Pristup podacima iz programskog koda

Projektni zadatak

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

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
- 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 2 – 10 bodova, Ishod 4 – 20 bodova, Ishod 5 – 20 bodova)

Vaš zadatak je implementirati vlastitu implementaciju *Object Relational Mapper* (**ORM**) alata u bilo kojem programskom jeziku (savjet je korištenje OOP jezika). Možete koristiti djelove koda objavljene na vježbama i konceptualna rješenja problema izvedena u Entity Framework biblioteci kao inspiraciju za vaše rješenje.

Značajke koje Vaše programsko rješenje mora podržati:

- Mapiranje klasa na tablice u bazi podataka – osnovni koncept čitanja redova iz baze podataka kao instance entitetski klasa u aplikativnom kodu
- Mapiranje podatkovnih tipova za barem sljedeće podatkovne tipove:
    - VARCHAR
    - INT
    - DECIMAL
    - FLOAT
    - UNIQUE
- Barem sljedeća ograničenja na stupcima:
    - NULL / NOT NULL
    - DEFAULT
    - UNIQUE
    - Auto-generirani primarni ključ
- Upravljanje konekcijom s bazom podataka
- Upravljanje transakcijskim kontekstom (korištenjem *Unit of Work* obrasca)
- Osnovne CRUD operacije korištenjem pristupa po želji
    - Obuhvatiti filtriranje (idealno putem *fluent* API)
    - Podržati sortiranje (opcionalno i grupiranje :))
- Navigacijska svojstva za dohvaćanje povezanih podataka
- Migracije
    - Stvaranje i izvršavanje
    - *Rollback* na prijašnje stanje

Demonstrirajte potrebne funkcionalnosti implementacijom jednostavne web ili konzolne aplikacije koja koristi vašu ORM implementaciju za CRUD operacije na relacijskoj shemi baze podataka. Aplikacija treba podržavati relacijski model podataka koji obuhvaća scenarij medicinskog sustava odgovornog za upravljanje pacijentima, njihovim medicinskim kartonima, pregledima i receptima. Aplikacija mora omogućiti CRUD operacije na pacijentima, pregledima, lijekovima. Pregledi mogu biti određene vrste (liječnik opće prakse, krv, rendgen, CT, MR, ULTRA, EKG, ECHO, OKO, DERM, DENTA, MAMMO, EEG), a recepti detaljno opisuju koju dozu lijekova pacijent uzima.

Značajke navedene u nastavku dokumenta potrebno je demonstrirati na obrani projekta. Kao što je već spomenuto, možete koristiti jednostavnu konzolu ili web aplikaciju kako biste demonstrirali kako vaša ORM implementacija funkcionira u realnom scenariju upotrebe. Također možete očekivati pitanja vezana uz razumijevanje kako temeljni mehanizmi funkcioniraju za različite pohrane podataka koje ste koristili u projektu i koje smo obradili u vježbama.

Ishod učenja 1 (Minimalni – 10 bodova) Izradite bazu podataka s potrebnim entitetima i relacijama smještenim u Postgres bazi podataka u oblaku koju osigurava Supabase servis u oblaku. Obavezno je objasniti kako je baza podataka kreirana te kako se povezati s danom instancom. Koristeći vlastitu ORM implementaciju, implementirajte funkcionalnost za generiranje DDL upita na temelju sheme baze podataka definirane u aplikativnom kodu.

Ishod učenja 1 (Željeni – 10 bodova) Implementirajte podršku za CRUD operacije na potrebnim povezanim entitetima, koristeći bilo koji pristup koji želite. Potrebno je dodatno demonstrirati sljedeće:

- detaljno objašnjenje komunikacije između aplikacije i baze podataka
- razumijevanje arhitekture Postgresa

Ishod učenja 4 (Minimalni – 10 bodova) Stvorite konceptualni model podataka uvođenjem potrebnih entiteta povezanih različitim odnosima, slijedeći definiciju 3. normalne forme. Osigurajte integritet podataka implementacijom podrške za ograničenja stupaca baze podataka u vašoj implementaciji ORM alata.

Ishod učenja 4 (Željeni – 10 bodova) Implementirajte dohvaćanje povezanih podataka korištenjem navigacijskih svojstava za različite kardinalnosti (1-više,više-1, 1-1).

Ishod učenja 5 (Minimalni – 10 bodova) Implementirajte strategiju upravljanja konekcija s bazom podataka kao i upravljanje transakcijama (*Unit of Work* obrazac). Demonstrirajte ove značajke u vašoj implementaciji ORM-a kroz svoju aplikaciju.

Ishod učenja 5 (Željeni – 10 bodova) Implementirajte vlastitu funkcionalnost migracije sheme baze podataka. Idealno bi bilo implementirati zasebnu konzolnu aplikaciju koje je zadužena za stvaranje novih migracija, izvršavanje postojećih migracija (s praćenjem) i vraćanje na prethodnu migraciju (eng. *rollback*).

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 pipeliena je generirati skup podataka koji sadrži informacije o pticama i njihovim opažanjima na temelju:

- audio datoteka koje sadrže moguće pjevove 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 obzervacijama 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 koji 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 tijek 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.