

Pristup podacima iz programskog koda

Projektni zadatak

Ishodi učenja					UKUPNO
I1	I2	I3	I4	I5	
20	20	20	20	20	100

Nastavnik: Borna Skračić, pred.

UPUTE

- Obrana projekta odvija se za vrijeme ispitnih rokova
- Student radi prijavu kao i za pismene ispite
- Rješenja projektnih zadataka predaju se putem GitHub platforme te je obavezno korištenje Git alata prilikom izrade istih
- OBAVEZNO JE SERVIS ZA POHRANU (POSTGRES, MONOGO, MINIO) POSTAVITI KAO DOCKER KONTEJNERE** (možete slobodno ignorirati dio s "cloud" konfiguracijom)
- Projekt je potrebno poslati **tri dana prije roka definiranog na IE**

ISHODI UČENJA:

Ishod 1 (20 bodova):

- minimalni ishod učenja (10 bodova):* Izraditi softversko rješenje uporabom relacijske baze podataka u oblaku kao izvora podataka
- željeni ishodu učenja (10 bodova):* Izraditi relacijsku bazu podataka u oblaku i softversko rješenje uporabom relacijske baze podataka u oblaku kao izvora podataka

Ishod 2 (20 bodova):

- minimalni ishod učenja (10 bodova):* Izraditi softversko rješenje uporabom rješenja za pohranu nestrukturiranih podataka u oblaku kao izvora podataka
- željeni ishodu učenja (10 bodova):* Izraditi bazu za pohranu nestrukturiranih podataka u oblaku i softversko rješenje uporabom rješenja za pohranu nestrukturiranih podataka u oblaku kao izvora podataka

Ishod 3 (20 bodova):

- minimalni ishod učenja (10 bodova):* Izraditi softversko rješenje uporabom nerelacijske baze podataka u oblaku kao izvora podataka
- željeni ishodu učenja (10 bodova):* Izraditi nerelacijsku bazu podataka u oblaku i softversko rješenje uporabom nerelacijske baze podataka u oblaku kao izvora podataka

Ishod 4 (20 bodova):

- minimalni ishod učenja (10 bodova):* Odabrati i implementirati optimalni konceptualni model podataka
- željeni ishodu učenja (10 bodova):* Odabrati i implementirati optimalni složeni konceptualni model podataka

Ishod 5 (20 bodova):

- minimalni ishod učenja (10 bodova):* Implementirati softversko rješenje uporabom odabranih alata ORM
- željeni ishodu učenja (10 bodova):* Implementirati složeno softversko rješenje uporabom odabranih alata ORM

Prvi projekt (Ishod 1 – 20 bodova, Ishod 4 – 20 bodova, Ishod 5 – 20 bodova)

Vaš zadatak je, za minimalni ishod učenja, izraditi programsko rješenje koristeći gotovu implementaciju *Object Relational Mapper (ORM)* alat po Vašem izboru (npr. *Entity Framework*). Za željeni ishod potrebno je izraditi vlastitu implementaciju **ORM** alata u bilo kojem programskom jeziku (obaveza je koristiti OOP jezik koji sadrži implementaciju refleksije, ukoliko planirate koristiti nekakav „egzotičan“ jezik, **obavezni ste se javiti unaprijed – najkasnije do kraja polovice semestra**). Možete koristiti dijelove koda objavljene na vježbama i na repozitoriju kolegija (<https://github.com/bornaskracic/adpc-examples>) te konceptualna rješenja problema izvedena u *Entity Framework* biblioteci ukoliko ste se odlučili za vlastitu implementaciju ORM-a (željeni ishod učenja).

Demonstrirajte potrebne funkcionalnosti implementacijom jednostavne konzolne, desktop ili Web aplikacije koja koristi ORM alat za implementaciju CRUD operacija na relacijskoj shemi baze podataka (ovisno o Vašim preferencijama, koristite ili gotovu implementaciju ili vašu implementaciju ORM alata). Aplikacija treba podržavati relacijski model podataka izdvojenog na osnovi sljedećeg projektnog scenarija:

Potrebno je implementirati medicinski sustav odgovoran za upravljanje pacijentima (ime, prezime, OIB – jedinstvena vrijednost, datum rođenja, spol, adresa boravišta i adresa prebivališta). Korisnici ovog sustava su liječnici, definirani imenom, prezimenom i specijalizacijom. Liječnicima je cilj pratiti povijest bolesti pacijenata (od čega su bolovali, u kojem vremenskom periodu) te trenutne lijekove koje su im prepisane za određena stanja. Lijekovi se piju u specifičnim dozama (npr. tablete, mg, jedinice, itd.) u definiranoj učestalosti (npr. 3 puta dnevno, svaki drugi dan, jednom u dva tjedna, itd.). Također, liječnici mogu zakazati specijalističke preglede (CT, MR, ULTRA, EKG, ECHO, OKO, DERM, DENTA, MAMMO, EEG) u određenom terminu, kod određenog liječnika specijalista.

