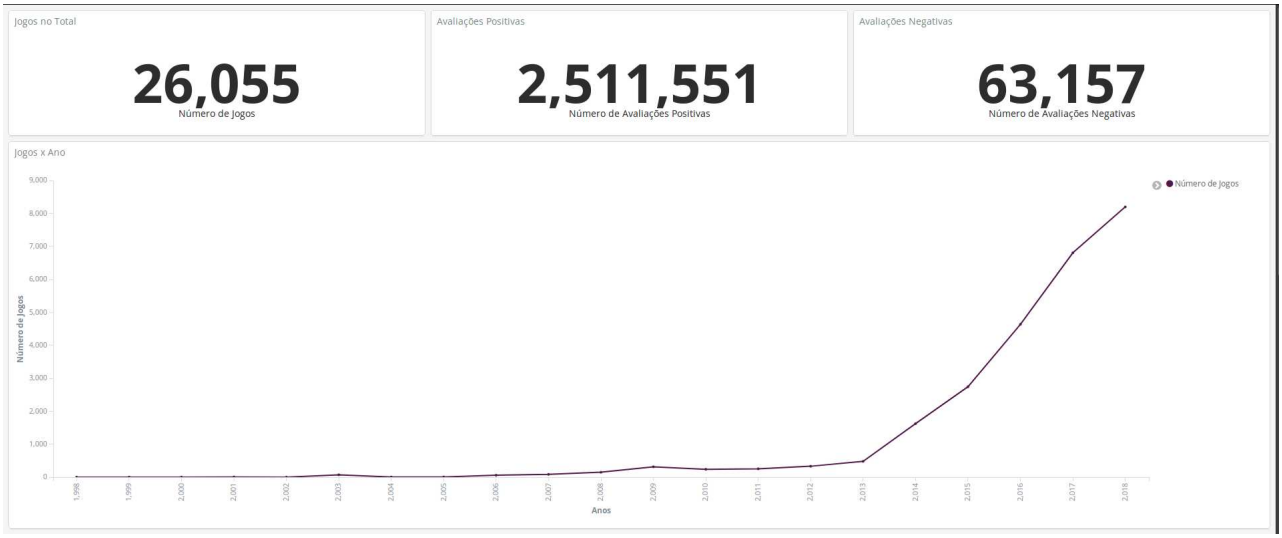

Figura 8 – *Dashboard* Total

Neste *dashboard* é possível notar que, há muito mais avaliações positivas que negativas, a partir do ano de 2013 a Steam, praticamente dobrou o número de jogos inseridos por ano, a cada ano. Também nota-se que a grande maioria dos jogos possuem suporte a língua inglesa e são feitos para plataforma Windows[1]. Outras métricas foram levantadas, como, gênero e categoria mais utilizado, desenvolvedora com mais jogos e meses com mais lançamentos.

Dashboard de Gênero

Neste *dashboard* é mostrado as relações entre os jogos de um determinado gênero. Um exemplo deste *dashboard* pode ser visto na Figura 9.

[1] <https://www.microsoft.com/pt-br/windows>

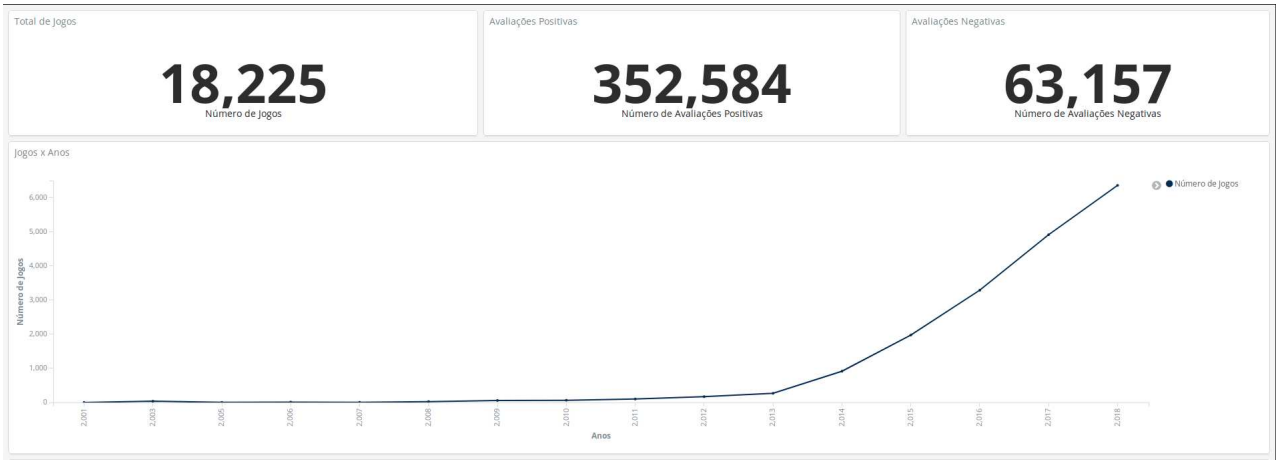

Figura 9 – *Dashboard* do Gênero Ação

Neste *dashboard* foi utilizado o gênero *indie*, pois o mesmo e o mais utilizado na Steam, nele é possível notar que foram criados mais jogos *indies* a partir do ano de 2013, coincidindo com o sucesso da Steam. Outras métricas foram levantadas, como, categorias, linguagens e plataformas mais utilizadas, desenvolvedora com mais jogos, meses com mais lançamentos e gêneros que mais foram utilizados em conjunto com o gênero *indie*.

