Interoperabilnost informacijskih sustava i otvorenost podataka

Hanić, Marin

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Marin Hanić

INTEROPERABILNOST INFORMACIJSKIH SUSTAVA I OTVORENOST PODATAKA

Diplomski rad

Voditelj rada:
dr.sc Ognjen Orel

Zagreb, rujan, 2022

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Sadržaj

Uvod

U današnjem svijetu informacijski sustav ne može doseći značajnu razinu uporabljivosti ako ne razmjenjuje informacije s drugim sustavima, aplikacijama ili platformama. Prvi potreban korak koji svaki sustav mora napraviti kako bi razmijenjivao informacije je uspostava interoperabilnosti s drugim sustavima. Za današnje društvo su posebno važni sustavi koji sadrže velike količine organiziranih i uređenih podataka koji su javno dostupni. Važno svojstvo tih podataka je upravo to da su javno dostupi čime je omogućeno njihovo korištenje i publikacija u drugim sustavima i platformama.

Jedan takav sustav za razmjenu i obradu podataka je Open Access Infrastructure for Research in Europe poznatiji kao OpenAIRE. OpenAIRE je Europski sustav osnovan 2018. godine koji trenutno okuplja više od 30 zemalja članica. Jedna od tih zemalja članica je i Hrvatska. Kako bi se neki sustav povezao s OpenAIRE platformom, on mora zadovoljavati Open Archives Initiative Protocol for Metadata Harvesting ili skraćeno OAI-PMH protokol.

U idućim poglavljima ćemo objasniti značenje i važnost otvorenih podataka. Posebno ćemo spomenuti otvorene javne podatke i ulogu otvorenih podataka u znanosti. Kao primjer uređenog skupa podataka ćemo opisati CRIS (Current Research Information System). U posebnom poglavlju ćemo detaljnije opisati OpenAIRE i njegovu ulogu u otvorenoj znanosti, a također ćemo osim OpenAIRE-a objasniti i OAI-PMH protokol. Detaljno ćemo opisati njegovu specifikaciju i implementaciju te kako se povezuje s OpenAIRE platformom. Nakon što smo se upoznali s OpenAIRE-om i OAI-PMH protokolom, prikazat ćemo kako povezivanje izgleda u praksi na primjeru aplikacije koja povezuje Informacijski sustav znanosti Republike Hrvatske (CroRIS) i OpenAIRE. U radu ćemo pokazati važnije isječke koda u programskom jeziku Java i rezultate praktičnog rada.

Poglavlje 1

Otvoreni podaci

1.1 Što su otvoreni podaci?

Definiciju otvorenih podataka možemo pronaći na stranici [6] i ona glasi: ”Otvoreni podaci su podaci koje može besplatno koristiti, mijenjati i dijeliti bilo tko za bilo koju svrhu”. Tu definiciju bismo mogli bolje pojasniti ako objasnimo značenje riječi otvoreni. Riječ otvoreni u našem kontekstu možemo poistovjetiti s tri svojstva: dostupnost, mogućnost ponovne uporabe i univerzalna pristupačnost.

Dostupnost podataka nam govori da otvoreni podaci moraju biti u potpunosti dostupni u pristupačnom i promjenjivom formatu te je poželjno imati mogućnost da ih se može preuzeti s interneta. To znači da niti jedan dio otvorenih podataka ne bi smio biti skriven i da bi podaci morali biti dostupni u formatima u kojima ih korisnici mogu besplatno čitati.

Drugo svojstvo je mogućnost ponovne uporabe. Pod tim svojstvom podrazumijevamo da otvoreni podaci smiju biti korišteni neograničen broj puta te da se smiju dalje dijeliti drugim korisnicima i izmjenjivati i miješati s drugim podacima.

Posljednje i možda najvažnije svojstvo je pristupačnost. Pristupačnost otvorenih podataka nam kaže da podatke smije preuzimati, koristiti i dijeliti bilo tko bez ikojeg oblika diskriminacije. Pod diskriminacijom se u svijetu podataka najčešće misli na ograničenja poput: samo za privatne svrhe, samo za edukacijske svrhe i tako dalje.

1.2 Otvoreni podaci u znanosti

Otvoreni podaci u znanosti su vrsta otvorenih podataka koja se usredotočuje na objavljivanje postupaka i rezultata znanstvenih istraživanja. Svrha otvorenih podataka u znanosti je mnogo važnija nego kod ostalih podataka. Njihovo objavljivanje služi za provjeru istinitosti podataka i pomaže u budućim istraživanjima i napretku znanosti. Osim toga objave

takvih radova su i dokaz o autorstvu nad objavljenim radovima, a kroz citiranost radova se pruža uvid u značajnost samog otkrića.

U prošlosti se zbog nedostatka definicije znanstvenih podataka pojavio problem prirode znanstvenih podataka. Problem je postavljao pitanje: koji sve podaci mogu biti znanstveni podaci? Prvi pokušaj definiranja znanstvenih podataka je 1999. godine dala Nacionalna akademija znanosti u SAD-u i opisala ih kao: *"Činjenice, slova, brojevi ili simboli koji opisuju objekte, stanja, situacije ili ostale faktore"* [1]. Takva definicija je bila nepotpuna pa su 2011. u popis dodani razni objekti poput genomskih sekvenci, elektronskih reprezentacija literature, podataka koje generiraju ili obrađuju računala i slični objekti.

Najveći pomak u otvorenosti znanstvenih podataka se dogodio 2016. godine objavom *"FAIR Guiding Principles for scientific data management and stewardship"* [2] , odnosno smjernica koje govore kako treba izgledati "dobro" upravljanje podataka. Akronim *FAIR* na engleskom označava upravo ključne principe otvorenih podataka: mogućnost pronalaženja (Findability), pristupačnost (Accessibility), interoperabilnost (Interoperability) i mogućnost ponovnog korištenja (Reuse). Ovi principi su osobito naglašeni u okvirima strojnog djelovanja, odnosno mogućnostima računalnih sustava da pronađu podatke, pristupe im i bez posebne prilagodbe ih koriste uz minimalnu ljudsku pomoć. U moderno vrijeme su te kvalitete jako važne zbog sve većeg povećanja veličine podataka, njihove složenosti i brzine stvaranja novih podataka. Zbog tolike važnosti FAIR principa u računalnom svijetu ukratko ćemo opisati svaki princip i dati par smjernica kako svaki od principa uspješno ostvariti u praksi.

Mogućnost pronalaženja je prvo i očigledno ključno svojstvo otvorenih podataka. Sve otvorene podatke bi trebalo biti jednostavno pronaći i za ljude i za računala. Upravo zbog lakšeg pronalaska za računala, otvoreni podaci moraju imati metapodatke koje računala razumiju i iz njih mogu automatizmom pronaći sve ostale skupove podataka. Kako bi to bilo ostvarivo, otvoreni podaci bi trebali zadovoljiti držati četiri važne i korisne smjernice:

- Metapodacima i podacima moraju biti pridruženi univerzalni i konzistentni jedinstveni identifikatori. Dobar primjer takvih identifikatora su ORCID (Open Researcher and Contributor ID)[3] i DOI(Digital Object Identifier)[4].

- Podaci moraju biti opisani bogatim metapodacima, odnosno morali bi imati opisne informacije o kontekstu, kvaliteti, stanju i karakteristikama podataka.

[1] DC: The National Academies Press Washington, A Question of Balance: Private Rights and the Public Interest in Scientific and Technical Databases (1999) [9]

[2] Mark D. Wilkinson, et.al, The FAIR Guiding Principles for data management and stewardship [10]

[3] https://info.orcid.org/what-is-orcid/

[4] https://www.nsk.hr/doi/

- Metapodaci jasno i eksplicitno sadrže identifikatore podataka koje opisuju. Metapodaci i podaci se često opisuju u različitim datoteka zbog čega je ovo izrazito važno.

- Metapodaci i podaci bi trebali biti indeksirani i lako pretraživi. Samo postojanje metapodataka i podataka ne garantira da ih se može pronaći, ali zato indeksiranost tome uvelike pridonosi. Indeksiranjem podaci postaju računalima lako pretraživi, a to je upravo ono čemu težimo.

Pristupačnost je drugo važno svojstvo otvorenih podataka. Jednom kada pronađemo podatke koje tražimo želimo im pristupiti. Iz tog razloga je važno da korisnik ili računalo koje je tražilo podatke zna kako im pristupiti, što može uključivati i autorizaciju. Kako bi to olakšali preporuka je držati se dvije smjernice:

- Metapodaci i podaci moraju biti dohvatljivi pomoću njihovog identifikatora koristeći standardiziran komunkacijski protokol. Primjer takvih protokola su TCP ili https. Uz dohvatljivost podataka pomoću protokola važno je da ukoliko podaci nisu otvoreni i univerzalno dostupni, protokol mora omogućiti autorizaciju gdje je potrebna.

- Metapodaci moraju ostati dostupni čak i nakon što podaci više nisu. Na prvu ruku princip zvuči nejasno, ali metapodaci bi trebali sadržavati informacije o osobama, ustanovama i/ili identifikatorima koji bi mogli pomoći u pronalasku podataka koje tražimo na drugom mjestu. Važno je razumjeti da su bogati metapodaci koje smo opisivali kod prethodnog svojstva gotovo jednako važni kao i sami podaci.

Treće svojstvo koje otvoreni podaci moraju imati je interoperabilnost. Interoperabilnost je važna jer podaci koje pronađemo često moraju biti uklopljeni s drugim sličnim podacima. Također, te podatke će koristiti razne aplikacije za analizu, preradu i spremanje i podaci ne bi smjeli ometati te procese. Kako bi to bilo moguće podaci bi morali zadovoljavati:

- Metapodaci i podaci moraju koristiti modele podataka i rječnike koji su formalni, pristupačni i široko primjenjivi za prikazivanje podataka. Time omogućujemo aplikacijama da lako koriste podatke bez potrebe za dodatnim obradama i mapiranjima podataka. Primjeri dobrih formata podataka bi bili JSON ili XML.

- Metapodaci i podaci moraju koristiti rječnike koji zadovoljavaju FAIR principe. Već smo spomenuli da rječnici moraju biti formalni, pristupačni i primjenjivi, a rječnici koji zadovoljavaju FAIR principe su upravo takvi. Tu najčešće govorimo o kontroliranim rječnicima koji olakšavaju razumijevanje podataka koji ih koriste.

- Metapodaci i podaci sadrže dobre poveznice na druge metapodatke i podatke. Ovdje govorimo o dva svojstva: postoji poveznica između podataka i poveznica je dobro

opisana. Postojanje povezanosti među podacima je često i korisno jer podaci nerijetko opisuju jedni druge. Dobro opisanu poveznicu možemo razumjeti na primjeru. Dobra poveznica podataka X i Y je: ”X je poglavlje u knjizi Y”, dok bi loša poveznica bila: ”X je u vezi s Y” ili poveznica koju često pronalazimo: ”X see also Y”. Dobra poveznica pruža bolje razumijevanje i snalaženje u podacima.

Posljednje svojstvo je ujedno i razlog nastanka FAIR principa: mogućnost ponovnog korištenja. Upravo optimizacija ponovnog korištenja je cilj svih FAIR principa. Kako bi se to postiglo metapodaci i podaci moraju sadržavati mnoštvo preciznih i relevantnih atributa, sadržavati jasnu i pristupačnu licencu za korištenje, biti povezani s podacima o izvoru podataka i zadovoljavati standarde domene u kojoj su relevantni. Atributi u metapodacima i podacima moraju pružiti kontekst kako bi korisnik, bilo to računalo ili čovjek, mogao odlučiti jesu li mu podaci zapravo korisni. Kako bi se to lakše postiglo atributi bi trebali dati odgovor na nekoliko točaka:

- Koji je bio razlog prikupljanja ili stvaranja podataka?
- Postoje li neki detalji ili ograničenja podataka kojih bi drugi korisnici morali biti svjesni?
- Jesu li podaci već obrađeni?
- Koje je značenje pojedinih varijabli? Odnosno, sva imena varijabli bi trebala biti objašnjena.
- Koja je verzija podataka? Potrebno je dokumentirati i specificirati verziju pohranjenih ili korištenih podataka.

Gore opisani principi se odnose na tri tipa entiteta: podatke (ili digitalne objekte), metapodatke (informacije o digitalnim objektima) i infrastrukturu. Više o procesu prilagodbe podataka da zadovoljavaju FAIR principe možemo pronaći u [3] i njima se više nećemo posebno baviti. Kako smo već spomenuli, težnja FAIR principa je optimizacija ponovnog korištenja otvorenih podataka. U današnjem svijetu sve je veća potreba potreba prikupiti podatke i omogućiti njihovo korištenje sve većem broju ljudi, a to se osobito odnosi na znanstvene podatke. Iz tog razloga su podaci koji zadovoljavaju FAIR principe izuzetno vrijedni i važni mnogim modernim servisima čija je svrha objediniti i širiti znanstvene podatke. Upravo jedan takav servis koji čini srce Otvorene znanosti u Europi[5] je OpenAIRE.

[5] https://openscience.eu/

Poglavlje 2

OpenAIRE

2.1 Što je OpenAIRE?

OpenAIRE[1] ili punim nazivom "Open Access Infrastructure for Research in Europe" (Istraživačka infrastruktura za otvoreni pristup u Europi) je europski projekt potpore pokretu Otvorene znanosti u Europi, ali i ostatku svijeta. OpenAIRE kao organizacija je posvećena promoviranju, obrazovanju i prakticiranju otvorene znanosti. Osim toga OpenAIRE služi i kao tehnička infrastruktura koja skuplja znanstvene podatke i radove od povezanih pružatelja podataka. Upravo to skupljanje znanstvenih radova je omogućilo OpenAIRE-u da postane otvorena i održiva infrastruktura za obrazovanje.

OpenAIRE je osnovala Europska komisija 2008. godine u sklopu "7th Framework Programme (FP7)" s ciljem potpore u osnivanju Otvorene znanosti u Europi. Osnovni ciljevi bili su promoviranje i prihvaćanje načela Otvorene znanosti koje su donijeli Europska znanstvena organizacija ERC (European Research Council[2]) i pilot pokreta "Otvoreni pristup" koje je pokrenula Europska komisija[3]. OpenAIRE je napravio infrastrukturu za znanstvenike koja im pomaže da, u skladu s načelima koje su donijeli, mogu povezivati svoje znanstvene časopise. To je dovelo do stvaranja velikog distribuiranog sustava pomoći u čijem središtu se nalazi 27 nacionalnih podrška za Otvorenu znanost, takozvanih NOAD-ova (National Open Access Desks), koji su rasprostranjenih po cijeloj Europi. U tim počecima OpenAIRE je stvorio i Zenodo, digitalni repozitorij za sve znanstvenike koji nemaju pristup repozitorijima u odgovarajućim ustanovama.

U idućoj iteraciji napretka, 2012. godine je pokrenut program OpenAIREplus. Cilj novog projekta je bio unaprijediti i proširiti OpenAire i omogućiti povezivanje raznih podataka od već postojećih publikacija do shema financiranja. Ovaj projekt je obuhvaćao

[1] OpenAIRE [8]

[2] https://erc.europa.eu/homepage

[3] https://ec.europa.eu/info/index_en

više od četrdeset uglavnom europskih partnera. Mreža publikacija je proširena tako da uklopi pružatelje podataka raznih domena, zbog čega je u djelovanje puštena nova tehnička infrastruktura. Nova infrastruktura je omogućila povezivanje i upravljanje znanstvenim podacima.

