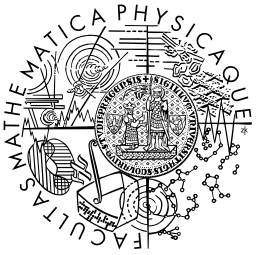


MATEMATICKO-FYZIKÁLNÍ FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Richard Fedák

Webová aplikace pro vyhledávání a editaci hráčů basketbalu ve Wikidata

Katedra softwarového inženýrství

Vedoucí bakalářské práce: doc. Mgr. Martin Nečaský, Ph.D.
Studijní program: Informatika
Studijní obor: Programování a vývoj software

Praha 2022

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ....

Podpis autora

Chcel by som sa poďakovať môjmu vedúcemu doc. Mgr. Martinovi Nečaskému, Ph.D. za jeho vedenie, pomoc, rady a čas, ktorý mi venoval. Taktiež by som chcel poďakovať mojej rodine, ktorá ma podporovala počas vypracovania tejto práce, ale aj počas celej doby štúdia.

Název práce: Webová aplikace pro vyhledávání a editaci hráčů basketbalu ve Wikidata

Autor: Richard Fedák

Katedra: Katedra softwarového inženýrství

Vedoucí bakalářské práce: doc. Mgr. Martin Nečaský, Ph.D., Katedra softwarového inženýrství

Abstrakt: Existuje veľa spôsobov, ako na internete zvejerňovať dáta v strojovo čitateľnej podobe. Jeden zo spôsobov je prostredníctvom entít a ich vzájomným prepojením, kde dve entity sú prepojené ak majú medzi sebou nejaký vzťah. Takáto datová štruktúra sa označuje pojmom znalostný graf, ktorý nachádza v súčasnoti uplatnenie vo svete otvorených dát.

Cieľom tejto práce je vytvoriť webovú aplikáciu, ktorá sa dotazuje nad dvoma takýmito grafmi a to znalostný graf Wikidata a DBpedia. Aplikácia je schopná vyhľadať hráča basketbalu a získať o ňom dáta z daných znalostných grafov. Umožniť užívateľovi objavovať rôzne prepojenia medzi vyhľadaným hráčom a ostatnými hráčmi pomocou spoločných kategórií a jednoducho doplniť základné chýbajúce informácie pomocou aplikácie do znalostného grafu Wikidata.

Klíčová slova: basketbal wikidata rdf sparql dbpedia

Title: Web application for searching and editing basketball players in Wikidata

Author: Richard Fedák

Department: Department of Software Engineering

Supervisor: doc. Mgr. Martin Nečaský, Ph.D., Department of Software Engineering

Abstract: There are many ways to publish data in a machine-readable form on the Internet. One way is through entities and their interconnection, where two entities are interconnected if they have a relationship with each other. Such a data structure is referred to as a knowledge graph, which is used in the world of open data.

The aim of this work is to create a web application that queries two such graphs, namely the knowledge graph Wikidata and DBpedia. The application is able to search for a basketball player and obtain data about him from the given knowledge graphs. It allows the user to discover various connections between the searched player and others using shared categories and simply add basic missing information using the application to the Wikidata knowledge graph.

Keywords: basketball wikidata rdf sparql dbpedia

Obsah

Úvod

Používanie internetu sa stalo bežnou súčasťou každodenného života. Webové stránky a aplikácie sprístupňujú a poskytujú rozličné informácie, ktoré sú uložené v systéme WWW (World Wide Web). Problémom je, že WWW nijako nepopisuje formát, v akom by mali byť dáta zverejnené na internete. Často môžeme naraziť na rôznych stránkách na rovnaké dáta, ale samotné dáta sú zverejnené odlišne. Jedny môžu byť reprezentované, ako tabuľka, ďalšie, vo forme viet. Bežným používateľom internetu postačuje, že našli hľadanú informáciu a forma, v akej bola prezentovaná ich až tak nezaujíma. Pre programátora však zmena formátov pri získavaní dát predstavuje komplikáciu.

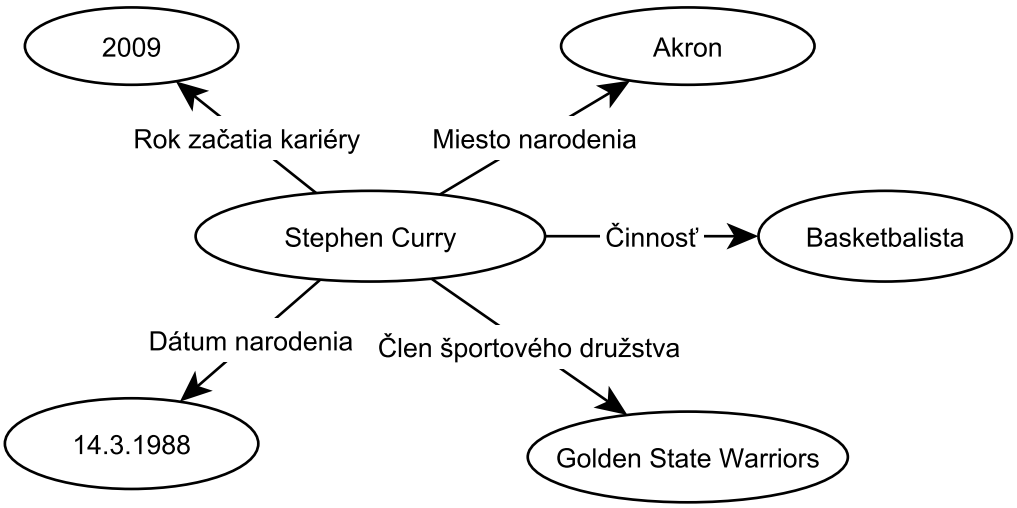
Ideálne by bolo, keby dáta boli reprezentované vo formáte, ktorý je strojovo čitateľný, voľne dostupný a aby sme dáta mohli využívať pre svoju potrebu. V tejto súvislosti pre dáta existuje kategorizácia „otvorenosti“ dát, ktorá porovnáva dáta od najmenej po najviac otvorené (a teda najpoužiteľnejšie).

1. Dáta nie sú v strojovo čitateľnom formáte. (PDF)
2. Dáta sú v strojovo čitateľnom formáte. (XLS)
3. Dáta sú vo formáte so slobodnou špecifikáciou. (CSV)
4. Dáta majú taktiež svoju vlastnú URI adresu. (RDF)
5. Dáta sú taktiež prepojená s ostatnými dátami. (LOD - Linked Open Data)

Prepojené otvorené dáta [1] (*anglicky Linked Open Data*) nám ponúkajú práve to, čo potrebujeme. Sú to verejne prístupné dáta zverejnené na internete, ktorých hlavnými vlastnosťami sú, že sú medzi sebou prepojené a strojovo čitateľné.

Táto práca sa zaoberá vyhľadávaním, zobrazovaním a doplňaním informácií o hráčoch basketbalu, kde získané informácie sú práve zo zdrojov prepojených otvorených dát. Informácie sú popísané pomocou RDF [2] formátu (bližie popísaný na nasledujúcej strane), ktorý je použitý pre strojovú reprezentáciu znalostných grafov Wikidata [3] a DBpedia [4]. Zo znalostného grafu Wikidata sa čerpajú základné informácie o hráčovi (dátum narodenia, začiatok profesionálnej kariéri a pod.) a informácie pre časové osi, ktoré zobrazia ocenenia, tímy a turnaje hráča. Na obrázku 1 je znázornená reprezentácia dát vo Wikidata.

[1] `https://www.w3.org/egov/wiki/Linked_Open_Data`
[2] Raimond a Schreiber (2014)
[3] https://www.wikidata.org
[4] https://www.dbpedia.org/about/



Obr. 1: Znázornenie prepojenia dát. Stephen Curry reprezentuje vyhľadaného hráča.

Zo zdroja DBpedia sa čerpajú taktiež základné informácie o hráčovi, ktoré slúžia na prípadné doplnenie chýbajúcich dát z Wikidata. No hlavný dôvod dotazovania sa tohto znalostného grafu je, že zaraďuje jednotlivé entity do kategórií a to umožňuje vyhľadávať skupiny hráčov, ktorí zdieľajú rovnaké kategórie. Výber kategórií je možné ľubovoľne kombinovať a tak objavovať aj nečakané spojitosti medzi hráčmi.

Resource Description Framework

Resource Description Framework (RDF), je rodina špecifikácií, ktorá sa používa na popis informácií v strojovo čitateľnej podobe. RDF reprezentuje dáta v podobe grafu tvoreného vrcholmi a hranami, ktoré ich prepájajú. Jednotlivé vrcholy sú reprezentované pomocou IRI (Internationalized Resource Identifier) no RDF pre identifikáciu vrcholov pri znalostných grafoch väčšinou používa jeho špeciálnu podobu - URL (Uniform Resource Locator), ktorý bežne používame ako identifikátor webových stránok. Vrcholy môžu byť taktiež vyjadrené ako literály, čo sú primitívne datové hodnoty uvedené v úvodzovkách alebo ako prázdne vrcholy.

Napríklad basketbalista Stephen Curry je jednoznačne reprezentovaný následovne:

- **Wikidata:** https://www.wikidata.org/wiki/Q352159
- **DBpedia:** https://dbpedia.org/page/Stephen_Curry

Základom u RDF je tvrdenie (*anglicky statement*), ktoré je vyjadrené formou trojice: **subjekt**, **predikát** a **objekt**. Subjekt a objekt reprezentujú vrcholy a predikát orientovanú hranu od subjektu k objektu.

Ako ukážkové tvrdenie budeme používať:

Stephen Curry má miesto narodenia Akron.

Tvrdenie v RDF:

```
<https://www.wikidata.org/wiki/Q352159>
<https://www.wikidata.org/wiki/Property:P19>
<https://www.wikidata.org/wiki/Q163132>
```

Prvý riadok reprezentuje subjekt „Stephen Curry“. Druhý riadok predikát „má miesto narodenia“. A tretí objekt „Akron“.

Takéto tvrdenie je strojovo čitateľné, no zápis sa nám určite nečíta ľahko. Z tohto dôvodu existuje serializácia **RDF Turtle**[5], ktorá nám umožňuje definovať a využiť prefixy, ktoré nám výrazne pomôžu skrátiť zapisovanie bežných RDF tvrdení.

Tvrdenie v RDF Turtle:

```
@prefix wd: <http://www.wikidata.org/entity/> .
@prefix wdt: <http://www.wikidata.org/prop/direct/> .

wd:Q352159 wdt:P19 wd:Q163132 .
```

Formát Turtle nám taktiež umožňuje skrátiť zápis tvrdení, v ktorých sa opakuje subjekt alebo predikát. Ak by sme chceli rozšíriť naše základné tvrdenie a pridať k miestu narodenia aj dátum narodenia, tvrdenie by sme vedeli skrátene zapísať nasledovne.

Rozšírené tvrdenie v RDF Turtle:

```
@prefix wd: <http://www.wikidata.org/entity/> .
@prefix wdt: <http://www.wikidata.org/prop/direct/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

wd:Q352159 wdt:P19 wd:Q163132;
           wdt:P569 "1988-03-14T00:00:00Z"^^xsd:dateTime.
```

Pre vynechanie zápisu opakovaného subjektu na konci tvrdenia napíšeme bodkočiarku (;) a pre vynechanie opakovaného subjektu a predikátu píšeme čiarku (,).

XML schema (xsd) [6] sa v RDF využíva pre širokú definíciu datových typov.

SPARQL

SPARQL [7] je štandardný dotazovací jazyk, ktorý slúži na dotazovanie dát reprezentovaných vo formáte RDF. SPARQL endpointy, ktorých sa budeme dotazovať:

- https://dbpedia.org/sparql

[5] Beckett a kol. (2014)
[6] https://www.w3.org/TR/swbp-xsch-datatypes/
[7] Harris a kol. (2013)

- https://query.wikidata.org/

V SPARQL popisujeme, aké časti dotazovaného znalostného grafu chceme extrahovať pomocou podmienok, ktoré musia vrcholy extrahovaných častí spĺňať. Ako príklad môžeme uviesť dotaz určený pre Wikidata endpoint, ktorým chceme zistiť miesto narodenia a dátum narodenia Stephena Curryho.

SPARQL dotaz:

```
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>

SELECT ?birthPlace ?birthDate
WHERE
{
  wd:Q352159 wdt:P19 ?birthPlace;
             wdt:P569 ?birthDate.
}
```

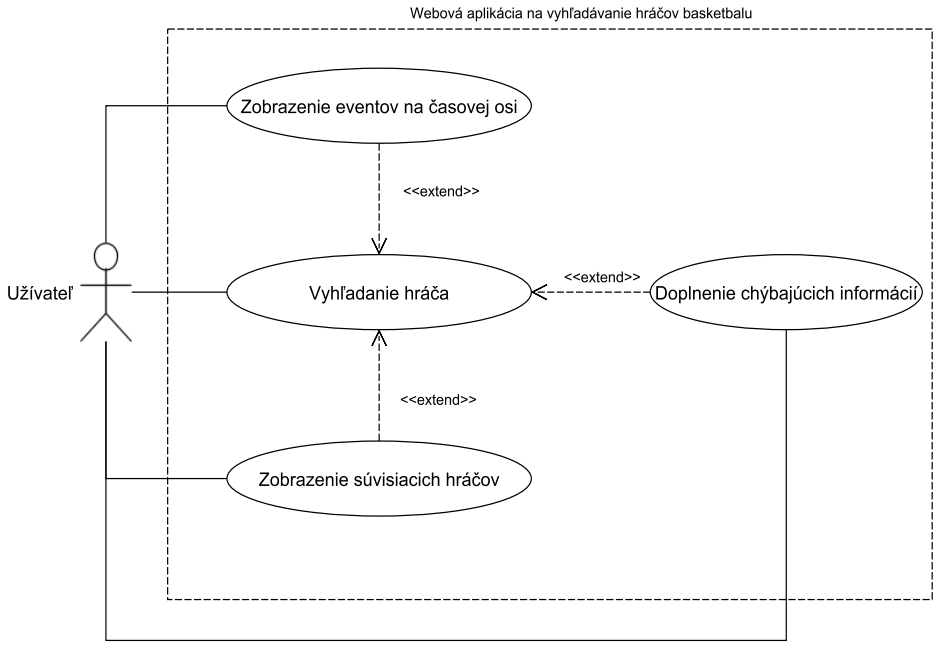
Jedná sa o jednoduchý `SELECT` dotaz, ktorý nám ako výsledok vráti tabuľku so stĺpcami `birthPlace` a `birthDate`, je to zoznam premenných, ktoré sa nachádzajú za slovom `SELECT`. Nasleduje kľúčové slovo `WHERE`. V jeho bloku ohraničenom zátvorkami sa definuje grafový vzor, kde premenné sú reprezentované ako `?[nazov]`.

