****

*Large Scale and Multi-structured Databases*

*PokeMongo: Project Documentation*

Edoardo Fazzari, Mirco Ramo, Olgerti Xhanej

Summary

[1. Introduction 3](#_Toc56958848)

[2. Interface Mock-up 4](#_Toc56958849)

[3. Requirements 7](#_Toc56958851)

[3.1 Functional requirements and use cases 7](#_Toc56958852)

[3.2 Non-functional requirements 8](#_Toc56958853)

[3.3 Uml Relation Diagram 8](#_Toc56958854)

[4.Queries and Database Structure 9](#_Toc56958855)

[4.1 Main DB queries 9](#_Toc56958856)

[4.2 Json collections 9](#_Toc56958857)

[5 Further Ideas 10](#_Toc56958858)

# Introduction

PokeMongo is a gaming application in which users compete each other to build up the best Team choosing from the set of Pokemon available in the environment. Every user can make just one single Team.

Every Team is composed by up to 6 distinct Pokemons and is assigned to a numerical value based on features and properties of the chosen Pokemons, for ranking purposes.

Users can also navigate through the ranking in order to visualize the best teams (according to the values cited before), most used/caught Pokemons.

The user can also search a specific Pokemon using the Pokedex tool, in which he/she can browse Pokemons according to specific search filters (e.g. Pokemon name, Type, Points…).

Moreover, as a “real” Pokemon Trainer, the user is invited to “Catch ‘em ‘all”, i.e. to catch Pokemon in order to create/update his own team. Thus, it is provided to the user a prefix number of daily Pokeball to be used to try to catch them.

At each Pokemon is associated a probability to catch it, the higher the Pokemon’s value, the lower the probability.

-Creating a “social” structure in which users can follow each other in order to share his/her own team

-Dalla GUI vengono consigliati amici in base agli attuali amici, (pokemon in comune)

-Sfruttare I tipi diversi per punti bonus

-Punti = (255-catchRateBase \* 1.5) if tipi tutti diversi in 6 pokemon

-catchRate = catchRateBase \* (1-%utentiChePossiedono)

Analytics:

-Ranking di punti per tutti gli utenti

-Ranking di punti per i tuoi amici

-Ranking di punti per country

-Ranking pokemon più posseduti nei team per tutti gli utenti

-Ranking pokemon più posseduti nei team per country

-Andamento(windowing) del catchRate nel tempo

-Numero di utenti iscritti nel tempo/login a giorno(Admin only)

