

Department of Informatics

Web Applications

Report Phase 2

Group 20

27 March, 2017

Members:

- Alexander José Pereira Fernandes (MEI) - 35931, TP21;
- André Filipe Bernardes Oliveira (MI) - 45648, TP21;
- Ricardo Ferreira Vallejo (MEI) - 43338, TP21;
- Tânia Sofia Guerreiro Maldonado (MBBC) - 44745, TP22

Index

Index	2
Introduction	3
Planning	4
3. Architecture	5
4. Knowledge Base	6
4.1. Data Sources	6
4.2. Data schema (from phase 2)	7
4.3. Data Collection	7
5. Data Access Point	8
5.1. API Reference	8
5.2. Examples	8
6. Data Visualization and Interaction	11
6.1. Functionalities	11
6.2. Web Services used	12
6.3. Screenshots	13
7. Discussion	18
7.1. Work done by Alexander	18
7.2. Work done by André	18
7.3. Work done by Ricardo	18
7.4. Work done by Tânia	18
7.5. Critical analysis	18
8. Attachments	19

1. Introduction

The history of movies goes back as far as centuries ago. In the late 19th century, a single compact reel of film stored several minutes of action; today, thousands of movies can be stored in a device as small as a USB drive.

Like movies, the Internet has grown as time goes by, and as of today, it is no longer seen as an independent and disconnected information repository. What decades ago needed a full room to be stored, can now be saved, searched and viewed in a single web page, interacting with all kinds of information about that specific movie.

In this project, the aim is to explore the large set of information related with movies to build a web service that allows its users to interact with information at worldwide level, ie, searching for a particular movie will link its viewer to any information that it is attached to it - its cast, location, quotes and social media feedback.

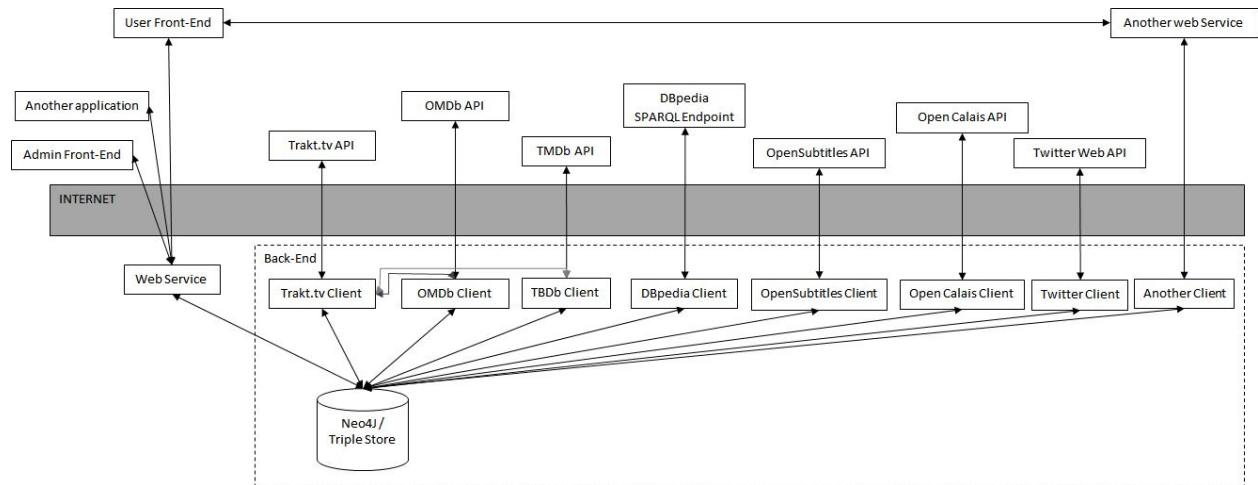
That being said, our main focus is to deliver a match between social media provided by Twitter through movie quotes and user posts, geographic locations provided by Google Maps based on movie shootings and at last, movie informations provided by imdb and track.tv.

