



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-
40 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. Fábio Macedo Mendes
Convidado 1

Prof. Dra. Carla Silva Rocha Aguiar
Convidado 2

Brasília, DF
2018

Lista de ilustrações

Figura 1 – Arquitura Geral do Projeto	23
Figura 2 – Arquitura de Extração	24
Figura 3 – Requisições na <i>Steam WEB</i>	26
Figura 4 – Requisições na <i>Steam Store</i>	27
Figura 5 – Requisições na <i>Steam Spy</i>	27
Figura 6 – Requisições na <i>Youtube API</i>	28
Figura 7 – <i>Dashboard</i> do Gênero Ação	32
Figura 8 – <i>Dashboard</i> da Desenvolvedora Telltale Games	33

Lista de tabelas

Tabela 1 – Dados extraídos de cada API	28
Tabela 2 – Frequência de Extração de Dados	29

Lista de Algoritmos

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>
JSON	<i>JavaScript Object Notation</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	Arquitetura do Projeto	23
3.1.1	Arquitetura de Extração	24
3.2	Análise de Ferramentas	25
3.2.1	Banco de Indexação	25
3.2.2	Frequência na Extração dos Dados	25
3.2.3	Visualização dos Dados	25
3.3	Fontes de Dados	26
3.3.1	Tipos de Dados	28
3.3.1.1	Dados Temporais	29
3.3.1.2	Dados Estáticos	29
4	RESULTADOS OBTIDOS	31
4.1	Software de Extração	31
4.2	Visualização dos Dados	32
5	CONSIDERAÇÕES FINAIS	35
5.1	Trabalhos Futuros	35
	REFERÊNCIAS	37

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 plataforma que disponibiliza jogos para downloads, 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 diferenças, 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, disponibiliza-los num só lugar, possibilitando a criação de ferramentas de análise dos dados.

1.1 Objetivos

O objetivo deste trabalho é uma arquitetura de extração de variadas fontes para a criação de um banco comum.

Os objetivos específicos são:

- Elaborar uma arquitetura responsável por fazer a extração, 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.

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 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 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*

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, foram desenvolvidas 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 política de escolhas para diferentes cenários de negócios, a 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:

- 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.

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 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 uso da telemetria para obtenção 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](#)).

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 outras. Essas métricas são chamadas de *game metrics*. *Game metrics* possuem os mesmo potências 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 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.

2.4.1 Steam DB

Steam DB¹ 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.

2.4.2 Steam Spy

Steam Spy² 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*.

¹ <<https://steamdb.info/>>

² <<http://steamspy.com/>>

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

2.4.3 DFC Intelligence

DFC Intelligence³ 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/>>

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 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 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.

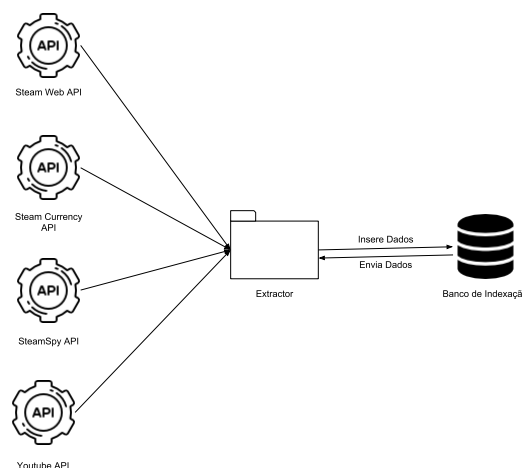


Figura 1 – Arquitetura Geral do Projeto

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.

3.1.1 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.

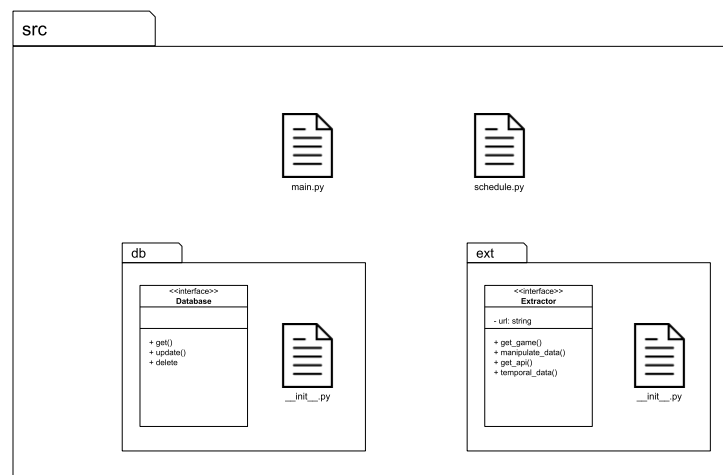


Figura 2 – Arquitetura de Extração

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 a ser utilizado. Esta interface possui funções responsáveis por inserir/atualizar, deletar e mostrar os dados guardados pelo banco.

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

3.2 Análise de Ferramentas

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

3.2.1 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 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).

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.

3.2.2 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. 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 Python². Como Celery trabalha com a utilização de tarefas, podemos agendar seus funcionamentos utilizando *cronjobs*³ para suas frequências.