1. Analýza požiadaviek

1.1 Prípady použitia

Najskôr sa pozrieme na prípady použitia (*anglicky use cases*), ktoré sa používajú na zachytenie požiadaviek a opisujú vzťahy a interakciu medzi aplikáciou a používateľom.

Obrázok 1.1 popisuje diagram prípadov použitia pre túto aplikáciu.



Obr. 1.1: Use case diagram podľa uživateľských požiadaviek.

Prípady použitia sú značené oválne. Väzby označené ako «extend» medzi prípadmi znamenajú, že prípad rozširuje iný prípad. Napríklad je možné pre vyhľadaného hráča si nechať zobraziť aj údaje o jeho eventoch na časovej osi.

1.2 Analýza užívateľských požiadavok

Táto časť popisuje a analyzuje užívateľské požiadavky, ktoré boli kladené na vývoj aplikácie.

Vyhľadanie hráča

Užívateľ vie vyhľadať basketbalového hráča.

Analýza Vyhľadanie hráča je umožnené pomocou vstupného poľa, kam užívateľ píše meno hráča a tlačidla, ktorým užívateľ potvrdí vyhľadanie hráča. Vyhľadávanie je doplnené o našepkávanie hráčov pod textovým poľom. Našepkávaní sú

hráči, ktorých meno začína rovnako ako vstup, ktorý užívateľ zadal. Zhodná časť mena so vstupom je zvýraznená tučným písmom. Užívateľ si vie kliknutím zvoliť jedného z ponúkaných hráčov. Po zvolení je hráč doplnený do textového poľa a užívateľ môže hráča vyhľadať kliknutím na tlačidlo pre vyhľadávanie.

Ak by pre vstup užívateľa mohlo byť našepkané veľké množstvo hráčov (vstup je napríklad jedno písmeno), tak je maximálny počet navrhnutých hráčov obmedzený na 5.

Nie je možné vyhľadať hráča, ktorý nie je užívateľovi zobrazený pomocou našepkávania. Ak sa o to užívateľ pokúsi, zobrazí sa mu informácia o chybnom vstupe. Po spustení vyhľadávania a čakania na výsledok, nie je možné zadávať a vyhľadávať ďalšieho hráča.

Pre vyhľadaného hráča sa zobrazia základné informácie. Základné informácie sa primárne získavajú z Wikidata. Obsahujú fotografiu hráča a zoznam informácií o hráčovi, ktoré zahŕňajú: *meno hráča, dátum narodenia, miesto narodenia, štátne občianstvo, váha (kg), výška (cm), rok začatia a prípadne skončenia profesionálnej kariéry*. Pre každého hráča sa taktiež zobrazí mapa, na ktorej je vyznačené miesto narodenia.

Ak datový zdroj neposkytuje fotografiu hráča, zobrazí sa predvolený obrázok, ktorý reprezentuje hráča bez fotky. Ak je miesto narodenia neznáme, t.j. datový zdroj nám ho neposkytne, namiesto mapy sa zobrazí nápis o absencii daného údaju.

Pre každý chýbajúci údaj zo zoznamu základných informácií sa zobrazí textové pole a tlačidlo. Údaje zo zoznamu informácií sa získavajú aj z DBpedia. V prípade, že daný údaj nie je na Wikidata, ale DBpedia ho pre daného hráča poskytuje, daný údaj je automaticky doplnený do textového poľa, ktoré je zobrazené namiesto daného údaju.

Doplnenie základných informácií

Užívateľ má možnosť doplniť chýbajúce údaje zo zoznamu informácií o hráčovi.

Analýza Do textového poľa pre chýbajúcu informáciu môže užívateľ dopísať údaj a pomocou tlačidla údaj doplniť na stránke a do Wikidata. Doplnený údaj si aplikácia pamätá počas doby, kým sa nezmení hráč alebo neobnoví stránka.

Nie je možné odoslať údaj nevyhovujúci formátu danej položky zo zoznamu informácií. Každý pokus o doplnenie údaju prejde základnou kontrolou, aby sa obmedzili šance doplniť chybný údaj. Užívateľ bude pri pokuse o doplnenie chybného údaju informovaný a chybný údaj sa nedoplní ani na stránke, ani do Wikidata.

Keďže údaje o štátnom občianstve a mieste narodenia by bolo nevhodné doplniť ako literály, užívateľ je požiadaný o doplnenie QID entity, pre zachovanie prepojených dát. Pri doplňaní daného údaju sa užívateľovi zobrazí nápoveda, ako QID získať.

Zobrazenie eventov na časovej osi

Užívateľ si vie pre vyhľadaného hráča zobraziť časovú os pre jeden z troch eventov.

Analýza Eventy majú tri typy:

- **Turnaje**, ktorých sa hráč zúčastnil.

  Ak je to možné, okrem názvu turnaja sa zobrazí rok a krajina, v ktorej sa turnaj konal.

- **Ocenenia**, ktoré hráč získal.

  Ak je to možné, okrem názvu ocenenia sa zobrazí rok, kedy hráč dané ocenenie získal.

- **Tímy**, za ktoré hráč hral

  Ak je to možné, okrem názvu tímu sa zobrazí rok, kedy hráč začal a prípadne skončil hrať za daný tím.

Časová os s eventami sa zobrazí, ako samostatná stránka, z ktorej je možné taktiež vyhľadávať nového hráča alebo spätne zobraziť základné informácie o vyhľadanom hráčovi, pre ktorého sa eventy zobrazujú.

Eventy s časovým údajom sa získavajú z Wikidata. Sú zoradené podľa roku, začínajúc najstarším. Keďže DBpedia pre hráčov neposkytuje eventy typu turnaje a neposkytuje ani časové údaje pre eventy ako Wikidata, sú eventy z DBpedia zobrazené po všetkých eventoch Wikidata aby nenarušili štruktúru časovej osi.

Z DBpedia sú doplnené len tie eventy, ktoré nie sú obsiahnuté v eventoch získaných z Wikidata. V prípade, že k vyhľadanému hráčovi nebol nájdený ani jeden event, zobrazí sa o tom užívateľovi informácia.

Zobrazenie súvisiacich hráčov

Užívateľ si vie pre vyhľadaného hráča zobraziť súvisiacich hráčov.

Analýza Pod pojmom súvisiaci hráč chápeme hráča, ktorý zdieľa s vyhľadaným hráčom isté kategórie, ktoré sú získavané z datového zdroja DBpedia. Súvisiaci hráči sa vyhľadávajú pomocou predvoleného vyhľadávania. To hľadá hráčov, ktorí zdieľajú dvojicu spoločných kategórií. Užívateľ môže taktiež špecifikovať, na základe ktorých kategórií sa majú hráči vyhľadávať. Pre oba druhy vyhľadávania je na stránke tlačidlo, ktoré ich reprezentuje. Pri prejdení myšou na tlačidlo sa zobrazí informácia, ktorá popíše funkciu tlačidla.

Pre každého súvisiaceho hráča sa zobrazí „box“, ktorý hráča reprezentuje. Box môže mať dve podoby:

- Ak bolo použité predvolené vyhľadávanie, box obsahuje meno hráča a dve kategórie, ktoré má spoločné s vyhľadaným hráčom.

- Ak bolo použité vyhľadávanie na základe užívateľom zvolených kategórií, box obsahuje len meno hráča.

Súvisiaci hráči sa zobrazú po skupinách aby v prípade veľkého množstva výsledkov nedošlo k nadbytočnému čakaniu. Užívateľ má pod súvisiacimi hráčmi k dispozícii tlačidlo na zobrazenie ďalšej skupiny hráčov. Ak už nie sú ďalší hráči k dispozícii, tlačidlo zmizne a užívateľovi sa o tom zobrazí informácia.

Užívateľ si vie zvoliť kategórie, na základe ktorých sa zobrazia súvisiaci hráči. Vie si zvoliť ľubovoľnú kombináciu kategórií. Kategórie obsahujú taktiež údaje zo zoznamu základných informácií o hráčovi. Pre možné rozdielne údaje medzi Wikidata a DBpedia sa pre výšku a váhu hráča nevyhľadávajú hráči s rovnakou hodnotou ale hráči, ktorý spadajú do istého intervalu, pre danú hodnotu.

Nie je možné vyhľadávať súvisiacich hráčov bez toho aby bola zvolená aspoň jedna kategória. Ak hráč nemá žiadne kategórie k dispozícií a ani údaje v zozname základných informácií, užívateľovi sa pri pokuse o výber kategórií zobrazí informácia, že výber nie je možný.

2. Analýza datových zdrojov

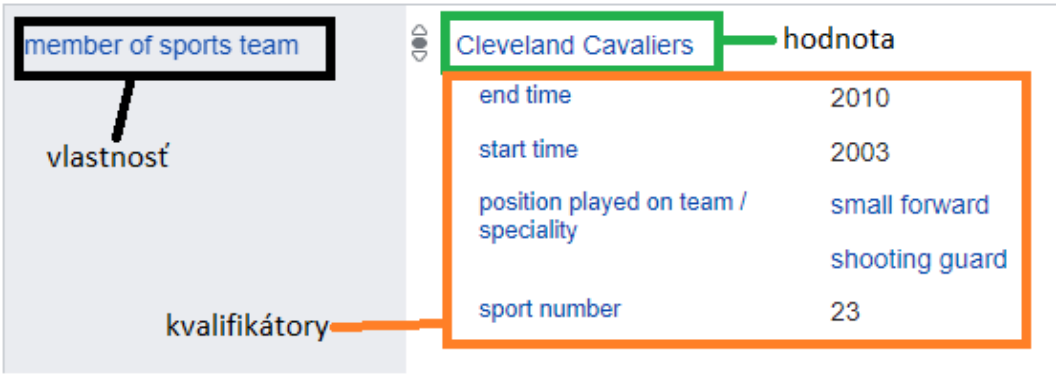
V tejto kapitole sa pozrieme na dva datové zdroje, z ktorých čerpáme dáta. Rozoberieme, v čom sú špecifické a akú majú štruktúru v zameraní sa na hráčov basketbalu. Budú to, už viackrát spomínané Wikidata a DBpedia. Vieme, že oba datové zdroje sú znalostné grafy a dotazovanie prebieha pomocou SPARQL dotazov pričom oba majú svoj vlastný SPARQL endpoint, nad ktorým sa aplikácia dotazuje.

2.1 Wikidata

Wikidata [1] je projekt kolektívne upravovanej podpornej databázy pre Wikipédiu [2]. Je prevažne zložená z entít, ktoré majú svoj unikátny identifikátor, QID (písmeno „Q“ nasledované číslom). Každá entita má svoje označenie (*anglicky label*). Label slúži ľudom pre jednoduchšie vyhľadávanie a identifikáciu. Pre jednoduchú strojovú identifikáciu entity slúži QID.

V kontexte aplikácie si stačí namiesto entity predstaviť hráča basketbalu. Wikidata poskytujú pre hráča tvrdenia (*anglicky statements*). Príklad tvrdenia možno vidieť na obrázku 2.1. Tvrdenie popisuje charakteristiky entity. Pozostáva z vlastnosti (*anglicky property*) a z hodnoty (*anglicky value*). Wikidata vlastnosti sú vo forme písmeno „P“ nasledované číslom. Napríklad, miesto narodenia je P19. Hodnoty môžu byť literály alebo ďalšie entity.

Každá hodnota môže byť doplnená o prívlastky/kvalifikátory (*anglicky qualifiers*) a referencie. Referencia slúži na odkázanie sa na zdroj, z ktorého informácia pochádza. Kvalifikátory nám môžu poskytnúť doplňujúce informácie k danej hodnote. Využijeme ich najmä pri eventoch, kde nám napríklad poskytnú časový údaj, v ktorom roku hráč získal ocenenie alebo, v ktorom roku začal a prípadne skončil hrať za istý tím.



Obr. 2.1: Príklad tvrdenia hráča LeBron James.

[1] Vrandečić a Krötzsch (2014)
[2] https://www.wikipedia.org/

2.1.1 Analýza Wikidata z pohľadu aplikácie

Cieľové entity sú hráči basketbalu. Pod pojmom hráč basketbalu chápeme každú entitu, ktorá má dve nasledujúce tvrdenia.

- **wdt:P106(Činnosť) wd:Q3665646(Basketbalista)**
- **wdt:P31(Je) wd:Q5(Človek)**

U hráčoch basketbalu nás zaujímajú dva druhy tvrdení. Prvý nám poskytne dáta pre základné informácie o hráčovi. Druhý zas informácie pre časové osi.

Prefixy, ktoré využívame pri vlastnostiach sú:

- **wdt:** odkazuje na entitu/literál tvrdenia
- **p:** neodkazuje na entitu ale na hodnotu tvrdenia. Táto hodnota môže byť použitá ako subjekt pre nasledujúce tvrdenia a tak sa z nej môžeme odkazovať na **kvalifikátory**.
  - **ps:** použitý v predikáte pre hodnotu tvrdenia nám umožní získať entitu/literál.
  - **pq:** nám zas umožní získať informácie z kvalifikátora.

Základné informácie

Zoznam vlastností a ich hodnôt, s ktorými pracujeme:

- **wdt:P569 - Dátum narodenia**

  Odkazuje na literál, ktorý reprezentuje dátum narodenia.