In order to accomplish the demanded request, we will communicate with multiple diverse API's ensuring updated information and persisting this same information with the guarantee of easy and faster access for a better user experience.

2. Planning

Week	Task	Student	State	Notes	Delivery
3	Exploration of Neo4J	André	Finished		Phase 0 (6-Mar)
3	Exploration of Trakt API	Alexander	Finished		
3	Exploration of OMDb and TMDB APIs	André	Finished		
3	Exploration of Twitter API	Tânia	Finished		
3	Exploration of OpenSubtitles	Ricardo	Finished		
3	Exploration of Open Calais API	Ricardo	Finished		
4	Java and html/javascript project setup	Alexander	Finished		Phase 1 (27-Mar)
4	Database setup	André	Finished		
4	Trakt data collection service	Alexander	Finished		
4	OMDb/TMDB data collection service	André	In Progress		
4	OpenSubtitles data collection service	Ricardo	Finished		
4	Initial deployment of our data API	André	Finished		
5	Twitter data collection service	Tânia	Finished		
5	front-end user feedback and crowdsourcing				
6	Initial implementation of front-end interface	Alexander	Finished		Phase 2 (4-May)
8	Exploration of Wikipedia API	Ricardo		Easter	
9	Collect data from Wikipedia	Ricardo		After SPARQL class	
10	Our API improvements	André	Finished		
10	Identify quotes from tweets	Tânia			
11	Front-end interface improvements	Alexander			
12					
13					
14					Phase 3 (18-May)

3. Architecture



1. System Architecture Schema.

The above schema displays our system organization with most of its components (Frontend, Backend and NoSQL Database). For the backend we have chosen to use Java as the main programming language because every member of the group has prior experience.

For the network-based software architecture style we have chosen the Representational State Transfer more commonly called REST, because of its simplicity. More specifically, a RESTful web service where every resource is identified by an URI and resources are handled using POST, GET, PUT, DELETE operations (verbs) which are similar to Create, Read, Update and Delete (CRUD) operations. Our external web service communication will be made (preferably) with the MIME-type JSON (if it is not available, by XML) via the HTTP protocol and our internal web service communication is restricted to JSON. Our web services will be tested using Postman.

The chosen API for the REST support is JAX-RS (Java API for RESTful Web Services), where the use of annotations defines the REST relevance of Java classes. We will be using a library to implement this RESTful webservices in a Java servlet container called Jersey which is the reference implementation for the Java Specification Request (JSR) 339.

As web server and web container we have chosen Tomcat (who combines them and can handle HTTP requests/responses and implements Java Servlet API into one).

On the Persistence side we will be using Neo4j graph database, which allow us to query complex matches through semantic. Alternatively a Triple Store can be used, but Neo4j is preferable because of its simplicity and because it has a graphical interface which enables the visualization of the database's content.

For the communication between backend and frontend, we will use the Java Server Pages (JSP),

beautified with HTML, Javascript and CSS.

4. Knowledge Base

4.1. Data Sources

General movie information:

- Trakt.tv API (<http://docs.trakt.apiary.io>) - main source of movie information including IMDb's id. The methods used are: GET /movies/trending (Get trending movies); GET /movies/popular (Get popular movies); GET /movies/id (Get a movie); GET /movies/id/people (Get all people for a movie)
- IMDb information - using OMDb API (<http://www.omdbapi.com/>) and/or The Movie Database API (<https://www.themoviedb.org/documentation/api>) with the id's returned from Trakt.tv request. Information will be retrieved specifying the id of the movie: parameter i for OMDb API and GET /movie/{movie_id} for TMDb.
- Wikipedia information - access DBpedia SPARQL Endpoint; data is obtained through SPARQL queries
- OMDb (<http://www.omdbapi.com/>) - website that makes available additional movie information and movie images.

