Sveučilište u Zagrebu

Fakultet elektrotehnike i računarstva

Otvoreno računarstvo

Skripta za laboratorijske vježbe

Autori:

Branimir Pervan, Emanuel Guberović, Ivana Bosnić, Igor Čavrak

Zagreb, 2021.

Ovo djelo je dano na korištenje pod licencom Creative Commons Imenovanje-Nekomercijalno-Dijeli pod istim uvjetima 4.0 međunarodna.

Sadržaj

1. laboratorijska vježba - Izrada otvorenog skupa podataka

Podaci su sirovi materijali iz kojih se mogu dobiti informacije i znanje. Kada se podacima pridruži kontekst, oni postaju informacije iz kojih se kroz spoznaje stvara određeno znanje. **Otvoreni podaci** su podaci kojima bilo tko može slobodno pristupiti, koristiti ih i dijeliti ih. Otvaranjem podataka stvaramo mogućnost kreiranja novog znanja koje originalni kreator podataka još nije spoznao.

Cilj je ove vježbe upoznavanje s procesom **kreiranja otvorenog skupa podataka** te **njegovog dijeljenja** u obliku javnog *git* repozitorija.

Priprema za vježbu

Ponovite znanje o **modeliranju i izradi jednostavne baze podataka** te proučite spremanje, odnosno izvoz podataka iz baze u formatima **CSV** i **JSON**. Za dijeljenje otvorenog skupa podataka morate se upoznati i s **izradom javnog git repozitorija na sustavu Github.** S obzirom na to da repozitorij morate popuniti i opisom skupa podataka u formatu **markdown**[1], potrebno je upoznati se i s tim jednostavnim oblikom izrade kratkih zapisa.

Proces kreiranja otvorenih podataka[2] uključuje sljedeće korake:

1. Odabir skupa podataka od kojih se kreiraju otvoreni podaci
2. Odabir licencije otvorenih podataka (ako se ne može definirati otvorena licencija vratiti se na 1. korak)
3. Osiguravanje pristupačnosti podataka
4. Osiguravanje da su podaci vidljivi, tj. da se mogu pronaći/otkriti (*discoverable data*).

Da bi otvoreni podaci bili pristupačni (*accessible*), moraju biti u formatu[3]:

- kojega korisnik može razumjeti
- koji je strojno čitljiv
- koji podržava jednostavnu izmjenu podataka
- koji ne zahtijeva korištenje licenciranih alata

Formati prikladni za različite skupove podataka mogu biti različiti. Dokumenti poput PDF-a mogu olakšati razumijevanje podataka, ali nisu strojno čitljivi. Iz tog se razloga pristupačnost često postiže stavljanjem istih podataka u raspon različitih formata (HTML, CSV, JSON, XLSX, PDF...)[4].

Pronalaženje/otkrivanje podataka osigurava se objavljivanjem na portalima otvorenih podataka kao i tražilicama skupova podataka.

Zadatak za vježbu

Za potrebe 1. laboratorijske vježbe trebate napraviti skup podataka kojega je potrebno **spremiti u bazu podataka po vašem odabiru**, pri čemu je potrebno voditi računa o tome da podaci unutar baze moraju biti prikladno normalizirani. Preporučuje se korištenje baze podataka s kojom ste najbolje upoznati, te koja omogućava jednostavno izdvajanje (*export*) podataka u formatima *CSV* i *JSON*. Npr. baza *PostgreSQL* omogućava jednostavno izdvajanje podataka u obliku *CSV* koristeći naredbu *COPY*, dok baza *MongoDB* omogućava jednostavno izdvajanje podataka u oblicima *CSV* i *JSON* koristeći alat

[1] https://www.markdownguide.org/basic-syntax/
[2] https://opendatahandbook.org/guide/en/how-to-open-up-data
[3] https://www.europeandataportal.eu/elearning/en/module9/#/id/co-01
[4] http://opendatahandbook.org/guide/hr/appendices/file-formats

mongoexport. U kontekstu odabira baze podataka, važno je naglasiti da ćete odabranu bazu vjerojatno koristiti do zadnje laboratorijske vježbe, te da bi vam eventualna promjena baze tijekom kolegija mogla uzrokovati nepotrebne latencije prilikom izrade vježbi. Postupak izdvajanja podataka u oblicima CSV i JSON mora biti automatiziran, npr. *shell* naredbama, skriptama ili SQL upitima. Izdvajanje CSV ili JSON datoteka iz grafičkih sučelja za administraciju baza podataka ili korištenje *online* konvertera nije prihvatljivo za ovu svrhu. Prilikom izdvajanja podataka u oblik CSV koji načelno ne dopušta oblikovanje hijerarhije i odnosa podataka, za prikaz odnosa roditelj-dijete možete koristiti ponavljanje podataka roditelja (Primjer 1).

```
// Primjer 1
// JSON
{
      "Naziv":"Genijalni um",
      "Glumci":[
            { "Ime":"Russell", "Prezime":"Crowe" },
            { "Ime":"Jennifer", "Prezime":"Connelly" }
      ]
}
// CSV
Genijalni um, Russel, Crowe
Genijalni um, Jennifer, Connelly
```

Otvaranjem svog skupa podataka izradit ćete otvorene podatke koje trebate izdvojiti u datoteke **naziv_skupa.csv** i **naziv_skupa.json.**

U datoteci naziva ***README.md*** u *markdown* formatu u nekoliko rečenica kratko opišite skup podataka te dodajte barem 10 metapodataka koji moraju sadržavati:

- Opis otvorene licencije, npr. Creative Commons uz specificiranje podvrste
- Naziv autora: Vaše ime i prezime
- Verzija skupa podataka (npr. 1.0 za 1. laboratorijsku vježbu)
- Jezik u kojemu se nalaze podaci (npr. hrvatski, engleski, ...)
- Opis atributa koji se nalaze u skupu podataka (oni odgovaraju stupcima u *CSV* datotaci ili ključevima u *JSON* datoteci)
- Bilo koji drugi podaci koji mogu donijeti koristan kontekst skupu podataka. Za inspiraciju može poslužiti primjer otvorenih podataka autobusnih stanica.

Napravite **javni repozitorij u sustavu *Github***, u kojega ćete postaviti svoj otvoreni skup podataka u formatima *JSON* i *CSV*, zajedno s opisom skupa u formatu *markdown*. Repozitoriju je potrebno ispuniti metapodatke poput opisa repozitorija i licencije konzistentne s opisom skupa podataka.

Sadržaj skupa podataka odaberite proizvoljno. Neki dobri primjeri su:

- **kataloški skupovi podataka**: podaci koji sadrže karakteristike različitih entiteta (npr. skupovi koji opisuju listu filmova, knjiga, sportaša, itd.)

- **podaci vremenskog niza**: podaci koji zapisuju određenu promjenu u vremenu (npr. skupovi koji zapisuju promjenu temperature u vremenu, broja osoba zaraženih COVID-19 po danu, itd.)
- **geoprostorni podaci**: skupovi koji sadrže katalog podataka specificiranih svojom prostornom lokacijom (npr. lokacija kanti za smeće, solarnih klupa, nogometnih stadiona, itd.)

