UNIVERSITÀ DI PISA

Computer Engineering, Artificial Intelligence and Data Engineering

Large-Scale and Multi-Structured Database

PokèMongo

Project Documentation

TEAM MEMBERS:
Edoardo Fazzari
Mirco Ramo
Olgerti Xhanej

Academic Year: 2020/2021

Contents

1 Introduction

PokeMongo is a gaming application in which users compete each other to build up the best Team choosing between the set of Pokémon available.

1.1 Description

Every **User** can build up his own team. Every **Team** is composed by up to 6 distinct **Pokémon** and is assigned to a numerical value (points) based on features and properties of the chosen Pokémon, for ranking purposes.

A **User** can also follow other users in order to make new friends basing on common friends or common interests. Moreover users can express sentiments on **Pokémon**, choosing their favorite ones and posting or commenting on them.

Users can also navigate through the ranking in order to visualize the best teams (according to the values cited before) and the most used/caught **Pokémon**, both among their friends, grouped by country and among worldwide players.

User can browse for a specific **Pokémon** using the *Pokédex* tool, in which he/she can lookup for **Pokémon** according to search filters like *Pokémon name*, *Type* or *Point*s.

Moreover, as a "real" Pokémon Trainer, the **User** is invited to *Catch 'em' all*, i.e. to try to get a new **Pokémon** in order to create/update his/her own Team. Thus, it is provided to the **User** a prefix number of *daily Pokéball* to be used to try to capture them. At each **Pokémon** is associated a probability to catch it, the higher the Pokémon's value, the lower the probability.

Furthermore, the **User** can exploit the social network structure of the application to make new **Friends** and discover new **Pokémon**. Indeed, he/she can search for new friends by *username* or choosing them among the provided recommended friends list. The **User** can choose his/her **favorite Pokémon**, obtaining in this way a shortcut to catch it faster, and can post or answer to **Posts** in order to express his/her opinion on that **Pokémon**.

In addition, to extend the dynamic behavior of the application, the *catch rate* (i.e. the probability to get a Pokémon using a Pokéball) changes in time depending on the number of **Users** who have that **Pokémon**: *the more it is popular, the harder will be to catch it.* Since the rankings' points are computed based on the catch rate, the winning strategy could be on predicting which **Pokémon** will become popular in the near future and try to get it

early! Every **User** has access to the visualization of the temporal drift of the catch rate.

The safeguard and the improvement of the application is in charge of **Admin** users. They are able to *ban mischievous users*, *delete inappropriate posts or comments*, *add/remove Pokémon* to the collection, *consult geo-temporal usage statistics* which are useful to make new business plans.

2 Analysis

2.1 Functional Requirements and Use Cases

2.1.1 Use Cases List

2.1.2 UML Use Cases Diagram

2.2 Non-Functional Requirements

2.3 Sources, Velocity properties and Volume of data

2.4 UML Entities Diagram

2.5 Main application queries

3 Project

3.1 Adopted Databases

3.2 Document Database

3.2.1 Queries Handled

3.2.2 Entities handled

3.2.3 Collections structure

3.2.4 Indexes

3.3 Graph Database

3.3.1 Queries Handled

3.3.2 Entities handled

3.3.3 Graph structure

3.3.4 Indexes

3.4 Redundancies and consistency management

3.5 Database Properties

3.5.1 Availability

3.5.2 Replicas

3.5.3 Eventual Consistency

3.5.4 Sharding

3.5.5 Pros and Drawbacks

3.6 Client, Server and Daemon Thread

3.7 Technologies and Frameworks

4 Implementation

4.1 Package structure and information hiding

4.1.1 Packaging strategy and information hiding

4.1.2 UML package diagram

4.2 APIs and SPIs

4.3 Main tools

4.3.1 GSON

4.3.2 Caching mechanism and multimedia management

4.3.3 Password Encryptor

4.3.4 Logger

4.4 Analytics queries

4.4.1 User Rankings

4.4.2 Pokémon Rankings

4.4.3 Usage Statistics

4.4.4 Dynamic Catch Rate

4.5 Business logic

4.5.1 Points computing

4.5.2 Dynamic Catch Rate Computing

5 Test

5.1 Privacy and Security

5.2 Unit Test

5.3 Robustness

5.4 Performance