Movies subtitles:

- OpenSubtitles API (<http://trac.opensubtitles.org/projects/opensubtitles/wiki/XMLRPC>) - subtitles will be obtained using the search method SearchSubtitles (providing the IMDb ID) and after they are found, they can be downloaded through the DownloadSubtitles method.
- Google Cloud Natural Language API (<https://cloud.google.com/natural-language/>) - to parse and retrieve important content from the subtitles (semantic). English subtitles can be sent to Google Cloud Natural Language API with POST method available. Possible alternatives are Geographic Ontology like W3C (<https://www.w3.org/2003/01/geo/>) or GeoNames Ontology (<http://www.geonames.org/ontology/documentation.html>).
- quodb (<http://www.quodb.com/>) - website that returns a movie list that match a given string (quote). Optional, the idea was to use the subtitles file. Currently unavailable API, right now, data can only be obtained by Web scraping.

Social network:

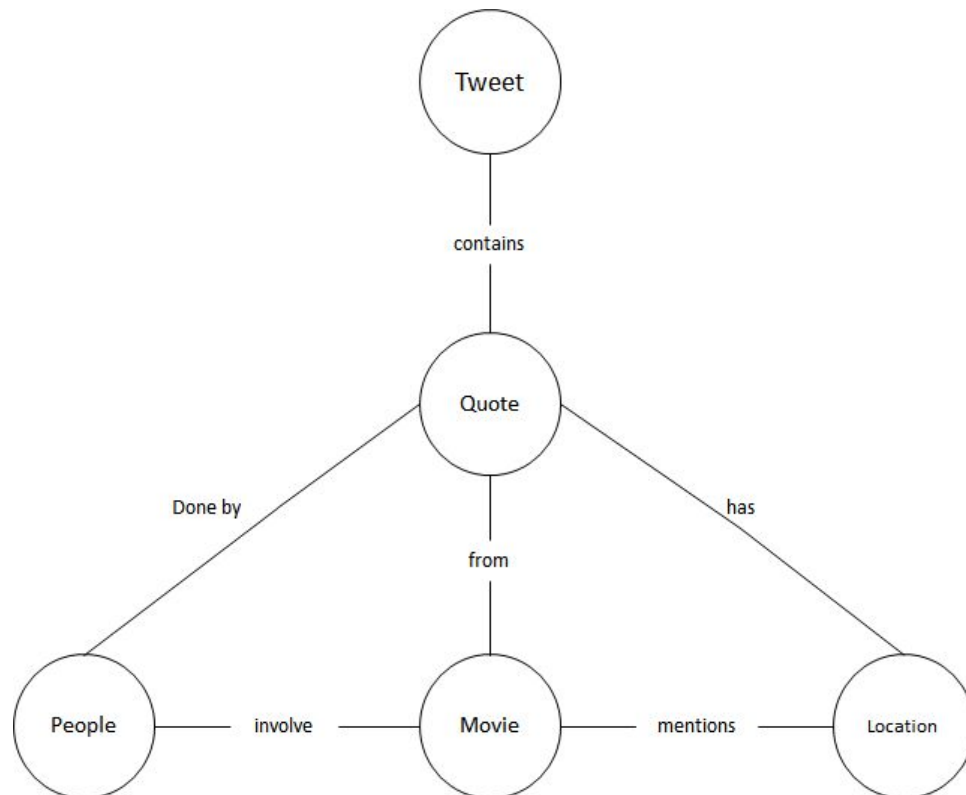
- Twitter (<https://dev.twitter.com/overview/api>) - to movie related searches: movie title, cast members or quotes (GET search/tweets), or to tweet automatically about a given movie (POST statuses/update).

RSS Feed:

No RSS feed was used because no adequate use (that fitted our view of the project) was found to justify using it. The most appropriate findings were Metacritic reviews (<http://www.metacritic.com/rss>) or future premiers ("Coming Soon"), but they were not

relevant enough to be included in our system. If an adequate RSS feed is found that contains relevant information (critics, news, etc) about the movies, this section will be updated in future deliveries.

4.2. Data schema (from phase 2)



2. Data Schema with most relevant entities, and their associations.

4.3. Data Collection

For the general movie information the various Rest APIs will be used for data collection, minimal information (e.g. references and key information) will be stored on the database from these, the remaining will be fetched as needed, for the users.