Aplikacija mora omogućiti CRUD operacije nad svim entitetima osim liječnika koje je potrebno definirati prilikom inicijalizacije aplikacije (potrebno je omogućiti spomenuto dodavanje samo prilikom prvog pokretanja aplikacije). Ako se odlučite za ostvarivanje samo minimalnih ishoda, dovoljno je koristiti gotovu implementaciju ORM-a kako bi ste zadovoljili kriterije ishoda učenje. Ukoliko Vam nije cilj samo prolazna ocjena, već i ona viša (željeni ishodi), potrebno je implementirati vlastiti ORM u obliku biblioteke koja podržava sljedeće značajke:

- mapiranje klasa na tablice u bazi podataka – osnovni koncept čitanja redova iz baze podataka kao instance entitetskih klasa u aplikativnom kodu (savjet: koristiti refleksiju)
- mapiranje podatkovnih tipova za barem sljedeće podatkovne tipove: (INT, DECIMAL, FLOAT, VARCHAR, CHAR, TEXT, TIMESTAMP WITH TIMEZONE, TIMESTAMP WITHOUT TIMEZONE), ukoliko je potrebno možete koristiti i atribute (anotacije)
- ograničenja na stupcima - NULL / NOT NULL, UNIQUE, DEFAULT kao i PRIMARY KEY (uz mogućnost navođenja njegovog auto-inkrementalnog ponašanja)
- osnovne CRUD operacije nad objektima (odnosno mapiranim tabličnim redovima), korištenjem pristupa po želji
 - obuhvatiti sortiranje i filtriranje (idealno putem parsiranja ekspresija) prilikom dohvaćanja podataka
 - implementirati automatsko generiranje odgovarajućih modifikacijskih upita na osnovi stanja praćenog mapiranog objekta (primjer je *change tracking* koncept implementiran u *DbContext* klasi *Entity Frameworka*)
- dohvat povezanih podataka, korištenjem *eager* i/ili *lazy* loading pristupa (savjet je implementirati navigacijska svojstva no i ostale implementacije su prihvatljive)
- automatsko generiranje migracija na osnovi razlike trenutnog stanja sheme u bazi podataka i klasnih definicija entiteta u OO programskom kodu kao i njihovo izvršavanje (unaprijed ili unazad)