# Interface Mock-up

LOGIN

## 

HOMEPAGE

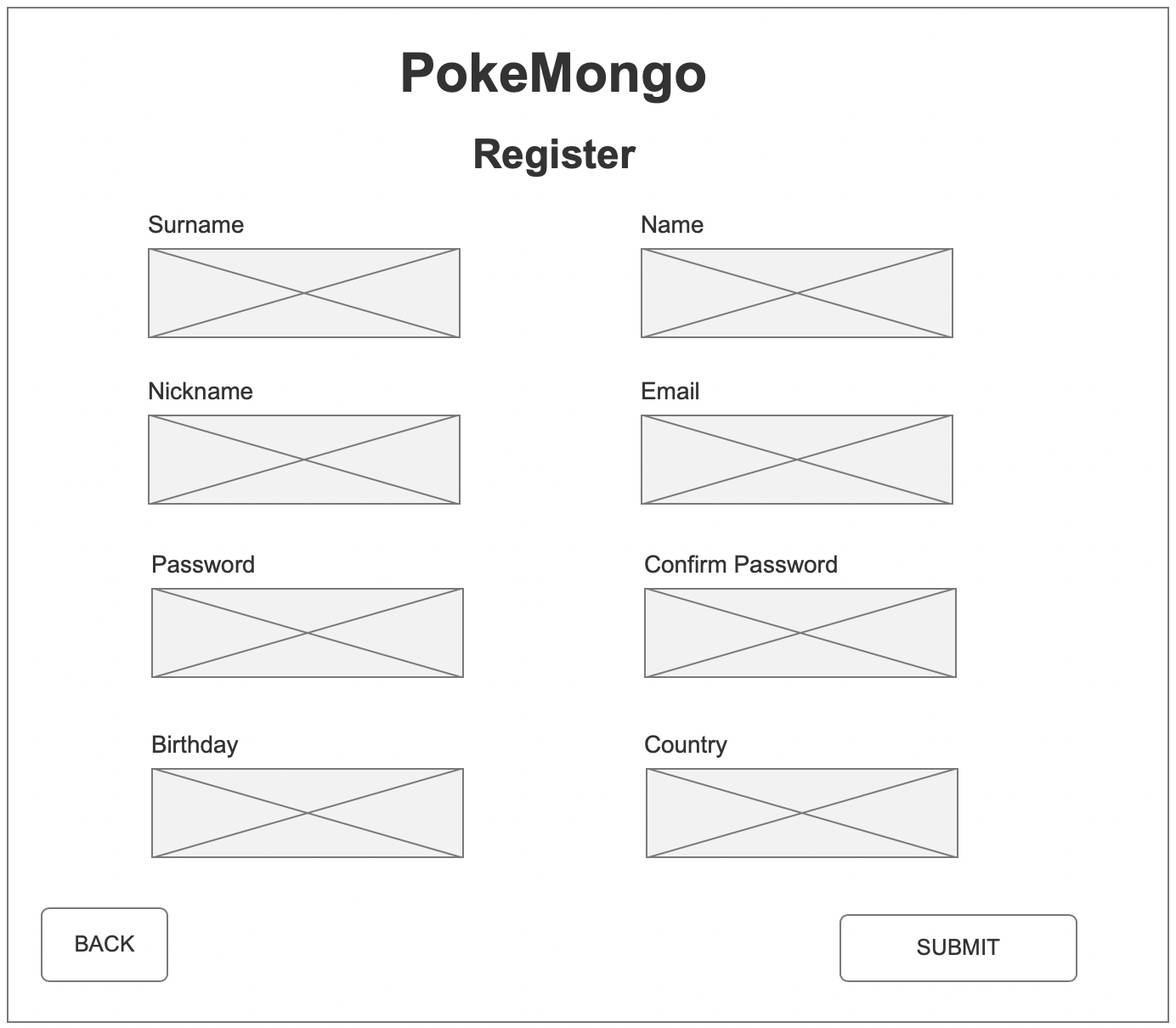
Black: everybody

Blue: just the normal user

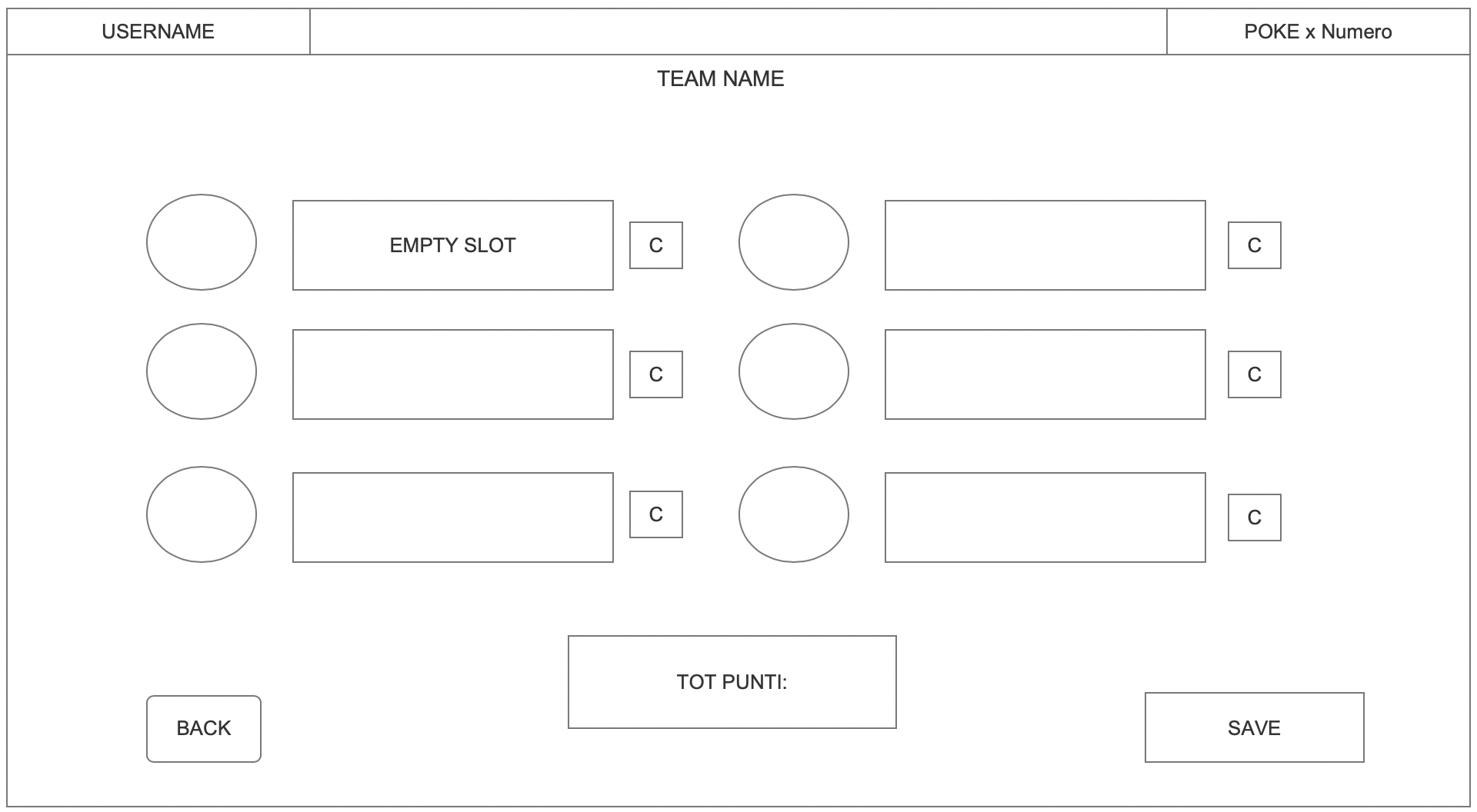
Red: just the admin



SIGNUP