- **wdt:P19 - Miesto narodenia**

  Odkazuje na Wikidata entitu, z ktorej vieme získať názov miesta narodenia pomocou Wikidata label.

  Taktiež potrebujeme zistiť súradnice miesta narodenia pre zobrazenie na mape. Spojením vlastností `p:P625` a `psv:P625` získame súradnice entity miesta narodenia. P625 reprezentuje zemepisné súradnice.

- **wdt:P27 - Štátne občianstvo**

  Odkazuje taktiež na Wikidata entitu, z ktorej získame názov pomocou Wikidata label.

- **p:P2067 - Hmotnosť**

  Pri hmotnosti nastáva menší problém, pretože literál na ktorý sa odkazuje môže byť v rôznych jednotkách (kg, libry). Wikidata nám umožňuje hodnotu previesť na normalizovanú (v kilogramoch).

  Na hodnotu v kilogramoch dostaneme spojením vlastností `psn:P2067` (normalizovaná hodnota) a `wikibase:quantityAmount` (hodnota), čím získame normalizovanú hodnotu v kilogramoch.

- **p:P2048 - Výška**

  Hodnota literálu môže byť taktiež v rôznych jednotkách (cm, palce). No tak, ako pri hmotnosti nám Wikidata umožňuje hodnotu previesť na normalizovanú (v metroch).

  Hodnotu v metroch dostaneme spojením vlastností `psn:P2048` (normalizovaná hodnota) a `wikibase:quantityAmount` (hodnota), čím získame výšku v metroch.

- **wdt:P2031 - Pracovné obdobie (začiatok)**

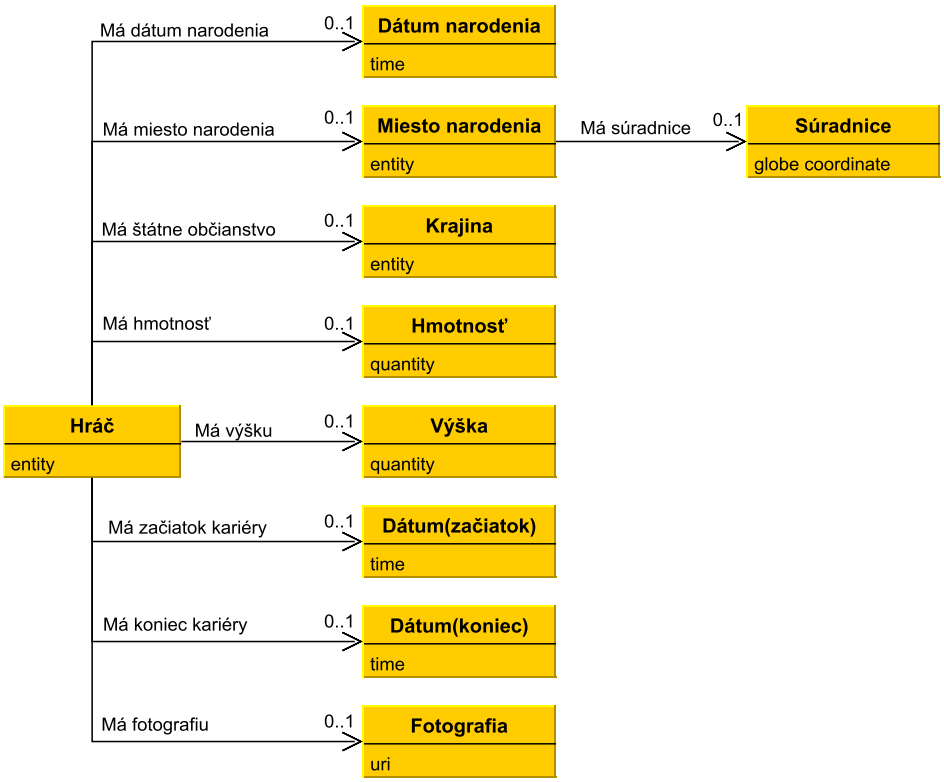
  Odkauje na literál. Zvyčajne na Wikidata je táto hodnota len špecifikovaná na rok (deň a mesiac je 1.1.), ktorý je postačujúci pre túto informáciu.

- **wdt:P2032 - Pracovné obdobie (koniec)**

  Rovnaké, ako predchádzajúca vlastnosť P2031.

- **wdt:P18 - Obrázok**

  Odkazuje sa na URI, fotografiu hráča.



Obr. 2.2: Grafické znázornenie štruktúry údajov - základné informácie hráča z Wikidata a ich typy.

Na obrázku 2.2 môžeme vidieť údaje, ktoré nás zaujímajú a ich typy. Údaj je buď entita alebo jeden z datových typov Wikidata [3].

[3] `https://www.wikidata.org/wiki/Help:Data_type`

Eventy

Zoznam vlastností na získanie informácií o jednotlivých eventoch:

- **p:P166 - Ocenenie**

  Odkazuje na entitu ocenenia, pomocou Wikidata label získame názov ocenenia.

  Tvrdenie môže byť doplnené o kvalifikátor, ktorý nám poskytne údaj, v ktorom roku hráč dané ocenenie získal (pq:P585).

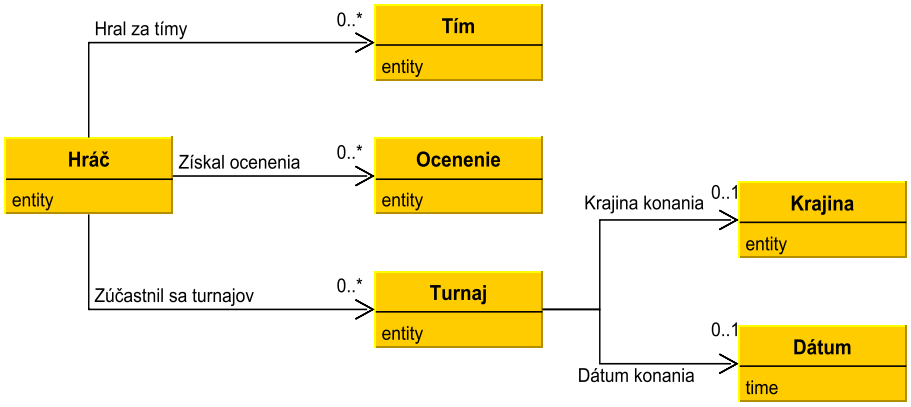
- **p:P54 - Člen športového mužstva**

  Odkazuje na entitu tímu, pomocou Wikidata label získame názov tímu.

  Tvrdenie môže byť taktiež doplnené o kvalifikátory, ktoré nám poskytnú údaj o tom, v ktorom roku hráč začal (pq:P580) a prípadne skončil(pq:P582) hrať za daný tím.

- **wdt:P1344 - Účasť (turnaje)**

  Odkazuje na entitu turnaja, pomocou Wikidata label získame názov turnaja. Pre entitu turnaja Wikidata môže poskytovať krajinu, kde (wdt:P17) sa turnaj uskutočnil a rok, v ktorom (wdt:P585) sa turnaj konal.



Obr. 2.3: Grafické znázornenie štruktúry údajov - eventy hráča z Wikidata.

2.1.2 Dotazovanie sa Wikidata z pohľadu aplikácie

Základné informácie

```
SELECT distinct
?itemLabel ?birthDate ?birthPlaceLabel
?birthPlaceCoordsLat ?birthPlaceCoordsLong
?citizenshipLabel ?imageURL (ROUND(?weightKG_) as ?weight)
(ROUND(?height_*100)/100 as ?height)
(year(?careerStartYear) as ?careerStart)
(year(?careerEndYear) as ?careerEnd)
WHERE
```

```
{
  values ?item {wd:Q36159}
  OPTIONAL {?item wdt:P569 ?birthDate}
  OPTIONAL {?item wdt:P19 ?birthPlace.
          ?birthPlace p:P625/psv:P625 [
                 wikibase:geoLatitude ?birthPlaceCoordsLat;
                 wikibase:geoLongitude ?birthPlaceCoordsLong
               ] .}
  OPTIONAL {?item wdt:P2031 ?careerStartYear}
  OPTIONAL {?item wdt:P2032 ?careerEndYear}
  OPTIONAL {?item wdt:P18 ?imageURL}
  OPTIONAL {?item wdt:P27 ?citizenship.}
  OPTIONAL {
    ?item p:P2067 ?weight_.
    FILTER NOT EXISTS{?weight_ pq:P582 []. }
    FILTER NOT EXISTS{?weight_ pq:P585 []. }
    ?weight_ psn:P2067/wikibase:quantityAmount ?weightKG_.
    }
  OPTIONAL {
    ?item p:P2048/psn:P2048/wikibase:quantityAmount ?height_.
    }

  SERVICE wikibase:label { bd:serviceParam wikibase:language "en". }
}
```

Eventy

Dotaz na získanie informácií na časovú os o tímoch, za ktoré hráč hral.

```
SELECT distinct
?eventLabel ?eventYear
?eventEndYear ?eventDescr
WHERE
{
  values ?item {wd:Q36159}
  ?item p:P54 ?statement.
  ?statement ps:P54 ?event;
             pq:P580 ?ContractStart.
  OPTIONAL{?statement pq:P582 ?ContractEnd}
  ?event rdfs:label ?eventLabel.
  filter(lang(?eventLabel)="en")
  ?item rdfs:label ?itemLabel.
  filter(lang(?itemLabel)="en")

  bind(year(?ContractStart) as ?eventYear)
  bind(year(?ContractEnd) as ?eventEndYear)
  bind(BOUND(?eventEndYear) && strlen(?eventEndYear)>0 as ?endCheck)
  bind(concat(" and stopped playing in ",str(?eventEndYear), ".") as
     ?endDescr)
  bind(if(?endCheck,?endDescr,".") as ?eventEndDescr)
```

```
  bind(concat(str(?itemLabel)," started plaing for
     ",str(?eventLabel)," in ",str(?eventYear), ?eventEndDescr) as
     ?eventDescr)

  SERVICE wikibase:label { bd:serviceParam wikibase:language
     "[AUTO_LANGUAGE],en". }
}
ORDER BY (?eventYear)
```

2.2 DBpedia

DBpedia [4] je projekt, ktorého cieľom je ťažiť štrukturované dáta a sprístupniť ich v strojovo čitateľnej forme na webu ako súčasťou sémantického webu [5]. Zdroje dát sú Wikipedia a Wikidata.

Základným stavebným prvkom je, rovnako ako u Wikidata, entita. Narozdiel od Wikidata, DBpedia nepriraďuje svojím entitám QID ani žiadne podobné číslo ale každá entita má svoj unikátny názov.

Každá entita môže mať vlastnosti, ktoré odkazujú na hodnoty. Hodnoty môžu byť ďalšie entity alebo literály. V tomto sú si s Wikidata podobné, no hodnoty už nedisponujú nijakými kvalifikátormi či referenciami.

2.2.1 Analýza DBpedie z pohľadu aplikácie

DBpedia poskytuje dáta pre prípadné doplnenie chýbajúcich základných informácií získaných z Wikidata a doplnenie eventov (okrem turnajov a štátneho občianstva, tie DBpedia u hráčoch neposkytuje). Naopak, poskytuje niečo, čo Wikidata nie a to je zaradenie hráčov do kategórií. Vďaka kategóriám je možné nájsť súvisiacich hráčov.

Hráča basketbalu v DBpedií chápeme, ako entitu, ktorá obsahuje tvrdenie **rdf:type dbo:BasketballPlayer**.

Základné informácie

Zoznam vlastností a hodnôt, pomocou ktorých kontrolujeme a prípadne dopĺňame chýbajúce informácie z Wikidata:

- **dbo:birthDate - Dátum narodenia**

  Odkazuje na literál, ktorý reprezentuje dátum narodenia.

- **dbo:birthPlace - Miesto narodenia**

  Odkazuje na DBpedia entitu, z ktorej vieme získať názov miesta pomocou vlastnosti **dbp:name**.

- **dbo:weight - Hmotnosť**

[4] Auer a kol. (2007)

[5] https://en.wikipedia.org/wiki/Semantic_Web

Odkazuje na literál. Narozdiel od Wikidata, DBpedia neurčuje u hodnôt, v akých sú jednotkách. To nám určuje samotná vlastnosť. Hodnoty dbo:weight sú vždy v gramoch.

- **dbo:height - Výška**

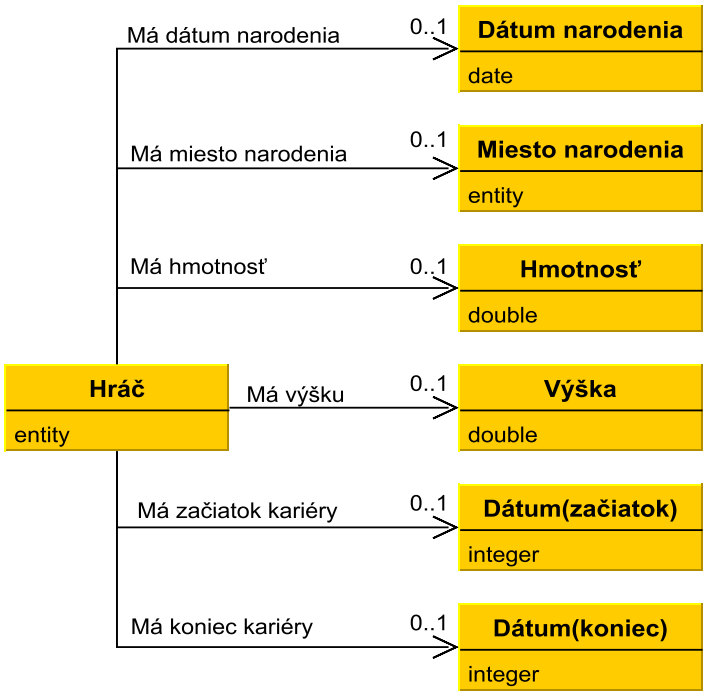
  Odkazuje na literál. Rovnako ako pri hmotnosti, vlastnosť dbo:height nám určuje, že jej hodnota je vždy v metroch.