U sklopu FP8 programa "Horizon 2020" Europska komisija je proširila pilot projekt "Otvoreni pristup" i pretvorila ga u projekt koji će obuhvatiti sve projekte vezane za otvorenu znanost i nazvala ga "Otvoreni podaci". U korak s time 2015. godine dolazi OpenAIRE2020 projekt koji je stvoren kako bi pomogao implementirati i nadgledati sve te aktivnosti. Ideja OpenAIRE2020 je uklopiti otvorena stipendiranja i značajno unaprijediti mogućnost pronalaska i ponovnog korištenja znanstvenih podataka i publikacija. Ta inicijativa je okupila više od pedeset stručnjaka raznih stručnosti iz institucija u cijeloj Europi kako bi zajedno proširili OpenAIRE platformu, prikupili podatke i pružili potporu upravljanju znanstvenim podacima.

2018. godine OpenAIRE-Advance nastavlja misiju OpenAIRE-a u podršci otvorenim podacima u Europi. OpenAIRE-Advance je projekt koji je naglasak stavio najviše na obrazovanje NOAD-ova i učinio ih ključnim dijelom nacionalnih podatkovnih infrastruktura. Osim obrazovanja NOAD-ova, OpenAIRE je i unaprijedio svoju tehničku potporu kroz značajnu optimizaciju performansi i skalabilnosti. Također OpenAIRE kroz novo partnerstvo s Centrom za operativnu analizu i istraživanje COAR (Center for Operational Analysis and Research[4]) proširuje svoju ulogu i pokreće suradnju s ustanovama iz Južne Amerike, SAD-a, Japana, Kanade i Afrike.

Trenutno posljednja iteracija napretka OpenAIRE-a je OpenAIRE-Nexus pokrenuta 2020. godine. U novoj iteraciji je cilj OpenAIRE-a implementirati i ubrzati Otvorenu znanost. Pod otvorenu znanost se misli na značajno proširenje OpenAIRE-a koje obuhvaća niz novih podataka kojima je cilj uspostaviti jedinstvenu komunikaciju Otvorene znanosti kroz sustav znanost. Novi sustav je podijeljen u tri velike cjeline: otkrivanje, nadgledanje i objavljivanje. Svaka od tih cjelina se sastoji od više raznih servisa i platformi koji pružaju različite usluge OpenAIRE-u. Na primjer za otkrivanje podataka se koriste OpenAIRE-ovi moduli Connect, Explore i Provide, za nadziranje OpenAIRE-Monitor, OpenCitations, ScholExplorer, OpenAIRE AAI i slični, dok se za objavljivanje koriste već spomenuti Zenodo, EpiSciences, Amnesia i ostali.

[4]https://coar-global.org/

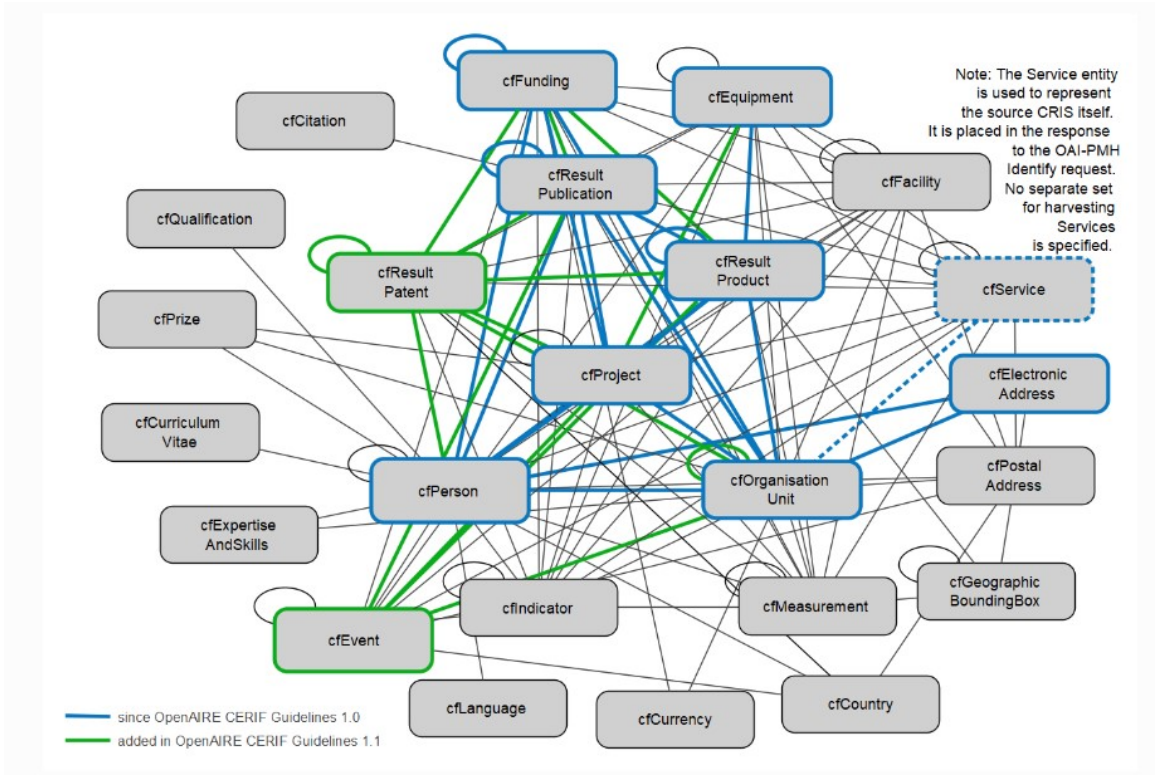
2.2 Model podataka

Kako smo opisali u prethodnom poglavlju, OpenAIRE se neprestano širi i obuhvaća sve veći opseg podataka koje pohranjuje. OpenAIRE također teži povezivanju svih tih podataka na logičkoj razini, čime dolazi do potrebe za standardizacijom modela podataka. Povećanje opsega podataka dovelo je OpenAIRE do razine CRIS-a. Kako bi olakšao normalizaciju podataka, OpenAIRE je preuzeo već postojeći CRIS model podataka koji se koristi u Europi zvan CERIF. CERIF (Common European Research Information Format[5]) ili Normalizirani europski format informacijskog sustava je preporučen za korištenje od strane Europske Unije, a trenutno ga održava euroCRIS[6], neprofitabilna organizacija koja je posvećena interoperabilnosti informacijskih sustava. CERIF osim domene i formata podataka uključuje i mehanizme za stvaranje XML profila za razmjenu podataka. To znači da OpenAIRE profili bazirani na CERIF-u podržavaju izvlačenje i uvoz podataka iz CRIS-eva. CERIF-ovi modeli za CRIS-eve su daleko opsežniji nego je potrebno za OpenAIRE.

Također je važno naglasiti da su CERIF-ovi modeli samo preporuka dobre prakse za CRIS-eve, odnosno pri izgradnji vlastitog CRIS-a moguće je imati razlike u semantici. Uz te razlike, prilikom spajanja vlastitog CRIS-a na OpenAIRE potrebno je pojasniti semantiku i spojiti podatke iz vlastitog CRIS-a s njihovom reprezentacijom u CERIF-ovom modelu. Kako bi to uparivanje bilo lakše, nužno je i potrebno u XML datoteci generiranoj za spajanje OpenAIRE sustava s CRIS-evima, pridržavati se definiranih elemenata i rječnika iz OpenAIRE CERIF profila.

Kako smo već spomenuli, OpenAIRE CERIF profili obuhvaćaju samo dio CERIF modela. Taj dio modela možemo vidjeti na slici 2.1. Na slici su plavom i zelenom bojom označeni moduli i veze među modulima koji se nalaze u OpenAIRE CERIF modelu. Možemo uočiti da su središnji pojmovi u OpenAIRE modelu publikacije, projekti, osobe, organizacijske jedinice, patenti i produkti. Osim tih elemenata tu su još podaci o financiranjima, opremi, događajima, elektroničkim adresama i podaci o servisu koji pruža CRIS. Iz slike možemo vidjeti još mnoštvo drugih pojmova koji trenutno nisu dio OpenAIRE modela, ali se nalaze u većini trenutnih CRIS-eva baziranih na CERIF-ovom modelu. Te pojmove nećemo opisivati, niti posebno spominjati, ali ćemo zato detaljnije opisati pojmove koji se nalaze u OpenAIRE-ovom modelu. Kod opisa pojmova fokusirat ćemo se na elemente koji sadrže, njihove tipove podataka, njihovo značenje te jesu li oni obavezni.

[5] https://eurocris.org/services/main-features-cerif
[6] euroCRIS [1]



Slika 2.1: OpenAIRE CERIF model[7]

2.3 Elementi OpenAIRE CRIS sustava

Već smo spomenuli da se OpenAIRE s vremenom širio i isto se dogodilo i s njegovim modelom podataka. Ovdje ćemo opisati posljednju verziju OpenAIRE CRIS-a i njegove elemente. Tu verziju možemo naći pod nazivom: *OpenAIRE CERIF Guidelines 1.1*[8].

Publication

Prvi pojam OpenAIRE CRIS-a i ujedno najstariji element su *publikacije*. Pod pojmom publikacije podrazumijevamo obrazovne materijale ili rezultate istraživanja. Entitet *publikacija* obično predstavlja pojedinu objavljenu stavku poput knjige ili članka, ali može biti korišten i za izvore koji predstavljaju više publikacija poput serija knjiga ili časopisa. U

[7] https://openaire-guidelines-for-cris-managers.readthedocs.io/en/v1.1.1/cris_elements_openaire.html
[8] OpenAIRE CERIF Guidelines 1.1 [7]

već spomenutoj XML datoteci za izvoz CRIS-a, odnosno uvoz istog u OpenAIRE se entitet *publikacije* prezentira u zasebnom XML elementu *Publication*.

Svaki entitet *publikacija* mora sadržavati dva obavezna elementa: Internal Identifier i Type. Prvi od ta dva elementa Internal Identifier predstavlja jedinstveni identifikator publikacije u CRIS-u, a predstavljen je u XML datoteci atributom id. Drugi element Type predstavlja tip publikacije, a u XML datoteci je predstavljen s istoimenim elementom. Tip elementa Type je text, a vrijednosti mogu biti iz skupa: annotation, bibliography, book, conference object, lecture, letter, periodical, preprint, report, research proposal, review, technical documentation, working paper, thesis i musical notation. Naravno, neke od tih vrijednosti su općenite pa onda postoje njihove konkretnije vrijednosti koje se mogu koristiti poput book part za book ili za periodical imamo vrijednosti journal, review article i slične. Više o tim vrijednostima možemo pronaći na web-stranici pod [7].

Zbog velike složenosti publikacija, one mogu sadržavati mnogobrojne neobavezne elemente. Za većinu tih elemenata je iz naziva jasno što oni predstavljaju pa nećemo posebno objašnjavati, ali ćemo elemente podijeliti po tipovima kojima su predstavljeni u bazi. Tako na primjer imamo jednostavne tipove poput brojeva, teksta i datuma, a njima su predstavljeni elementi poput: Number, Volume, Issue, Edition, StartPage, EndPage, Abstract, PublishedDate te neki specifični identifikatori kao DOI, Handle, ISI-Number, SCP-Number, ISSN i ISBN. Od kompleksnijih elemenata imamo Title, Subtitle, Keyword i Description kao višejezične tipove koji sadrže vrijednost tekst i atribut jezik teksta, a imamo i elemente koji su tipa drugih entiteta. Među takvim entitetima imamo tipove *Person* i *OrgUnit* koje ćemo kasnije opisati. Takvim tipovima predstavljamo elemente poput Author, Editor i Publisher. Od ostalih tipova još možemo pronaći tipove poput *Project*, *Funding*, *Event*, *Patent*, *Product* i *Publication*. Vidimo da su *publikacije* jako složen entitet koji je povezan sa skoro cijelim CRIS-om pa popis svih elemenata nećemo navoditi, ali možemo pronaći na [7].

Još jedan važan element koji entitet *Publication* sadrži je ns4:Access. On predstavlja razinu otvorenosti publikacije pa tako imamo mogućnosti od toga da je publikacije u potpunosti besplatna i otvorena za čitanje do toga da o njoj možemo dobiti pristup samo metapodacima, odnosno osnovnim podacima poput naziva i imena autora. Ovaj element će se ponavljati u više entiteta pa ćemo objasniti značenja njegovih mogućih vrijednosti. Moguće vrijednosti elementa ns4:Access su:

- **open access**. Ona označava da je publikacija otvorena za čitanje bez ikakvih restrikcija za korisnike.

- **embargoed access**. Vrijednost embargoed access nam kaže da je publikacija trenutno u stanju da možemo pristupiti samo metapodacima, ali je tako samo do nekog određenog trenutka, poput dana objave. Nakon toga publikacija mora doći u stanje open access.

- **restricted access**. Publikacija koja je u stanju restricted access je otvorena samo za određenu grupu ljudi. Taj tip pristupa pruža širok spektar mogućnosti za publikaciju. Tako publikacija može biti dostupna samo osobama koje se mogu ulogirati u neki sustav ili se može dobiti pristup publikaciji ako se zatraži od autora ili neke nadležne organizacije.

- **metadata only access**. Zadnja vrijednost elementa ns4:Access je metadata only access. Takva publikacija korisnicima nije dostupna u slobodnom čitanju, ali može postojati u fizičkom obliku ili možda može biti dostupna u nekim uvjetima. Neki od tih uvjeta su na primjer plaćanje ili promjena lokacije. Pod promjenom lokacije možemo podrazumijevati da dane web-adrese nemaju otvoren pristup, ali postoje neke druge adrese koje imaju otvoren ili ograničen pristup.

Očito je iz opisa tih vrijednosti da je težnja OpenAIRE-a da što više publikacije bude u otvorenom pristupu.

Product

Drugi entitet u OpenAIRE-u je *Product*, koji označava bilo koji rezultat znanstvenog istraživanja koji nije publikacija ili patent. To znači da entitet *produkt* uključuje objekte poput skupa podataka, programa, slika, videa, kartografskih materijala, audio snimaka i sličnih objekata koji su nastali kao rezultati istraživanja.

Kao i prethodni entitet, obavezni elementi su Internal Identifier i Type. Razlika je naravno u elementu Type koji može poprimiti drugačije vrijednosti nego kod entiteta *Publication*. Vrijednosti koje element Type može poprimiti su: interactive resource, website, dataset, image, video, software, workflow, cartographic material, map, sound, musical composition i other.

Neki od neobaveznih elemenata jednostavnijeg tipa entiteta *Product* su ARK, DOI, Handle, URL, URN, Language i ns4:Access, dok su Name, VersionInfo, Description i Keyword višejezični elementi. Od složenijih elemenata imamo elemente Creators i Publishers koji su lista elemenata tipa *Person* ili *OrgUnit*. Neki elementi poput PartOf i References mogu biti tipa *Publication*, *Patent* ili čak *Product*. Preostali su elementi još tipova poput *Event*, *Equipment*, *Project* ili *Funding*. Kao što možemo zaključiti entitet *Product* je još jedan od kompleksnijih i bolje povezanih entiteta s ostatkom CRIS-a čime upravo uz entitete *Publication* i *Patent*, koji ćemo uskoro navesti, čini središnji dio OpenAIRE-ovog CRIS-a.

Patent

Treći središnji entitet OpenAIRE-ovog CRIS-a je gore spomenuti *Patent*. Kao što ime kaže, taj entitet predstavlja upravo patente, odnosno on opisuje skup ekskluzivnih prava koje imaju investitor ili osoba na određeno vrijeme. Ta prava im osigurava njihova država u zamjenu za potpunu objavu podataka javnosti nakon što prođe unaprijed određeno vrijeme.