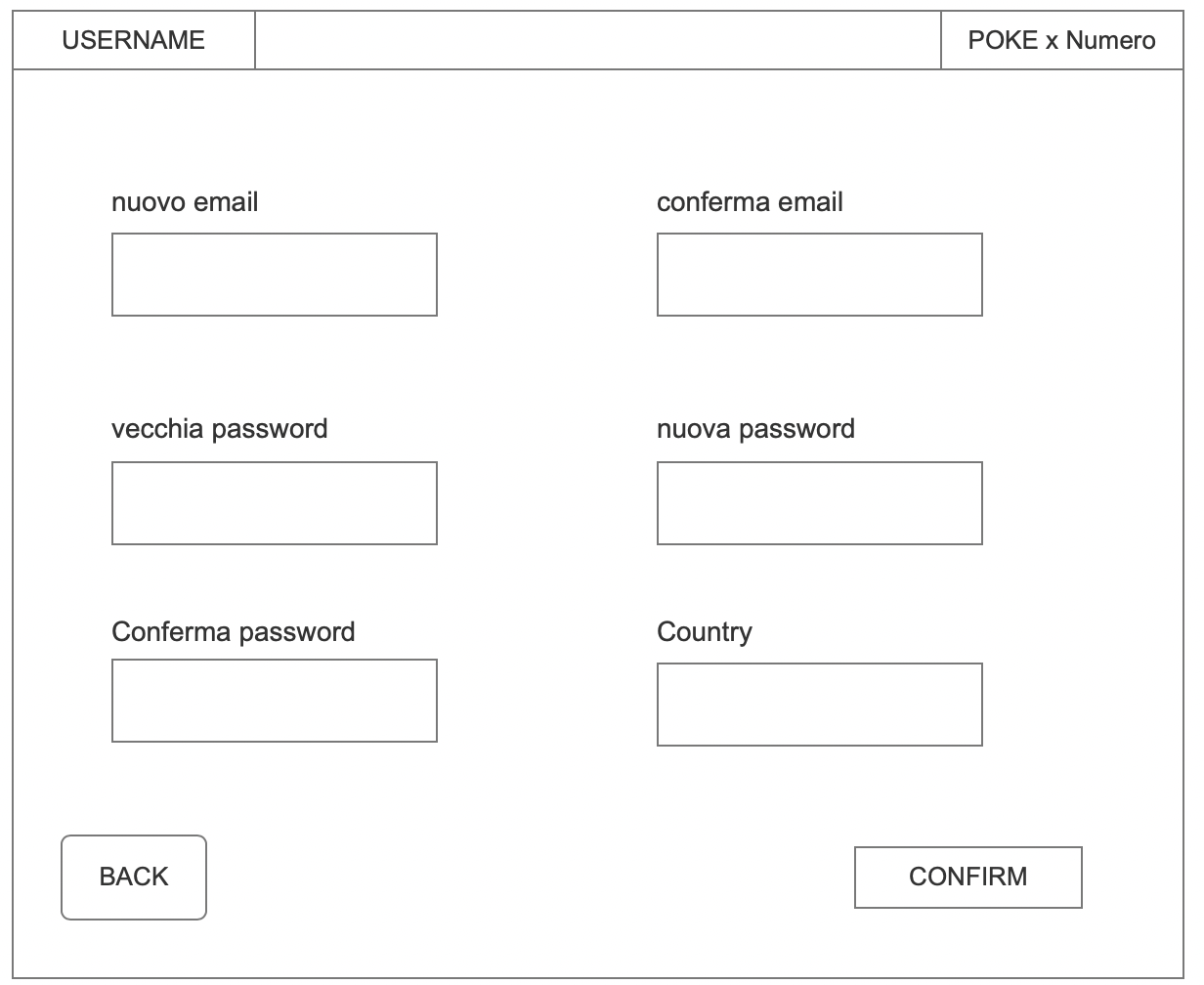
TEAM



POKEBALL NUMBER

POINTS

SETTINGS



POKEBALL NUMBER

Confirm E-mail address

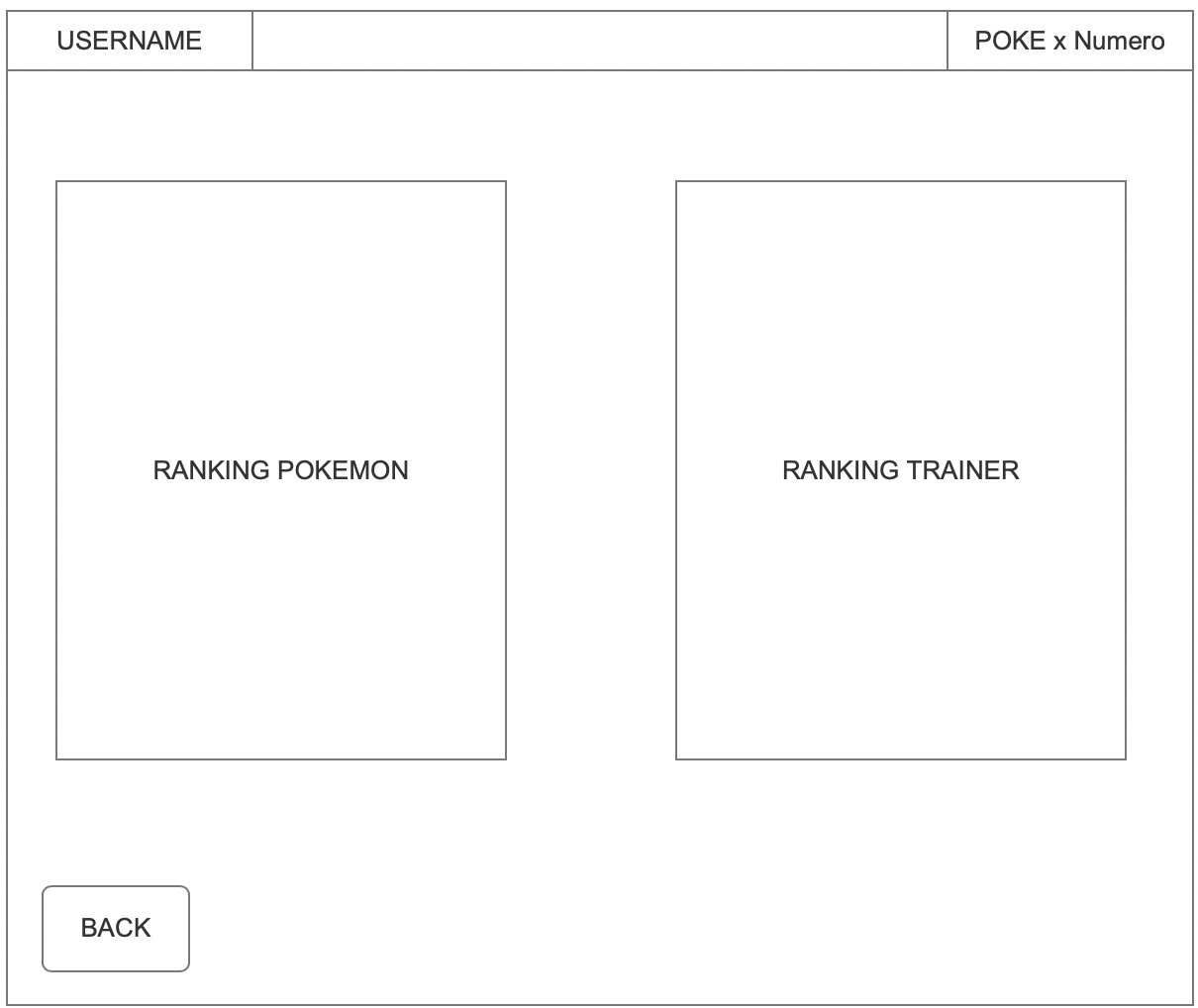
E-mail address

New Password

Old Password

Confirm Password

RANKING



*Note:* the remaining windows (Pokédex, catch’em all, add/remove pokemon) will be done after projecting the database, in order to make the most appropriate solution.