OpenSubtitles API will be used to fetch subtitles which will be later supplied to Open Calais API to parse and return key entities such as location and people mentioned in the movie, which will be stored on the database.

QuoDB web site can be used to fetch movie's quotes which will be stored and later compared from tweets (from twitter API) to identify movie popularity statistics.

We will develop most of the clients to consume the external API's since no good alternative was found.

5. Data Access Point

5.1. API Reference

The following resources and collections are provided:

Resource	Method	Returns
/rest/movie/{m_uri}	GET	movie information
/rest/movie/{m_uri}	PUT	insert/edit movie information
/rest/movie/{m_uri}	DELETE	delete movie information
/rest/movie/{m_uri}/locations	GET	locations mentioned on the movie
/rest/movie/{m_uri}/locations	POST	insert locations mentioned on the movie
/rest/movie/{m_uri}/locations/{l_uri}	GET	information about and specific location
/rest/movie/{m_uri}/locations/{l_uri}	PUT	edit information about an specific location
/rest/movie/{m_uri}/locations/{l_uri}	DELETE	delete information about an specific location
/rest/movie/{m_uri}/people	GET	People mentioned on the movie
/rest/movie/{m_uri}/people/{p_uri}	PUT	insert new people mentions on the movie
/rest/movie/{m_uri}/people/{p_uri}	DELETE	delete information about an specific person mentioned on the movie
/rest/movie/{m_uri}/quote	GET	quotes from the movie
/rest/movie/{m_uri}/quote	POST	insert new information about a quote from the movie
/rest/movie/{m_uri}/quote/{q_uri}	GET	information about an specific quote
/rest/movie/{m_uri}/quote/{q_uri}	PUT	edit information about an specific quote from the movie
/rest/movie/{m_uri}/quote/{q_uri}	DELETE	delete information about an specific quote from the movie
/rest/movie/{m_uri}/quote/{q_uri}/tweet	GET	tweets related to thes specific quote
/rest/movie/{m_uri}/quote/{q_uri}/tweet	POST	add tweets related to the specific quote
/rest/movie/{m_uri}/quote/{q_uri}/tweet	DELETE	remove tweet from quote relation
/rest/movie/{m_uri}/tweet	POST	write tweet about a given movie

All requests use REST architecture, and when the output is required, it follows JSON format.

5.2. Examples

1. Get movie list

```
[
  {
    "title":"The Dark Knight",
    "uri":"TheDarkKnight"
  },
  {
```



```

    "title":"Inception",
    "uri":"Inception"
  },
  {
    "title":"Guardians of the Galaxy",
    "uri":"GuardiansOfTheGalaxy"
  },
  {
    "title":"The Avengers",
    "uri":"TheAvengers"
  },
  {
    "title":"Suicide Squad",
    "uri":"SuicideSquad"
  },
  {
    "title":"The Matrix",
    "uri":"TheMatrix"
  },
  {
    "title":"Interstellar",
    "uri":"Interstellar"
  },
  {
    "title":"Doctor Strange",
    "uri":"DoctorStrange"
  },
  {
    "title":"Star Wars: The Force Awakens",
    "uri":"StarWarsTheForceAwakens"
  },
  {
    "title":"Deadpool",
    "uri":"Deadpool"
  }
]

```

2. Get movie details

```

{
  "overview":"Based upon Marvel Comics's™ most unconventional anti-hero, DEADPOOL tells the origin story of former Special Forces operative turned mercenary Wade Wilson, who after being subjected to a rogue experiment that leaves him with accelerated healing powers, adopts the alter ego Deadpool. Armed with his new abilities and a dark, twisted sense of humor, Deadpool hunts down the man who nearly destroyed his life."
}

```

```

"runtime":108.0,
"title":"Deadpool",
"uri":"Deadpool",
"certification":"R",
"trailer":"http://youtube.com/watch?v\u003dONHBaC-pfsk",
"genres":[
  "action",
  "adventure",
  "comedy",
  "romance"
],
"id_trakt":"190430",
"id_imdb":"tt1431045",
"tagline":"Witness the beginning of a happy ending",
"imdb_rating":8.20508,
"url_trakt":"https://trakt.tv/movies/deadpool-2016",
"released":"2016-02-12",
"homepage":"http://www.foxmovies.com/movies/deadpool",
"id_tmdb":"293660"
}