Ishod učenja 1		Bodovi	Ukupno
MINIMALNI	Objasniti koncepte vezane za Postgres arhitekturu (WAL, vacuum, checkpoint, ...).	2,00	20,00
MINIMALNI	Demonstrirati razumijevanje ACID principa i objasniti njihovu realizaciju u Postgresu.	2,00	
MINIMALNI	Demonstrirati postavljanje Postgres baze podataka putem Dockera i Supabase platforme.	2,00	
MINIMALNI	Detaljno objasniti značaj i funkciju indeksa u bazi podataka, kao i objasniti zašto neki upiti hoće, odnosno neće, koristiti indeks.	2,00	
MINIMALNI	Objasniti specifičnosti vezano za povezivanje s bazom podataka kao i upravljanje konekcijom.	2,00	
ŽELJENI	Implementirati osnovno mapiranje klasa na tablice (savjet: koristiti refleksiju).	2,50	
ŽELJENI	Implementirati osnovno dohvaćanje i umetanje podataka u bazu, mapirajući programski kod (npr. pozive funkcija nad objektom) u izvršavajući SQL upit.	2,50	
ŽELJENI	Implementirati filtraciju podataka prilikom dohvaćanje (mapiranje programskih poziva na WHERE dio SQL upita, savjet: koristiti parsiranje ekspresija).	2,50	
ŽELJENI	Implementirati ograničenja (eng. <i>constraints</i>) nad stupcima (NULL/NOT NULL, UNIQUE, DEFAULT kao i PRIMARY KEY)	2,50	
Ishod učenja 4		Bodovi	Ukupno
MINIMALNI	Implementirati programsko rješenje za traženi projektni scenarij definiran u tekstu projektnog zadatka (ukoliko ovaj dio implementirate za minimalni ishod učenja, bez vlastite implementacije ORM-a, koristite gotovo rješenje u obliku ORM alata implementiranog u jeziku koji koristite)	7,50	20,00
MINIMALNI	Demonstrirati funkcionalnost dohvaćanje povezanih podataka koristeći <i>eager</i> ili <i>lazy</i> pristup. Objasniti razliku i nedostatke oba pristupa.	2,50	
ŽELJENI	Implementirati dohvaćanje povezanih podataka koristeći implementaciju navigacijskih svojstava (moguće je implementaciju ostvariti i nekom drugom strategijom, dok god implementacija dohvaća povezane podatke - 1:1, 1:N, N:1 - implementacije se smatra prihvatljivom). Implementaciju je potrebno objasniti i demonstrirati na obrani.	5,00	
ŽELJENI	Implementirati mogućnost praćenja stanja objekta i automatsko generiranje odgovarajućih upita na osnovi promjena nad praćenim objektom (nalik na primjere pokazane kroz EntityFramework :: ChangeTracker koncept)	5,00	
Ishod učenja 5		Bodovi	Ukupno
MINIMALNI	Implementirati prethodno spomenuto programsko rješenje korištenjem <i>code first</i> pristupa.	5,00	20,00
MINIMALNI	Demonstrirati stvaranje, izvršavanje i korištenje migracija te objasniti u kojim slučajevima migracije nije moguće izvršiti.	5,00	
ŽELJENI	Implementirati algoritam za automatsko generiranje adekvatnih migracija temeljenih na razlici trenutnog stanja sheme baze podataka i klasnih definicija entiteta u OOP kodu (najzahtjevniji dio, tako da je prihvatljivo uzeti u obzir neke pretpostavke koje olakšavaju implementaciju).	7,00	
ŽELJENI	Implementirati mehanizam za izvršavanje migracija (unaprijed i unazad) te praćenja trenutnog stanja izvršenih migracija.	3,00	

Drugi projekt (Ishod 2 – 20 bodova, Ishod 3 – 20 bodova)

Vaš je zadatak implementirati *pipeline* za obradu podataka o opažanju i taksonomiji ptica koji se sastoji od četiri koraka. Glavni cilj ovog pipelinea je generirati skup podataka koji sadrži informacije o pticama i njihovim opažanjima na temelju:

- audio datoteka koje sadrže moguće pjevoe ili zovove ptica
- podataka konzumiranih iz vanjskih ornitoloških izvora (servisa).

Za ovo rješenje obavezno je koristiti **MinIO** (ili bilo koju drugu S3 kompatibilnu pohranu podataka – I2) i **MongoDB** (I3) kao pružatelje usluga pohrane. Međutim, ako vam je potreban bilo koji drugi vanjski alat ili biblioteka, dopušteno vam je njihovo korištenje u vašem rješenju.

Prvo je potrebno prikupiti sve podatke o vrstama ptica (lat. *aves*) i stvoriti temelj projekta pohranjivanjem prikupljenih podataka u **MongoDB** kolekciju. Podaci su dostupni na <https://aves.regoch.net> (mock web stranica koja koristi javno dostupne GBIF podatke). Ako podaci već postoje u kolekciji, ovaj korak je potrebno preskočiti.

Nadalje, potrebno je pročitati sve poruke prisutne na Kafka brokeru u trenutku izvršavanja. Poruke na Kafki sadrže informacije o observacijama ptica koje su objavili ornitolozi, a nazivaju se opažanjima. Ove poruke ne sadrže nikakve audio datoteke. Umjesto toga, moraju sadržavati identifikacijski taksonomski kod opažene ptice i pripadajući geografski položaj (dužina i širina), ali trenutni podaci biološkog opažanja mogu se razlikovati između različitih ornitoloških izvora (npr. veličina tijela, tjelesna temperatura, status migracije, obrazac leta, stanište itd.). Sva opažanja potrebno je pohraniti u vašu bazu podataka, korištenjem **MongoDB**-a.