- **dbp:careerStart - Začiatok profesionálnej kariéry**

  Odkazuje na literál. Vyjadrený je ako celé číslo, ktoré reprezentuje rok začatia kariéry.

- **dbo:careerEnd - Koniec profesionálnej kariéry**

  Rovnaké, ako u vlastnosti dbp:careerStart.



Obr. 2.4: Grafické znázornenie štruktúry údajov - základné informácie hráča z DBpedia.

Eventy

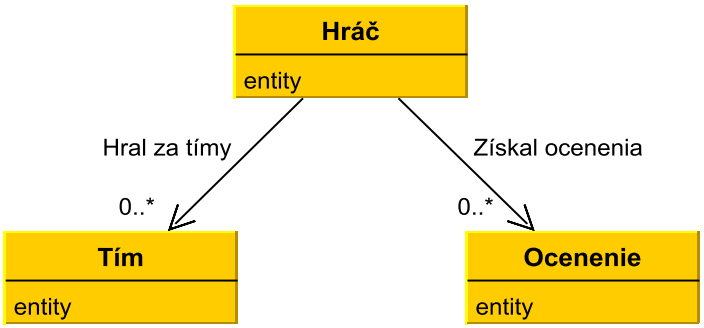
Zoznam vlastností na získanie názvov jednotlivých eventov:

- **dbo:award - Ocenenie**

  Odkazuje na entitu ocenenia, ktoré hráč získal. Názov ocenenia získame pomocou vlastnosti rdfs:label na entite ocenenia.

- **dbp:team - Člen tímu**

  Odkazuje na tím, za ktorý hráč hral. Názov tímu získame pomocou vlastnosti rdfs:label na entite tímu.



Obr. 2.5: Grafické znázornenie štruktúry údajov - eventy hráča z DBpedia.

Kategórie

DBpedia pre hráča poskytuje zaradenie do kategórií (Obrázok 2.6). Pomocou nich dokážeme nájsť hráčov, ktorí hrali za rovnaký tím, chodili na rovnakú školu apod. Kategórie nájdeme pod vlastnosťou **dct:subject**, ktorá sa odkazuje na rôzne entity, ktoré reprezentujú kategórie.

Obr. 2.6: Príklad kategórií hráča Tomáš Vyoral.

Každá kategória nám dá istú množinu entít, z ktorej je potrebné vybrať basketbalistov.

2.2.2 Prepojenie DBpedia a Wikidata entít

Vieme, že DBpedia a Wikidata entity majú rozdielny spôsob identifikácie. V DBpedií sú jednoznačne určené pomocou názvu a Wikidata majú pre každú entitu unikátne QID.

A tak sa ponúka otázka, ako entity, ktoré popisujú rovnakého hráča prepojiť medzi týmito datovými zdrojmi. Našťastie, DBpedia entity majú vlastnosť `owl:sameAs`, ktorá má ako jednu z hodnôt URL entity vo Wikidata. Vďaka nej sa vieme jednoznačne odkazovať na entitu, ktorá popisuje rovnakého hráča.

2.2.3 Dotazovanie sa DBpedia z pohľadu aplikácie

Základné informácie

```
select ?birthDate ?birthPlaceLabel ?careerStart ?careerEnd
    (ROUND(?mass_/1000) as ?weight) (ROUND(?height_*100)/100 as ?height)
WHERE
{
  ?player owl:sameAs wikidata:Q36159.
  OPTIONAL {?player dbo:birthDate ?birthDate}
  OPTIONAL {?player dbo:birthPlace ?birthLocation.
            ?birthLocation dbp:name ?birthPlace.
            FILTER (lang(?birthPlace) = "en").
            BIND (STR(?birthPlace) AS ?birthPlaceLabel)}
  OPTIONAL {?player dbp:careerStart ?careerStart}
  OPTIONAL {?player dbp:careerEnd ?careerEnd}
  OPTIONAL {?player dbo:weight ?mass.
            BIND(xsd:integer(?mass) as ?mass_).}
  OPTIONAL {?player dbo:height ?height_}
}
```

Eventy

Dotaz na získanie tímov, za ktoré hráč hral.

```
select distinct (str(?teamName) as ?eventLabel)
where {
  ?player owl:sameAs wikidata:Q36159;
          dbp:team/rdfs:label ?teamName.
  FILTER (lang(?teamName) = "en").
}
```

Súvisiaci hráči

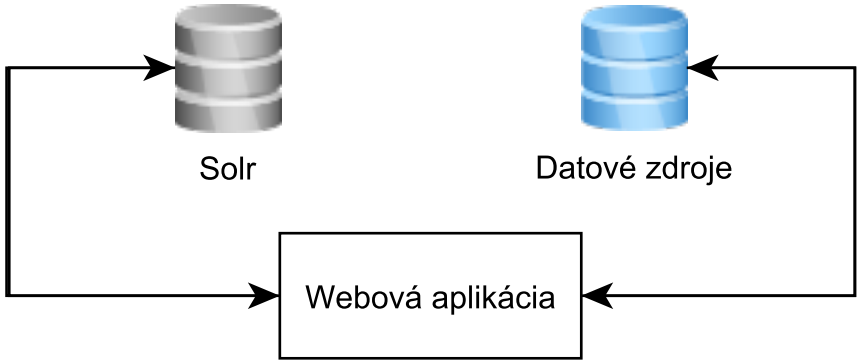
Vyhľadanie piatich súvisiacich hráčov k hráčovi LeBron James, ktorí sa narodili v roku 1984 a boli vyhlásení ako NBA All-Stars.

```
select distinct ?similarPlayerLabel ?similarPlayerWikidataULR
where {
    ?similarPlayer rdf:type dbo:BasketballPlayer;
                   rdfs:label ?similarPlayerLabel;
                   owl:sameAs ?similarPlayerWikidataULR;
                   dct:subject dbc:1984_births,
                   dbc:National_Basketball_Association_All-Stars.
    FILTER (regex(str(?similarPlayerWikidataULR), "www.wikidata.org" ) )
    FILTER (lang(?similarPlayerLabel) = "en").
    FILTER NOT EXISTS { ?similarPlayer owl:sameAs wikidata:Q36159}
}
limit 5
```

3. Návrh architektúry

Aplikácia je postavená na klientskej časti. Klient priamo komunikuje so Solr serverom a s datovými zdrojmi. Solr sa využíva na rýchle vyhľadávanie hráčov, ktorí sú našepkaní.

Klient využíva bohaté datové rozhrania Wikidata a DBpedia, ktoré mu poskytujú potrebné informácie. Získané dáta si dokáže spracovať a využiť pre potreby aplikácie.



Obr. 3.1: Komunikácia medzi aplikáciou, Solr serverom a datovými zdrojmi.

3.1 Solr

Apache Solr [1] je vyhľadávací server postavený na Apache Lucene [2]. Vytvorením Solr jadra (*anglicky core*) a nastavením konfigurácie dokážeme zaindexovať vlastné dáta pre rýchle vyhľadávanie basketbalistov v aplikácii na základe ich mena. Získanie dát a nastavenie Solr servera je podrobnejšie popísané v časti 4.1 Solr server.

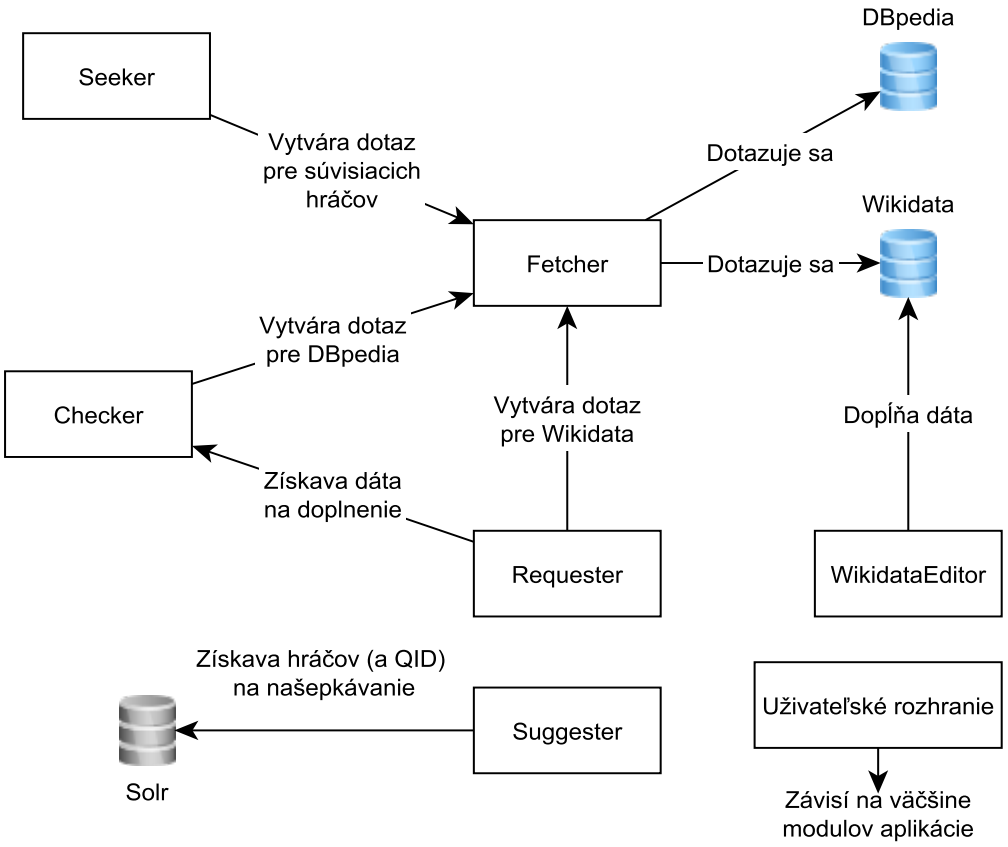
3.2 Moduly

V tejto časti si opíšeme jednotlivé moduly aplikácie, ich funkcie a vzájomnú komunikáciu. Obrázok 3.2 popisuje rozvrhnutie klientskej časti do modulov.

Suggester

Modul komunikuje so Solr serverom. Získava a udržuje si zoznam hráčov, ktorí vyhovujú vstupu užívateľa pri vyhľadávaní. Umožňuje pre každého hráča zo zoznamu získať jeho QID.

[1] https://solr.apache.org/
[2] https://lucene.apache.org/core/



Obr. 3.2: Závislosti a komunikácia modulov aplikácie.

Fetcher

Modul zabezpečuje dotazovanie sa Wikidata a DBpedia endpointov. Moduly, ktoré ho využívajú, mu posielajú SPARQL dotazy. Fetcher dotazy posiela na endpoint, ktorý si moduly zvolili. Modulom potom vráti odpoveď z daných endpointov vo formáte JSON[3].

Requester

Modul komunikuje s modulom Fetcher, ktorému posiela dotazy pre Wikidata a spracováva prijaté dáta. Dotazy sú zamerané na získanie základných informácií o hráčovi a na získanie eventov s časovými informáciami.

Okrem toho, Requester získava dáta z DBpedia pomocou modulu Checker. V prípade, že dáta od modulu Checker obsahujú niečo, čo nebolo získané z Wikidata, tak sú dáta doplnené.

Checker

Modul posiela dotazy modulu Fetcher určené pre DBpedia. Sú na získanie základných informácií no aj pre eventy na získané ocenenia a tímy hráča.

[3]https://www.json.org/

Postup (detailne zobrazený na obrázku 3.3) získania základných informácií v aplikácií je nasledovný:

1. Requester pošle dotaz pre Wikidata modulu Fetcher, ktorý mu vráti základné informácie a tie si uloží.
2. Requester informuje Checker o tom aby získal dáta z DBpedia.
3. Checker pošle dotaz pre DBpedia modulu Fetcher, ktorý mu vráti základné informácie. Checker informácie vráti modulu Requester
4. Informácie získané z oboch zdrojov sa porovnajú a každá informácia, ktorá nebola získaná z Wikidata ale bola z DBpedia sa doplní.

Seeker

Modul slúži na získanie kategórií hráča a vyhľadanie súvisiacich hráčov v užívateľom zvolenom alebo predvolenom vyhľadávaní. Modul si v sebe uchováva stav vyhľadávania, do ktorého patria nájdení súvisiaci hráči.

Pre vyhľadávanie súvisiacich hráčov si taktiež pre dotaz pamätá `offset`. Ten sa využíva, ak je k dispozícií väčší počet súvisiacich hráčov. Pre každý dotaz rovnakého typu vyhľadávania sa `offset` mení a tak sa získavajú noví hráči.

WikidataEditor

Modul slúži na doplnenie chýbajúcich základných informácií do Wikidata. Jeho úlohou okrem doplnenia informácií je poskytnúť metódy na kontrolu a úpravu údajov zo zoznamu základných informácií.

Užívateľské rozhranie

Zoznam modulov, s ktorými rozhranie priamo komunikuje a čo dané moduly rozhraniu poskytujú.

- Suggester

  Od užívateľského rozhrania dostáva vstupy užívateľa a vracia mu hráčov na našepkávanie. Pri potvrdení vyhľadávania daného hráča poskytne QID, ktoré je následne využité modulom Requester.