Kao i prethodna dva entiteta, *Patent* također ima dva obavezna elementa: Internal Identifier i Type. Jedna značajna razlika u ovom entitetu je ta što element Type može poprimiti samo jednu vrijednost: patent. Spomenuli smo već da je *Patent* jedan od entiteta koji su dobro povezani s ostatkom sustava pa tako možemo pronaći gotovo sve druge entitete kao tipove podataka.

Tako imamo opet jednostavne tipove poput datuma, brojeva i stringova za elemente RegistrationDate, ApprovalDate, CountryCode, PatentNumber i Subject te višejezične tipove za elemente Title, VersionInfo, Abstract i Keyword. Kompleksnije tipove možemo pronaći u svim ostalim elementima pa tako je element Issuer tipa *OrgUnit*, Inventor tipa *Person*, a Holder tipa *Person* ili *OrgUnit*. Od ostalih elemenata OriginatesFrom može biti tipa *Project* ili *Funding*, Predecessor tipa *Patent*, a References tipa *Publication*, *Patent* ili *Product*.

Person

Iako idući entitet ne ubrajamo u jedan od tri središnja entiteta, on se nalazi u skoro svim ostalim entitetima pa možemo reći da je barem jednako važan kao i prethodni entiteti. Govorimo o entitetu *Person*, odnosno entitetu koji opisuje pojedinu osobu. Pod pojmom osoba ovdje mislimo samo na prave osobe, odnosno na znanstvenike, studente koji rade istraživanja, autore, razne suradnike u stvaranju publikacija, podataka ili programa, osobe koje rade na projektima, razne ovlaštene osobe na ustanovama ili projektima te ostale osobe koje mogu imati razne uloge na ustanovama ili opremi. Kao i u stvarnom svijetu, u CRIS-evima je očekivano da jedna osoba ima više uloga.

Elementi entiteta *Person* se sastoje od jednog obavezno elementa, odnosno atributa Internal Identifier i mnoštva neobaveznih elemenata. Od neobaveznih elemenata imamo kompleksniji element PersonName koji se sastoji od tri elementa tipa String: FamilyNames, FirstNames i OtherNames. Također imamo neobavezni element Gender koji može poprimiti vrijednosti ”m” ili ”f” te imamo velik broj elemenata koji označavaju razne identifikatore poput ORCID, ResearcherID, ScopusAuthorID, ISNI, DAI i njihove alternativne vrijednosti u istoimenim elementima koji imaju dodatni prefiks Alternative kao na primjer AlternativeORCID. Preostali neobavezni elementi su ElectronicAddress i jako važan element Affiliation koji je tipa *OrgUnit* i spaja pojedinu osobu s ustanovama. Element Affiliation može sadržavati ustanove na kojima je osoba zaposlena, stekla zvanje ili neku stručnost.

OrgUnit

Sljedeći entitet *OrgUnit* sadrži podatke o organizacijskim jedinicama. Sam pojam organizacijske jedinice je ponešto kompleksniji i manje intuitivan od ostalih pojmova s kojima se susrećemo. Ta neintuitivnost slijedi iz popisa svih mogućnosti koje se smatraju organizacijskom jedinicom. Tako je poprilično intuitivno da ćemo pod organizacijskom ustanovom smatrati neke vrste ustanova poput sveučilišta, instituta, fakulteta, škola i sličnih znanstvenih ili obrazovnih ustanova.

Neintuitivni dio pojma organizacijske ustanove bi bile razne udruge ljudi koje imaju neku zajedničku svrhu u znanstvenom istraživanju poput nekih pokreta ili odbora. Sve te moguće elemente pojma organizacijske jedinice može podijeliti u neke smislene grupe. Te grupe bi bile:

- organizacije koje izvode istraživanja (instituti, fakulteti)
- organizacije koje financiraju istraživanja (sveučilište, ministarstvo, kompanije)
- znanstvene udruge (na primjer: Hrvatsko matematičko društvo)
- ustanove koje pružaju neku vrstu usluga u znanstvenom svijetu(nakladnici)
- organizacije koje dovode zakone ili pomažu u provođenju (ministarstva, uredi za patente)
- grupa svih ostalih(razni drugi odbori i udruge)

Iako je pojam organizacijske jedinice obuhvaća širok spektar značenja, sam entitet *OrgUnit* ima relativno malo atributa. Tako imamo kao i uvijek obavezan atribut Internal Identifier. Svi ostali elementi su neobavezni i većinom mogu biti mnogobrojni. Prvi u nizu je Type koji predstavlja tip ustanove, zatim su tu još višejezični element Name, element Acronym koji može postojati najviše jedan, element ElectronicAddress, lista ostalih mogućih identifikatora Identifier i PartOf element koji predstavlja poveznicu na organizacijsku jedinicu kojom pripada organizacijska jedinica koju opisujemo.

Project

Entitet *Project*, kao što ime kaže opisuje pojam projekta. Projekt definiramo kao privremeni rad kojemu je svrha stvoriti neki novi jedinstveni produkt, uslugu ili dobiti neki rezultat. Entitet *Project* je kao i sam OpenAIRE usmjeren prema znanstvenom svijetu pa će upravo takvi projekti biti značajni. Iz tog razloga važna su četiri tipa projekata:

- istraživački projekti kojima je rezultat proširivanje baze ljudskog znanja

- projekti tehnološkog razvoja kojima je rezultat neka tehnologija ili program
- inovativni projekti koji unapređuju već postojeće produkte i procese
- projekti koji stvaraju ili poboljšavaju infrastrukturu koja služi za istraživanja, razvoj tehnologije ili inovacija.

Sam entitet *Project* osim samih projekata može opisivati i njihove podentitete poput stadija projekta, individualnih zadataka i sličnih dijelova, ovisno o granulaciji podataka koje posjedujemo.

Entitet *Project* kao i posljednjih nekoliko entiteta ima samo jedan obavezan atribut Internal Identifier. Ostale elemente možemo podijeliti u dvije skupine: elementi koji opisuju projekt i elementi predstavljeni drugim entitetima iz OpenAIRE-a koji popisuju osobe, ustanove i opremu te njihove uloge na projektu. Iz prve skupine imamo elemente Type, Acronym, Title, Identifier, StartDate, EndDate, Subject, Keyword, Abstract, Status i OAMandate. Većina ovih elemenata nam je poznata i ima identično značenje kao u nekim prethodim entitetima. Jedini element s kojim se nismo prije susreli je Status koji, kako možemo i sami zaključiti, govori u kojem stadiju se projekt nalazi.

Druga skupina elemenata se sastoji od četiri elementa. od kojih su tri kompleksnija sa svojim podelementima:

- **Uses** je jedini jednostavniji element koji je tipa *Equipment* i predstavalja nam popis opreme koja se koristila na projektu
- **Consortium** predstavlja konzorcij projekta, odnosno ustanove i osobe koje su ugovorom vezane za rad na projektu. On se sastoji od pet mogućih elemenata koji su svi tipa *OrgUnit* i/ili *Person*. Ti elementi su: Coordinator, Partner, Contractor, InKindContributor i Member. Iz prijevoda naziva je jasno na koju ulogu se koji element odnosi pa ih nećemo posebno opisivati.
- **Team**, odnosno popis osoba povezanih s ulogama na projektu. Team se sastoji od tri elementa: PrincipalInvestigator odnosno osoba odgovorna za cijeli projekt, Contact i Member. Elementi Contact i Member predstavljaju odgovornu kontakt osobu za projekt, odnosno ostale članove tima koji rade na projektu. Intuitivno je da su svi elementi tipa *Person*.
- **Funded** pruža informacije o financiranju projekta. On se sastoji od elemenata By i As. Element By je tipa *OrgUnit* ili *Person* i predstavlja popis ustanova i osoba koje pružaju financijska sredstva za projekt. Drugi element As govori o načinu financiranja projekta. Način financiranja nam kaže kako je projekt dobio sredstva za financiranje, na primjer kroz nagrade, ugovore ili na neke druge načine. Element As je predstavljen entitetom *Funding* koji ćemo opisati u idućem potpoglavlju.

Funding

Kako smo spomenuli u opisu zadnjeg elementa prethodnog entiteta, entitet *Funding* govori o načinu financiranja. Način financiranja je u ovom slučaju vrsta izvora iz kojeg su dodijeljena financijska sredstva. Što smatramo izvorima financiranja ćemo najbolje shvatiti iz popisa mogućih vrijednosti elementa Type. Spomenute moguće vrijednosti elementa Type su:

- Funding Programme
- Call
- Tender
- Gift
- InternalFunding
- Contract
- Award
- Grant

Značenje tih vrijednosti je jasno iz samog naziva pa nećemo dodatno objašnjavati. Vidimo da je element Type izuzetno važan u entitetu *Funding* pa je uz element Internal Identifier jedan od dva obavezna elementa. Ostali elementi u ovom entitetu su neobavezni i većina njih su jednostavne ili višejezične string vrijednosti. Tako imamo elemente Name, Acronym, Identifier, Description, Subject, Keyword, OAMandate i Amount. Sve ove elemente smo već prije vidjeli i njihovo značenje je isto ili slično u kontekstu entiteta *Funding*. Novi element je Amount. Jasno nam je što on predstavlja, ali isto tako vrijedi spomenuti da bi Amount trebao biti tipa string i oblika brojčana vrijednost koju slijedi naziv ili simbol valute. Entitet *Funding* ima još dva elementa složenijeg tipa: Funder i PartOf.

Element Funder je tipa *OrgUnit* ili *Person* i predstavlja popis ustanova ili osoba koje financiraju ovo financiranje. Drugi element je *PartOf* i u njemu pohranjujemo poveznicu na drugi entitet *Funding* koji je na neki način nadređen našem entitetu. Primjer toga bi bili programi i potprogrami za financiranje.

Equipment

Entitet *Equipment* smo već spominjali kao tip elementa u entitetima *Product* i *Project*. Ovaj entitet nije toliko povezan s drugim entitetima kao većina prijašnjih entiteta pa iz tog razloga ne sadrži puno elemenata i skoro svi su tipa String.

Jedini obavezni element je Internal Identifier. Većinu ostalih neobaveznih elemenata smo susreli već prije u drugim entitetima, poput Name, Acronym, Description, Identifier i Type. Svi ovi elementi poprimaju slične vrijednosti kao prije, osim elementa Type koji može poprimati više različitih vrijednosti. Element Type je u entitetima *OrgUnit* i *Project* također mogao, ali u rijetkim slučajevima je poprimao različite vrijednosti, dok je u slučaju entiteta *Equipment* to izrazito čest slučaj. Posljedni element entiteta *Equipment* je Owner. Element Owner može biti tipa *OrgUnit* ili *Person* i označava vlasnika opreme koju opisujemo.

Event

Pretposljednji entitet u OpenAIRE-u je *Event*. *Event* bilježi podatke o neki događanjima koja su važna i često vezana za neke publikacije. Iako je jedan od entiteta koji su posljednji dodani u skup entiteta koje prikuplja OpenAIRE, entitet *Even* ima velik broj elemenata pa čak i onih s kojima se nismo prije susreli.

Jedini obavezni entitet je Internal Identifier. Elementi s kojima smo se prije susreli su Name, Acronym, Description, Type, Subject, Keyword, StartDate i EndDate i njih nećemo detaljnije objašnjavati. Dva nova jednostavna elementa su Place i Country. Element Place sadrži naziv mjesta ili grada u kojem se događaj održao, dok element Country predstavlja podatke, poput naziva, skraćenice i koda države u kojoj se događaj održao. Preostala tri elementa su svi tipa *OrgUnit* i/ili *Project* i iz njihovog naziva nam je jasno što oni predstavljaju. Ti elementi su Organizer, Sponsor i Partner.

Service

Posljednji entitet značajan OpenAIRE-u je *Service*. Možemo uočiti da taj entitet nismo nigdje spominjali u prethodnim entitetima i da nije baš povezan s ostatkom sustava. Razlog za to je što taj entitet predstavlja podatke o CRIS-u koji pruža podatke OpenAIRE-u i koji je usklađen s OpenAIRE preporukama za CRIS menadžere [7] koje ovdje opisujemo.

Entitet *Service* kao i većina entiteta sadrži jedan obavezan atribut Internal Identifier. Neobavezni elementi s kojima smo se ranije susreli su Name, Acronym, Description, Identifier i Owner koji je jedini element koji povezuje *Service* s ostatkom CRIS-a. Od preostalih elemenata imamo Compatibility koji trenutno može poprimiti vrijednosti ”OpenAIRE Guidelines 1.1 compatible CRIS” ili ”OpenAIRE Guidelines 1.0 compatible CRIS” ovisno s kojom verzijom OpenAIRE-a je CRIS kompatibilan te tri URL elementna WebsiteURL, OAIPMHBaseURL i SubjectHeadingsURL. Prvi element WebsiteURL predstavlja poveznicu na početnu stranicu CRIS-a. Drugi element OAIPMHBaseURL pruža poveznicu na osnovni URL za OAI-PMH pristupnu točku CRIS-a o čemu ćemo više pričati u idućem

poglavlju. Posljedni od tri URL elementa je SubjectHeadingsURL koji sadrži poveznicu na popis klasifikacija koje su korištene u CRIS-u.

2.4 Tehnička implementacija

OpenAIRE za preuzimanje podataka iz CRIS-a koristi OAI-PMH 2.0 protokol za prikupljanje metapodataka. OAI-PMH protokol ćemo opisati u idućem poglavlju, ali neke posebnosti koje zahtijeva OpenAIRE ćemo ovdje navesti.

Prva takva posebnost proizlazi iz verzije preporuka s kojima je CRIS usklađen. Ukoliko je CRIS usklađen s ”OpenAIRE Guidelines 1.1” tada bi se u XML datoteci trebao koristiti prefix ”oai_cerif_openaire”. Primjer takvog odgovora možemo pronaći na poveznici [2]. Gore smo opisivali sve moguće entitete koje OpenAIRE prihvaća u svoj CRIS i to je bilo važno upravo zato što svaki od tih entiteta OpenAIRE zahtijeva po jedan zahtjev za svaki skup podataka. Iz tog razloga imamo devet mogućih zahtjeva oblika: **”openaire_cris_ImeEntiteta”**. Također mora postojati popis svih zahtjeva za sve skupove koje pruža CRIS-a i on se nalazi na **”openaire_oaipmh_ListSets”**. Jedini entitet koji ima poseban zahtjev je *Service*. Njegov zahtjev je drugačiji zato što ne predstavlja skup podataka nego samo jedan podatak i glasi: **”openaire_oaipmh_Identify”**. Zadnji od posebnih zahtjeva koje OpenAIRE ima na OAI-PMH protokol je da u XML datoteci svi Internal Identifier elementi budu predstavljeni kao XML atribut ”id”. Nakon što smo nabrojali posebnosti tehničke implementacije, sada možemo detaljnije opisati OAI-PMH protokol.