Skup podataka kojeg kreirate mora imati:

- **barem 10 instanci podataka** (10 redaka u *CSV*-datoteci, 10 objekata u *JSON*-polju)
- **barem 10 atributa** (10 stupaca u *CSV*-datoteci, 10 ključeva u *JSON*-objektima)
- skup podataka mora imati **barem jednu roditelj-dijete vezu između entiteta** (npr. skup opisa filmova može sadržati vezu roditelj-dijete između filmova i glumaca, jer jedan film može imati više glumaca)

Skup podataka koji se nalazi u formatu JSON i koji zadovoljava sve uvjete ilustriran je Primjerom 2.

```
// Primjer 2
[...,
      {
            "Naziv":"Genijalni um",
            "Zemlja":"Sjedinjene Američke Države",
            "Prosječna_ocjena_(TMDB)":79,
            "Godina":2001,
            "Trajanje_(min)":135,
            "Žanr":["Drama", "Biografija"],
            "Redatelj":{"Ime":"Ron", "Prezime":"Howard"},
            "Glumci":[
                  { "Ime":"Russell", "Prezime":"Crowe" },
                  { "Ime":"Jennifer", "Prezime":"Connelly" }
            ]
      },
...]
```

Predaja vježbe

Za uspješnu predaju 1. laboratorijske vježbe potrebno je na sustav *Moodle* postaviti poveznicu na javno dostupan *Github* repozitorij čija oznaka *Release 1.0* sadrži:

- ***README.md, LICENCE***
- ***naziv_skupa.json, naziv_skupa.csv***
- ***dump baze podataka s podacima***

Ispitno gradivo vježbe

- Koje podatke sadrži vaš skup podataka i pod kojom licencijom?
- Koju bazu podataka koristite za spremanje vašeg skupa podataka i zašto?
- Kako ste modelirali vašu bazu podataka?
- Kako iz baze podataka generirate skup podataka u formatima *CSV* i *JSON*?
- Koja su prednosti i mane spremanja podataka u formatima *CSV* i *JSON*?
- Kako ste omogućili da vaši podaci budu pristupačni?

Poveznice i iteratura

- Open Data Handbook - https://opendatahandbook.org/
- Discovering Open Data - https://www.europeandataportal.eu/elearning/en/#/id/co-01
- PostgreSQL Tutorial - https://www.postgresqltutorial.com/
- MongoDB Tutorial - https://docs.mongodb.com/manual/tutorial/
- Difference Between JSON and CSV - https://www.geeksforgeeks.org/difference-between-json-and-csv/

2. laboratorijska vježba - Pristupačnost i vidljivost otvorenih podataka

Za pristupačnost podataka i vidljivost podataka, korisno je kreirati web grafičko sučelje u kojemu će biti moguće pristupiti različitim verzijama skupova podataka te njihovim metapodacima u formatima čitljivim ljudima i strojevima.

Cilj ove vježbe je proširenje znanja u procesu izrade otvorenog skupa podataka s naglaskom na **povećanje pristupačnosti i vidljivosti** kroz izradu tabličnog *HTML* web sučelja uz obrazac za filtriranje zapisa, te strukturiranog strojno čitljivog oblika metapodataka u obliku *JSON Schema*.

Priprema za vježbu

S ciljem pripreme za vježbu potrebno je ponoviti znanje o **izradi web sučelja s naglaskom na izradu *HTML/CSS*-obrazaca i podatkovnih tablica čiji se podaci dohvaćaju asinkrono (AJAX, bez osvježavanja stranice)** kao i o **procesu izrade pozadinskog poslužiteljskog sučelja koje filtrira sadržaj baze podataka** kroz upite nad bazom podataka. Potrebno je proučiti specifikaciju ***JSON Schema*** za anotaciju i validiranje dokumenata u formatu JSON.

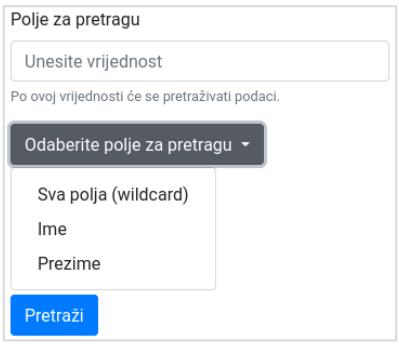
AJAX (*Asynchronous JavaScript and XML*) podrazumijeva bilo koji asinkroni način dohvaćanja sadržaja s udaljenog poslužitelja bez osvježavanja stranice, npr. *XMLHttpRequest*, *fetch()*, *jQuery.ajax()* itd. Radni okviri za razvoj *front-end* aplikacija najčešće imaju vlastite komponente za asinkrono dohvaćanje sadržaja. Ako ste odlučili za razvoj aplikacije u sklopu ove vježbe koristiti neki od takvih radnih okvira, onda se preporučuje iskoristiti njihove ugrađene mehanizme za asinkrono dohvaćanje. Iako svojim opsegom mogućnosti **znatno izlaze** iz okvira potreba ove laboratorijske vježbe pa ih radi toga ne preporučujemo, dozvoljeno je koristiti i radne okvire (*frameworks*) za razvoj *front-end* aplikacija, npr. *Angular* ili *React*.

Zadatak za vježbu

Za potrebe 2. laboratorijske vježbe potrebno je osigurati dostupnost podataka u strojno čitljivom obliku tako što ćete u web sučelje, **u datoteci index.html,** staviti **poveznicu na verzije podataka u oblicima *CSV* i *JSON* koristeći** *HTML hyperlink* **oznake**. Poveznice poslužuju podatke u oblicima CSV i JSON s datotečnog sustava.

Za prikaz podataka u ljudima čitljivom formatu potrebno je podatke prikazati u obliku **HTML tablice koja se zajedno s obrascem za filtriranje i asinkronim pozivom (AJAX) za dohvaćanje podataka nalazi u datoteci datatable.html.** Za izvedbu tabličnog prikaza podataka možete koristiti standardnu HTML-tablicu koju ćete ručno popuniti iz HTTP-odgovora dobivenog asinkronim pozivom vašega servisa ili neku od predefiniranih podatkovnih tablica ostvarenih kao komponente (npr. *data table*[5]). Podatke prikazane u tablici potrebno je dohvatiti iz baze podataka, a čitanje ostvariti korištenjem pozadinskog poslužiteljskog jezika po vašem izboru. Prikaz podataka dohvaćanjem iz prethodno stvorenog *dumpa* baze u obliku CSV ili JSON datoteka s datotečnog sustava nije dozvoljen.