- Requester

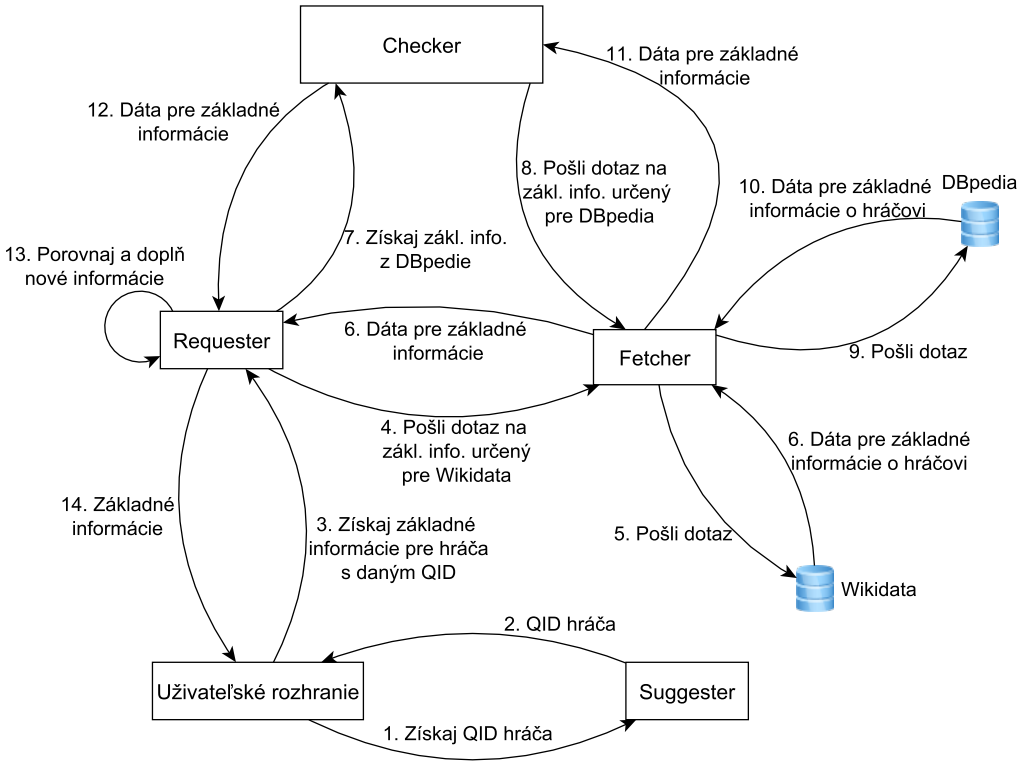
  Užívateľské rozhranie využíva tento modul pre získavanie základných informácií o hráčoch. Eventy hráča sa taktiež získavajú prostredníctvom tohto modulu.

- Seeker

  Užívateľské rozhranie využíva tento modul pre získanie súvisiacich hráčov a získanie kategórií hráča.

- WikidataEditor

  Modulu posiela užívateľom doplnené údaje a využíva jeho metódy, pomocou ktorých kontroluje vstup od užívateľa.



Obr. 3.3: Znázornenie komunikácie modulov po tom, ako užívateľ potvrdí vyhľadanie hráča.

4. Implementácia

4.1 Solr server

4.1.1 Získanie dát

Skript, napísaný v jazyku Python[1], ktorý slúži na získanie dát pre Solr je súbor `BasketballPlayers.py`. Nachádza sa v priečinku `data`. Využíva SPARQLWrapper[2] na dotazovanie sa Wikidata.

Skript ukladá dáta do JSON súboru `players.json`. Príklad časti výsledných dát možno vidieť na obrázku 4.1. Dáta sú reprezentované zoznamom objektov, kde každý obejkt má položky `id` (QID bez „Q“) a `name` (meno hráča).


```
[

{"id": 24519, "name": "Tony Wroten"},
{"id": 25278, "name": "Dwight Howard"},
{"id": 25369, "name": "Kobe Bryant"},
{"id": 25515, "name": "Coleman Collins"},
{"id": 25573, "name": "Jared Newson"},
{"id": 25580, "name": "Mark Tyndale"}

]
```


Obr. 4.1: Štruktúra dát na indexovanie. Príklad so šiestimi hráčmi.

4.1.2 Konfigurácia servera

Pre využitie `Solr suggester`[3] bolo potrebné pridať vo vytvorenom `core` nasledovný kód do konfiguračného súboru `conf/solrconfig.xml`. Pridaním danej `searchComponent` špecifikujeme fieldy, na ktoré sa dotazujeme či typ algoritmu použitého na vyhľadávanie. Pomocou daného `requestHandler` zas koľko výsledkov nám má Solr vrátiť.

```
<searchComponent name="suggest" class="solr.SuggestComponent">
  <lst name="suggester">
    <str name="name">mySuggester</str>
    <str name="lookupImpl">AnalyzingInfixLookupFactory</str>
    <str name="dictionaryImpl">DocumentDictionaryFactory</str>
    <str name="field">name</str>
    <str name="payloadField">id</str>
    <str name="suggestAnalyzerFieldType">text_general</str>
    <str name="buildOnStartup">true</str>
  </lst>
</searchComponent>
<requestHandler name="/suggest" class="solr.SearchHandler"
    startup="lazy">
  <lst name="defaults">
    <str name="suggest.dictionary">mySuggester</str>
```

[1] https://www.python.org/
[2] https://sparqlwrapper.readthedocs.io/en/stable/
[3] `https://solr.apache.org/guide/8_9/suggester.html`

```
    <str name="suggest">true</str>
    <str name="suggest.count">5</str>
  </lst>
  <arr name="components">
    <str>suggest</str>
  </arr>
</requestHandler>
```

V prílohe A.1 Solr server je odkaz na stiahnutie už nakonfigurovaného servera a základné inštrukcie.

4.2 React framework a TypeScript

Webová aplikácia je implementovaná v jazyku TypeScript[4] a pomocou React frameworku[5]. Typescript je nadstavbou jazyka JavaScript a rozširuje ho najmä o statické typovanie. React je určený pre tvorbu užívateľských rozhraní. Keďže mnohé jeho funkcionality boli použité pri tvorbe aplikácie, bolo by vhodné najkôr ukázať základné princípy fungovania tohto frameworku.

Komponenty

Pri návrhu React aplikácie je na začiatku vhodné rozdeliť aplikáciu na časti, z ktorých má byť aplikácia zložená. Tieto časti sa v terminológií Reactu nazývajú komponenty.

Komponenty môžeme chápať, ako stavebné bloky aplikácie. Ich základné vlastnosti sú, že sa môžu skladať z ďalších komponentov, môžu si medzi sebou predávať hodnoty známe ako `props` a taktiež môžu mať svoj stav `state`, v ktorom si uchovávajú dáta meniace sa počas behu aplikácie. Existujú dva druhy komponentov: funkčné a komponenty založené na triede.

Príklad komponentov:

```
type Props = {
    value: string;
};
type State = {
    value: string;
};
//Funkčný komponent
const FuncComponent = (props: Props) => <div>{props.value}</div>;

//Komponent založený na triede
class ClassComponent extends React.Component<Props, State> {
    constructor(props: Props) {
        super(props);
        this.state = { value: "state" };
    }
```

[4] https://www.typescriptlang.org/
[5] https://reactjs.org/

```
    render() {
        return (
            <div>
                Props: {this.props.value}
                State: {this.state.value}
                <FuncComponent value="props" />
            </div>
        );
    }
}
```

JSX

JSX[6] je rozšírenie syntaxe jazyka JavaScript Reactom, ktoré poskytuje spôsob, ako štruktúrovane vykresľovať komponenty. Vzhľadovo je podobný HTML. `ClassComponent` a `FuncComponent` z príkladu vracajú JSX element.

Metódy životného cyklu

Metódy životného cyklu [7] (*anglicky Lifecycle methods*) sú funkcie, ktoré môžeme využívať v komponentoch. Volajú sa v stanovených bodoch počas života komponentu. Medzi významné metódy pre komponenty založené na triede, patria:

- **render** - je najdôležitejšia metóda životného cyklu. Volá sa, keď sa aktualizuje komponent napríklad zmenou `state`. Využíva sa najmä na vrátenie JSX elementu.
- **componentDidUpdate** - volá sa okamžite po tom, ako bol komponent aktualizovaný, po `render`. Nie je volaná pri prvom volaní `render`.
- **componentWillUnmount** - volá sa pred tým, ako je komponenta odstránená z DOMu.

React Context

Ako už vieme, komponenty si medzi sebou vedia predávať dáta pomocou `props`, no ak sa zamyslíme, čo ak by sme mali do seba vnorených niekoľko komponent a chceli by sme predať dáta z najvyššej komponenty do najnižšej. Museli by sme manuálne predávať dané dáta medzi komponentami ako `props`, čo by nebolo veľmi príjemné. Z takéhoto dôvodu React poskytuje `Context`[8], ktorý vie tento problém vyriešiť.

[6] https://reactjs.org/docs/introducingjsx.html
[7] https://reactjs.org/docs/react-component.html#commonly-used-lifecycle-methods
[8] https://reactjs.org/docs/context.html

Príklad React context:

```
const context = React.createContext<{value: string} | null>(null);

class A extends React.Component{
    render() {
        return (
            <context.Provider value={{value : "ahoj"}}>
                <B />
            </context.Provider>
        );
    }
}
class B extends React.Component {

    render() {
        return (
            <C />
        );
    }
}
class C extends React.Component {

    render() {
        return (
            <context.Consumer >
                {(context) => (
                    <div>{context?.value}</div>
                )}
            </context.Consumer>
        );
    }
}
```

Styled components

Na stavbu užívateľského rozhrania bol použitý framework Styled Components[9]. Umožňuje písať vlastný štýl pre komponenty pomocou CSS syntaxe.

4.3 Inštalácia

Projekt využíva správcu balíčkov `npm`[10]. Pre preloženie projektu je nutné spustiť nasledujúce príkazy:

1. `npm install`

   Pre stiahnutie všetkých potrebných balíčkov.

[9] https://styled-components.com/
[10] https://www.npmjs.com/

2. `npm run build`

   Pre preloženie projektu. Tým sa vytvorí nový priečinok `build`. Nachádza sa v ňom súbor `index.html`, ktorým je možné spustiť aplikáciu.

4.4 Programátorská dokumentácia

Dôležité typy

Aplikácia má množstvo typov pre `state` a `props` jednotlivých komponentov. No je potrebné nejako reprezentovať aj dáta získané z datových zdrojov a tak uvedieme typy, ktoré reprezentujú získané základné informácie a eventy. Ak typ končí na „Validatable“, znamená to, že obsahuje položky, ktoré možno porovnať t.j. oba datové zdroje ich ponúkajú.

```
type PlayerInfoValidatable = {
    birthDate: {
        value: string;
    };
    birthPlaceLabel: {
        value: string;
    };
    weight: {
        value: string;
    };
    height: {
        value: string;
    };
    careerStart: {
        value: string;
    };
    careerEnd: {
        value: string;
    };
    itemLabel: {
        value: string;
    };
};

type PlayerInfo = PlayerInfoValidatable & {
    imageURL: {
        value: string;
    };
    birthPlaceCoordsLat: {
        value: string;
    };
    birthPlaceCoordsLong: {
        value: string;
    };
    citizenshipLabel: {
        value: string;
```

```
    };
};

type PlayerEventValidatable = {
    eventLabel?: {
        value: string;
    };
};

type PlayerEvent = PlayerEventValidatable & {
    eventYear?: {
        value: string;
    };
    eventEndYear?: {
        value: string;
    };
    eventDescr: {
        value: string;
    };
};
```

Použitie znaku „&“ znamená, že všetky položky typu budú položkami definovaného typu. Teda typ `PlayerEvent` má taktiež `eventLabel` od `PlayerEventValidatable`.

Kód aplikácie sa delí na komponenty a triedy. Komponenty tvoria najmä užívateľské rozhranie. Triedy zas implementujú logiku aplikácie. Dokumentácia popisuje najkôr všetky hlavné komponenty aplikácie a po nich rozoberieme jednotlivé triedy a ich logiku.

4.4.1 Komponent App

Ide o koreňový komponent celej aplikácie. Poskytuje `Context` ostatným komponentom, ktorý obsahuje súčasného vyhľadaného hráča (inštanciu triedy Player). `Context` tiež obsahuje informáciu o tom, či má byť zobrazená časová os. Pre obe hodnoty má `Context` funkcie na ich nastavenie. Hodnota `Context` je uložená ako `state` komponentu App a preto je typ `state` nasledujúci:

```
type AppState = {
    currPlayer: Player;
    isTimeline: boolean;
    setTimeline: (showTimeline: boolean) => void;
    setCurrPlayer: (player: Player) => void;
}
```

Ak ešte nedošlo k vyhľadaniu hráča, komponent zabezpečuje zobrazenie úvodnej stránky s vyhľadávacím poľom (komponent TextBox). Inak sa zobrazujú aj rôzne iné informácie o hráčovi, no o to sa starajú už ďalšie komponenty.

4.4.2 Komponent TextBox

Je to hlavný komponent na vyhľadávanie hráča, v `state` má uložený vstup z textového poľa, ktorý sa pri každej zmene vstupu mení. Zobrazuje našepkaných hráčov získaných s pomocou triedy Requester. Pri úspešnom potvrdení vyhľadávania hráča volá funkciu SetNewPlayer, ktorá nastaví nového hráča t.j. inštanciu triedy Player v `Context`.

4.4.3 Komponent InfoView

Komponent rozhoduje, aký pohľad sa má zobraziť. Využíva `Context`, z ktorého získa informáciu o tom či má byť zobrazená časová os hráča (komponent Timeline) alebo jeho základné informácie (komponent PlayerComponent). Na základe daného rozhodnutia zobrazí potrebné informácie.

```
class InfoView extends React.Component {

  render() {
    return (
      <playerContext.Consumer>
        {({ isTimeline }) =>
          isTimeline ? <Timeline /> : <PlayerComponent />
        }
      </playerContext.Consumer>
    );
  }
}
```

4.4.4 Komponenty Timeline, TimelineButtonComponent a PlayerInfoButton

Komponent Timeline zabezpečuje zobrazenie pohľadu pre časové osi. Má metódu `ShowEvents`, ktorá vráti eventy vo forme JSX elementu. Eventy sú od seba oddelené podľa zdroja. Prípadne vráti informáciu o tom že eventy pre hráča nie sú k dispozícií.

`TimelineButtonComponent` slúži na reprezentáciu tlačidiel pre časové osi. Kliknutím na daný komponent a teda tlačidlo, sa nastaví `Context` aplikácie na zobrazenie eventov hráča na časovej osi.

Pohľad je možné zmeniť pomocou komponentu `PlayerInfoButton`, ktorý nastaví `Context` aplikácie na zobrazenie základných informácií hráča.