Poglavlje 3

Open Archives Initiative Protocol for Metadata Harvesting

3.1 Open Archives Initiative

Open Achives Initiative (OAI) je neformalna organizacija koju su krajem 20.stoljeća osnovali Herbert Van de Sompel, Carl Lagoze, Michael Nelson i Simeon Warner. OAI se bavi razvojem standarda za interoperabilnost sustava koji bi olakšao učinkovito širenje sadržaja. Izvorni cilj OAI-a je bio razvoj i implementacija tehničkih standarda za interoperabilnost fakultetskih arhiva kako bi mogli međusobno dijeliti informacije o prikupljenim katalozima, odnosno metapodacima. Svojim djelovanjem, odnosno razvojem programskih okvira i standarda za interoperabilnost, OAI je omogućio učinkovitu razmjenu podataka poput znanstvenih članaka, čime je direktno pomogao pokretu Otvorene znanosti. Iako je prvotna svrha OAI-a bila omogućiti razmjenu metapodataka u fakultetskim arhivima, tehnologija i standardi koje su razvili su primjenjivi u puno širem spektru podataka od samo akademskih publikacija. OAI je u svojem djelovanju razvio tri projekta:

- ResourceSync
- Protocol for Metadata Harvesting (OAI-PMH)
- Object Reuse and Exchange (OAI-ORE)

3.2 OAI-PMH - uvod i definicije

The Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) je mehanizam za interoperabilnost podatkovnih skladišta. On pruža interoperabilnost programskih ok-

vira koja je bazirana na prikupljanju metapodataka i neovisna od aplikacije. U OAI-PMH protokolu postoje dvije klase sudionika:

- *Data Providers* ili pružatelji podataka. Oni su administratori sustava koji izlaže svoje metapodatke.
- *Service Providers* ili pružatelji usluga. Pružatelji usluga koriste metapodatke koje dobiju od pružatelja podataka kao osnovu za izgradnju usluga s novim vrijednostima.

Jedan od pojmova s kojim ćemo se često susretati je *harvester*, odnosno *pobirač*. *Pobirač* je klijentska aplikacija koja obrađuje OAI-PMH zahtjeve. On služi *pružatelju usluga* kao sredstvo za prikupljanje metapodataka iz sustava.

Repository

Sljedeći u nizu pojmova je *repository* (*repozitorij*, odnosno server kojemu se može pristupiti putem mreže i koji može obraditi sve OAI-PMH zahtjeve koje ćemo kasnije opisati. *Repozitorij* je dio protokola koji pruža metapodatke *pobiraču* i njime upravljaju pružatelji podataka. Kako bi dopustili različite postavke *repozitorija*, OAI-PMH razlikuje tri disjunktna pojma koja su povezana s metapodacima koje OAI-PMH čini dostupnima:

- *resource*, odnosno objekt o kojem metapodaci govore.
- *item* ili *predmet* je sastavnica repozitorija iz koje metapodaci o nekom objektu mogu biti rasprostranjeni. To je konceptualno neki kontenjer koji pohranjuje ili dinamički generira metapodatke o svakom pojedinom objektu u više različitih formata. Svi formati koje *predmet* sadrži moraju moći biti prikupljeni kao *zapis* kroz OAI-PMH. Važno je spomenuti da svaki *predmet* mora sadržavati identifikator koji je jedinstven u *repozitoriju* u kojem se nalazi.
- *record* ili *zapis* je metapodatak u specifičnom formatu. On služi kao povratna vrijednost u XML obliku na zahtjev protokola da proširi specifični format metapodatka iz odabranog objekta. Više o *zapisima* ćemo govoriti kasnije.

Unique Identifier

Unique Identifier, odnosno *jedinstveni identifikator* je još jedan neizostavan pojam u OAI-PMH protokolu. On nedvosmisleno određuje *predmet* unutar *repozitorija*. *Jedinstveni identifikator* se koristi u OAI-PMH zahtjevima za dohvaćanje metapodataka o *predmetu*. U slučaju da *predmet* sadrži metapodatke u više formata, *jedinstveni identifikator* je svima zajednički, što znači da s njime možemo dohvatiti sve *zapise* koji su dostupni za traženi *predmet*. Format u kojem se nalazi *jedinstveni identifikator* mora odgovarati URI (Uniform

Resource Identifier[1]) sintaksi. *Jedinstveni identifikatori* imaju dvije uloge u OAI-PMH protokolu:

- *Response* (*Odgovor*): Identifikatore vraćaju zahtjevi **ListIdentifiers** i **ListRecords** koje ćemo kasnije opisati.
- *Request* (*Zahtjev*): Identifikator se u kombinaciji s vrijednošću elementa **metadataPrefix** koristi u zahtjevu **GetRecord** za dohvaćanje određenog *zapisa* u specifičnom formatu.

Važno je naglasiti da *jedinstveni identifikator* koji ovdje spominjemo se odnosi na objekt u protokolu, a ne na objekt koji se nalazi u bazi. Protokol nema nikakvo znanje o vrsti identifikatora koje sadrže objekti u bazi. Dobra praksa je da *repozitoriji* koriste neki element iz metapodataka kako bi povezali *zapise* i identifikatore iz baze poput URL-a ili DOI-a.

Record

Idući pojam smo već nekoliko puta spomenuli i definirali, a to je *record* ili *zapis*. Definirali smo *zapis* kao podatak u specifičnom formatu koji služi kao povratna vrijednost na zahtjeve OAI-PMH protokola i koji se može dohvatiti kombiniranjem *jedinstvene vrijednosti* i *metadataPrefix* vrijednosti. Iako smo rekli da se *zapis* može dohvatiti pomoću te dvije vrijednosti, to ne znači da ga one jednoznačno određuju. Za nedvosmislen pronalazak nekog *zapisa* nam je potreban i podatak *datestamp*, odnosno *vremenska oznaka*. Naime ne možemo očekivati da se podaci neće mijenjati pa nam je bitan podataka za identifikaciju točnog *zapisa* upravo vrijeme dohvata podatka, odnosno njegova *vremenska oznaka*. Spomenuli smo da *zapis* služi kao povratna vrijednost na zahtjeve i da se tada nalazi u XML obliku. U XML-u se *zapis* sastoji od tri elementa:

- **header** ili *zaglavlje* - sadrži *jedinstveni identifikator* objekta i elemente potrebne za birani dohvat podatka. *Zaglavlje* se sastoji od četiri elementa:
  - *jedinstvenog identifikatora*
  - *oznake vremena* - označava datum stvaranja, izmjene ili brisanja *zapisa* za svrhe dohvata.
  - *setSpec* - popis *skupova* u kojima se nalazi *zapis* koji dohvaćamo. Može vratiti nula ili više elemenata.
  - *status* - element koji nije obavezan, a može sadržavati samo vrijednost *"deleted"* koja označava da je zapis u traženom formatu nedostupan ili obrisan.

[1] https://www.ietf.org/rfc/rfc2396.txt?number=2396

- **metadata** - označava u kojem formatu je objekt prikazan, zato što OAI-PMH protokol ima podršku za prikaz više različith formata metapodataka.
- **about** - neobavezan i ponovljiv skup podataka koji sadrži informacije o metapodacima za traženi *zapis*. Ovaj skup podataka se često koristi kako bi se prikazali podaci o uvjetima korištenja ili naglasio izvor podataka, odnosno jesu li podaci izvorno iz ovog *repozitorija* ili su uvezeni iz nekog drugog i ako jesu iz kojeg *repozitorija*

```
<header>
  <identifier>oai:arXiv:cs/0112017</identifier>
  <datestamp>2002-02-28</datestamp>
  <setSpec>cs</setSpec>
  <setSpec>math</setSpec>
</header>
<metadata>
 <oai_dc:dc
     xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/"
     xmlns:dc="http://purl.org/dc/elements/1.1/"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/oai_dc/
     http://www.openarchives.org/OAI/2.0/oai_dc.xsd">
   <dc:title>Using Structural Metadata to Localize Experience of Digital
             Content</dc:title>
   <dc:creator>Dushay, Naomi</dc:creator>
   <dc:subject>Digital Libraries</dc:subject>
   <dc:description>With the increasing technical sophistication of both
    information consumers and providers, there is increasing demand for
    more meaningful experiences of digital information. We present a
    framework that separates digital object experience, or rendering,
    from digital object storage and manipulation, so the
    rendering can be tailored to particular communities of users.
   </dc:description>
   <dc:description>Comment: 23 pages including 2 appendices,
                  8 figures</dc:description>
   <dc:date>2001-12-14</dc:date>
   <dc:type>e-print</dc:type>
   <dc:identifier>http://arXiv.org/abs/cs/0112017</dc:identifier>
 </oai_dc:dc>
</metadata>
<about>
  <provenance
      xmlns="http://www.openarchives.org/OAI/2.0/provenance"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/provenance
      http://www.openarchives.org/OAI/2.0/provenance.xsd">
    <originDescription harvestDate="2002-02-02T14:10:02Z" altered="true">
      <baseURL>http://the.oa.org</baseURL>
      <identifier>oai:r2:klik001</identifier>
      <datestamp>2002-01-01</datestamp>
      <metadataNamespace>http://www.openarchives.org/OAI/2.0/oai_dc/</metadataNamespace>
    </originDescription>
  </provenance>
</about>
```

Slika 3.1: XML odgovor na zahtjev za dohvat pojedinog *zapisa*[2]

Primjer jednog XML-a s gore navedenim elementima možemo vidjeti na slici 3.1.

Na primjeru nije prikazano, ali rekli smo da *zapis* u XML *zaglavlju* može sadržavati *status* i da njegova vrijednost može biti samo *"deleted"*. Naime ako neki *zapis* više nije dostupan, za njega smatramo da je u statusu *"deleted"*. Ukoliko *zapis* ima taj status, tada XML ne smije sadržavati elemente *metadata* i *about*. OAI-PMH protokol definira tri razine podrške za obrisane *zapise* i svaki sustav bi trebao izjasniti koji od te tri razine podržava. Taj podatak se treba nalaziti u *deletedRecord* elementu odgovora na **Identify** zahtjev protokola i može poprimiti jednu od vrijednosti:

- *no* - ta vrijednost označava da sustav ne prati informacije o bisanju *zapisa*. Takav sustav nikad ne smije vratiti status *"deleted"*.

- *persistent* - u suprotnosti s prethodnom vrijednosti, ovaj sustav čuva informacije o brisanju *zapisa*. Takav sustav mora za svaki obrisani *zapis* vratiti status *"deleted"*. To znači da zapis ne smije samo nestati iz odgovora na zahtjev.

- *transient* - ova vrijednost se najrjeđe pojavljuje u sustavima, a ona kaže da sustav ne garantira da se lista brisanja održava, niti da je konzistentna. Sustavi koji se izjasne za ovu vrijednosti mogu, ali ne moraju vratiti vrijednost *"deleted"* za status *zapisa*.

Set

Sljedeći pojam *set* ili *skup* nam označava grupiranje objekata u svrhu biranog dohvata. To grupiranje je neobavezno i u potpunosti proizvoljno u okvirima OAI-PMH protokola, ali nama izuzetno značajno za razmjenu podataka s OpenAIRE-om. U OAI-PMH protokolu objekti mogu biti organizirani u *skupove* po hijerarhiji ili mogu biti obične liste. Ukoliko sustavi definiraju *skupove*, oni moraju uključiti podatke o pripadnosti u *zaglavlju* objekta koji se vraća kao odgovor na zahtjeve **ListIdentifiers**, **ListRecords** i **GetRecord**. Svaki element u skupu mora sadržavati:

- *setSpec* - lista čvorova u *skupu* koji sadrži hijerarhiju od korijena do čvora koji sadrži traženi *zapis*. Svaki element liste je String koji je odvojen stupcem [:]. U slučaju da *skup* ne sadrži hijerarhiju nego je obična lista, tada *setSpec* ne sadrži stupce [:].

- *setName* - sadrži ime *skupa*

- *setDescription* - neobavezan, ali ponovljiv skup podataka koji može sadržavati podatke o *skupu*.

[2]https://www.openarchives.org/OAI/openarchivesprotocol.html

Važno je naglasiti da objekti u sustavu ne moraju pripadati niti jednom *skupu*, ali mogu biti sadržani i u više *skupova*. U našem slučaju za potrebe OpenAIRE-a će ipak biti slučaj da će svaki podatak koji prikazujemo biti član barem jednog *skupa*. Kako *skupove* nisu unaprijed određeni od strane protokola, OAI-PMH protokol ima poseban zahtjev **ListSets** koji mora vratiti listu svih *skupova*. Ta lista mora sadržavati elemente *setSpec* i *setName*, a može sadržavati i *setDescription*. Postojanjem *skupova*, zahtjevi **ListRecords** i **ListIdentifiers** mogu primiti dodatan argument *set* čija vrijednost je iz liste svih čvorova koji se nalaze u *setSpec*. Dodavanjem tog argumenta, zahtjev kroz protokol traži dohvat samo objekata koji pripadaju traženom *skupu*. Takav slučaj će biti česta pojava pri razmjeni podataka s OpenAIRE-om.

3.3 Značajke OAI-PMH protokola

HTTP zahtjevi

OAI-PMH je obično implementiran koristeći standardne web servise zbog čega su njegovi zahtjevi izraženi u obliku HTTP zahtjeva. OAI-PMH zahtjevi od HTTP-a moraju koristiti ili metodu **GET** ili metodu **POST**. Sustavi koji koriste OAI-PMH protokol moraju podržavati obje metode. Također mora postojati jedan zajednički osnovni dio URL-a za sve zahtjeve. Taj URL određuje internetskog poslužitelja i port, a može sadržavati i putanju do HTTP servera na kojemu je smješten sustav. Sustavi svoj bazni URL prikazuju kao vrijednost elementa *baseURL* u odgovoru na *Identify* zahtjev.

Osim osnovnog dijela URL, svaki zahtjev sadrži i listu argumenata od kojih je svaki u obliku para *"key=value"*, odnosno *"ključ=vrijednost"*. Argumenti se kao i u svakom HTTP zahtjevu mogu pojaviti u bilo kojem poretku i moraju biti odvojeni znakom '&'. OAI-PMH ima dodatan uvjet na svoje zahtjeve, a to je da mora u svakom zahtjevu postojati barem jedan par *"ključ=vrijednost"* koji određuje o kojem je zahtjevu riječ. Štoviše u tom dodatnom zahtjevu je uvijek na mjestu ključne riječi *ključ* riječ *"verb"*, dok se na mjestu *vrijednost* nalazi jedan od definiranih OAI-PMH zahtjeva. **GET** i **POST** metode se također razlikuju i u načinu slanja argumenata. U **GET** metodi se ključne riječi lijepe na osnovni URL i od njega su odvojene znakom '?'. Isti ti argumenti se u **POST** metodi nalaze u tijelu poruke. Tada se u zaglavlju mora nalaziti element *Content-Type* i njegova vrijednost mora biti oblika *"application/x-www-form-urlencoded"*. Odgovori na zahtjeve OAI-PMH protokola moraju biti formatirani kao HTTP odgovori s odgovarajućim poljima u HTTP zaglavljima. Element *Content-type* vraćen za sve OAI-PMH zahtjeve mora imati vrijdenost "text/xml".

XML odgovori

Svi odgovori na OAI-PMH zahtjeve moraju biti dobro formirani XML dokumenti. Kodiranje XML-a također mora koristiti UTF-8 reprezentaciju Unicode-a što je standardna praksa u današnje vrijeme. Osim tih uvjeta, u svim odgovorima na OAI-PMH zahtjeve, podaci u XML dokumentima moraju poštovati XML shemu koja se može naći na web-stranici[3]. Iz sheme možemo vidjeti da odgovori na OAI-PMH zahtjeve imaju sljedeće zajedničke oznake i elemente:

1. Prva oznaka je XML deklaracija gdje je atribut *version* uvijek ”1.0” i *encoding* je uvijek ”UTF-8”, odnosno to izgleda ovako: *<?xml version=”1.0” encoding=”UTF-8” ?>*.