```

3. Get movie locations

```

{
  "movie": "movie_id",
  "locations":
  [
    {
      "id": "location_id_1",
      "latitude": 36,
      "longitude": -6,
      "location": "somewhere on portugal",
      "quote": ["quote_id_1", "quote_id_2",...]
    },
    ...
  ]
}

```

4. Get movie cast members

```

{
  "movie": "movie_id",
  "people":

```

```

{
  {
    "id": "people_id_1",
    "name": "The character/person name",
    "external_ids":
    {
      "wiki": "wiki_id",
      ...
    }
  },
  ...
}

```

5. Get movie quotes

```

{
  "movie": "movie_id",
  "quotes":
  [
    {
      "id": "quote_id",
      "text": "the quote from the movie",
      "timing": "1:39:45",
      "location": ["location_id_1", "location_id_2", ...]
    },
    ...
  ]
}

```

6. Data Visualization and Interaction

6.1. Functionalities

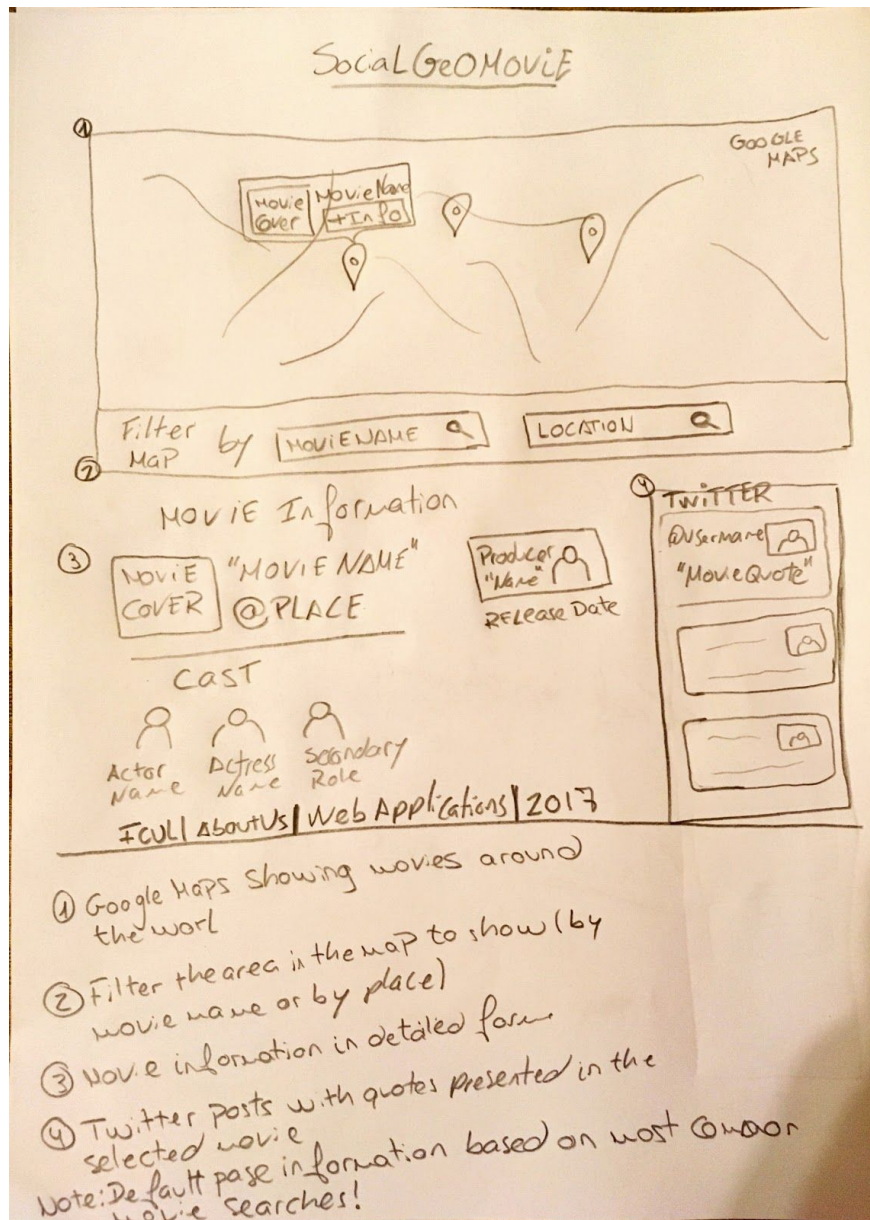
Our website will allow users to see a correlation between movies and places that are connected, in some way, with those movies (e.g. mentioned in the movie/subtitle, places the movie was shot on, etc). This will allow users to explore movies in a different way (navigating through places) and find movies that the user might like, because a similar set of places is present/is mentioned.