[5] https://www.datatables.net/



Pomoću prethodno opisane tablice i jednostavnog HTML-obrasca potrebno je omogućiti obavljanje **osnovnog filtriranja po vrijednostima podataka**. Jednostavni HTML-obrazac mora sadržavati polje za unos teksta i element (ili elemente) za odabir atributa po kojem se pretražuje. Zadano pretraživanje podataka podrazumijeva pretragu unesenog teksta po svim atributima (tzv. *wildcard*), a pretraga se može suziti odabirom atributa po kojem se pretražuje. Primjerice, ako vaši podaci sadržavaju ime i prezime osoba, upisivanjem u polje za unos teksta zadano će se pretraživati podaci i po imenu i po prezimenu, a ako u padajućem izborniku odaberete jedan od ta dva atributa, onda će se podaci pretraživati samo po tom atributu.

Filtrirane podatke koji se prikazuju u podatkovnoj tablici potrebno je moći preuzeti u oblicima CSV i JSON. Filtriranje i ostvarivanje mogućnosti preuzimanja trenutnog prikaza u oblicima CSV i JSON možete ostvariti i na poslužitelju (*back-end*) i na klijentu (*front-end*). Tehnologija kojima ćete ostvariti zahtjeve nije zadana, ali se svakako preporučuje korištenje one tehnologije s kojom ste najbolje upoznati.

Kako bi metapodaci bili čitljivi ljudima, dovoljno je prenijeti jednostavan i pristupačan **opis sadržaja datoteke *README.md* iz prethodne vježbe u datoteci index.html**. Pri tome pripazite da metapodaci koji se nalaze u <*head*> dijelu web-stranice, poput vrijednosti “*title*”, "*description*” i “*author*”, odgovaraju vrijednostima u opisu skupa podataka.

Kako bi metapodaci skupa podataka bili strojno čitljivi potrebno je koristiti dobro definiran format. Za potrebe 2. laboratorijske vježbe, skup podataka u formatu JSON obogaćuje se **metapodacima u formatu JSON Schema** u datoteci *schema.json,* koji mora imati ključne riječi:

- *$id*: identificira *JSON* resurs s njegovim URI-em
- $schema: identifikator *JSON Schema* seta karakteristika
- *title, description* i *type* anotacije za naziv i opis instanci objekata i njegovih atributa
- *properties* i *required* za opis atributa i njihove nužnosti postojanja

Za pomoć pri generiranju JSON scheme možete koristiti online generator poput jsonschema.net. Isječak ispravnog primjera *JSON scheme* za primjer podataka u formatu JSON iz uputa za 1. laboratorijsku vježbu je:

```
// Primjer 1
{
	"$schema": "http://json-schema.org/draft-07/schema",
	"$id": "https://or.fer.unizg.hr/orfilmovi.json",
	"type": "array",
	"title": "OR Filmovi",
	"description": "Skup filmova za otvoreno računarstvo.",
	"items": [
		"type": "object",
		"title": "Film",
		"description": "Objekt koji sadrži sve podatke o jednom filmu.",
		"required": [ "Naziv filma", "Wikipedia Handle" ],
		"properties": {
			"Naziv filma": {
				"type": "string",
				"title": "Naziv filma",
				"description": "Naziv filma."
			},
			...
```

Predaja vježbe

Za polaganje 2. laboratorijske vježbe potrebno je predati na pregled poveznicu na javno dostupan github repozitorij čiji Release 2.0 sadrži:

- ***README.md, LICENCE***
- ***naziv_skupa.json, naziv_skupa.csv, schema.json***
- ***index.html, datatable.html***
- ***dump baze podataka s podacima***
- ***pozadinski (back-end) kôd***
- ***klijentski (front-end) kôd***

S obzirom na to da izvedba filtriranja i preuzimanja filtriranih podataka nije strogo ograničena na poslužiteljsku ili klijentsku stranu, potrebno je u sklopu *Release 2.0* predati i sav kôd koji realizira te funkcionalnosti. Datoteka *index.html* mora sadržavati poveznice za preuzimanje cjelokupnog skupa podataka u CSV i JSON obliku (poslužuju se s datotečnog sustava), a datoteka *datatable.html* mora sadržavati obrazac za filtriranje, prikaz filtriranih podataka te poveznice za preuzimanje filtriranog skupa podataka (aktualni prikaz filtera), također u oblicima CSV i JSON.

Ispitno gradivo vježbe

- Kako ste omogućili pristupačnost podataka i metapodataka u ljudskom i strojnom obliku?
- Kako ste napravili HTML podatkovnu tablicu i obrazac za filtriranje?
- Kako i u kojoj tehnologiji ste izveli filtriranje podataka?
- Kako i zašto se koristi *JSON Schema*?

Poveznice i literatura

- Data on Web Best Practices - https://www.w3.org/TR/dwbp/
- HTML Tables - https://www.w3schools.com/html/html_tables.asp
- HTML Forms - https://www.w3schools.com/html/html_forms.asp
- NodeJS MongoDB - https://www.w3schools.com/nodejs/nodejs_mongodb.asp
- Node Postgres - https://node-postgres.com/
- Understanding JSON Schema - https://json-schema.org/understanding-json-schema/index.html
- JSON Schema Tool - https://www.jsonschema.net/

3. laboratorijska vježba – REST API

API (*Application Programming Interface*) je posrednička komponenta programske podrške koja omogućuje interakciju s drugim (udaljenim) komponentama programske podrške. API-ji pojedinih aplikacija apstrahiraju implementaciju i komponente izlažući klijentskoj aplikaciji sučelje za upravljanje podacima. Danas API-ji obično slijede arhitekturalni stil REST (*Representational State Transfer*) za izgradnju raspodijeljenih sustava.

Priprema za vježbu

Proučite arhitekturalni obrazac REST te specifikaciju *OpenAPI*. S obzirom na to da laboratorijsku vježbu možete implementirati korištenjem programskog jezika (i evenutalno radnog okvira) po želji, proučite dokumentaciju vezanu za odabrane tehnologije. Prilikom izrade ove laboratorijske vježbe mogu vam pomoći i sljedeće dobre prakse (tzv. pravila palca):

- Iskoristite protokol HTTP i sve ono što on nudi
    - Koristite HTTP-metode (glagole) za operacije nad resursima

Ako želite nad nekim resursom (kolekcijom) osigurati CRUD (*Create, Read, Update, Delete*), obično vrijedi sljedeće mapiranje metoda HTTP-a na operacije nad podacima:

Operacija	**HTTP metoda**	**Opis**
Create	POST	Obično se poziva nad kolekcijom, tijelo zahtjeva sadržava novi objekt bez jedinstvenog identifikatora koji je potrebno ubaciti u kolekciju. HTTP-odgovor sadržava isti objekt zajedno s kreiranim identifikatorom resursa.
Read	GET	HTTP-zahtjev sadržava jedinstveni identifikator resursa kao *Path parameter* (http://{url}/api/v1/departments/{ID}). HTTP-odgovor sadržava objekt dohvaćen korištenjem jedinstvenog identifikatora resursa.
Update	PUT	Poziva se nad resursom, tijelo zahtjeva sadržava objekt s jedinstvenim identifikatorom resursa i poljima čije je vrijednosti potrebno osvježiti. Jedinstveni identifikator nalazi se u URL-u kao *Path parameter* (http://{url}/api/v1/departments/{ID}), a može biti sadržan i u tijelu zahtjeva.
Delete	DELETE	Poziva se nad resursom, jedinstveni identifikator resursa koji se briše sadržan je u URL-u kao *Path parameter (*http://{url}/api/v1/departments/{ID})

Iako je tehnički izvedivo, nije prihvatljivo npr. koristiti metodu GET za brisanje podataka iz kolekcije.