2. Sljedeća oznaka je priložena uz korjenski element s imenom **OAI-PMH**. Taj element mora sadržavati tri atributa koji definiraju XML nazivlje koje se koristi u ostatku odgovora i lokaciju na kojoj se nalazi shema za validaciju. Ti atributi su:

   - *xmlns* - njegova vrijednost mora sadržavati URI (Uniform Resource Identifier) nazivlja OAI-PMH protokola.
   - *xmlns:xsi* - njegova vrijednost mora sadržavati URI za XML shemu koja se koristi.
   - *xsi:schemaLocation* - vrijednost ovog atributa je uređeni par u kojemu prvi dio čini URI nazivlja OAI-PMH protokola, isto kao *xmlns*, a drugi dio je URL na kojem se nalazi XML shema za validaciju odgovora.

3. Za sve odgovore, prva dva elementa korijena su *responseDate* i *request*. Prvi element *responseDate* sadrži vrijeme i datum slanja odgovora i mora biti izražen u UTC (Coordinated Universal Time) vremenu. Element *request*, kako možemo zaključiti iz imena, nam govori koji zahtjev iz protokola je generirao ovaj odgovor.

4. Treći element korijena je uvijek jedan od dva moguća elementa:

   - *error* element koji se mora pojaviti u slučaju pogreške.
   - element s istim imenom kao vrijednost argumenta *verb* u OAI-PMH zahtjevu koji je generirao ovaj odgovor.

Spomenuli smo već da OAI-PMH ima podršku za vraćanje zapisa u više različitih formata. Listu svih formata koji su dostupni za ovaj sustav se može dohvatiti zahtjevom **ListMetadataFormats**. Svaki format metapodataka mora imati definirano sljedeće:

[3] http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd

- *metadataPrefix* - string koji navodi format metapodataka u OAI-PMH zahtjevu. *metadataPrefix* se kao argument koristi u zahtjevima poput **ListRecords**, **ListIdentifiers** i **GetRecord** kako bi se naveo format u kojem se očekuju podaci i koristi se u zaglavlju povratne vrijednosti *zapisa* u kojem kaže u kojem se formatu taj *zapis* nalazi.

- *metadata schema URL* - URL na XML shemu koja se može koristiti za validaciju metapodataka koji su prikazani u toj shemi.

- *URI XML nazivlja* - jedinstveni globalni identifikator formata metapodataka.

Zadnji element koji se mora moći pojaviti u XML-u je *error*. Naravno, element *error* se mora pojaviti u slučaju da se u OAI-PMH protokolu dogodi greška ili iznimka. Također, pošto se radi o HTTP servisu preko kojeg se odvija razmjena zahtjeva, greške koje su specifične za OAI-PMH protokol se moraju razlikovati od standardnih HTTP status kodova. Za razliku od normalnih grešaka koje se pojavljuju u HTTP protokolima gdje se pojavljuje samo poruka prve pogreške, sustav bi u slučaju pogrešaka prouzrokovanih u OAI-PMH protokolu trebao ispisati sve pogreške i iznimke koje je uhvatio. Svaki element *error* treba prijaviti po jednu pogrešku i mora sadržavati atribut *code* te poprimiti vrijednost tipa string koja opisuje pogrešku. Popis svih mogućih vrijednosti *kodova* i pripadnih opisa tih vrijednosti možemo vidjeti u tablici 3.3, a primjer jednog odgovora koji prijavljuje grešku na slici 3.2.

```
<?xml version="1.0" encoding="UTF-8"?>
<OAI-PMH xmlns="http://www.openarchives.org/OAI/2.0/"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/
         http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd">
  <responseDate>2002-05-01T09:18:29Z</responseDate>
  <request verb="ListSets">http://arXiv.org/oai2</request>
  <error code="noSetHierarchy">This repository does not
    support sets</error>
</OAI-PMH>
```

Slika 3.2: Primjer XML odgovora na pogrešku[4]

[4]https://www.openarchives.org/OAI/openarchivesprotocol.html

Error Codes	*Description*
`badArgument`	The request includes illegal arguments, is missing required arguments, includes a repeated argument, or values for arguments have an illegal syntax.
`badResumptionToken`	The value of the `resumptionToken` argument is invalid or expired.
`badVerb`	Value of the `verb` argument is not a legal OAI-PMH verb, the verb argument is missing, or the `verb` argument is repeated.
`cannotDisseminateFormat`	The metadata format identified by the value given for the `metadataPrefix` argument is not supported by the item or by the repository.
`idDoesNotExist`	The value of the `identifier` argument is unknown or illegal in this repository.
`noRecordsMatch`	The combination of the values of the `from`, `until`, `set` and `metadataPrefix` arguments results in an empty list.
`noMetadataFormats`	There are no metadata formats available for the specified item.
`noSetHierarchy`	The repository does not support sets.

Slika 3.3: Tablica kodova i opisa greške[5]

3.4 Zahtjevi i odgovori OAI-PMH protokola

Za kraj opisa OAI-PMH protokola ćemo navesti zahtjeve koje protokol može tražiti i pripadajuće odgovore na njih. Svaki zahtjev protokola je definiran s vrijednošću ključne riječi *verb* pa ćemo ime zahtjeva poistovjetiti s tom vrijednosti.

GetRecord

GetRecord zahtjev se koristi za dohvaćanje pojedinog zapisa iz sustava. Zahtjev prima dva argumenta koji jedinstveno određuju o kojem zapisu se radi. Ti argumenti su *identifier* i *metadataPrefix*. Argument *identifier* je obavezan i prima *jedinstveni identifikator* koji određuje o kojem objektu se radi. Argument *metadataPrefix* je također obavezan i prima pojedinosti o formatu u kojem se traži objekt. **GetRecord** može prouzročiti tri greške:

[5]https://www.openarchives.org/OAI/openarchivesprotocol.html

- *badArgument* - zahtjevu nedostaje obavezni argument ili nije ispravnog oblika.
- *cannotDisseminateFormat* - vrijednost argumenta *metadataPrefix* nije podržana za objekt sa zadanom vrijednosti argumenta *identifier*.
- *idDoesNotExist* - vrijednost argumenta *identifier* nije ispravna, odnosno objekt s traženim *jedinstvenim identifikatorom* ne postoji ili nije dostupan.

Identify

Zahtjev **Identify** se koristi za dohvaćanje podataka o sustavu. Za razliku od prethodnog zahtjeva, on ne prima nikakve argumente i može postoji samo jedna greška koju može prouzrokovati, a to je *badArgument* u slučaju da zahtjev primi argumente. Format odgovora na ovaj zahtjev sadrži točno određene elemente koje mora zaprimiti i još neke koje može, ali nisu obavezni u odgovoru. Osim tog uvjeta, odgovor mora sadržavati točno po jedan primjerak elemenata:

- *repositoryName* - vrijednost elementa sadrži ime sustava.
- *baseURL* - osnovni URL na kojem se sustav nalazi.
- *protocolVersion* - verzija OAI-PMH protokola koju sustav podržava.
- *earliestDatestamp* - vrijednost tipa UTCdatetime koja je donja granica za sve datume u sustavu.
- *deletedRecord* - jednu od vrijednosti *"no"*, *"transient"* ili *"persistent"* koja govori o razini praćenja brisanja zapisa u sustavu.
- *granularity* - vrijednost ovog elementa označava najprecizniju valjanu vrijednost datuma u sustavu. Moguće vrijednosti su *YYYY-MM-DD* i *YYYY-MM-DDThh:mm:ssZ* čije značenje je definirano u ISO8601[6].

Osim gore navedenih postoji još jedan obavezan element kojeg odgovor na zahtjev **Identify** mora sadržavati, a to je *adminEmail*. Taj element nije u gornjem popisu jer se u sustavu može pojaviti više puta, ali mora barem jednom. On sadrži e-mail adresu administratora sustava.

Osim obaveznih elemenata, odgovor može sadržavati još dva elementa proizvoljan broj puta: *compression* i *description*. Element *compression* sadrži popis kodiranja koji sustav podržava, a element *description* sadrži opis sustava. Svaki element *description* mora sadržavati URL na XML shemu koja opisuje strukturu skupa podataka koju *description* sadrži.

[6] https://www.w3.org/TR/NOTE-datetime

ListIdentifiers

Zahtjev **ListIdentifiers** je skraćena verzija zahtjeva **ListRecords** koji ćemo kasnije opisati. Naime u zahtjevu **ListIdentifiers** se dohvaćaju samo zaglavlja svih zapisa, dok će se kod zahtjeva **ListRecords** dohvaćati cijeli zapisi. Ovaj zahtjev može primit neobavezne argumente kojima se može suziti dohvat zapisa s obzirom na pripadnost traženom skupu i/ili datumu zapisa. Zahtjev naravno može imati više argumenata, od kojih je samo jedan obavezan. Ti argumenti su:

- *metadataPrefix* - obavezan argument koji specificira format u kojem zapis mora biti vraćen. Ukoliko zapis ne postoji u traženom formatu, ovisno o razini praćenja brisanja zapisa u sustavu, vraća se drugačiji odgovor. Vrste odgovora se može potražiti u pododjeljku Record u prethodnom odjeljku.
- *set* - neobavezan argument koji prima vrijednost definiranu u zahtjevu **ListSets** s vrijednošću iz polja *setSpec*. Ovaj argument smo već spomenuli, a on služi za dohvat zapisa koji pripadaju skupu koji je naveden kao vrijednost argumenta.
- *from* - neobavezan argument koji prima vrijednost tipa UTCdatetime i služi za birani dohvat zapisa čiji datum je veći od vrijednosti argumenta.
- *until* - također neobavezan argument, koji kao i *from* prima UTCdatetime tip vrijednosti. Njegova svrha je birani dohvat zapisa čiji datum je manji od vrijednosti argumenta.

U **ListIdentifiers** zahtjevu se mogu pojaviti ukupno četiri različite greške:

- *badArgument* - vrijednost argumenta u zahtjevu nije dopuštena ili nedostaje.
- *cannotDisseminateFormat* - vrijednost argumenta *metadataPrefix* nije podržava u sustavu
- *noRecordsMatch* - kombinacija neobaveznih argumenata *set*, *from* i *until* te obaveznog argumenta *metadataPrefix* ne vraća niti jedan zapis, odnosno lista zapisa je prazna.
- *noSetHierarchy* - sustav ne podržava skupove, a argument *set* je naveden.

ListMetadataFormats

ListMetadataFormats zahtjev kao što ime kaže služi za dohvat formata metapodataka koji su dostupni u sustavu. Zahtjev osim popisa svih dostupnih formata u sustavu može poslužiti i za dohvat formata metapodataka za pojedini zapis. Kako bi to postigao, zahtjev

može sadržavati jedan neobavezan argument *identifier* čija vrijednost je *jedinstveni identifikator* objekta čije formate tražimo. Važno je naglasiti da svaki objekt ne mora imati prikaz u svim dostupnim formatima u sustavu, doduše čak ne mora imati prikaz ni u jednom zapisu. U slučaju pokušaja dohvata formata za objekt koji nema zapis ni u jednom formatu, OAI-PMH protokol će baciti grešku *noMetadataFormats*. Osim te greške, protokol može prouzrokovati još dvije greške: *badArgument* i *idDoesNotExist*. Obje greške smo već opisali u zahtjevu **GetRecord** i u potpunosti su analogne za oba zahtjeva.

ListRecords

Zahtjev **ListRecord** je proširenje zahtjeva **GetRecord**. Naime kako u zahtjevu **GetRecord** dohvaćamo jedan zapis, ovdje dohvaćamo listu zapisa. Ta lista zapisa u zahtjevu **ListRecords** može biti lista svih zapisa ili može biti sužena pomoću argumenata. U **ListRecord** zahtjevu postoje isti argumenti kao i u zahtjevu **ListIdentifiers**, prošireni dodatnim argumentom *identifier* iz **GetRecord** zahtjeva. Tako zahtjev **ListRecords** ima jedan obavezan argument *metadataPrefix*, a može primiti i neobavezne argumente: *identifiers*, *set*, *from* i *until*. Sve te argumente smo već spomenuli i opisali, a kako u ovom zahtjevu imaju u potpunosti isto značenje nećemo ih ponovno navoditi. Lista mogućih grešaka u zahtjevu **ListRecords** je ista kao i lista mogućih grešaka u **ListIdentifiers** zahtjevu. To znači da su moguće greške: *badArgument*, *cannotDisseminateFormat*, *noRecordsMatch* i *noSetHierarchy*. Sve greške su identične kao u **ListIdentifiers** zahtjevu pa ih nećemo ponovno opisivati.

ListSets

Posljednji zahtjev u OAI-PMH protokolu je **ListSets**. On se koristi za dohvat liste svih skupova koji postoje u sustavu. Zahtjev ne prima nikakve argumente i može proizvesti samo dvije greške: *badArgument* u slučaju da primi neki argument i *noSetHierarchy* ako sustav ne podržava skupove.

Poglavlje 4

OAI-PMH-API

4.1 Tehnologije

OAI-PMH-API je naziv aplikacije koja je napravljena kao praktični dio ovog diplomskog rada. Aplikacija je bazirana na OAI-PMH protokolu, odnosno ona služi za povezivanje Informacijskog sustava znanosti Republike Hrvatske (CroRIS[1]) i OpenAIRE-a. Aplikacija ima implementirane neke metode OAI-PMH protokola, ali je specijalizirana za potrebe izvoza podataka u OpenAIRE.

OAI-PMH-API je web aplikacija napisana u programskom jeziku Java uz korištenje Java Spring programskog okvira i Spring Boot[2] alata. Java Spring programski okvir je otvoreni okvir za stvaranje samostalnih aplikacija koje se izvršavaju na Java Virtualnom Stroju[3]. Spring Boot je alat koji olakšava i ubrzava izradu web aplikacija i mikroservisa u Spring programskom okviru pomoću tri ključne mogućnosti: automatske konfiguracije, samostalanog pristupa konfiguraciji i mogućnosti stvaranja samostalnih aplikacija.

OAI-PMH-API je također REST API[4], odnosno aplikacija koja odgovara REST arhitekturi. API (*application programming interface*) je skup pravila koja definiraju način spajanja i komunikacije među uređajima ili aplikacijama. REST(*representational state transfer*) stil arhitekture je skup implementacijskih pravila koja služe za povezivanje komponenata i aplikacija u arhitekturu mikroservisa. Kako bi aplikacija bila REST-API, ona mora zadovoljavati šest pravila:

1. *Uniformno sučelje* - svi zahtjevi za isti resurs moraju isto izgledati, bez obzira odakle dolaze.

[1] https://croris.hr/
[2] https://www.ibm.com/cloud/learn/java-spring-boot
[3] https://www.javatpoint.com/jvm-java-virtual-machine
[4] https://www.ibm.com/cloud/learn/rest-apis

2. *Neovisnost klijenta i servera* - klijentska i serverska aplikacija moraju biti u potpunosti neovisne. Jedina informacija koju klijentska aplikacija treba znati je URL zahtjeva.

3. *Statelessness* - Svaki zahtjev treba sadržavati sve informacije za njegovu obradu. Drugim riječima REST API ne dopušta serverskoj aplikaciji pohranjivanje podataka vezanih za zahtjev klijenta.

4. *Mogućnost predmemoriranja* - Ako je moguće, resursi bi se trebali spremiti u predmemoriju na klijentskoj ili serverskoj strani. Cilj ovog pravila je poboljšanje performansi.

5. *Slojevita sistemska arhitektura* - pozivi i odgovori u REST API-ima prolaze kroz različite slojeve. Moguće je postojanje međuslojeva između klijenta i servera pa ni klijet ni server u REST API-ju ne mogu znati komuniciraju li direktno ili s posrednikom.

6. *Izvršavanje na zahtjev* - U slučajevima da odgovor sadrži izvršim kod, a ne samo statičke resurse, taj kod se mora pokretati samo na zahtjev klijenta.