4.4.5 Komponent PlayerComponent

Komponent obsahuje všetky komponenty súvisiace so základnými informáciami. zabezpečuje zobrazenie fotky hráča, zoznamu informácií (komponent PlayerBasicInfo) a mapy (komponent Map). Obsahuje taktiež komponenty pre zobrazenie časových osí (komponenty TimelineButtonComponent) a súvisiacich hráčov (komponent SimilarPlayers).

Metódy

- `ExtractPlayerInfo(player: Player): PlayerInfoValidatable`

  Metóda zabezpečuje vytvorenie objektu typu `PlayerInfoValidatable` z Player objektu. Objekt sa predáva ako `props` komponente SimilarPlayers ktorá ho využíva.

4.4.6 Komponenty PlayerBasicInfo a PlayerBasicInfoFrag

Komponent `PlayerBasicInfo` obsahuje zoznam základných informácií o hráčovi. Pomocou inštancie hráča z `Context` získa údaje na zobrazenie.

Každá informácia je reprezentovaná komponentom `PlayerBasicInfoFrag`. Daný komponent príjíma v `props` niekoľko vlastností, ktoré využíva pri zobrazovaní a prípadnom doplnení informácií o hráčovi.

Podľa hodnoty `infoValue` získanej v `props` rozhodne či sa má zobraziť informácia alebo textové pole na doplnenie. Hodnota môže byť `"-unknown-"`, čo znamená, že ani jeden datový zdroj ju neposkytuje. Hodnota môže mať taktiež prefix `"-dbpedia-"`, ktorý znamená, že daná hodnota bola doplnená z DBpedia. Takáto hodnota je predvyplnená do textového poľa na doplnenie informácie. Inak je to hodnota získaná z Wikidata a zobrazí sa bez možnosti jej úpravy.

Pri pokuse o doplnenie hodnoty sa hodnota skontroluje príslušnou funkciou, ktorá bola predaná prostredníctvom `props`. Ak spĺňa podmienky doplnenia, je upravená do formy tvrdenia, ktoré sa pošle na doplnenie do Wikidata.

4.4.7 Komponenty SimilarPlayers, CustomViewButton a DefaultViewButton

Komponent `SimilarPlayers` obsahuje všetky komponenty, ktoré vzťahujú na súvisiacich hráčov. Vždy sú v ňom zobrazené komponenty `CustomViewButton` a `DefaultViewButton`, ktoré reprezentujú tlačidlá umožňujúce užívateľovi prepínať medzi pohľadmi.

`DefaultViewButton` zabezpečuje počiatočné zobrazenie súvisiacich hráčov, ktorý sa získajú pomocou predvoleného vyhľadávania v triede Seeker.

`CustomViewButton` zas zabezpečuje zobrazenie kategórií, z ktorých si užívateľ môže vybrať vlastné kategórie na vyhľadávanie.

Komponent má `state`, v ktorom hlavné fieldy sú zoznam súvisiacich hráčov a metóda na získanie nových súvisiacich hráčov, ktorá sa mení s ohľadom na ne/zvolené kategórie.

4.4.8 Komponenty SimilarPlayersViewer a SimilarPlayerComponent

Komponent `SimilarPlayesViewer` reprezentuje časť, kde sa zobrazujú súvisiaci hráči. Ako `state` si udržuje zoznam `similarPlayers: SimilarPlayer[]`, čo sú hráči na zobrazenie.

Vytvorí komponent `SimilarPlayerComponent` pre každého hráča v zozname. Kliknutím na daný komponent sa nastaví nový hráč v `Context` prostredníctvom SetNewPlayer funckie.

Typ SimilarPlayer má nasledujúcu štruktúru:

```
type SimilarPlayer = {
    similarPlayerLabel: {
        value: string;
    };
    similarPlayerWikidataULR: {
        value: string;
    };
    similarCategories: Category[];
};

type Category = {
    category: {
        value: string;
    };
    categoryLabel: {
        value: string;
    };
};
```

4.4.9 Komponent SimilarPlayersSelector

Komponent reprezentuje výber kategórií. Každá kategória je zobrazená ako vstup typu `checkbox`. Ako `state` si komponent najmä udržuje všetky užívateľom zvolené kategórie.

Ak je užívateľom potvrdený výber kategórií, nájde sa nová skupina súvisiacich hráčov pomocou triedy Seeker a nastaví sa nová metóda na získanie nových hráčov v `state` komponentu SimilarPlayers. Nová metóda je nastavená tak, aby sa nový hráči vyhľadávali podľa zvolených kategórií.

4.4.10 Komponenty Loader a MainSpin

S využitím balíka `react-loading`[11] sa tieto komponenty využívajú pre zobrazenie animácie načítavania. Rôzne iné komponenty aplikácie ich používajú najmä pri volaní asynchrónnych funkcií.

4.4.11 Komponent Map

Komponent využíva balíček `react-google-maps`[12], ktorý umožňuje zobraziť mapu a vyznačiť na nej miesto pomocou získaných súradníc miesta narodenia hráča.

[11] `https://www.npmjs.com/package/react-loading`
[12] `https://www.npmjs.com/package/@react-google-maps/api`

4.4.12 Trieda Suggester

Trieda sa využíva na komunikáciu so Solr serverom. Komponent TextBox ju využíva na získanie hráčov na našepkávanie a na získanie ich QID.

Fieldy

- `suggestList: PlayerName[]` je zoznam hráčov v neupravenej forme, tak ako sme ich dostali od Solr servera.
- `listOfPlayers: SuggestedPlayer[]` je zoznam hráčov na našepkanie.

  `SuggestedPlayer` reprezentuje objekt s položkami `id` a `name`.

Metódy

- `async MakeCall (inputName: string)`

  Parameter `inputName` je vstup užívateľa pri vyhľadávaní. Daný vstup je poslaný Solr serveru, ktorý vráti zoznam piatich objektov, ktoré reprezentujú hráčov na našepkanie.
- `GetSuggestedPlayers (): SuggestedPlayer[]`

  Vráti zoznam maximálne piatich objektov typu `SuggestedPlayer`
- `GetPlayerQID (playerName: string): string`

  Komponent TextBox volá metódu pri potvrdení vyhľadania hráča. Metóda vráti hráčove QID alebo prázdny `string`.
- `ClearList()`

  Vymaže záznamy o hráčoch ak boli uložené vo fieldoch.

4.4.13 Trieda Player a funkcia SetNewPlayer

Inštancia triedy Player reprezentuje súčasného hráča a je uložená v `Context`.

Fieldy

- Fieldy typu **string**: `playerQID`, `playerPhotoUrl`, `playerName`, `birthDate`, `birthPlace`, `citizenship`, `weight`, `height`, `careerStart`, `careerEnd`.
- `birthPlaceCoords`: objekt, ktorý ma fieldy `lat` a `lng` typu number.
- `requester: Requester`

  Slúži na získavanie základných informácií pre hráča a údajov pre časové osi.
- `events: PlayerEvent[]`

  Zoznam eventov hráča.

Funkcia SetNewPlayer

Funkcia slúži na nastavenie nového hráča a získanie základných informácií, ktorého QID dostane ako parameter. Nový hráč je nastavený v `Context`.

4.4.14 Trieda Requester

Slúži na získanie informácií z Wikidata a doplnenie nových informácií získaných z DBpedia triedou DBpediaChecker. Informácie z Wikidata získava pomocou triedy Fetcher, ktorej posiela dotazy.

Fieldy

- `playerBasicInfo: PlayerInfo`

  Základné informácie o hráčovi.

- `playerEventInfo: PlayerEvent[]`

  Zoznam eventov hráča.

Metódy

- `InitializeQuery(playerQID: string)`

  Získa základné informácie pre hráča z Wikidata a uloží ich do fieldu `playerBasicInfo`. Následne s využitím triedy DBpediaChecker sa doplnia prípadné nové informácie.

- `AwardsQuery(playerQID: string)`

  Získa ocenenia hráča z Wikidata a uloží ich do `playerEventInfo`. Následne s využitím triedy DBpediaChecker sa doplnia prípadné nové ocenenia.

- `TournamentsQuery(playerQID: string)`

  Získa turnaje, na ktorých sa hráč zúčastnil z Wikidata a uloží ich do `playerEventInfo`. Turnaje sa nedopĺňajú pretože DBpedia ich pre hráčov neposkytuje.

- `PlayerTeamsQuery(playerQID: string)`

  Získa tímy, za ktoré hráč hral z Wikidata a uloží ich do `playerEventInfo`. Následne s využitím triedy DBpediaChecker sa doplnia prípadné nové tímy.

- `FixPlayerBasicInfo(dataDBPedia: PlayerInfoValidatable)`

  Doplní prípadné chýbajúce údaje do základných informácií o hráčovi.

- `AddEventInfo(newEvents: PlayerEventValidatable[],`
  `eventText: string)`

  Doplní nové eventy z DBpedia do zoznamu eventov.

4.4.15 Trieda DBpediaChecker

Slúži na získanie informácií z DBpedia. Posiela dotazy pre DBpedia pomocou triedy Fetcher. Pomocou triedy Validator porovnáva informácie získané z oboch datových zdrojov na žiadosť triedy Requester.

Fieldy

- `infoDBpedia: PlayerInfoValidatable`

  Základné informácie o hráčovi získané z DBpedia.

- `eventsDBpedia: PlayerEventValidatable[]`

  Zoznam eventov hráča získaných z DBpedia.

Metódy

- `async QueryDBpediaBasicInfo(playerQID: string)`

  Získa základné informácie pre hráča z DBpedia.

- `async QueryDBpediaPlayerTeams(playerQID: string)`

  Získa tímy, za ktoré hráč hral z DBpedia.

- `async QueryDBpediaPlayerAwards(playerQID: string)`

  Získa ocenenia hráča z DBpedia.

- ```
  CheckPlayerInfoFill(WikidataInfo: PlayerInfoValidatable):
      PlayerEventValidatable[]
  ```

  K základným informáciám z Wikidata od triedy Requester pridá základné informácie získané z DBpedia ako parameter do volania funkcie `ValidatePlayerInfo` triedy Validator.

- ```
  CheckPlayerEventsFill(
          WikidataEvents:PlayerEventValidatable[]
      ):PlayerEventValidatable[]
  ```

  K eventom z Wikidata od triedy Requester pridá dáta o eventoch získané z DBpedia ako parameter do volania funkcie `ValidatePlayerEvents` triedy Validator.

4.4.16 Trieda Validator

Trieda slúži na porovanie informácií získaných z Wikidata a DBpedia.

Metódy

- ```
  ValidatePlayerInfo(infoWikidata: PlayerInfoValidatable,
   infoDBpedia: PlayerInfoValidatable):PlayerInfoValidatable
  ```

  Metóda slúži na porovnanie základných informácií hráča, ak detekuje rozlišné informácie vráti DBpedia údaje na doplnenie do údajov získaných z Wikidata.

- ```
  ValidatePlayerEvents(eventsWikidata: PlayerEventValidatable[],
  eventsDBpedia: PlayerEventValidatable[]
  ):PlayerEventValidatable[]
  ```

  Metóda slúži na porovnanie názvov eventov hráča. Vráti zoznam eventov, ktoré sa v `eventsWikidata` nenachádzajú ale v `eventsDBpedia` sú.

4.4.17 Trieda Seeker

Trieda sa využíva na získanie kategórií hráča a na získanie súvisiacich hráčov. Dáta získava pomocou triedy Fetcher, ktorej posiela dotazy pre DBpedia.

Fieldy

- `similarPlayers: SimilarPlayer[]`

  Zoznam nájdených súvisiacich hráčov.

- `categories: Category[]`

  Zoznam kategórií pre súčasného hráča.

- `randomCategoryPairs: Category[][]`

  Zoznam všetkých dvojíc kategórií pre súčasného hráča. Využíva sa pri predvolenom vyhľadávaní súvisiacich hráčov.

- `offset: number`

  Slúži na posunutie výsledkov dotazu, ak je výsledkov viac.

- `firstDefaultView: boolean`

  Slúži na zmazanie všetkých hodnôt ak sa vyhľadávajú súvisiaci hráči pre hráča prvý krát.

Metódy

- `async GetPlayerCategories(playerQID: string)`

  Získa všetky kategórie hráča.

- ```
  async QueryForSpecifiedSimilarPlayers(
          categories: string[],
          basicInfo: { infoName: string; infoValue: string }[],
          playerQIDToIgnore: string
      )
  ```

  Získa skupinu súvisiacich hráčov na základe zvolených kategórií. Taktiež zvýši `offset`. Zvolené kategórie (a základné informácie) zakóduje do formátu aby vyhovoval SPARQL dotazu pomocou metód:

  - `EncodeCategoriesToSPARQLQuery` - pre kategórie
  - `EncodeBasicInfoToSPARQLQuery` - pre základné informácie

- ```
  async GetRandomSimilarPlayers(categories: Category[],
          playerQIDToIgnore: string)
  ```

  Získa skupinu súvisiacich hráčov na základe dvojice kategórií. Dvojica je získaná z fieldu `randomCategoryPairs`.

- `async FindSimilarPlayers(playerQID:string):`
  `Promise<SimilarPlayer[]>`

  Slúži na získanie súvisiacich hráčov pri predvolenom vyhľadávaní. V tejto metóde sa inicializuje zoznam dvojíc kategórií pre predvolené vyhľadávanie. Hráči sa vyhľadávajú, kým počet nájdených hráčov nie je aspoň 5 alebo kým sa neskúsilo viac ako 200 možných kombinácií dvojíc.

- `EncodeBasicInfoToSPARQLQuery(`
  `basicInfo: { infoName: string; infoValue: string }[]`
  `)`

  Metóda každú položku zo zoznamu základných informácií upraví tak, aby bola vo forme tvrdenia pre SPARQL dotaz. Pre výšku a váhu hráča sa určí interval pre prípadné menšie výchylky v údajoch.

- `EncodeCategoriesToSPARQLQuery(categories: string[])`

  Metóda každú položku zo zoznamu kategórií upraví tak, aby bola vo forme tvrdenia pre SPARQL dotaz.