- Postavljajte HTTP kôd odgovora (*response codes*). Kôdove je potrebno ispravno koristiti, npr. u slučaju da je klijent postavio zahtjev za neki resurs:
    - ako resurs postoji, treba ga vratiti klijentu uz kôd *200 OK*
    - ako resurs ne postoji, vratiti klijentu poruku s pogreškom uz postavljanje kôda *404 Not Found*
- Postavljajte HTTP-zaglavlje *Content-Type* na odgovarajući tip podataka u tijelu odgovora

- Sve odgovore omotajte (*wrap*) u objekt *Response* s porukama čitljivima i za čovjeka i za stroj (**Error! Reference source not found.**)

```
// Primjer 1.
// GET https://api.local/v2/departments/1

// Content-Type: application/json
// 200 OK
{
    "status": "OK",
    "message": "Fetched department object",
    "reponse": {
        "id": 1,
        "name": "Department Name",
        "links": [
            {
                "href": "1/employees",
                "rel": "employees",
                "type": "GET"
            }
        ]
    }
}
```

- U slučaju pogreške ili iznimke (*Exception*) generirajte odgovarajući povratni objekt s opisom. Na isti način dobro je obraditi i nepostojeće krajnje točke (*endpoint*), neprimjenjive HTTP-metode ili nepostojeće resurse (**Error! Reference source not found.**).

```
// Primjer 2
// GET https://api.local/v2/departments/1001
// Resurs s identifikatorom 1001 ne postoji

// Content-Type: application/json
// 404 Not Found
{
    "status": "Not Found",
    "message": "Department with the provided ID doesn't exist",
    "reponse": null
}

// PATCH https://api.local/v2/departments/1001/groups
// Poslužitelj nije implementirao metodu PATCH za resurs

// Content-Type: application/json
// 501 Not Implemented
{
    "status": "Not Implemented",
    "message": "Method not implemented for requested resource",
    "reponse": null
}
```

Zadatak za vježbu

API

U sklopu ove laboratorijske vježbe, skup otvorenih podataka iz prijašnjih laboratorijskih vježbi potrebno je izložiti kroz RESTful API. Tehnologija i programski jezik u kojem ćete izvesti API je proizvoljan. API mora zadovoljiti nekoliko uvjeta:

1. API **mora izložiti sljedeće krajnje točke** čije HTTP-metode moraju odgovarati prikladnim akcijama nad podacima (CRUD):
    a. **jednu GET** krajnju točku za dohvaćanje **cjelokupne kolekcije** vaših podataka.
    b. **jednu GET** krajnju točku za dohvaćanje **pojedinačnog resursa** iz kolekcije temeljem jedinstvenog identifikatora resursa.
    c. barem **tri dodatne GET** krajnje točke po vlastitom odabiru.
    d. **jednu POST** krajnju točku za ubacivanje pojedinačnog resursa u kolekciju.
    e. **jednu PUT** krajnju točku kojom će se osvježiti elementi resursa.
    f. **jednu DELETE** krajnju točku za brisanje pojedinog resursa iz kolekcije temeljem jedinstvenog identifikatora resursa.
2. Svaki odgovor mora biti omotan u *Response* omotač (*wrapper*). Oblik omotača možete odabrati sami.
3. Aplikacija mora biti **otporna na pogreške** i na odgovarajući način vraćati poruke o eventualnim iznimkama ili pogreškama. Aplikacija **ne smije prestati s radom** u slučaju pogreške ili prikazati zadane poruke o pogreškama odabranih radnih okvira.

Nazive krajnjih točaka, kolekcija, resursa odaberite samostalno u skladu s tematikom vaše laboratorijske vježbe. API-sučelje izvršava operacije **nad podacima u bazi podataka**.

Dokumentiranje uporabom specifikacije OpenAPI

Specifikacija *OpenAPI* definira standard koji olakšava čitljivost mogućnosti API-ja, pritom definirajući sučelje za strojnu čitljivost. Dobro dokumentiran API korištenjem ove specifikacije olakšava korisniku API-ja razumijevanje i interakciju s API-jem bez implementacije čime se bitno može skratiti vrijeme razvoja programske podrške koja koristi uslugu izloženu API-jem. Specifikacija se može primijeniti neovisno o tehnologiji implementacije (*language-agnostic*), a sama specifikacija koja opisuje API je JSON (ili YAML) objekt, odnosno dokument.

Vršni objekt u datoteci sa specifikacijom *OpenAPI* neke usluge je *OpenAPI* objekt s tri obavezna polja (*field*):

1. *openapi* (string) – Verzija specifikacije *OpenAPI* koja se koristi
2. *info* (Info Object) – Metapodaci o usluzi koja se opisuje specifikacijom
3. *paths* (Paths Object) – Objekt koji opisuje sve putanje i njihove operacije nad podacima za API koji opisuje

S obzirom na to da su *info* i *paths* objekti, tj. kompleksni tipovi podataka, dokumentacijom specifikacije određena su njihova obavezna polja.

U sklopu ove laboratorijske vježbe potrebno je **dokumentirati ostvareni API po specifikaciji *OpenAPI***. Vršni objekt mora sadržavati **obavezna polja *openapi, info* i *paths***. Objekt *info* mora sadržavati obavezna polja i dodatno **podobjekte *contact* i *licence***, a objekt *paths* mora sadržavati **podobjekt (*path item object*) za svaku krajnju točku koju API izlaže**. *Path item* object sadržava podobjekt za svaku HTTP-metodu dostupnu nad resursom (*Operation Object*). Obratite pozornost **na obavezna polja koja objekt *paths* mora sadržavati** za svaku krajnju točku, a za više informacija konzultirajte samu specifikaciju dostupnu na poveznici u zadnjem poglavlju ove vježbe. Primjer implementacije vršnog *OpenAPI* objekta za neku uslugu dan je u Primjeru 3. Dodatan primjer specifikacije dostupan je u poveznicama. Specifikaciju je potrebno spremiti u datoteku **openapi.json** i predati kao dio repozitorija. Dodatno, API mora izložiti **krajnju točku** (HTTP-metoda GET) kojom će na zahtjev poslužiti specifikaciju.

```
// Primjer 3
{
    "openapi": "3.0.3",
    "info": {
        "info": "Moja usluga",
        "version": "1.2"
    },
    "paths": {
        "/kolekcija/{id}": {
            "get": {
                "summary": "Get resource from collection by ID",
                "description": "Returns a single collection resource",
                "produces": [
                    "application/json"
                ],
                "parameters": [
                    {
                        "name": "id",
                        "in": "path",
                        "description": "ID of the resource",
                        "required": true,
                        "type": "integer",
                        "format": "int64"
                    }
                ],
                "responses": {
                    "200": {
                        "description": "Fetch successfull",
                        "schema": {
                            "$ref": "#/definitions/Resource"
                        }
                    },
                    "400": {
                        "description": "Invalid ID"
                    },
                    "404": {
                        "description": "Resource not found"
                    }
                }
            },
            "post": {...},
            "delete": {...}
        }
    }
}
```