4.2 Implementacija protokola

U ovom odjeljku ćemo prikazati tok rada OAI-PMH-API aplikacije od primanja zahtjeva kroz pristupnu točku do vraćanja XML datoteke s potrebnim zapisima. Tok koji ćemo opisati je zajednički za gotovo sve zahtjeve pa ćemo razlike za pojedine zahtjeve napomenuti u pojedinim segmentima toka.

Kontroler

Nakon što klijent pošalje zahtjev, on se automatski prosljeđuje kontroleru. Klasa kontrolera je anotirana s **@RestController** anotacijom, što znači da aplikacija zna da je ona kontroler i da njoj prosljeđuje zahtjev kao u MVC (Model-View-Controller) arhitekturi te da će povratna vrijednost biti neki Http odgovor koji će biti u obliku JSON ili XML datoteke. Naš kontroler ima samo jednu metodu koja je anotirana s **@GetMapping("")**. Ta anotacija označava da metoda prima **GET** zahtjev koji je od baznog URL-a aplikacije odvojen praznom putanjom. Ta metoda može primiti tri parametra: *verb*, *metadataPrefix* i *set* kao što je definirano u OAI-PMH protokolu od koji je jedino parametar *verb* nužan. Primjere slanja podržanih zahtjeva ćemo vidjeti u idućem odjeljku.

Nakon primanja zahtjeva u metodu kontrolera, provjeravaju se priloženi parametri kako bi se odredio servis koji tražimo. Prvo provjeravamo vrijednost obaveznog parametra *verb*.

Postoje četiri podržana slučaja za taj parametar i ovisno o vrijednosti se zove jedan od tri servisa: dohvat liste skupova, dohvat metapodataka o formatima i dohvat podataka o servisu ili se zahtjev prosljeđuje na daljnju provjeru parametara ukoliko parametar *verb* je zahtjev za listom zapisa.

U daljnjoj provjeri, provjeravamo samo vrijednost parametra *set* zato što je to dovoljno za potrebe OpenAIRE-a. Naime OpenAIRE traži podatke u samo jednom formatu pa je cijela aplikacija konstruirana da se podaci nakon dohvata iz baze spreme u klase koje zadovoljavaju taj format. Također za dohvat liste zapisa, vrijednost parametra *set* mora biti unesena jer OpenAIRE traži isključivo podatke u traženim skupovima. Trenutno su implementirana četiri moguća skupa: *Equipments*(*Oprema*), *OrgUnits*(*Ustanove*), *Persons*(*Osobe*) i *Projects*(*Projekti*). Za svaki od skupova se pozivaju posebni servisi, odnosno metode klase servisa koje ćemo opisati u idućem pododjeljku.

Na kraju kontrolera, nakon što servisi dohvati odgovarajući odgovor na zahtjev, se u odgovor zahtjeva ugrađuju vrijednosti parametara koje su primljene, izgled zahtjeva koji je poslan i vrijeme slanja odgovora. S obzirom da su svi ti podaci pohranjeni u klasi *Response*, na kraju kontrolera pozivamo metodu koja pretvara naš objekt i sve njegove podobjekte u XML datoteku. Tu datoteku potom vraćamo kao odgovor na zahtjev.

Servis

Sljedeći korak nakon kontrolera je već spomenuti poziv servisa. Iako smo gore govorili o različitim servisima, u aplikaciji postoji samo jedna servis klasa s anotacijom **@Service**, a kao i obično pozivaju se njene metode. Ta anotacija omogućuje Spring Bootu da prepozna klasu kao servis klasu, odnosno klasu koja implementira poslovnu funkcionalnost što je u našem slučaju dohvat svih traženih podataka.

U našem servisu imamo ukupno sedam metoda, po jedna za svaki implementirani zahtjev. Imamo četiri slične metode za dohvat liste skupova zapisa i tri jednostavnije metode koje ćemo prvo opisati. Najjednostavnija metoda je dohvat formata metapodataka. U njoj stvaramo novu klasu koja sadrži podatke o formatu koji koristimo. Inače bi postojala lista, ali već smo spomenuli da OpenAIRE kojem pružamo podatke prihvaća samo jedan fiksan format pa smo njegove podatke isto tako fiksirali u klasu. Tu klasu potom vraćamo kontroleru. Druga metoda je dohvat liste setova. U njoj stvaramo klasu liste setova i dodajemo u nju sve skupove koje naša aplikacija podržava tako što svaki spremimo u klasu *Set* koja sadrži podatke *setSpec* i *setName* kao što je opisano u protokolu. Nakon što ju popunimo, listu vraćamo kontroleru. Zadnja od tri jednostavnije metoda je dohvat podataka o servisu. U toj metodi stvaramo klasu *CroRISService* koja sadrži podatke o CroRIS-u koji su opisanu u *Service* elemenut OpenAIRE-a. Toj klasi potom pridružujemo podatke o ustanovi koja je vlasnik, odnosno Sveučilišnom računskom centru u Zagrebu (SRCE) pozivajući odgovarajući repozitorij.

Sljedeće ćemo opisati četiri slične metode koje služe za dohvat zapisa o skupovima koje aplikacija sadrži. S obzirom da metode rade identično za dohvat svih skupova, opisat ćemo samo rad jedne metode. Metode se razlikuju u pozivima odgovarajućih repozitorija i u stvaranju odgovarajućih klasa s obzirom na tip podataka koji tražimo, drugim riječima ovisno o traženom skupu. Tako imamo četiri metode od kojih svaka dohvaća listu svih osoba, ustanova, opreme ili projekata. Kao primjer ćemo opisati metodu za dohvat svih osoba.

Metoda za dohvat svih osoba prima parametar *set* koji ima istu vrijednost kao istoimeni parametar u kontroleru. Na početku metode stvarano klasu *ListRecords* i dohvaćamo podatke o svim osobama pozivajući metodu *getAllPersons()* iz repozitorija *PersonRepository*. Ta metoda nam je vratila listu klasa *Person*, ali kao povratnu vrijednost želimo da servis vrati klasu liste zapisa koju smo stvorili. Metoda potom stvori listu zapisa *List<Record>* (nije isto što i klasa liste zapisa) te za svaku osobu u listi osoba radi sljedeće:

1. Stvori novi prazan zapis, odnosno klasu tipa *Record*

2. Stvori novu klasu *Metadata* kojoj pridruži podatke o osobi pozivom metode *setPerson(data)*, gdje je *data* varijabla tipa *Person* i sadrži podatke o jednom elementu iz liste osoba

3. klasu *Metadata* pridruži praznom zapisu

4. Stvori novo zaglavlje kojem postavi podatke o trenutnom vremenu i pripadnom skupu iz varijable *set* koju smo primili u metodi

5. zapisu pridruži zaglavlje

6. zapis se doda u listu zapisa

Na kraju klasa pridruži listu zapisa klasi liste zapisa jer želimo kontroleru vratiti jednu klasu, a ne listu. Tu klasu metoda potom vrati kontroleru. Za kraj napomenimo da koristimo klasu liste zapisa iz dva razloga:

1. Imamo jedan tip povratne vrijednosti za sve četiri metode. Time je jednostavnija implementacija kontrolera jer ne moramo imati drugačiju varijablu za svaki skup zapisa.

2. Naš pretvarač objekta u XML datoteku radi na razini klasa. Imajući istu klasu za zapise će nam pomoći da imamo jedinstven izgled skupa zapisa za sve skupove. Više o tome ćemo komentirati u pododjeljku **Konverter u XML**.

Repozitorij

Spomenuli smo u opisu zadnje metode servisa da pozivamo repozitorij *PersonRepository* i njegovu metodu *getAllPersons()*. Naime, za razliku od servisa gdje imamo jednu klasu i različite metode, ovdje imamo za svaki implementirani skup po jedan repozitorij. To znači da imamo trenutno četiri repozitorija: *PersonRepository*, *OrgUnitRepository*, *EquipmentRepository* i *ProjectRepository*. Svaki repozitorij ima svoje sučelje, odnosno *Interface* koje sadrži popis metoda i njihove povratne vrijednosti i ima klasu u kojoj su implementirane sve te metode koja je anotirana s **@Repository**. Ta anotacija govori Spring Bootu da klasa pruža mehanizme za rad nad objektima, što će u našem slučaju biti samo dohvat iz baze.

S obzirom na trenutne zahtjeve na aplikaciju, implementirani repositoriji nemaju sve metode identične. Tako repozitoriji za opremu i projekte imaju implementiranu samo metodu za dohvat sve opreme, odnosno projekata. Repozitorij za osobe ima osim dohvata svih osoba i metodu za dohvat pojedine osobe po njenom jedinstvenom identifikatoru iz baze. Taj dohvat se koristi za dohvat osoba koje sudjeluju u projektima, imaju neku ulogu u opremi ili na ustanovi. Zadnji repozitorij, odnosno repozitorij za ustanove sadrži metode za dohvat svih ustanova, pojedine ustanove po jedinstvenom identifikatoru i metodu za dohvat podataka o vlasniku servisa. Ova zadnja metoda je posebno odvojena zato da ne fiksiramo identifikator ustanove u aplikaciju.

Nakon što smo opisali koje sve metode postoje trebali bi opisati i implementaciju, ali ona je identična za sve metode u svim repozitorijima. Svaka od metoda radi točno dva koraka:

1. Postavlja parametre u varijablu *params*. Ukoliko metoda ne šalje parametre, postavi se prazan parametar.

2. poziva se metoda iz DAO klase s varijablom *params* i vraća se dobivena vrijednosti.

Za DAO (*Data Access Object*) klase nismo izdvojili poseban odjeljak jer su blisko povezane s repozitorijima i maperima pa ćemo ih opisati u sljedećem pododjeljku.

Maper

Za početak opišimo DAO klase. Kao i repozitoriji, svaki skup podatak ima svoju DAO klasu koja ima naziv oblika *"SkupDao"* gdje na mjestu "Skup" stoji naziv tipa objekta koji dohvaća, na primjer *PersonDao*. Svaka DAO klasa ima anotaciju **@Mapper** i sadrži popis svih metoda koje može pozvat iz odgovarajućeg mapera. Ta anotacija ima središnju ulogu u povezivanju podataka iz baze s klasama u koje ih želimo preslikati u našoj aplikaciji. Kako bi to preslikavanje bilo uspješno u maperima moramo pravilno dodijeliti tipove objekata, odnosno klase i imena varijabli u koje ih želimo preslikati.

Kao i repozitoriji i DAO klase, tako za svaki skup podataka postoji zaseban maper. To nije nužno, ali u aplikaciji poboljšava preglednost i snalaženje u kodu. Za razliku od ostatka projekta, maperi nisu pisani u programskom jeziku Java nego su kombinacija xml (*Extensible Markup Language*) i SQL (*Structured Query Language*) kodova. U skladu s time je njihov tip datoteke *.xml* i nalaze se u posebnom odjeljku aplikacije koji se zove *resources*.

Svaki maper se sastoji od dva dijela: popisa svih mapa *resultMap* i *select* upita. U *select* dijelovima imamo tri ključna elementa:

- atribut *id* - identifikator upita kojeg koristimo i kao ime upita. Koristi se u DAO klasama kao ime metode i u maperima za poziv upita koji popunjava komponente elementa *resultMap*.
- atribut *resultMap* - sadrži ime pripadne mape koja se popunjava izvršavanjem upita.
- SQL kod - sadrži kod koji se izvršava nad bazom. U našem slučaju su svi upiti SELECT.

Glavni dio mapiranja se nalazi u elementima *resultMap*, odnosno mapama. Već smo rekli da se mape popunjavaju izvršavanjem upita, ali također služe i za spajanje rezultata s Java klasama. Svaka mapa sadrži dva atributa: *id* koji koristimo kao ime mapera te kao atribut u upitima i *type* koji sadrži putanju u aplikaciji do klase koju stvaramo i popunjavamo. Svaka mapa sadrži jednak broj elemenata kao i klasa, a svaki od njih je jednog od četiri tipa:

- *id* - služi za označavanje identifikatora u klasi objekta i ima atribute: *property* koji sadrži ime elementa u klasi i *column* koji označava ime stupca u povratnoj vrijednosti SQL-a koji se spaja s elementom iz klase.
- *result* - služi za spajanje ostalih elemenata iz klase s vrijednostima iz upita ako se vrijednosti nalaze u točno jednom stupcu. Kao i *id* ima dva atributa *property* i *column* koji imaju isto značenje pa ih nećemo ponovno opisivati.
- *association* - koristi se za mapiranje više stupaca iz upita u jedan element klase. On sadrži atribute *property* i *javaType*. Prvi atribut kao i gore sadrži ime elementa iz klase, a drugi tip elementa koji je često neka druga klasa. Tip *association* osim atributa sadrži i elemente koji mogu biti istih tipova kao i elementi za *resultMap* te imaju analogno značenje, ali se spajaju u klasu čija putanja se nalazi u atributu *javaType*.
- *collection* - najkompleksniji tip elementa koji služi za popunjavanje elemenata pozivom drugih upita. Često se koristi za popunjavanje elemenata koji sadrže liste ili čiji tip varijable je neka klasa koja je element OpenAIRE-a poput *Person*, *OrgUnit* ili slično. Tip *collection* sadrži atribute:

- *property* - kao i prije, sadrži ime elementa koji se popunjava u klasi iz mapera.
- *javaType* - kao i kod *association* tipa označava klasu elementa.
- *column* - prenosi stupac iz mapera kao parametar u sql upit iz atributa *select* koji se poziva.
- *ofType* - sadrži ime drugog mapera koji sadrži elemente iste kao i *collection*. Atribut nije potreban ukoliko imamo popis elemenata kao i kod tipa *association*.
- *select* - sadrži ime upita kojeg *collection* poziva za popunjavanje elemenata.

Nakon što smo dohvatili sve vrijednosti te stvorili i popunili odgovarajuće objekte, vraćamo povratnu vrijednost u repozitorij pa u servis i na kraju u kontroler.

Konverter u XML

Zadnji korak koji moraju proći svi zahtjevi je pretvorba objekata u odgovarajući format, odnosno XML. Već smo spomenuli da se taj proces odvija neposredno prije vraćanja odgovora iz kontrolera. Kako bismo to postigli moramo pozvati metodu *convertToXML* iz klase *ObjectToXmlConverter*. Sama klasa stvara i koristi *JAXBContext*[5] klasu i pripadno *Marshaller*[6] sučelje koje je stvorila klasa *JAXBContext*. Sučelju naša metoda prosljeđuje objekt koji je primila, a ono onda prolazi kroz stablo klase objekta i stvara XML. Taj XML se potom potom upisuje u objekt tipa *StringWriter* koji verificira sadržaj i vraća ga kontroleru kao varijablu tipa String.

Kako bi sučelje znalo kako pravilno nazvati elemente u XML datoteci te razlikovati atribute objekta od pravih elemenata, potrebno je sve to definirati u klasama. Iz tog razloga smo morali u svim klasama koje se koriste u ispisu, dodatno definirati njihovu ulogu i naziv u XML-u. To smo ostvarili koristeći anotacije **@XmlAttribute(name="XMLIme")** i **@XmlElement(name="XMLIme")** nad metodama za dohvat pojedinih varijabli klase, odnosno *get* metodama.

4.3 Zahtjevi i odgovori

Za kraj ćemo pokazati zahtjeve koji su trenutno implementirani u aplikaciji. Odgovore na zahtjeve možemo usporediti s primjerima odgovora koji se nalaze na OpenAIRE-ovom GitHub repozitoriju[2].