# Requirements

## 3.1 Functional requirements and use cases

* An unregistered user can
  + - * + Register
* A registered user
  + - * + Sign in
        + Consult Pokédex:

Search by name

Search by type(s)

Search by Pokédex ID

Search by catch rate

Search by points

Search by Pokemon characteristics (i.e, height, weight,..)

* + - * + Consult ranking:

Most popular pokemon among all users

“” “” “” “” friends

Best world team

Best friends’ team

Best team by country

* + - * + Find users:

See recommended users

Find users by username

Follow them

Unfollow them

* + - * + Team handling:

Remove Pokemon from the team

View team

Save modified team

View the value of the team

* + - * + Catching:

Select a Pokemon you want to catch

Try to catch the Pokemon to add to your team

* + - * + Settings:

Change email

Change password

Change country

* + - * + Logout:

Exit from the account

Return to the sign in window

* An admin can

Add pokemon to the Pokédex

Remove pokemon from the Pokédex

See # of registered users in time

See # logins per day

Remove a user from the system

* The system should
  + Daily update Pokeball number of each user
  + Daily update (deferred) of Dynamic catch rates
  + Update (triggered) team points if 6 pokemons have all different types

//spazio per uml

## Non-functional requirements

//CAP scelta di due tra 3

//liv di availability

//liv di performance

//liv di consistency

//liv di atomicness

//liv di persistency

//AP

//eventual consistency: read-your-writes

//usability: very simple to use and enjoyable

//fast response time

//password must be crypted for security issues

//graphical interfaces with multimedia usage for a more interesting game experience

## Sources, velocity properties and volume of data

Source: PokeApi with some manipulations on data

Velocity guaranteed on daily update of catchrates

Volume of data is around 100Mb for a dataset of 250k users and almost 1k Pokemon

## 3.4Uml Relation Diagram

0..\*



A user can build up only 1 team: of course, each team has just one owner.

A team is composed of a maximum of 6 Pokemons, every Pokemon can be caught by anyone, so can belong to many teams.

//chiedere a Ducange come rappresentare la relazione follow Utenti-Utenti

# 4.Queries and Database Structure

## 4.1 Main DB queries

* Retrieve user information
* Retrieve team information based on user
* Retrieve Pokemon catch rates
* Retrieve Pokemon using several filters
* Retrieve recommended users
* Retrieve a user by username
* Retrieve list of a user’s friends
* Analytics: most popular Pokemons[by country], best ranked teams [among friends/by country], evolution on time of catchRates
* Analytics: evolution on time of # of users/logins per day (admin related)
* Create a user
* Remove a user (admin related)
* Modify user settings
* Update team (add/remove Pokemons)
* Remove/Add Pokemons from the database (admin related)

## 4.2 Json collections

Since they are very different Entities, User and Pokemon need their own collection.

In particular, for an admin user, some field are not needed: they have been eliminated.

As seen in the paragraph 3.3, the Team is strictly related with the user it was created by, so the most natural solution is to embed Team’s data into the collection *user*, so that to retrieve it faster.

Furthermore, a Team is an Array of Pokemon: that means that it is possible to replicate each Pokemon’s attribute into it. Anyway this solution is not scalable: the system is made of several Pokemon shared among many users, and each Pokemon is characterized by an high amount of attributes: the embedding of these documents would cause an exponential grow of storage occupation.

For this reason, it has been chosen to make the field Team an Array of Pokemon IDs.

Immagine che contiene monitor, schermo, computer, elettronico

Descrizione generata automaticamenteImmagine che contiene monitor, schermo, elettronico, computer

Descrizione generata automaticamenteThe final result is summarized in the two following collections, default value have the only purpose to show the type of fields.

# 

# 5 Graph db

1. Query su graph e traduzione delle query

Retrieve of Pokemons composing a team ==🡺 Retrieve Pokemon Nodes connected to a User node

Retrieve recommended users 🡺 Retrieve User nodes at distance 2 from a input User Node

Retrieve most popular pokemon 🡺 Count # of edges associated to each Pokemon Node

1. Struttura del db (nodi-archi)

Pokemon

{pokId}

User

User

{username}

has

follows