Predaja vježbe

Rezultat vježbe je REST sučelje koje poštuje navedene zahtjeve i navedene dobre prakse prilikom oblikovanja sučelja, zajedno sa dostupnom dokumentacijom po specifikaciji *OpenAPI*. Za uspješnu predaju 3. laboratorijske vježbe potrebno je predati poveznicu na repozitorij GitHub s oznakom Release 3.0 koji sadrži:

- README.md, LICENCE
- naziv_skupa.json, naziv_skupa.csv, shema_skupa.json
- index.html, datatable.html
- dump baze s podacima
- pozadinski kôd koji ostvaruje funkcionalnost API-ja
- openapi.json

Ispitno gradivo vježbe

- Koji ste programski jezik i radni okvir odabrali za implementaciju rješenja i zašto?
- Koje HTTP-metode bi se dodatno mogle iskoristiti prilikom oblikovanja API-ja?
- Koje su prednosti dokumentiranja API-ja korištenjem specifikacije *OpenAPI*?
- Kojim operacijama nad resursom odgovaraju pojedine HTTP-metode?

Poveznice i literatura

- JSON-LD APIs: Best Practices: https://json-ld.org/spec/latest/json-ld-api-best-practices/
- OpenAPI specifikacija: https://swagger.io/specification/
- The OpenAPI Specification (sekcija *Examples*): https://github.com/OAI/OpenAPI-Specification/blob/master/README.md
- Swagger Editor: https://editor.swagger.io/
- Primjer dokumentacije generirane iz specifikacije: https://petstore.swagger.io/

4. laboratorijska vježba – Otvorena autentifikacija i dodavanje semantike skupu podataka

Cilj izlaganja API-ja kao posredničke komponente programske podrške, interakcija je s drugim udaljenim komponentama. Te komponente mogu biti klijentske aplikacije koje se izvršavaju u *web* preglednicima, na mobilnim uređajima korisnika, ali i druge udaljene komponente koje su sakrivene od krajnjih korisnika te koje izlažu vlastite API-je. Neke od tih komponenti mogu biti i usluge za autentifikaciju korisnika te autorizaciju pristupa uslugama, poznatije kao *Single sign-on* usluge. Takve usluge omogućavaju korisnicima koji na njima već imaju korisničke račune, da delegiraju vlastite osobne podatke te podatke o pravima pristupa drugim aplikacijama. Na taj se način korisnicima omogućava pristup aplikacijama bez potrebe za otvaranjem korisničkih računa na tim aplikacijama. Prednosti *Single sign-on* paradigme su jednostavnost korištenja za korisnika koji može jednostavno ponovno iskoristiti neki od postojećih računa za prijavu te povećana razina sigurnosti jer korisnik ne ostavlja svoju lozinku još jednoj usluzi za koju nije siguran čuva li je na ispravan način. S negativne strane, na ovaj način stvara se jedna kritična točka kvara sustava. Ako u nekom vremenskom trenutku usluga koja pruža autentifikaciju i autorizaciju padne, korisnici pojedinih aplikacija privremeno se neće moći prijaviti u te aplikacije. Primjeri pružatelja *Single sign-on* usluga su npr. Google, Facebook, Apple, Microsoft i dr. U sklopu ove laboratorijske vježbe, cilj je **integrirati vanjsku uslugu autentifikacije u vašu web-aplikaciju** korištenjem sloja *OpenID Connect* nad protokolom *OAuth 2.0* (RFC6749).

Uz integraciju s vanjskom uslugom za autentifikaciju, cilj vježbe je i **dopuna vlastitog skupa podataka semantičkim povezivanjem s drugim vrstama podataka korištenjem specifikacije JSON-LD**.

Priprema za vježbu

OAuth 2.0 i OpenID Connect

Ponovite arhitekturalni obrazac REST iz prošle laboratorijske vježbe te proučite specifikacije *OAuth 2.0* i *OpenID Connect*. Također, proučite mehanizme koje nudi vaš odabrani programski jezik, odnosno radni okvir, za integraciju *Single sign-on* usluge u web-aplikaciju.

Protokol *OAuth 2.0* primarno je namijenjen autorizaciji pristupa, tj. utvrđivanju prava na pristup određenim resursima. Protokol omogućava korisniku da dozvoli drugim aplikacijama ili uslugama pristup tim zaštićenim resursima, bez otkrivanja vlastitih korisničkih podataka. Sam protokol definira četiri uloge (*roles*):

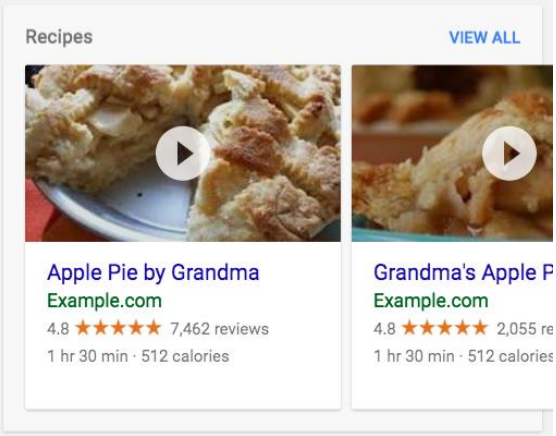
- Vlasnik resursa – vlasnik resursa ili korisnik koji ima pravo dozvoliti daljnji pristup resursu;
- Poslužitelj resursa – poslužitelj na kojemu se nalazi zaštićeni resurs;
- Klijent – aplikacija koja u ime krajnjeg korisnika traži pristup zaštićenom resursu;
- Autorizacijski poslužitelj – poslužitelj koji autentificira korisnika, utvrđuje razinu prava te u ovisnosti o identitetu i pravima dozvoljava ili brani pristup zaštićenom resursu.

Uloge nisu strogo odijeljene te se u praksi može naići na scenarij u kojem su autorizacijski poslužitelj i poslužitelj resursa smješteni unutar jednog fizičkog poslužitelja ili jedne poslužiteljske aplikacije.

Prilikom iniciranja zahtjeva za autorizacijom korisnika, klijentska aplikacija autorizacijskom poslužitelju, dojavit će i opseg autorizacije (eng. *scope*), primjerice *contact_data* ako se radi o zapisu u imeniku ili *account_balance_data* ako se radi o aplikaciji koja prikazuje bankovne podatke korisnika. Autorizacija klijentske aplikacije odnosi se samo na one podatke koji su navedeni u zahtijevanom opsegu.