Primjeri koje ćemo prikazati su izvršeni lokalno nad anonimiziranom bazom CroRIs-a kako bi zaštitili identitete osoba koje se nalaze u sustavu. Za slanje zahtjeva i primanje

[5] https://docs.oracle.com/javaee/7/api/javax/xml/bind/JAXBContext.html
[6] https://docs.oracle.com/javase/7/docs/api/javax/xml/bind/Marshaller.html

odgovora korištena je aplikacija Postman[7]. Postman je API razvojni alat za programere koji se koristi za lakše stvaranje, testiranje i dokumentiranje aplikacija. Nama je poslužio kao API klijent koji je slao HTTP zahtjeve aplikaciji i prikazivao dobivene odgovore.

Identify

Prvi zahtjev koji ćemo prikazati je **Identify**. Zahtjev smo već opisali na raznim mjestima pa ćemo samo prikazati primjer slanja zahtjeva i odgovore. Za slanje zahtjeva potrebno je poslati GET zahtjev oblika: **http://localhost:9862/oai-pmh-api/?verb=Identify**. Uočit ćemo da je u svim zahtjevima dio **http://localhost:9862/oai-pmh-api** zajednički. Prvi dio tog URL-a nam kaže kojem serveru je poslan zahtev, u našem slučaju **localhost** na portu **9862**,a drugi dio je ime aplikacije **/oai-pmh-api/** koji je bazni dio svake pristupne točke koju naša aplikacija prima. U ostatku URL-a vidimo parametre i njihove vrijednosti koje se prosljeđuju aplikaciji.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<OAI-PMH xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/ http://www.openarchives.org/OAI/2.0/
    OAI-PMH.xsd https://www.openaire.eu/cerif-profile/1.1/ https://www.openaire.eu/schema/cris/1.1/
    openaire-cerif-profile.xsd" xmlns="http://www.openarchives.org/OAI/2.0/" xmlns:xsi="http://www.w3.
    org/2001/XMLSchema-instance&quot;">
    <responseDate>2022-11-13T12:22:40.279+01:00</responseDate>
    <request verb="Identify">www.croris.hr/oai-pmh-api/</request>
    <Service>
        <Acronym>CroRIS</Acronym>
        <Compatibility>OpenAIRE Guidlines 1.1 CompatibleCRIS</Compatibility>
        <Name lang="hr">Informacijski sustav znanosti RH</Name>
        <Name lang="en">Croatian Research Information System</Name>
        <OAIPMHBaseURL>www.croris.hr/oai-pmh-api/</OAIPMHBaseURL>
        <Owner id="219" xmlns="https://www.openaire.eu/cerif-profile/1.1/">
            <Name lang="en">University Computing Centre - SRCE</Name>
            <Name lang="hr">Sveučilišni računski centar - Srce</Name>
            <Type Name="ostalo">1061</Type>
            <electronicAddress>ured@srce.hr</electronicAddress>
            <Identifier type="OIB">34016189309</Identifier>
            <Identifier type="Matični broj">3283020</Identifier>
            <Identifier type="Matični broj ustanove iz upisnika">093</Identifier>
        </Owner>
        <WebsiteURL>www.croris.hr</WebsiteURL>
    </Service>
</OAI-PMH>
```

Slika 4.1: Odgovor na zahtjev **Identify**

[7] https://www.postman.com/

ListSets

Zahtjev za listom implementiranih skupova je možda i najvažniji zahtjev za spajanje na OpenAIRE. Naime rekli smo već da OpenAIRE dohvaća zapise samo ako se nalaze u skupovima pa ovaj zahtjev zapravo daje popis podataka koje smo spremni izvesti. Izvršavanje ovog zahtjeva možemo pokrenuti slanjem GET zahtjeva:
http://localhost:9862/oai-pmh-api/?verb=ListSets

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<OAI-PMH xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/ http://www.openarchives.org/OAI/2.0/
    OAI-PMH.xsd https://www.openaire.eu/cerif-profile/1.1/ https://www.openaire.eu/schema/cris/1.1/
    openaire-cerif-profile.xsd" xmlns="http://www.openarchives.org/OAI/2.0/" xmlns:xsi="http://www.w3.
    org/2001/XMLSchema-instance&quot;">
    <responseDate>2022-11-13T12:26:18.757+01:00</responseDate>
    <request verb="ListSets">www.croris.hr/oai-pmh-api/</request>
    <ListSets>
        <set>
            <setName>OpenAIRE_CRIS_persons</setName>
            <setSpec>openaire_cris_persons</setSpec>
        </set>
        <set>
            <setName>OpenAIRE_CRIS_orgunits</setName>
            <setSpec>openaire_cris_orgunits</setSpec>
        </set>
        <set>
            <setName>OpenAIRE_CRIS_projects</setName>
            <setSpec>openaire_cris_projects</setSpec>
        </set>
        <set>
            <setName>OpenAIRE_CRIS_equipments</setName>
            <setSpec>openaire_cris_equipments</setSpec>
        </set>
    </ListSets>
</OAI-PMH>
```

Slika 4.2: Odgovor na zahtjev **ListSets**

ListMetadataFormats

Sljedeći zahtjev je **ListMetadataFormats**. Već smo komentirali da za potrebe OpenAIRE-a imamo samo jedan format i prema njemu su implementirane klase u aplikaciji. Za dobivanje podataka o formatu potrebno je poslati GET zahtjev:
http://localhost:9862/oai-pmh-api/?verb=ListMetadataFormats

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<OAI-PMH xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/ http://www.
    openarchives.org/OAI/2.0/OAI-PMH.xsd https://www.openaire.eu/cerif-profile/1.1/
    https://www.openaire.eu/schema/cris/1.1/openaire-cerif-profile.xsd" xmlns="http:/
    /www.openarchives.org/OAI/2.0/" xmlns:xsi="http://www.w3.org/2001/
    XMLSchema-instance&quot;">
    <responseDate>2022-11-13T12:27:16.886+01:00</responseDate>
    <request verb="ListMetadataFormats">www.croris.hr/oai-pmh-api/</request>
    <ListMetadataFormats>
        <metadataFormat>
            <metadataNamespace>https://www.openaire.eu/cerif-profile/1.1/</
                metadataNamespace>
            <metadataPrefix>oai_cerif_openaire</metadataPrefix>
            <schema>https://www.openaire.eu/schema/cris/1.1/openaire-cerif-profile.
                xsd</schema>
        </metadataFormat>
    </ListMetadataFormats>
</OAI-PMH>
```

Slika 4.3: Odgovor na zahtjev **ListMetadataFormats**

ListRecords: Osobe

Prvi **ListRecords** zahtjev za zapisima iz baze koji ćemo pokazati je zahtjev za ispisom osoba. Ovaj zahtjev dohvaća listu svih osoba, ali zbog veličine zahtjeva ćemo prikazati samo podatke o jednoj osobi. Također napomenimo da je za primjer odabrana osoba koja ima veću količinu podataka od ostalih zapisa, ali i dalje ne sadrži sve moguće podatke koje možemo izvesti. Zbog anonimizacije podataka će ime i prezime te elektronička adresa osobe u primjeru biti promijenjeni u serijalizirane vrijednosti. Ostali podaci su stvarni i pripadaju stvarnoj osobi. Podaci o ustanovama koje su vezane za osobu su prikazani u skraćenom obliku koji je dovoljan za identifikaciju ustanove i pronalazak iste u popisu svih ustanova. Kako bi se zahtjev izvršio potrebno je poslati GET zahtjev oblika:
http://localhost:9862/oai-pmh-api/?metadataPrefix=cerif_openaire &verb=ListRecords&set=openaire_cris_persons

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<OAI-PMH xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/ http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd https://www.openaire.eu/cerif-profile/1.1/ https://www.openaire.eu/schema/cris/1.1/openaire-cerif-profile.xsd" xmlns="http://www.openarchives.org/OAI/2.0/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance&amp;quot;">
    <responseDate>2022-11-13T12:40:27.571+01:00</responseDate>
    <request metadataPrefix="cerif_openaire" set="openaire_cris_persons" verb="ListRecords">www.croris.hr/oai-pmh-api/</request>
    <ListRecords>
        <record>
            <header>
                <datestamp>2022-11-13T12:40:30.642+01:00</datestamp>
                <setSpec>openaire_cris_persons</setSpec>
            </header>
            <metadata>
                <Person id="7">
                    <PersonName>
                        <FamilyNames>Prezime7</FamilyNames>
                        <FirstNames>Ime7</FirstNames>
                    </PersonName>
                    <Gender>Male</Gender>
                    <ElectronicAddress>znanstvenik7@croris.hr</ElectronicAddress>
                    <Affiliation Employment id="1" xmlns="https://www.openaire.eu/cerif-profile/1.1/">
                        <Name lang="en">Institute of Oceanography and Fisheries</Name>
                        <Name lang="hr">Institut za oceanografiju i ribarstvo, Split</Name>
                    </Affiliation Employment>
                    <Affiliation Employment id="66" xmlns="https://www.openaire.eu/cerif-profile/1.1/">
                        <Name lang="en">Ruđer Bošković Institute</Name>
                        <Name lang="hr">Institut Ruđer Bošković</Name>
                    </Affiliation Employment>
                </Person>
            </metadata>
        </record>
```

Slika 4.4: Primjer osobe iz odgovora na zahtjev **ListRecords** za dohvat osoba

ListRecords: Ustanove

Drugi implementirani zahtjev za dohvaćanje zapisa iz baze je zahtjev za popisom ustanova. Isto kao i u prošlom zahtjevu prikazat ćemo samo prvi zapis, a za to smo odabrali jednu od ustanova s većom količinom podataka. Također, analogno kao i gore su podaci o nadređenoj ustanovi prikazani u skraćenom obliku koji je dovoljan za njihovu identifikaciju. S obzirom da su ustanove i njihovi podaci javni, ovdje nije bilo potrebe za anonimizacijom. Zahtjev za popisom svih ustanova dobivamo slanjem GET zahtjeva:
http://localhost:9862/oai-pmh-api/?metadataPrefix=cerif_openaire&verb=ListRecords&set=openaire_cris_orgunits

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<OAI-PMH xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/ http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd https://www.openaire.eu/cerif-profile/1.1/ https://www.openaire.eu/schema/cris/1.1/openaire-cerif-profile.xsd" xmlns="http://www.openarchives.org/OAI/2.0/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance&quot;">
    <responseDate>2022-11-13T12:29:48.006+01:00</responseDate>
    <request metadataPrefix="cerif_openaire" set="openaire_cris_orgunits" verb="ListRecords">www.croris.hr/oai-pmh-api/</request>
    <ListRecords>
        <record>
            <header>
                <datestamp>2022-11-13T12:29:50.024+01:00</datestamp>
                <setSpec>openaire_cris_orgunits</setSpec>
            </header>
            <metadata>
                <OrgUnit id="79" xmlns="https://www.openaire.eu/cerif-profile/1.1/">
                    <Name lang="en">Faculty of Science</Name>
                    <Name lang="hr">Prirodoslovno-matematički fakultet, Zagreb</Name>
                    <Type Name="fakultet">590</Type>
                    <electronicAddress>dekanat@dekanat.pmf.hr</electronicAddress>
                    <Identifier type="OIB">28163265527</Identifier>
                    <Identifier type="Matični broj">080206953</Identifier>
                    <Identifier type="Matični broj ustanove iz upisnika">119</Identifier>
                    <partOf id="257" xmlns="https://www.openaire.eu/cerif-profile/1.1/">
                        <Name lang="en">University of Zagreb</Name>
                        <Name lang="hr">Sveučilište u Zagrebu</Name>
                        <Type Name="sveučilište">592</Type>
                        <electronicAddress>unizginfo@unizg.hr; rector@unizg.hr</electronicAddress>
                        <Identifier type="OIB">36612267447</Identifier>
                        <Identifier type="Matični broj">080209185</Identifier>
                        <Identifier type="Matični broj ustanove iz upisnika">312</Identifier>
                    </partOf>
                </OrgUnit>
            </metadata>
        </record>
```

Slika 4.5: Primjer ustanove iz odgovora na zahtjev **ListRecords** za dohvat ustanova

ListRecords: Oprema

Sljedeći **ListRecords** zahtjev koji smo implementirali je zahtjev za dohvat opreme. On izvršava slanjem GET zahtjeva:
http://localhost:9862/oai-pmh-api/?metadataPrefix=cerif_openaire &verb=ListRecords&set=openaire_cris_equipments

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<OAI-PMH xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/ http://www.openarchives.org/OAI/2.0/
    OAI-PMH.xsd https://www.openaire.eu/cerif-profile/1.1/ https://www.openaire.eu/schema/cris/1.1/
    openaire-cerif-profile.xsd" xmlns="http://www.openarchives.org/OAI/2.0/" xmlns:xsi="http://www.w3.
    org/2001/XMLSchema-instance&quot;">
    <responseDate>2022-11-13T12:31:40.077+01:00</responseDate>
    <request metadataPrefix="cerif_openaire" set="openaire_cris_equipments" verb="ListRecords">www.
        croris.hr/oai-pmh-api/</request>
    <ListRecords>
        <record>
            <header>
                <datestamp>2022-11-13T12:31:43.046+01:00</datestamp>
                <setSpec>openaire_cris_equipments</setSpec>
            </header>
            <metadata>
                <Equipment id="2" xmlns="https://www.openaire.eu/cerif-profile/1.1/">
                    <Name lang="hr">Mastersizer 2000, analizator veličina čestica</Name>
                    <Name lang="en">Mastersizer 2000, particles size analysator</Name>
                    <Identifier type="Inventory number"/>
                    <Description lang="hr">Uređaj služi za mjerenje veličina čestica metodom
                        raspršenja laserskog svjetla u području od 20 nm do 2 mm.</Description>
                    <Type Name="Medium Equipment (200.000 - 1.000.000 HRK)">611</Type>
                    <Type Name="srednja (200.000 - 1.000.000 kn)">611</Type>
                    <Type Name="Standalone">613</Type>
                    <Type Name="samostalan">613</Type>
                    <Type Name="Fully functional">620</Type>
                    <Type Name="potpuno funkcionalan">620</Type>
                    <owner id="66" xmlns="https://www.openaire.eu/cerif-profile/1.1/">
                        <Name lang="en">Ruđer Bošković Institute</Name>
                        <Name lang="hr">Institut Ruđer Bošković</Name>
                        <Type Name="javni znanstveni institut">599</Type>
                        <electronicAddress>ravnatelj@irb.hr</electronicAddress>
                        <Identifier type="OIB">69715301002</Identifier>
                        <Identifier type="Matični broj">3270289</Identifier>
                        <Identifier type="Matični broj ustanove iz upisnika">098</Identifier>
                    </owner>
                </Equipment>
            </metadata>
```

Slika 4.6: Primjer opreme iz odgovora na zahtjev **ListRecords** za dohvat oprema

ListRecords: Projekti

Posljednji zahtjev koji smo implementirali u aplikaciji je zahtjev za dohvat podataka o projektima. Za razliku od prošlih zahtjeva, zapisi o projektima imaju puno bolje popunjene podatake, a projekti su kako smo spomenuli u poglavlju o OpenAIRE-u jedan od glavnih elemenata CRIS sustava. Na sljedeće dvije slike možemo vidjeti podatke o jednom zapisu opreme koji za razliku od prethodnih primjera ima manju količinu podataka od većine drugih projekata. Možemo također uočiti da su podaci o osobama anonimizirani kao i kod dohvata podataka o osobama. Zahtjev za popisom opreme izvršavamo slanjem GET zahtjeva: **http://localhost:9862/oai-pmh-api/?metadataPrefix=cerif_openaire &verb=ListRecords&set=openaire_cris_projects**