Another key aspect about movie recommendations are social networks, therefore this system will allow users to identify the most talked movies in social networks. We will exhibit the most quoted movies on twitter as it can be viewed as a metric of popularity.

6.2. Web Services used

Data Visualization will require additional web services. We will use Maps Embed API (<https://developers.google.com/maps/documentation/embed/guide>) to display a map with the referenced locations for each movie in our database. This will require the use of Geocoding API (<https://developers.google.com/maps/documentation/geocoding/start>) to convert known locations to geographic coordinates.

6.3. Screenshots



3. Interface Sketch

```
localhost:8080/aw2017/rest/movie/
[{"title":"The Dark Knight","uri":"TheDarkKnight"}, {"title":"Inception","uri":"Inception"}, {"title":"Guardians of the Galaxy","uri":"GuardiansOfTheGalaxy"}, {"title":"The Avengers","uri":"TheAvengers"}, {"title":"Suicide Squad","uri":"SuicideSquad"}, {"title":"The Matrix","uri":"TheMatrix"}, {"title":"Interstellar","uri":"Interstellar"}, {"title":"Doctor Strange","uri":"DoctorStrange"}, {"title":"Star Wars: The Force Awakens","uri":"StarWarsTheForceAwakens"}, {"title":"Deadpool","uri":"Deadpool"}]
```

4. Json Response Example (as the first example)

Tabela com todos os filmes		Tabela com 20 pessoas		Tabela com cast do filme id: Deadpool	
Title	ID	Name	ID	Name	ID
The Dark Knight	TheDarkKnight	Ryan Reynolds	RyanReynolds	Dan Zachary	DanZachary
Inception	Inception	Morena Baccarin	MorenaBaccarin	Olesia Shewchuk	OlesiaShewchuk
Guardians of the Galaxy	GuardiansOfTheGalaxy	Ed Skrein	EdSkrein	Anthony J. Sacco	AnthonyJSacco
The Avengers	TheAvengers	T.J. Miller	TJMiller	Sean Quan	SeanQuan
Suicide Squad	SuicideSquad	Gina Carano	GinaCarano	Michael Neumeyer	MichaelNeumeyer
The Matrix	TheMatrix	Leslie Uggams	LeslieUggams	David Longworth	DavidLongworth
Interstellar	Interstellar	Brianna Hildebrand	BriannaHildebrand	Greg LaSalle	GregLaSalle
Doctor Strange	DoctorStrange	Karan Soni	KaranSoni	Tony Chris Kazoleas	TonyChrisKazoleas
Star Wars: The Force Awakens	StarWarsTheForceAwakens	Jed Rees	JedRees	Matthew Hoglie	MatthewHoglie
Deadpool	Deadpool	Stefan Kapičić	StefanKapicic	David Hardware	DavidHardware
		Randal Reeder	RandalReeder	Victoria De Mare	VictoriaDeMare
		Isaac C. Singleton Jr.	IsaacCSingletonJr	Fabiola Colmenero	FabiolaColmenero
		Stan Lee	StanLee	Heather Ashley Chase	HeatherAshleyChase
		Michael Benyaer	MichaelBenyaer	Rob Hayter	RobHayter
		Style Dayne	StyleDayne	Paul Lazenby	PaulLazenby
		Kyle Cassie	KyleCassie	Rachel Sheen	RachelSheen
		Taylor Hickson	TaylorHickson	Benjamin Wilkinson	BenjaminWilkinson
		Ayzee	Ayzee	Jason Day	JasonDay