Dashboard de Desenvolvedora

Neste *dashboard* é mostrado as informações de uma determinada desenvolvedora. Um exemplo deste *dashboard* pode ser visto na Figura 10.

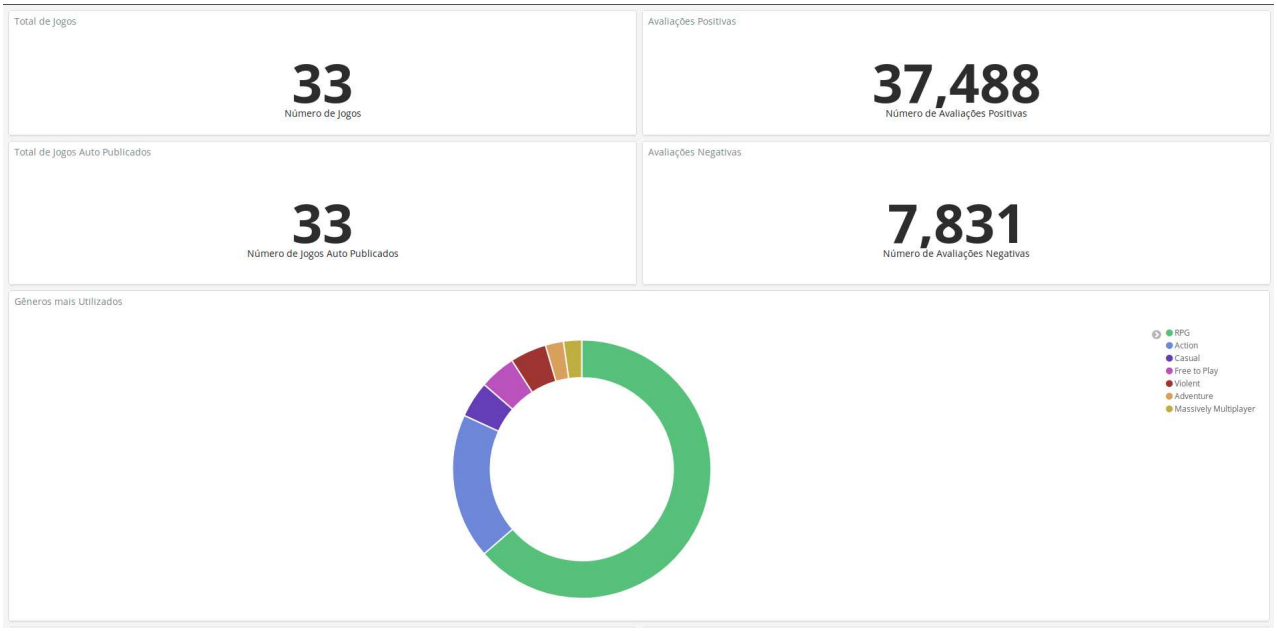

Figura 10 – *Dashboard* da Desenvolvedora Telltale Games

Neste *dashboard* foi utilizado a desenvolvedora Square Enix[2], nele é possível notar que todos os jogos desenvolvidos por ela, foram auto publicados. Outras métricas foram levantadas, como, categorias, linguagens, plataformas e gêneros mais utilizados, meses com mais lançamentos e desenvolvedoras que foram publicadas pela Square Enix.

[2] <https://www.square-enix.com/>

4 Considerações Finais

Neste capítulo são discutidos as conclusões levantadas sobre o desenvolvimento do projeto.

Durante a implementação da arquitetura, diversos desafios apareceram, muitos desses em consequência da pouco conhecimento da linguagem de programação escolhida. Dentre esses desafios, houve um que não foi resolvido no tempo dedicado a esse trabalho, sendo ele, a implementação da funcionalidade de manter os registros dos dados temporais.

Com mais maturidade sobre as ferramentas utilizadas no decorrer do projeto, percebo, que havia maneiras mais eficientes de desenvolver a arquitetura, e, por consequência, maneiras de implementar o registro de dados temporias, como, a cada chamada das tarefas de atualização das informações, fosse criado um novo documento no Elasticsearch, não inserido num já existente.

Finalmente, conclui-se que a proposta de arquitetura de extração de dados do mercado de jogos, insere e disponibilizá esses dados num só lugar, além de possibilitar a criação de ferramentas para a utilização dos mesmos.

4.1 Trabalhos Futuros

Nesta seção são levantados possíveis trabalhos que podem ser desenvolvidas a partir do presente. São eles:

- inserção de novas fontes de dados, disponibilizando uma maior número de informações sobre os jogos;
- implementação de uma API RESTful, possibilitando uma maneira mais fácil de obtenção destes dados;
- criação de métricas de negócios a partir dos dados extraídos;
- desenvolvimento de uma plataforma web para disponibilização destas métricas.

Referências

ANDERS, D.; EL-NASR, M. S.; CANOSSA, A. Game analytics: Maximizing the value of player data. Londres, 2013. Citado na página 25.

BECK, K. et al. *Manifesto Ágil*. 2001. Disponível em: <http://agilemanifesto.org/iso/ptbr/manifesto.html>. Citado na página 29.

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 31.

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

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 31.

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 27.

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 26.

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 29.

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 32.