4.4.18 Trieda WikidataEditor

Slúži na doplnenie základných informácií do Wikidata. Využíva nástroj QuickStatements [13], ktorý umožňuje dopĺňať tvrdenia do Wikidata. Obsahuje metódu `Edit` na doplnenie tvrdenia:
`async Edit (userName: string, token: string, statement: string).`

- `userName` je meno užívateľa na Wikidata.
- `token` je užívateľovi priradený token prostredníctvom QuickStatements.
- `statement` je tvrdenie v syntaxi QuickStatements, ktoré sa chce doplniť.

Obsahuje niekoľko metód na upravenie a kontrolu požadovanej hodnoty na doplnenie. Nasleduje príklad jednej z metód pre kontrolu a úpravu výšky.

```
ValidIntegerInput(input: string) {
       if (input[0] == "0" || input.includes(".") ||
           input.includes(",") || input.includes("e") ||
           input.includes("E") || input.includes("-")) {
           return false;
       }
       return true;
   }

DecoratePlayerHeight(height: string) {
       if (!this.ValidIntegerInput(height)) return "";

       const num = parseInt(height);
```

[13] https://www.wikidata.org/wiki/Help:QuickStatements

```
        if (Number.isInteger(num) && num > 0) {
            return height + QSsuffixes.CM;
        }
        return "";
    }
```

QSsuffixes.CM je jednotka pre centimetre v syntaxi QuickStatements. Centimeter má QID Q174728 a je potrebné nahradiť písmeno „Q“ za „U“. A preto QSsuffixes.CM má hodnotu U174728.

Ak by sme chceli doplniť tvrdenie u hráča LeBron James, v ktorom by sme chceli uviesť, že je vysoký 206 centimetrov, tvrdenie by vyzeralo následovne (znak „|“ sa používa na separáciu častí tvrdenia).

```
Q36159|P2048|206U174728
```

- `Q36159` je LeBron James (QID)
- `P2048` je Wikidata vlastnosť pre výšku
- `206U174728` je hodnota 206 centimetrov

4.4.19 Trieda Fetcher

Fieldy

- `WikidataEndpointUrl: string`
- `DBPediaEndpointUrl: string`

  Fieldy reprezentujú jednotlivé endpointy datových zdrojov.

Metódy

- `async MakeSPARQLQueryWikidata(sparqlQuery: string)`
- `async MakeSPARQLQueryDBpedia(sparqlQuery: string)`

Metódy slúžia na získanie údajov špecifikovaných v dotaze prijatom ako parameter. Dotaz sa zakóduje do pomocou JavaScriptovej funkcie `encodeURIComponent`. V hlavičke (Wikidata) alebo `format` (DBpedia) uvedieme, že požadujeme výsledok vo formáte JSON.

Implementácia `MakeSPARQLQueryWikidata` metódy:

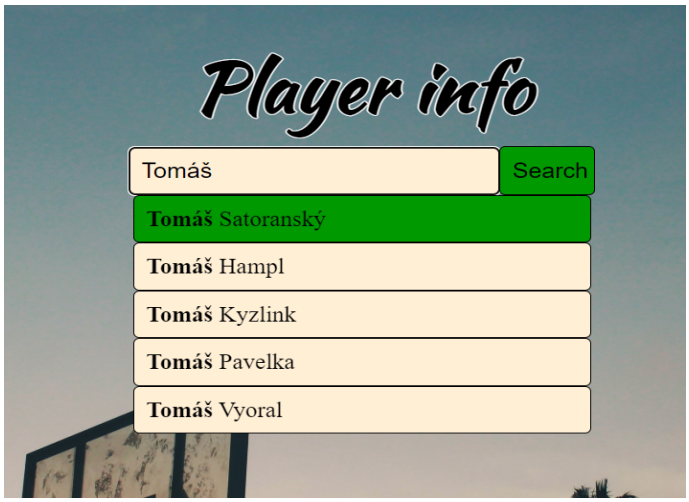
```
async MakeSPARQLQueryWikidata(sparqlQuery: string) {
        const fullUrl = this.WikidataEndpointUrl + "?query=" +
            encodeURIComponent(sparqlQuery);
        const headers = { Accept: "application/sparql-results+json" };

        return fetch(fullUrl, { headers })
        .then((data) => data.json())
        .catch(error => console.error("Error: ", error));
    }
```

5. Užívateľské rozhranie

5.1 Vyhľadávanie

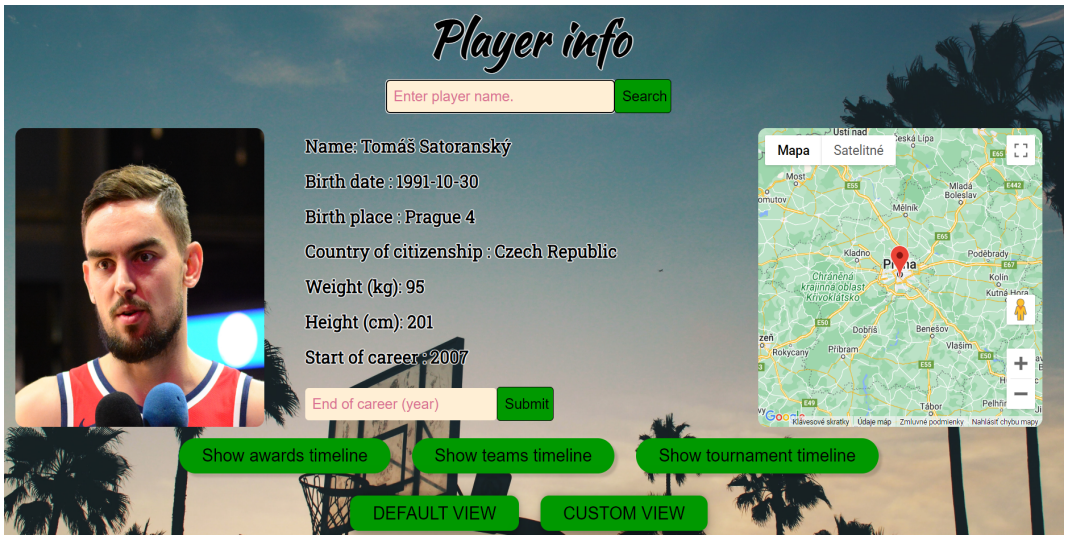
Po spustení aplikácie sa užívateľovi zobrazí textové pole na vyhľadávanie hráčov. Ako užívateľ zadáva meno hráča, tak sú mu našepkávaní hráči.



Obr. 5.1: Vyhľadávanie hráčov a našepkávanie.

5.2 Základné informácie

Po vyhľadaní hráča sa zobrazia základné informácie o hráčovi, tlačidlá na zobrazenie časových osí a tlačidlá pre vyhľadávanie súvisiacich hráčov. Taktiež je zobrazené textové pole na opätovné vyhľadávanie.



Obr. 5.2: Zobrazenie základných informácií o hráčovi.

5.3 Súvisiaci hráči

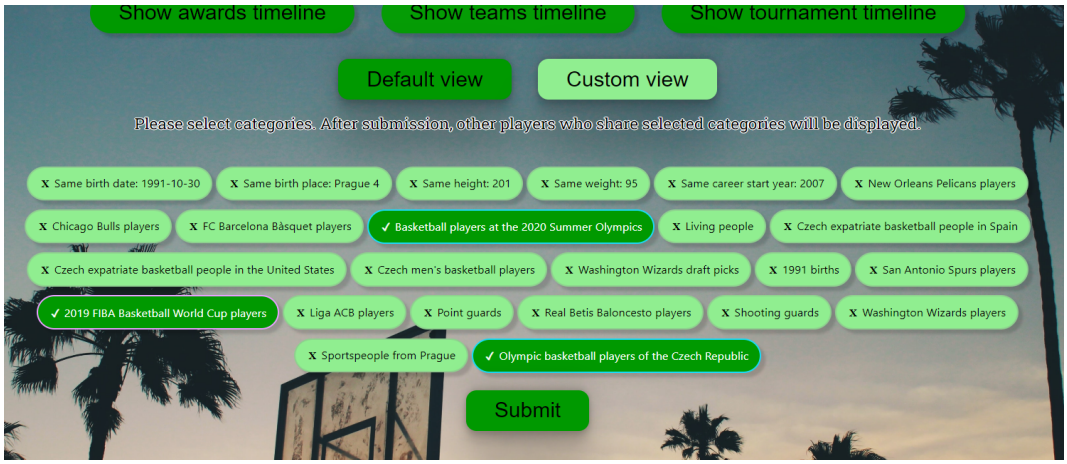
5.3.1 Default View

Po kliknutí na tlačidlo „Default view“ sa zobrazí skupina hráčov, ktorí s vyhľadaným hráčom zdieľajú dvojicu kategórií.

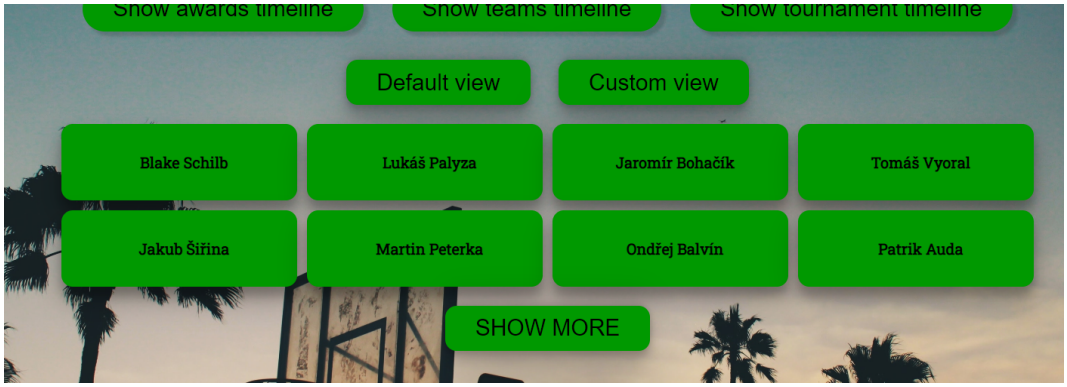
Obr. 5.3: Zobrazenie súvisiacich hráčov, ktorí zdieľajú dve spoločné kategórie.

5.3.2 Custom View

Po kliknutí na tlačidlo „Custom view“ si užívateľ môže vyberať zo zoznamu kategórií. Zoznam je tvorený aj základnými informáciami o hráčovi. Kategóriu užívateľ vyberie kliknutím na ňu. Kliknutím na tlačidlo „Submit“ spustí vyhľadávanie súvisiacich hráčov na základe zvolených kategórií.



Obr. 5.4: Výber spoločných vlastností a kategórií.



Obr. 5.5: Zobrazenie súvisiacich hráčov na základe voľby užívateľa.

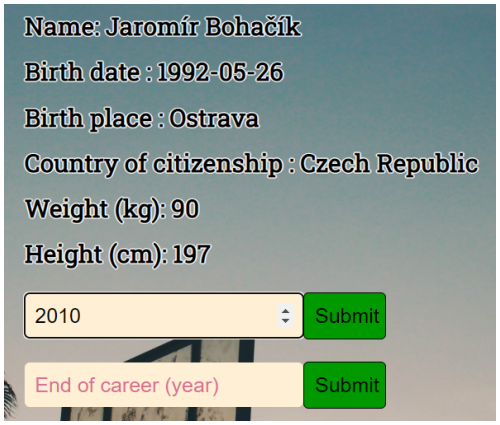
5.4 Časová os

Po kliknutí na jedno z tlačidiel pre časové osi sa zobrazia eventy hráča. Pre každý event je samostatné pole, v ktorom sú poskytnuté informácie o evente. Z tohto pohľadu sa užívateľ môže pomocou tlačidla vrátiť späť na základné informácie, zobraziť iné časové osi alebo vyhľadať nového hráča.

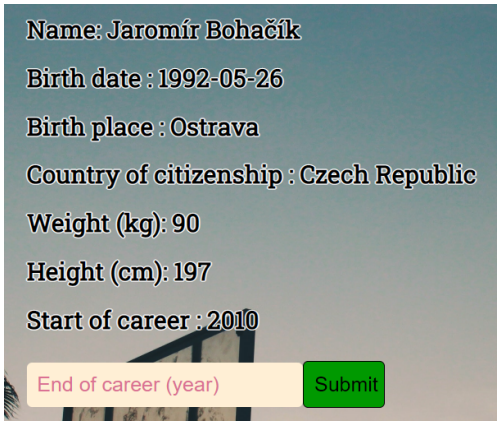
Obr. 5.6: Zobrazenie časovej osi hráča (turnaje).

5.5 Doplnenie základných informácií

Každú chýbajúcu informáciu môže užívateľ doplniť pomocou textového poľa na mieste danej informácie. Kliknutím na tlačidlo „Submit“ doplní danú informáciu ak je v správnom tvare.



Obr. 5.7: Doplnenie základných informácií o hráčovi. (PRED)



Obr. 5.8: Doplnenie základných informácií o hráčovi. (PO)

6. Užívateľské testovanie

6.1 System Usability Scale

System Usability Scale (SUS) [1] je test, ktorý testuje použiteľnosť systému. Je zložený z 10 otázok s piatimi možnosťami pre každú otázku. Možnosti sú na škále od „Silne nesúhlasím“ do „Silne súhlasím“.

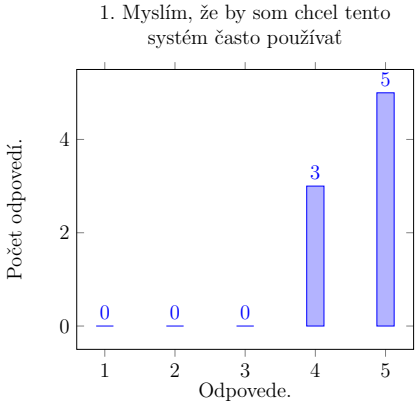
Osobám, ktoré sa zúčastnia testovania, bude zadaných nasledovných 5 úloh, ktoré sú napísané tak, aby testované osoby nemuseli mať nijaké znalosti z oblasti basketbalu. Každá úloha je zameraná na odlišnú funkcionalitu aplikácie a tak po úspešnom či neúspešnom prejdení všetkých úloh bude testovaná osoba schopná objektívne zhodnotiť použiteľnosť aplikácie a prívetivosť užívateľského rozhrania.