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<OAI-PMH xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/ http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd https://www.openaire.eu/cerif-profile/1.1/ https://www.openaire.eu/schema/cris/1.1/openaire-cerif-profile.xsd" xmlns="http://www.openarchives.org/OAI/2.0/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance&quot;">
    <responseDate>2022-11-13T12:33:53.170+01:00</responseDate>
    <request metadataPrefix="cerif_openaire" set="openaire_cris_projects" verb="ListRecords">www.croris.hr/oai-pmh-api/</request>
    <ListRecords>
        <record>
            <header>
                <datestamp>2022-11-13T12:34:02.278+01:00</datestamp>
                <setSpec>openaire_cris_projects</setSpec>
            </header>
            <metadata>
                <Project id="5">
                    <Titles lang="hr">Multimodalni pristup liječenju i dugoročnom praćenju tijeka depresivnog poremećaja metodom magnetske rezonancije</Titles>
                    <Titles lang="en">Multimodal approach to treatment and long-term assessment of depressive disorder using magnetic resonance imaging</Titles>
                    <Acronym>MODERN</Acronym>
                    <Types Name="Scientific research projects">385</Types>
                    <Types Name="Znanstveno-istraživački projekti">385</Types>
                    <Identifier type="Project identifier">IP-2014-09-2979</Identifier>
                    <StartDate>2015-07-01T00:00:00+02:00</StartDate>
                    <EndDate>2018-01-07T00:00:00+01:00</EndDate>
                    <Abstract lang="hr"/>
                    <Abstract lang="en">The aim of the proposal is to establish a unique and novel approach to therapy and assessment of long-term effects of depression. We plan to use a network of leading researchers in the fields of magnetic resonance imaging, spectroscopy, traditional psychiatric approach and ethics. As a result, we expect to bring a major contribution to personalized approach in treatment of depression, objectivising of treatment results and assessment of effects of the disease and the treatment on the wellbeing of the individual and its cost to the society.</Abstract>
                    <Team>
                        <PrincipalInvestigator>
                            <Person id="7506">
                                <PersonName>
                                    <FamilyNames>Prezime7507</FamilyNames>
                                    <FirstNames>Ime7507</FirstNames>
                                </PersonName>
                                <ElectronicAddress>znanstvenik7506@croris.hr</ElectronicAddress>
                            </Person>
                            <Role Name="Manager">415</Role>
                        </PrincipalInvestigator>
                    </Team>
                    <Consortium>
                        <Contractor>
                            <Role Name="Contractor">397</Role>
                            <OrgUnit id="70" xmlns="https://www.openaire.eu/cerif-profile/1.1/">
                                <Name lang="en">School of Medicine</Name>
                                <Name lang="hr">Medicinski fakultet u Zagrebu</Name>
                                <Type Name="fakultet">590</Type>
                                <electronicAddress>mf@mef.hr</electronicAddress>
                                <Identifier type="OIB">45001686598</Identifier>
                                <Identifier type="Matični broj">080159956</Identifier>
                                <Identifier type="Matični broj ustanove iz upisnika">108</Identifier>
                            </OrgUnit>
                        </Contractor>
                    </Consortium>
                    <OAMandated>true</OAMandated>
                </Project>
            </metadata>
        </record>
```

Slika 4.7: Primjer projekta iz odgovora na zahtjev **ListRecords** za dohvat projekata

Poglavlje 5

Zaključak

U ovom diplomskom radu smo trebali prikazati važnost otvorenih podataka te složenost prikupljanja i obrade podataka na primjeru platforme OpenAIRE. U sklopu prikupljanja podataka, bilo je potrebno prikazati Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) kao protokol za uspostavu interoperabilnosti sustava. Kako bi se lakše prikazala uspostava interoperabilnosti sustava i razmjena podataka, bilo je potrebno napisati aplikaciju za razmjenu podataka upravo putem OAI-PMH protokola.

U današnjem svijetu je razmjena informacija među sustavima gotovo neizostavan dio svakog informacijskog sustava. Štoviše, možemo reći da informacijski sustav koji ne razmjenjuje informacije ne može ni doseći značajnu razinu uporabljivosti. Kako bi sustavi mogli razmjenjivati informacije, oni moraju biti interoperabilni. Uspostava interoperabilnosti je zato jedan od prvih koraka koje svaki informacijski sustav mora napraviti. Upravo razmjena informacija među sustavima može značajno povećati količinu podataka koje sustav sadrži, a za današnje društvo su upravo sustavi s velikom količinom podataka osobito važni. Osim velike količine podataka ključna su još dva izrazito važna svojstva: dobra uređenost podataka i dostupnost podataka javnosti.

Otvoreni podaci su podaci koje može besplatno koristiti, mijenjati i dijeliti bilo tko za bilo koju svrhu. Osobito su važni otvoreni podaci u znanosti koji donose ubrzan razvoj znanosti i tehnologije u svijetu. Kako bi otvorene podatke jednostvanije razmjenjivali, potrebno je držati se FAIR smjernica koje služe za dobro strukturiranje i upravljanje otvorenim podacima. FAIR je akronim za četiri glavna svojstva otvorenih podataka: mogućnost pronalaženja (Findability), pristupačnost (Accessibility), interoperabilnost (Interoperability) i mogućnost ponovnog korištenja (Reusability).

Najveća platforma za prikupljanje otvorenih podataka o znanstvenim istraživanjima u Europi je OpenAIRE. OpenAIRE od osnivanja 2008.godine do danas se neprestano razvija i širi pa tako već sad okuplja više od trideset zemalja članica, a od 2018. godine se sklapanjem novih partnerstava proširio i na ostatak svijeta. OpenAIRE za otkrivanje, prikup-

ljanje i obradu podataka koristi desetke drugih aplikacija i platformi. Kako bi olakšao već izrazito složenu obradu podataka, OpenAIRE je definirao svoj model podataka za CRIS sustave koji je djelomično preuzet iz već provjerenog CERIF-ovog modela. Time je teret uspostave interoperabilnosti premješten na pružatelje podataka, ali isto tako taj model služi kao smjernica novim CRIS sustavima za izgradnju svoje baze podataka. Iz prikaza modela podataka i pojedinačnih elemenata, smo uočili izrazito gustu i povezanu mrežu podataka.

OAI-PMH protokol je niskogranični mehanizam za uspostavu interoperabilnosti podatkovnih skladišta. Protokol pruža interoperabilnost programskih okvira koja se bazira na prikupljanju metapodataka i koja je neovisna od aplikacija. To znači da je OAI-PMH protokol napisan općenito i da su za uspostavu interoperabilnosti između sustava potrebne minimalne ili nikakve izmjene. Sam protokol koristi HTTP GET i POST zahtjeve za slanje svojih zahtjeva, a odgovore prima u JSON ili XML formatu zapisa. Popis zahtjeva koje OAI-PMH protokol podržava je strogo definiran od izgleda HTTP zahtjeva i mogućih atributa do popisa elemenata u XML odgovoru.

U praktičnom dijelu ovog rada je napravljena aplikacija koja razmjenjuje informacije Informacijskog sustava znanosti Republike Hrvatske (CroRIS) i OpenAIRE-a koristeći upravo OAI-PMH protokol. Aplikacija je napisana u programskom kodu Java i korišteni su popularni programski okviri i alati poput Springa i Spring-Boota za lakšu izgradnju i povezivanje aplikacije. Implementiran je OAI-PMH protokola s minimalnim izmjenama koje su napravljene prateći OpenAIRE-ove primjere zahtjeva i odgovora. Također nisu implementirani svi zahtjevi nego samo oni koje propisuje OpenAIRE. Interoperabilnost OpenAIRE-a i CroRIS-ovog modela podataka je također napravljena unutar aplikacije. Interoperabilnost je u aplikaciji uspostavljena modeliranjem Java klasa kako bi odgovarale OpenAIRE-ovim elementima iz modela podataka.

Prilikom izrade ovog rada, upoznali smo važnost i vrijednost otvorenih podataka, te kako naše podatke učiniti dostupnijima i kako poboljšati njihovu uporabljivost. Također smo kroz primjer OpenAIRE platforme i njenog razvoja uvidjeli složenost pronalaska, prikupljanja i obrade podataka, a osobito na primjeru CRIS sustava. Upoznali smo se i sa složenom mrežom objekata koji čine CRIS sustave. Kroz OAI-PMH protokol smo naučili kako se uspostavlja interoperabilnost informacijskih sustava, a kroz izradu aplikacije smo vidjeli da to nije jednostavan proces. Izrada aplikacije je bila odličan način savladavanja svega što smo prije naučili, a upravo kroz samostalan rad smo uvidjeli da je razmjena informacija u sustavima izrazito složen proces za koji je potrebno puno vremena i truda.

Za kraj možemo zaključiti da usprkos složenom i zahtjevnom procesu, uspostava interoperabilnosti sustava je nužan preduvjet za izradu značajnog informacijskog sustava. Kako bi se olakšala uspostava interoperabilnosti, informacijski sustavi bi morali prilikom planiranja skladišta i modela podataka uzimati u obzir i slijediti već korištene modele poput CERIF-a. Osim samih sustava, veliki napredak u znanosti bi bila bolja popunjenost

podataka u sustavima, ali i više sadržaja u otvorenom pristupu. Štoviše, nakon izrade ovog rada, smatramo da je upravo nedostupnost podataka u javnosti najveći problem uspostave učinkovitijih informacijskih sustava i bržeg razvoja znanosti.

Bibliografija

[1] *euroCRIS*, https://eurocris.org/.

[2] *GitHub: Guidlines-CRIS-Managers/Samples*, https://github.com/openaire/guidelines-cris-managers/tree/v1.1/samples.

[3] *GO FAIR*, https://www.go-fair.org/fair-principles/.

[4] *Open Archives Initiative*, https://www.openarchives.org/.

[5] *Open Data Handbook*, https://opendatahandbook.org/guide/en/what-is-open-data/.

[6] *Open Definition*, https://opendefinition.org/.

[7] *OpenAIRE Guidelines for CRIS Managers*, https://openaire-guidelines-for-cris-managers.readthedocs.io/en/v1.1.1/index.html.

[8] *OpenAIRE*, https://www.openaire.eu/.

[9] DC: The National Academies Press Washington, *A Question of Balance: Private Rights and the Public Interest in Scientific and Technical Databases (1999)*, (1999), https://nap.nationalacademies.org/read/9692/chapter/3.

[10] Mark D. Wilkinson et al., *The FAIR Guiding Principles for data management and stewardship*, (2006), https://www.nature.com/articles/sdata201618?ref=https://githubhelp.com.

Sažetak

Informacijski sustavi su u današnje vrijeme postali središta razmjene novih postignuća u znanosti. Kako bi ta razmjena bila učinkovita, informacijski sustavi se moraju povezati. Jedan takav informacijski sustav koji prikuplja informacije iz svijeta znanosti je Open Access Infrastructure for Research in Europe poznatiji kao OpenAIRE. OpenAIRE je najveći takav sustav u Europi koji okuplja više od trideset zemalja članica. U praktičnom dijelu ovog rada je napravljena aplikacija i opisano je povezivanje OpenAIRE-a i Informacijskog sustava znanosti Republike Hrvatske (CroRIS) koristeći Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH).

U prvom poglavlju govorimo o otvorenim podacima i njihovim svojstvima. Potom definiramo otvorene podatke u znanosti i opisujemo FAIR smjernice za dobro upravljanje podacima. FAIR principi također služe i za lakše pronalaženje podataka pa detaljno objašnjavamo svaki od četiri principa: mogućnost pronalaženja (Findability), pristupačnost (Accessibility), interoperabilnost (Interoperability) i mogućnost ponovnog korištenja (Reuse).

Drugo poglavlje je posvećeno OpenAIRE-u. Za početak iznosimo pregled razvoja OpenAIRE-a od njegovih početaka do danas. Nakon toga govorimo o podacima koje OpenAIRE prikuplja i detaljno opisujemo izgled svakog elementa CRIS sustava kojeg OpenAIRE sadrži. Upravo ti elementi nam služe kao smjernice za oblikovanje objekata u našoj aplikaciji.

U trećem poglavlju govorimo o OAI-PMH protokolu na kojem se bazira rad naše aplikacije. Prvo definiramo osnovne pojmove i njihovu svrhu u protokolu. Potom opisujemo značajke protokola poput izgleda HTTP zahtjeva i odgovora u XML formatu. Za kraj poglavlja navodimo sve zahtjeve koje protokol podržava i opisujemo pripadne odgovore.

U zadnjem poglavlju detaljno objašnjavamo rad OAI-PMH-API aplikacije koju smo napravili. Prvo navodimo sve korištene tehnologije, a potom prolazimo kroz tok rada aplikacije. Detaljno opisujemo svaki korak od zaprimanja zahtjeva do vraćanja odgovora. Na kraju prikazujemo sve implementirane zahtjeve i primjere odgovora koje aplikacija vraća.

Summary

Nowadays, information systems have become centers of exchange of new achievements in science. In order for this exchange to be effective, information systems must be connected. One such information system that collects information from the world of science is Open Access Infrastructure for Research in Europe, better known as OpenAIRE. OpenAIRE is the largest such system in Europe, which gathers more than thirty member countries. In the practical part in this paper, we created an application and described the connection between OpenAIRE and the Research Information System of the Republic of Croatia (CroRIS) using the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH).

In first chapter, we talk about open data and its properties. Then we define open data in science and describe the FAIR guidelines for good data management. The FAIR principles are also used to make data easier to find, so we explain in detail each of the four principles: Findability, Accessibility, Interoperability and Reuse.

Second chapter is dedicated to OpenAIRE. To begin with, we present an overview of the development of OpenAIRE from its beginnings until today. After that, we talk about the data that OpenAIRE collects and describe in detail the appearance of each element of the CRIS system that OpenAIRE contains. It is these elements that serve us as guidelines for the design of objects in our application.

In third chapter, we talk about the OAI-PMH protocol on which the operation of our application is based. First, we define the basic terms and their purpose in the protocol. We then describe protocol features such as the appearance of HTTP requests and responses in XML format. At the end of the chapter, we list all requests that the protocol supports and describe the corresponding responses.

In last chapter, we explain in detail the operation of the OAI-PMH-API application we made. First, we list all technologies used, and then we go through the workflow of the application. We describe every step from receiving a request to returning a response. Finally, we show all implemented requests and example of responses that the application returns.

Životopis

Rođen sam 24.prosinca 1997. godine u Virovitici. Prve tri godine osnovne škole sam završio u Osnovnoj školi Ivane Brlić Mažuranić u Virovitici, a ostatak osnovnoškolskog obrazovanja u Osnovnoj školi August Cesarec u Špišić Bukovici. 2016.godine sam završio Prirodoslovno-matematički smjer Gimnazije Petra Preradovića u Virovitici. Tijekom osnovne i srednje škole sam sudjelovao na dva ekipna državna natjecanja i nekoliko regionalnih natjecanja iz matematike te županijskim natjecanjima iz matematike, informatike, kemije, fizike i biologije.

U akademskoj godini 2016./2017. sam upisao Preddiplomski sveučilišni studij matematike na Matematičkom odsjeku Prirodoslovno-matematičkog fakulteta Sveučilišta u Zagrebu. Studij sam završio 2020.godine i iste godine upisao Diplomski sveučilišni studij Računarstva i matematike koji se nadam završiti obranom ovog rada.

Od 2019. godine do danas pomažem u distribuciji i pakiranju materijala za matematičko natjecanje "Klokan bez granica", a od rujna 2021.godine radim kao student u Sveučilišnom računskom centru Sveučilišta u Zagrebu.