Jedan od mogućih (standardiziranih) opsega jest i *openid*, kojim se zahtjeva pristup osobnim podacima korisnika te time inicira autentifikacija po standardu *OpenID Connect*. Uz *openid* u opsegu zahtjeva, nije neuobičajeno da klijentska aplikacija neke dodatne podatke korisnika zatraži i eksplicitno, npr. *email*.

JSON-LD



JSON-LD (*JavaScript Object Notation – Linked Data*) je specifikacija čijim se korištenjem podaci bez značenja stavljaju u kontekst te im se pridjeljuje semantika. Dodjeljivanje semantike podacima koji se koriste na *web*-sjedištima i mreži općenito korisno je iz više aspekata, a obično se koristi u okviru web-stranica čime se tražilicama omogućava da semantički klasificiraju informacije i time poboljšaju iskustvo korisnika prilikom pretrage, ponude dodatne sadržaje, automatizirano se povežu s drugim izvorima podataka i sl. Primjer 4 ilustrira pridavanje semantičkog konteksta sadržaju korištenjem specifikacije JSON-LD. Korijenski objekt kao svoj kontekst koristi rječnik *Schema.org* (ključna riječ *@context*), a sam korijenski objekt je tipa *Recipe* (deklarirano ključnom riječi *@type*) iz rječnika Schema.org. Korijenski objekt sadrži podobjekt *author* koji označava autora recepta, a podobjekt *author* je tipa *Person*, također iz rječnika *Schema.org* (ponovno deklarirano ključnom riječi

```
// Primjer 4
<html>
  <head>
    <title>Party Coffee Cake</title>
    <script type="application/ld+json">
    {
      "@context": "https://schema.org/",
      "@type": "Recipe",
      "name": "Party Coffee Cake",
      "author": {
        "@type": "Person",
        "name": "Mary Stone"
      },
      "datePublished": "2018-03-10",
      "description": "This coffee cake is awesome and perfect for
                      parties.",
      "prepTime": "PT20M"
    }
    </script>
  </head>
  <body>
  <h2>Party coffee cake recipe</h2>
  <p>
    This coffee cake is awesome and perfect for parties.
  </p>
  </body>
</html>
```

@type). Primjer je preuzet sa stranice *Understand how structured data works* (poveznica se nalazi u literaturi).

U prethodnom primjeru kontekst je definiran rječnikom *Schema.org*. Kontekst se može prilagoditi ručnim navođenjem URI-ja do opisnika pojedinih tipova podataka.

```
// Primjer 5
// Company / Departments / Employees
{
    "@context": {
        "head": "https://schema.org/Person",
        "name": "https://schema.org/name",
        "location": "https://schema.org/GeoCoordinates"
    },
    ...
    "id": 1,
    "name": "Department 1",
    "head": {
        "familyName": "Perić",
        "givenName": "Pero"
    },
    "location": {
        "locationName": 123,
        "latitude": 123,
        "longitude": 123,
    },
    ...
}
```

U Primjeru 5, za skup podataka koji opisuje hijerarhiju neke tvrtke, atribut naziva *head* opisuje voditelja odjela, a odgovarat će objektu tipa *Person* iz rječnika *Schema.org*. Slično vrijedi i za atribute naziva *name* i *location* koji opisuje naziv i lokaciju navedenog odjela. Iako je u ovom slučaju kontekst prilagođen pojedinačnim navođenjem URI-ja, s obzirom na to da su kao opisnici tipova podataka korišteni elementi rječnika *Schema.org*, kontekst je jednostavno mogao biti definiran kao:

```
"@schema": "https://schema.org/"
```

Kako rječnik *Schema.org* koristi unaprijed definirane nazive atributa (*fields*) za semantičke objekte, moguće je da će atributi iz rječnika odudarati od naziva atributa za vaše resurse u kolekciji. Primjerice, ako vaša kolekcija sadržava osobe, prirodno je upotrijebiti objekt tipa *Person* iz rječnika *Schema.org*. U slučaju da ste svoje atribute kojima opisujete osobu unutar objekta nazvali različito od atributa objekta *Person* iz rječnika Schema.org, npr. (*first_name* umjesto *givenName*, *last_name* umjesto *familyName*), specifikacijom JSON-LD možete prikazati da su atributi ekvivalentni (primjer 6). Ključna

```
// Primjer 6
{
  "@context": {
    "@vocab": "http://schema.org/",
    "first_name": "givenName",
    "last_name": "familyName"
  },
  "first_name": "Pero",
  "last_name": "Perić"
}
```

riječ *@vocab* označava zadani rječnik iz kojeg se izvode ostali termini. Atributi *first_name* i *last_name* imat će semantičko značenje rječničkih tipova *givenName* i *familyName*.

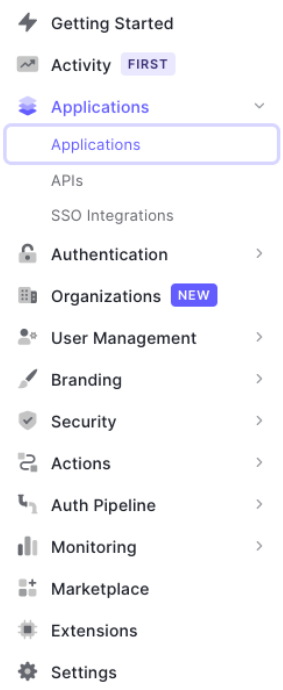
Kao pripremu za vježbu, proučite detaljno standard *JSON-LD* te se upoznajte s rječnikom *Schema.org*. Za sve navedeno, poveznice su dostupne u literaturi.

Zadatak za vježbu

Otvorena autentifikacija

Zadatak ovog dijela vježbe jest integrirati vlastitu web-aplikaciju s uslugom *Auth0* za *Single sign-on*. Za početak, potrebno je otvoriti korisnički račun na usluzi *auth0*[6]. Usluga je besplatna, a u slučaju potrebe za odabirom razine usluge, dovoljno je odabrati osnovnu, odnosno „*Starter*“ razinu.

Nakon stvaranja korisničkog računa, potrebno je vlastitu web-aplikaciju registrirati, tj. prijaviti usluzi *Auth0*. U upravljačkoj ploči, u izborniku s lijeve strane odaberite *„Applications“* iz padajućeg izbornika također pod naslovom *„Applications“*. Unesite željeni naziv vaše aplikacije, odaberite njezinu vrstu te potvrdite odabir. U postavkama aplikacije koje će uslijediti, na podkartici *Settings*, nalaze se podaci koji su vam potrebni za implementaciju protokola *OAuth 2.0*, a to su prije svega *Client ID* i *Client Secret*, te tekstualni okviri za unos dozvoljenih poveznica *Callback URL-ova* i *Logout URL-ova*. Dodatno, potrebno je stvoriti korisnika, odnosno korisnički račun, čija će se kombinacija korisničkog imena i lozinke koristiti za ispitivanje rada prijave. U izborniku s lijeve strane odaberite stavku *„User Management“*, a nakon nje *„Users“*. Stvaranje korisnika vrši se pritiskom na gumb *„Create“*. Taj korisnik postojat će samo u kontekstu vaše aplikacije, a kombinacija korisničkog imena i lozinke koristit će se samo za ispitivanje integracije. Kako bismo olakšali ispitivanje na laboratorijskoj vježbi, stvorite korisnika sa sljedećim podacima:




```
Email: or@or.hr
Password: Password1!
Connection: Username-Password-Authentication
```

Daljnja integracija ovisi o programskom jeziku i radnom okviru kojega koristite. Integraciju je moguće provesti ručno, ali je i moguće provjeriti postoji li za vaš odabrani jezik dostupan *plug-in* ili SDK što će bitno olakšati integraciju. Svako valjano rješenje koje uspješno integrira *Single sign-on* uslugu bit će prihvaćeno. Sve što nije definirano, slobodno možete odabrati sami.