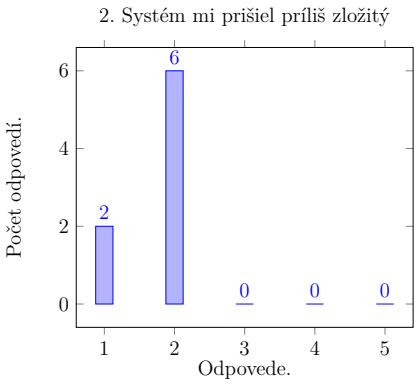
6.1.1 Úlohy

1. Zistite koľko kíl váži hráč Lebron James.
2. Overte či hráč Dirk Nowitzki získal ocenenie NBA MVP (most valuable player).
3. Zistite, od ktorého roku hrá hráč Stephen Curry za tím Golden State Warriors.
4. Zobrazte všetkých spoluhráčov Tomáša Satoranského z českého tímu, ktorí sa s ním zúčastnili posledných letných olympijských hier.
5. Hráčovi Jaromír Bohačík chýba údaj o roku, kedy začal hrať profesionálne basketbal (2010). Doplňte mu ho.

6.1.2 Výsledky testovania

Nižšie môžeme vidieť zoznam desiatich otázok. Pri každej otázke je graf, ktorý znázorňuje počet odpovedí pre jednotlivé možnosti. Odpovede sú na stupnici 1-5, kde 1 znamená „Silne nesúhlasím“ a 5 znamená „Silne súhlasím“.

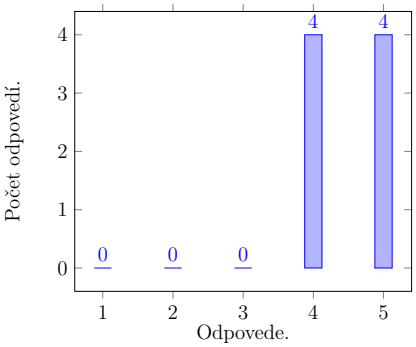




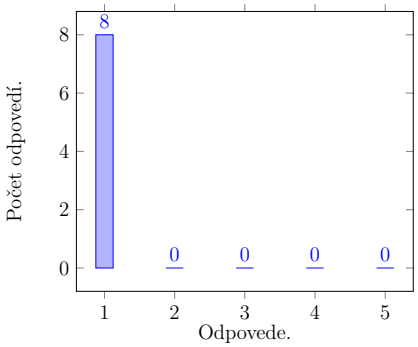
.

[1] Will (2017)

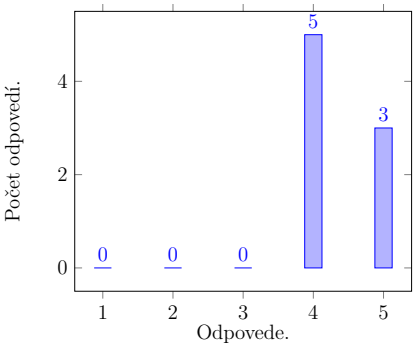
3. Systém mi prišel ľahko použiteľný



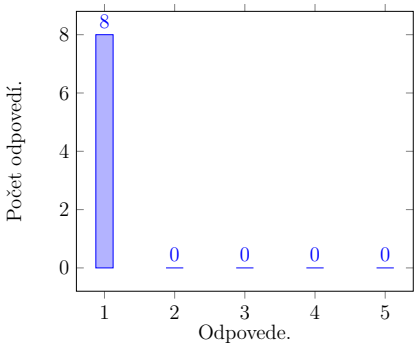
4. K tomu, aby som mohol/la používať tento systém by som potreboval(a) podporu technického personálu



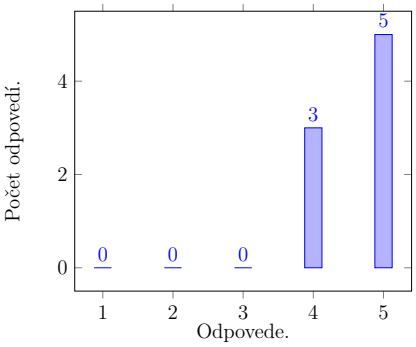
5. Myslím, že rôzne funkcie tohto systému sú dobre integrované



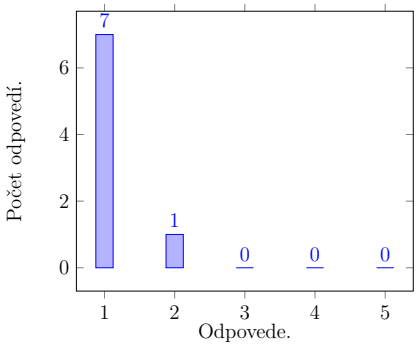
6. Myslím, že je systém príliš nekonzistentný (rozdielne názvy pre rovnaké veci, rozdielne ovládánie podobných prvkov...)



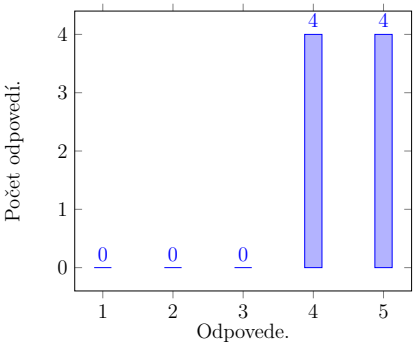
7. Myslím, že väčšina ľudí by sa naučila s týmto systémom pracovať veľmi rýchlo



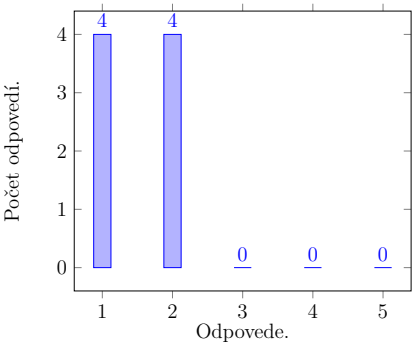
8. Systém mi prišiel veľmi ťažkopádny



9. Pri používání systému som sa cítil(a), že viem čo robím



10. Pred použitím systému som sa musel(a) naučiť veľa vecí



Výpočet: Zo zozbieraných odpovedí sa počíta výsledné skóre následovne:

- Pre každú otázku sa spraví priemer z bodov odpovedí
- Sčítajú sa body za nepárne otázky a od výsledku sa odpočíta 5
- Sčítajú sa body za všetky párne otázky a výsledok sa odpočíta od čísla 25
- Výsledky z prvých dvoch bodov sa sčítajú a vynásobia číslom 2.5

Podľa prieskumov priemerný systém skóre SUS dosahuje skóre **68**. Skóre pod 68 sa považuje ako podpriemer a skóre nad 68, ako nadpriemer. Na základe výsledku testovania, ktorého za zúčastnilo 8 osôb, vychádza skóre tejto aplikácie na **90,6**.

Záver

V rámci tejto práce bola navrhnutá, implementovaná a otestovaná webová aplikácia podľa zadaných požiadavok. Webová aplikácia umožňuje užívateľovi vyhľadávať basketbalových hráčov, kde pre každého nájdeného hráča sa zobrazí fotografia, základné informácie a miesto narodenia na mape, samozrejme len vtedy, ak to všetky potrebné dáta dotazované znalostné grafy obsahujú. V prípade ak znalostný graf Wikidata a ani DBpedia nedisponuje nejakou základnou informáciou, tak užívateľovi je umožnené danú informáciu doplniť prostredníctvom aplikácie.

Taktiež okrem základných informácií si užívateľ dokáže pozrieť tri druhy časových osí, ktoré sú zamerané na tímy, ocenenia a turnaje. Dokáže vyhľadávať súvisiacich hráčov, ktorý sa zobrazujú buď podľa dvojice kategórií, ktoré spolu zdieľajú alebo podľa užívateľom vybraných kategórií a základných vlastností.

Aplikácia bola taktiež úspešne otestovaná pomocou metódy SUS, ktorá nám vypočítala skóre použiteľnosti systému.

Nevýhodou práce je, že pre zobrazenie miesta narodenia hráča na mape a pre doplnenie údajov do Wikidata je potrebné dodať do zdrojového kódu aplikácie API kľúč a token, ktorých získanie je popísané v prílohe A.2 Doplnky aplikácie.

Aplikáciu by určite bolo možné ďalej rozširovať, v prvom rade by sa dalo zapracovať na lepšej užívateľskej prívetivosti, možnosti výberu jazyka a pod.

Tiež by sa mohol vylepšiť spôsob vyhľadávania súvisiacich hráčov, tak aby sa napríklad pri predvolenom vyhľadávaní zobrazovali ako prví takí hráči, ktorí zdieľajú s vyhľadaným hráčom najviac kategórií. Alebo poskytnúť užívateľovi viac možností pri výbere vlastných kategórií - umožniť určiť intervaly hodnôt na základe ktorých hľadať (váha, dátum narodenia, miesto narodenia - v istom okruhu), zobraziť súvisiacich hráčov na mape.

Rozšíriť by bolo možné aj dopĺňanie údajov o hráčovi. Namiesto QID doplniť názov miesta/krajiny, ktoré by mohlo byť našepkávané. Umožniť dopĺňanie eventov pre časové osi a pod.

Zoznam použitej literatúry

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R. a Ives, Z. (2007). Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer. URL https://link.springer.com/chapter/10.1007/978-3-540-76298-0_52.

Beckett, D., Berners-Lee, T., Prud'hommeaux, E. a Carothers, G. (2014). Rdf 1.1 turtle. *World Wide Web Consortium*, pages 18–31. URL https://www.w3.org/TR/2014/REC-turtle-20140225/.

Harris, S., Seaborne, A. a Prud'hommeaux, E. (2013). Sparql 1.1 query language. *W3C recommendation*, **21**(10), 778. URL https://www.w3.org/TR/sparql11-query/.

Raimond, Y. a Schreiber, G. (2014). Rdf 1.1 primer. *W3C Note. W3C*. URL https://www.w3.org/TR/rdf11-primer/.

Vrandečić, D. a Krötzsch, M. (2014). Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, **57**(10), 78–85. URL https://dl.acm.org/doi/fullHtml/10.1145/2629489.

Will, T. (2017). Measuring and interpreting system usability scale (sus). *UIUX Trend*. URL https://uiuxtrend.com/measuring-system-usability-scale-sus/.

Zoznam obrázkov

A. Prílohy

A.1 Solr server

Pre spustenie Solr servera (9.0) je potrebné mať stiahnutú Java 11 verziu alebo novšiu. Verziu 18 je možné stiahnuť na tomto odkaze: `https://www.oracle.com/java/technologies/downloads/#jdk18-windows`.

Link na stiahnutie už nakonfigurovaného servera je v A.3 Online odkazy.

Po rozbalení sa stačí dostať do priečinka `bin` v ktorom spustiť príkaz

```
solr start -all
```

Server by mal byť dostupný na `http://localhost:8983`.

Príkaz na vypnutie servera:

```
solr stop -all
```

Pre vytvorenie vlastného Solr `core`, je potrebné stiahnuť Apache Solr [1]. Nakonfigurovaný server má Solr verziu 9.0, no je možné použiť aj staršie verzie. Po stiahnutí je potrebné sa dostať do priečinka `bin` a vykonať nasledujúce kroky/príkazy.

- `solr create -c [nazov-core]`

  Pre vytvorenie jadra, kde [nazov-core] je meno jadra.

- `post -c [nazov-core] players.json`

  Kde `players.json` musí byť cesta k dátam na zaindexovanie, alebo stačí pre jednoduchosť premiestniť dáta do `bin` priečinka.

- Týmto je vytvorené jadro so zaindexovanými dátami. Následne je potrebné v súbore `server/solr/[nazov-core]/conf/solrconfig.xml` doplniť kód pre upravenie konfigurácie, ktorý je možné nájsť v časti 4.1 Solr server.

- Nakoniec je ešte potrebné zapnúť CORS [2].

A.2 Doplnky aplikácie

Aplikácia využíva Google Maps JavaScript API[3], ktoré umožňuje zobraziť mapu s miestom narodenia hráča. Taktiež využíva nástroj QuickStatements, pomocou ktorého dokáže doplniť predpripravené tvrdenia do Wikidata.

[1] https://solr.apache.org/downloads.html

[2] https://opensourceconnections.com/blog/2015/03/26/going-cross-origin-with-solr/

[3] https://developers.google.com/maps/documentation/javascript/overview

A.2.1 API kľúč

Pre zobrazenie mapy je potrebné získať API kľúč [4] a pridať ho ako návratovú hodnotu funkcie `GetApiKey()`, ktorá sa nachádza v súbore
`src/infoComponent/mainInfoComponent/map/api_key.ts`

A.2.2 QuickStatements token

Pre získanie tokena je potrebné mať účet na Wikidata, ktorý je starší ako 4 dni a má viac ako 50 editácií.

Token sa získa prihlásením sa na QuickStatements [5] v časti `user`. Je potrebné ho pridať spolu s prihlasovacím menom ako návratové hodnoty pre `GetQSToken()` a `GetQSUserName()`, ktoré sa nachádzajú v súbore

`src/infoComponent/mainInfoComponent/playerInfoComponent/QSlogin.ts`

A.3 Online odkazy

Link na stiahnutie nakonfigurovaného Solr servera: `https://drive.google.com/file/d/10loRIagyE7C4XbBi4iZAdtACpuRql0il/view?usp=sharing`

Repozitár aplikácie: `https://gitlab.mff.cuni.cz/fedakri/player-info`

A.4 Zdrojové kódy

Prílohou k tejto práci sú zdrojové kódy webovej aplikácie.

[4] https://developers.google.com/maps/documentation/javascript/get-api-key
[5] https://quickstatements.toolforge.org