3.2.3 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 ferramenta 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.

3.3 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 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.

Ao fazer uma requisição a **Steam WEB** irá retornar um JSON⁵ (*JavaScript Object Notation*), com as informações pedidas pela requisição. Esta é uma API dividida em interfaces, onde cada interface fornece determinados tipos de serviço, para cada interface é necessário passar qual método irá utilizar e qual a versão do mesmo, além das informações necessárias para o método funcionar. 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 passar 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 é possível passar parâmetros 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 retornar arquivos JSONs vazios, isso dificulta na inserção de dados, porém passados 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 acessar estes dados 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 Developers⁶, sendo assim oficial do Youtube. Ela dispõe de informações sobre cada vídeo como os números de likes, dislikes e visualizações. Vale resaltar 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 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.

Requisição da lista de vídeos

<https://www.googleapis.com/youtube/v3/search?part=snippet&maxResults=5&q=Don't+Starve&type=video>

Requisição de detalhes de um vídeo

<https://www.googleapis.com/youtube/v3/videos?part=statistics&id=ksLz-fwEzww>

Figura 6 – Requisições na **Youtube API**

A **Youtube API** possui um número máximo de requisições por dia, que são explicadas mais detalhadamente em sua documentação⁷. 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.

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

Tabela 1 – Dados extraídos de cada API

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

3.3.1 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.

3.3.1.1 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 2.

Tipo do Dado	Frequência
Média de Horas	1 vez por dia
Donos	1 vez por di
Jogadores Online	1 vez por dia
Preço	1 vez por dia
<i>Userscore</i>	1 vez por dia
Visualizações	1 vez por semana
<i>Likes</i>	1 vez por semana
<i>Dislikes</i>	1 vez por semana

Tabela 2 – Frequência de Extração de Dados

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.

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

3.3.1.2 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 acontece, o acompanhamento destas mudanças não agregam valor a arquitetura. Alguns exemplos de dados estáticos são: nome, id, data de lançamento, dentre outras.

4 Resultados Obtidos

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

4.1 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.

Inicialmente, não era de conhecimento, que a API do Youtube possuía limitações de requisições, este foi o gargalo da arquitetura, pois passada as 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 burlado as limitações da API do Youtube, para isso foi pensando em duas soluções. Há primeira, a qual foi implementada, foi diminuir a frequência que estes dados era requisitados, e com isso têm-se mais tempo para o tratamento dos erros que ocorreram.

Há segunda seria a aquisição de diversas chaves da API, assim seria feito uma alternância entre essas chaves para burla as limitações da API, no momento da criação do projeto existiam 26586 jogos no banco de dados da Steam, como cada chave inseri aproximadamente 8 mil jogos, eram preciso 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 Developers Google, assim quanto mais jogos houvessem, mais chaves seriam necessárias.

Para as limitações de requisições das outras APIs, como essas não atrapalhavam diretamente a inserção dos dados no Elasticsearch, há solução implementada é mais simples. Quando ocorria algum erro de alguma destas requisições, o id do jogo é guardado num arquivo, e futuramente, e realizado a inserção novamente dos jogos que estão gravados no arquivo, caso ocorra erro novamente, o jogo é deletado do arquivo.

Após feita a divisão dos dados, foi possível notar que, para fazer a primeira inserção dos dados estáticos, foram inseridos 12813 jogos, em 7 horas. Nota-se então que houve uma melhora na *performace* do algoritmo implementado.

4.2 Visualização dos Dados

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

Dashboard de Gênero

O primeiro *dashboard* foi criado para mostrar as relações entre o gênero de um jogo, com os outros dados estáticos. Com isso foram disponibilizados gráficos que mostram a relação de um gênero, com outros gêneros, com suas categorias, com as linguagens suportadas e com as desenvolvedoras deste gênero. Um exemplo deste *dashboard* pode ser visto na Figura 7.