5. Draft HTML

Social Geo Movie

Home

Geo Movies

About Us



6. Initial page with movie list

Social Geo Movie

[Home](#)[Geo Movies](#)[About Us](#)

Deadpool

Witness the beginning of a happy ending

Rating: 8.19905663010411

Runtime: 108

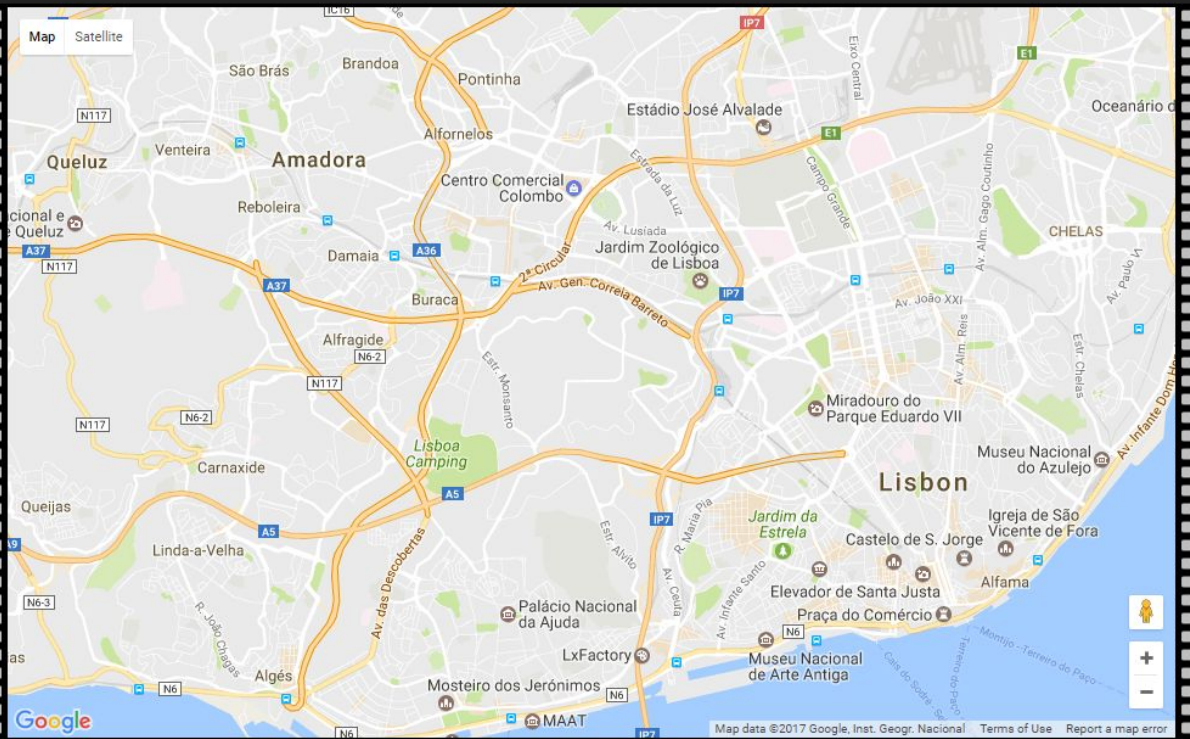
Release: 2016-02-12T00:00:00.000Z

Based upon Marvel Comics' most unconventional anti-hero, DEADPOOL tells the origin story of former Special Forces operative turned mercenary Wade Wilson, who after being subjected to a rogue experiment that leaves him with accelerated healing powers, adopts the alter ego Deadpool. Armed with his new abilities and a dark, twisted sense of humor, Deadpool hunts down the man who nearly destroyed his life.