Kako bi integracija u kontekstu vaših dosadašnjih aplikacija bila vidljiva, potrebno je napraviti zaštićeni dio aplikacije za dohvaćanje profila prijavljenog korisnika. Na web stranici, dodajte poveznicu s tekstom *„Prijava“*. Pritisak na poveznicu, korisnika vodi na okvir za prijavu usluge *Auth0*, a nakon upisivanja gore navedenih podataka, korisnik se vraća na vašu web stranicu kojoj su sada u izborniku dostupne poveznica *„Korisnički profil“* i *„Osvježi preslike“*. Pritiskom na poveznicu *„Korisnički profil“*, otvara se podstranica na kojoj su navedeni podaci o korisniku dohvaćeni iz *openid* objekta dok se

[6] https://auth0.com/

pritiskom na poveznicu *„Osvježi preslike“* programski izvodi izvoz podataka iz vaše baze podataka u oblicima CSV i JSON te se sprema na datotečni sustav. Sva buduća posluživanja tih dviju datoteka s datotečnog sustava moraju koristiti nove verzije podataka.

Eventualan izravan odlazak na URL podstranice s korisničkim profilom ili pozivanje osvježavanja preslike, a bez prethodne prijave korisnika, mora rezultirati pogreškom. Uz poveznicu za pregled korisničkog profila, u izbornik dodajte poveznicu s natpisom *„Odjava“* kojom se korisnik odjavljuje s vaše aplikacije, ali ne nužno i sa servisa *Auth0*. U tom slučaju naknadni pokušaj prijave rezultirat će prijavom u aplikaciju bez potrebe za upisivanjem korisničkog imena i lozinke na servisu *Auth0*, radi postojeće sesije.

S obzirom na sve navedeno, okvirni hodogram prijave korisnika na vašoj web stranici jest:

1. Korisnik pritišće poveznicu s natpisom „Prijava“
2. Vaša web stranica, odnosno klijentska aplikacija, preusmjerava korisnika na okvir za prijavu usluge Auth0
3. Korisnik upisuje korisničko ime i lozinku
4. Usluga Auth0 vraća korisnika na klijentsku aplikaciju
5. Klijentska aplikacija dodaje poveznice za pregled korisničkog profila, osvježavanje preslika baze i odjavu

Dodavanje semantike korištenjem specifikacije JSON-LD

U ovom dijelu vježbe, potrebno je semantički opisati barem dva atributa pojedinačnog resursa iz vašeg skupa podataka. Izmijenite JSON objekt kojega vraća vaš API tako da sadrži potrebna polja kojima će se odabranim atributima pridijeliti semantika. Za pregled dostupnih semantičkih tipova podataka koristite rječnik *Schema.org*. Za provjeru valjanosti strukturiranih podataka korištenjem specifikacije JSON-LD, možete se poslužiti službenim validatorom *Schema.org*. Poveznica je dostupna u literaturi.

Predaja vježbe

Rezultat vježbe jest web-aplikacija integrirana s uslugom *auth0* za *Single sign-on*, dopunjen API i web sučelje mogućnošću prikaza elementarnog profila prijavljenog korisnika, te semantička dopuna za barem dva atributa resursa iz kolekcije po specifikaciji JSON-LD. Za uspješnu predaju 4. laboratorijske vježbe, potrebno je predati poveznicu na oznaku Release 4.0 javnog Github repozitorija koji sadrži:

- README.md, LICENCE
- naziv_skupa.json, naziv_skupa.csv, shema_skupa.json
- index.html, datatable.html
- dump baze s podacima
- pozadinski kod koji ostvaruje osnovnu i dopunjenu funkcionalnost API-ja
- openapi.json

Za ispitivanje valjanosti rada API-ja možete se poslužiti nekim od besplatnih alata, npr. *Postman* ili *cURL*.

Ispitno gradivo vježbe

- Što je protokol OAuth i za što se koristi?
- Koja je razlika između prokola OAuth i OpenID Connect?
- Kako ste integrirali uslugu vanjske autentifikacije u vašu aplikaciju?
- Što je specifikacija JSON-LD i za što se koristi?
- Koje ste atribute vašeg resursa semantički opisali?

Poveznice i literatura

- The Oauth 2.0 Authorization Framework: https://www.rfc-editor.org/rfc/rfc6749.html
- OpenID Connect Messages: https://openid.net/specs/openid-connect-messages-1_0-20.html
- auth0: https://auth0.com/
- JSON-LD: https://json-ld.org/
- Schema.org: https://schema.org/
- Understanding how structured data works: https://developers.google.com/search/docs/guides/intro-structured-data
- Schema.org validator: https://validator.schema.org/

5. laboratorijska vježba – Sudjelovanje u projektu otvorenog kôda

Projekti otvorenog kôda namijenjeni su – i otvoreni – **raznim oblicima suradnje**. Cilj ove laboratorijske vježbe je upoznavanje s radom na projektima otvorenog kôda, svim fazama suradnje, i jednostavan doprinos **projektu po vlastitom odabiru**.

Priprema za vježbu

Općeniti koraci za sudjelovanje u ovakvom projektu otvorenog kôda su:

1. **Odabir prikladnog projekta** – prilikom odabira projekta, obratite pažnju na:
    - Programski jezik kojeg već poznajete
    - Postojanje dobre dokumentacije za razvoj projekta, poput:
        - Postojanje jasnih uputa za početak – „*Getting started*“ dokumenti i tutorijali
        - Postojanje jasnih uputa za instalaciju potrebnog okruženja za rad s izvornim kôdom
        - Postojanje jasnih uputa za proces uključivanja izvornog kôda
    - Zrelost projekta
        - Redovito održavanje i izdavanje novih verzija projekta
        - Aktivna baza korisnika – postojanje foruma ili drugih kanala komunikacije
    - Organiziran način prijave pogrešaka i novih značajki – *Issue Tracker*
        - Dovoljan broj prijavljenih pogrešaka i trenutačno predloženih značajki
    - Vlastita zainteresiranost za projekt
        - Bilo bi jako dobro da taj program koristite redovito, kako biste znali njegove prednosti, mane i način rada
2. **Temeljito proučavanje dokumentacije i uputa**
    - Standardni *README*-dio (npr. na *GitHubu* ili na drugoj platformi)
    - Provjera otvorenosti licencije
    - Dokument „*Getting started*“ i slični
3. **Instalacija radnog okruženja za rad na izvornom kôdu**
    - Preporučeni alati i dodaci za programiranje i automatizirano testiranje
    - Lokalna baza podataka, poslužitelj Weba i slično
        - Možda postoji mogućnost preuzimanja *Docker* "slike" razvojnog okruženja koja integrira većinu potrebnih alata za razvoj (npr. web poslužitelj, bazu, itd.)
    - Dodatne biblioteke potrebne za rad
4. **Upoznavanje sa strukturom i arhitekturom sustava**
    - Raspored datoteka, struktura baze podataka
    - Osnovne funkcije koje se često koriste
        - npr. način prikazivanja nekog elementa u sučelju
    - Dobro je – uz izvorni kôd i postavljeno radno okruženje – proći osnovni tutorijal za razvoj (naziva poput „moj prvi modul“ i slično)
5. **Odabir već postojeće pogreške/značajke (*issue*) (ili prijava svoje) na kojoj želite raditi**
    - Odabrati dovoljno jednostavnu, ali ne prejednostavnu stvar
        - Za rješenje je potrebno barem desetak linija kôda
        - Nije dozvoljeno „dodati razmak“, zamijeniti poredak riječi, popraviti pogrešku u prijevodu i sl.
        - Dozvoljen je npr. „mali“ ispravak CSS-a (ako nije baš jedna linija)

- Pripaziti da *issue* već nije dodijeljen nekoj osobi, što obično znači da ona radi na njemu pa će vas preduhitriti
- OBAVEZNO provjeriti možete li reproducirati problem kod sebe, prije nego ga krenete rješavati
- Pratiti (*watch, subscribe, vote...*) razvoj događaja za određeni *issue*
  - Ako ste sigurni da ga želite riješiti, možete se i prijaviti za njega, kako ga netko ne bi „preoteo“ u međuvremenu

6. **Rad, rad, rad...**
   - Ne očajavajte ako ne ide iz prve, ili ako ste sasvim izgubljeni u programskom kôdu
   - Slobodno dodajte komentar u *issue*, pitajte iskusnije na forumu
   - Dobro je, dapače i poželjno, da prvi doprinos ne bude odmah cjelokupni P*ull request* s gotovim rješenjem. Predložite rješenje, stavite *screenshot*, prijavite se za rješavanje, komunicirajte...
7. **Slanje svog doprinosa**
   - Doprinos se može poslati i prije nego je sasvim gotov
   - Detaljno proučiti način formiranja *Pull requesta* i pripadajućih akcija, provjeriti kako se na ispravan način u odabranom projektu dodaje doprinos
   - Moguće je da postoje alati koji automatizirano rade kontinuiranu provjeru i integraciju (CI/CD). U tom slučaju, doprinos može biti odbijen zbog. npr. nepoštivanja konvencija o pisanju kôda, ili testova kojima rezultati nisu prošli. Potrebno je popraviti pogreške, pa ispočetka...
   - Moguće je da osobe koje održavaju projekt neće prihvatiti doprinos. Za smanjenje takvih problema, potrebno je i prije samog dodavanja kôda komunicirati s osobama zaduženim za *issue*, predložiti rješenje, itd...
   - Ako se i dogodi da iz nekog razloga doprinos ne bude prihvaćen – nije kraj svijeta. Bit će dovoljno za dobivanje bodova na ovoj vježbi.

Zadatak za vježbu

Izrada ove vježbe bit će drugačija nego prethodne. Podsjećamo da je vježba **neobavezna** (ali ulazi u broj bodova). Ova se vježba **NE MOŽE NAPRAVITI TIJEKOM JEDNE NOĆI**, jer je potrebno vrijeme za komunikaciju s održavateljima projekta i slično. Koraci:

1. Do nedjelje, 9. siječnja 2022. – prijaviti **sudjelovanje na vježbi** (način prijave će biti objavljen naknadno). Trebate prijaviti projekt na kojemu ćete raditi i (poželjno!) problem/funkcionalnost na kojemu ćete raditi. Dobro je projekt **prijaviti i ranije** – u slučaju velike navale na isti projekt pribjeći ćemo metodi FCFS – *First Come, First Served* pa ćemo vas zamoliti da promijenite projekt. Također, što ranije prijavite projekt, postoji veća vjerojatnost da dobijete dobar savjet zašto to ne bi bio dobar projekt, pa ga stignete promijeniti. Bilo bi dobro da, primjerice radi čekanja na odgovor projektne zajednice, puno ranije od ovoga datuma započnete raditi.
2. Na ovom labosu trebate raditi **kontinuirano** – u više navrata. Pokušajte već tijekom prosinca imati pripremljenu radnu okolinu, istraženu arhitekturu (minimum koji vam treba), osmišljen koncept rješavanja, itd.

Predaja vježbe

1. Do nedjelje, 16. siječnja 2022. – **predaja vježbe u obliku KRATKE prezentacije sa snimljenim zvukom** (video nije potreban, po želji), koja će trajati **najviše 5 minuta.** Ovisno o broju prijavljenih, na posljednjim predavanjima zajedno ćemo prodiskutirati o prezentacijama, ili ćemo taj dio odraditi asinkrono preko foruma.
2. U kratkoj prezentaciji treba **opisati cijeli proces izrade doprinosa**
    a. Kako/zašto ste odabrali baš taj projekt
    b. Koje su karakteristike projekta – popularnost, suradnici, licenca, itd.
    c. Što je od dokumentacije bilo dostupno
    d. Kako je protekla priprema radne okoline
    e. Kako ste odabrali baš taj problem kojeg ste rješavali
    f. Kako je proteklo rješavanje, što ste na kraju napravili
    g. Jeste li komunicirali s drugim osobama na projektu
    h. Kako je protekao *Pull request* (ako ste do njega uopće stigli)
    i. Što ste naučili? Što je bilo najteže? Na što treba obratiti pažnju?
    j. ……..

U cijeloj vježbi, posebno u prezentaciji, osim „učinka“ – ostvarenog doprinosa, zanimaju nas vaša **iskustva tijekom rada na projektu otvorenog kôda**. Ponavljamo da uspješno prihvaćanje doprinosa i njegovo uključivanje u izvorni kôd projekta **nije uvjet za predaju vježbe**. Ne morate stići do kraja, možete dobiti bodove / dio bodova i za „dobar pokušaj“, ako ga uspješno iznesete u svojoj prezentaciji.

SAVJET: Dok radite na labosu, paralelno pripremajte prezentaciju (radite screenshotove, pišite natuknice). Prezentaciju shvatite kao kratki „dnevnik rada“ u kojeg redovito dodajete poneku sitnicu – to će vam **jako** olakšati predaju.

Ispitno gradivo vježbe

Ova vježba nema uobičajeni način ispitivanja; bodovi se dodjeljuju temeljem rada na odabranoj temi.

Poveznice i literatura

Literatura ovisi o odabranom projektu otvorenog kôda.