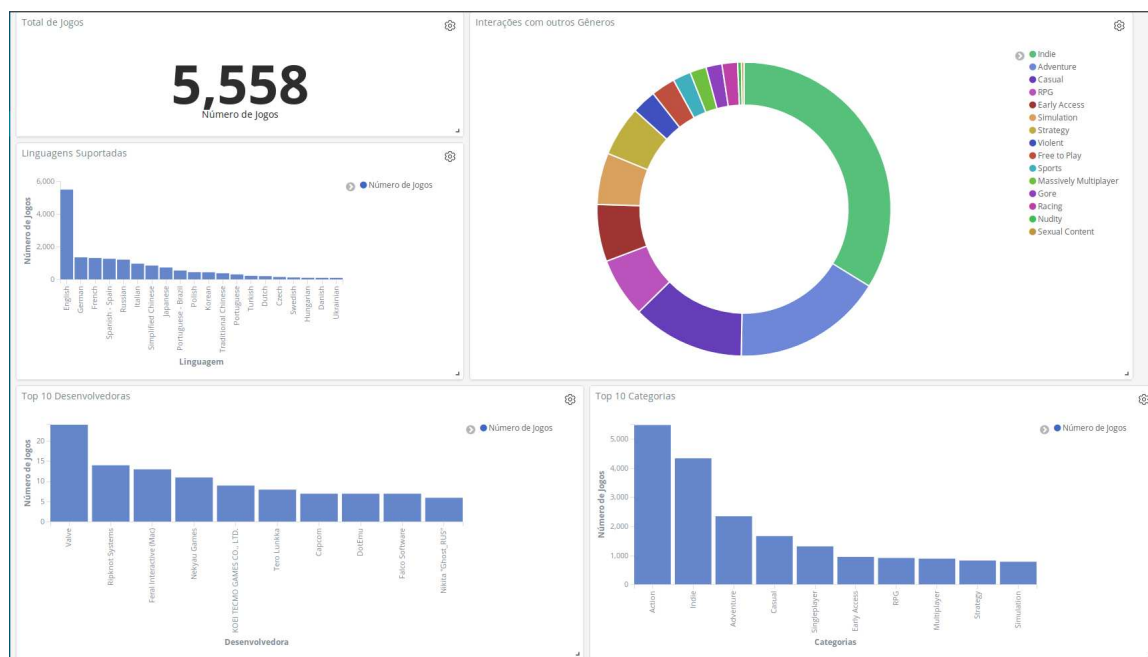
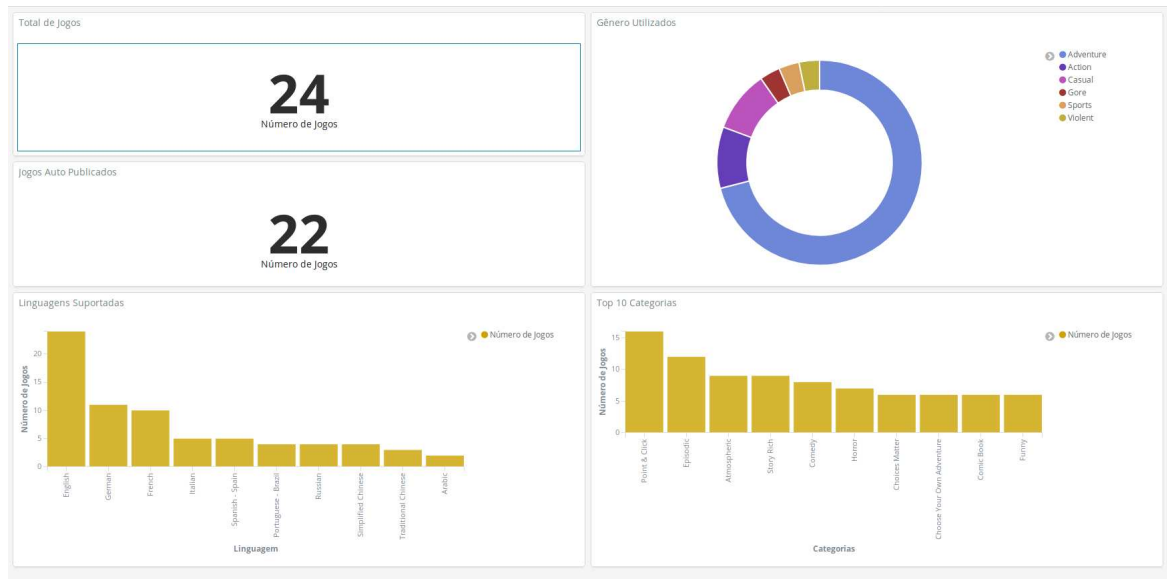


Figura 7 – *Dashboard* do Gênero Ação

Dashboard de Desenvolvedora

Também foi criado um *dashboard* para mostrar as relações entre uma desenvolvedora, com os outros dados estáticos. Para mostrar essa relação foram criadas métricas e gráficos, métricas para mostrar quantos jogos a desenvolvedora auto publicou, e gráficos para mostrar relações entre uma desenvolvedora e os gêneros utilizados, linguagens suportadas e categorias mais comuns. Um exemplo deste *dashboard* pode ser visto na Figura 8.

Figura 8 – *Dashboard* da Desenvolvedora Telltale Games

5 Considerações Finais

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

Sobre a implementação da arquitetura proposta, é concluído que, apesar dos gargalos encontrados do decorrer do projeto, a arquitetura proposta, mostrou-se capaz de completar as funcionalidades a ela atribuída. Após a divisão dos tipos de dados, em estáticos e temporais, o uso do Elasticsearch tornou-se muito mais vantajoso, o que aumento a *performance* de inserção dos dados, como também diminuiu o número de gargalos do projeto. Finalmente, posso concluir que a proposta de arquitetura de extração de dados do mercado de jogos precisa de algumas modificações, porém, consegue criar um lugar onde todos esses dados se concentre.

5.1 Trabalhos Futuros

Nesta seção são levantados possíveis trabalhos futuros no desenvolvimento do projeto.

- 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 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 25.
- DAVEPORT, T. H.; HARRIS, J. G. Competing on analytics: The new science of winning. Boston, 2007. Citado na página 19.
- 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 25.
- ELASTIC. *Kibana*. 2010. Disponível em: <<https://www.elastic.co/products/kibana>>. Citado na página 25.
- 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 21.
- 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 26.

Anexos