Treći korak trebao bi obraditi audio datoteke u ciljnom direktoriju, što znači prijenos svih datoteka u MinIO (ili bilo koju drugu S3 kompatibilnu pohranu) i slijedno poslati API zahtjev za klasifikaciju ptice na javno dostupni model klasifikacije ptica temeljem audio zapisa. Odgovor klasifikatora pružit će informacije o pticama prisutnima u audio snimci, kao i ocjenu pouzdanosti klasifikacije. Audio datoteka i rezultati klasifikacije moraju se shodno tome pohraniti u bazu podataka kako bi se omogućio kasnije preuzimanje.

Ciljni direktorij koji sadrži audio datoteke može biti lokalni direktorij, ali opcionalno možete podržati i pohranu u oblaku (npr. *Google Drive*). Svaka datoteka mora biti povezana s geografskim položajem, slično prethodnom koraku (radi jednostavnosti, možete pretpostaviti da su sve datoteke u određenoj mapi povezane s jednim geografskim položajem).

Završni korak trebao bi generirati statistiku za sve vrste ptica koje imaju barem jednu pozitivnu klasifikaciju. Prije stvaranja konačnog rezultata, obavezno očistite podatke i primijenite odgovarajuće transformacije. Izrađeni CSV trebao bi sadržavati naziv vrste, broj klasificiranih opažanja i sve relevantne podatke o promatranju. Prilikom izvršavanja ovog pipelinea, može se postaviti opcionalni parametar kako bi se omogućio fuzzy filter prema nazivu vrste ptica.

Da bi se postigao maksimalan broj bodova, opisani tijekom rada treba podijeliti u više manjih skripti orkestriranih bilo kojim alatom koji omogućuje izvršavanje s jednom ulaznom točkom (eng. *entrypoint*) te s mogućnošću definiranja opcionalnih parametara za vrijeme izvođenja (budući da je Python preporučeni jezik za ovaj zadatak, [Snakemake](#) je preporučeni orkestrator izvršavanja skripti). Za dodatne bodove, omogućite izvršavanje skripte putem ručno okinutog [Github Actions](#) workflowa (kao što je prikazano u vježbama), kao i vizualizaciju generiranog izvješća, koristeći alate po želji.

U nastavku teksta, opisano su koraci za svaki uspješno implementiran segment pipelinea. Redoslijed je izmiješan kako bi se prilagodio definicijama i redoslijedu ishoda učenja, a za stvarni redoslijed koraka, pogledajte prethodni opis.

Ishod učenja 2 (Minimalni – 10 bodova) Implementirajte obradu datoteka koje se nalaze u danom direktoriju tako da ih prenesete u **MinIO** (ili drugu S3-kompatibilnu pohranu) te osigurate da se svaka datoteka može dohvatiti i jedinstveno identificirati. Prenesene datoteke trebaju biti povezane s metapodacima, kao što su lokacija i naziv datoteke, a ta povezanost mora biti pohranjena u **MongoDB**-u. Po želji možete dodati podršku za obradu datoteka pronađenih na drugim udaljenim lokacijama (npr. *Google Drive*).

Ishod učenja 2 (Željeni – 10 bodova) Za svaku prenesenu datoteku, pošaljite API poziv klasifikacijskom modelu ptica POST <https://aves.regoch.net/api/classify> te pohranite log zahtjeva u **MinIO** (definirajte bilo koji format koji smatrate prikladnim), a rezultate klasifikacije spremite u odgovarajuću **MongoDB** kolekciju, povezujući taksonomske podatke o pticama s njihovim opažanjima.

Ishod učenja 3 (Minimalni – 10 bodova) Pohranite taksonomske podatke o vrstama ptica prikupljene s <https://aves.regoch.net> u odgovarajuću **MongoDB** kolekciju izbjegavajući duplicirane unose. Nakon uspješno provedenog zahtjeva za klasifikacijom ptice u audio zapisu, pohranite rezultate klasifikacije u odgovarajuću kolekciju povezujući opažanja ptica s informacijama o vrstama.

Ishod učenja 3 (Željeni – 5 bodova) Konzumirajte Kafka poruke koje sadrže opažanja ptica i pohranite ih u **MongoDB** kolekciju, uključujući ID vrste, lokaciju i sve navedene podatke o biološkim opažanjima. Budite oprezni, jer različita opažanja mogu sadržavati različita biološka svojstva.

Ishod učenja 3 (Željeni – 5 bodova) Implementirajte filtriranje (fuzzy sting matching) na temelju naziva vrsta za generiranje konačnog CSV izvješća i primijenite odgovarajuću metodu čišćenja i transformacije podataka prilikom generiranja finalne CSV datoteke.