Cast

[Dan Zaccary](#)
[Olivia Shewchuk](#)
[Anthony J. Sacco](#)
[Sean Quan](#)
[Michael Neumeyer](#)
[David Longworth](#)
[Greg LaSalle](#)
[Tony Clana Kazoleas](#)
[Matthew Hogle](#)
[David Hardware](#)
[Victoria De Marc](#)
[Fabiola Cobencro](#)
[Heather Ashley Chase](#)
[Rob Hayter](#)
[Paul Lazebny](#)
[Rachel Sheen](#)
[Benjamin Wilkinson](#)
[Jason Day](#)
[Kyle Rideout](#)
[Darcy Johnson](#)
[Paul Liebato](#)
[Chad Riley](#)

7. Movie information with corresponding cast



8. Map page for showing geographic information

7. Discussion

7.1. Work done by Alexander

Implemented the first draft of the front end to display available information from the database.

The frontend is now able to display all movies within our database and information about them, including their descriptions, ratings, runtime, release and complete cast.

Implemented a some changes on the server to import some additional information from OMDb (poster).

7.2. Work done by André

Improvements in data collection: now we can update the information that is collected without overwriting it. Get new data from trakt.tv and process the existing one.

Development of the remaining REST methods of our API to interact with the available data.

7.3. Work done by Ricardo

Added Google Cloud Natural Language API support, explored and started implementation of DBPedia service. Helped in the development of REST methods.

7.4. Work done by Tânia

Improvements in the Twitter collection client, that can now properly return tweets based on movie names or persons (or any other filter, like language or localization, if desired). Further work on filtering quotes from tweets is still required.

7.5. Critical analysis

At this point we don't have much more data than in the previous phase, but we are continuing our pipeline with the existing one: creating web pages to show the list of movies and movie details and also process the data better. The focus of this phase was to draft the website and improve the methods developed in the previous step.

8. Attachments

Examples from the API's that will be used:

1. Find Movie Quotes within Twitter Posts:

<https://api.twitter.com/1.1/search/tweets.json?q=portugal>

```
{
  "created_at": "Sat Mar 04 11:50:39 +0000 2017",
  "id": 837993690182402000,
  "id_str": "837993690182402049",
  "text": "RT @SouBenfica1904: 4 de Março de 1992... A data em que nascia uma lenda, sem igual em Portugal. Muitos parabéns, rapazes. 25 anos! "Sempre...",
  "truncated": false,
  "entities": {
    "hashtags": [],
    "symbols": [],
    "user_mentions": [
      {
        "screen_name": "SouBenfica1904",
        "name": "Sou Benfica",
        "id": 2842529048,
        "id_str": "2842529048",
        "indices": [
          3,
          18
        ]
      }
    ],
    "urls": []
  },
  "metadata": {
    "iso_language_code": "pt",
    "result_type": "recent"
  }
}
```

2. Query movie information on Trakt.tv API:

<https://api.trakt.tv/movies/tron-legacy-2010>

```
{
  "title": "TRON: Legacy",
}
```

```
"year": 2010,
"ids": {
  "trakt": 343,
  "slug": "tron-legacy-2010",
  "imdb": "tt1104001",
  "tmdb": 20526
},
"tagline": "The Game Has Changed.",
"overview": "Sam Flynn, the tech-savvy and daring son of Kevin Flynn, investigates his father's disappearance and is pulled into The Grid. With the help of a mysterious program named Quorra, Sam quests to stop evil dictator Clu from crossing into the real world.",
"released": "2010-12-16",
"runtime": 125,
"updated_at": "2014-07-23T03:21:46.000Z",
"trailer": null,
"homepage": "http://disney.go.com/tron/",
"rating": 8,
"votes": 111,
"language": "en",
"available_translations": [
  "en"
],
"genres": [
  "action"
],
"certification": "PG-13"
}
```