Projektni zadatak P5 – Communication Bus

Osnovna namjena projekta je mijenjanje formata podataka prilikom njihovog prenosa kroz mrežu. Predstavljena su tri formata: *XML*, *JSON* i *SQL*. U tekstu je objašnjena svaka komponenta, komunikacije između komponenti kao i na koji način se vrše konverzije iz jednog formata u drugi. Korišćena je *MySQL* baza podataka. Tabele koje su proizvoljno dodate su *bolnica*, *odnos_radnika*, *radnik*, *tip_veze* i *vrsta_radnika*. Iz ovih tabela mogu da se čitaju ali i mijenjaju podaci kao i same tabele. Projekat sadrži podatke o zaposlenima u određenoj bolnici i njihovim vezama.

1. Web-Client

Web Client predstavlja komponentu koja na određeni vremenski period šalje zahtjeve ka *Communication Bus* komponenti. Sadrži *connect_to_server*, *print_response* i *client funkciju*. c*onnect_to_server* omogućava konektovanje na server *Communication Bus*-a. Nakon uspiješno uspostavljenog *TCP* komunikacionog kanala, šalju se zahtjevi klijenta. Korisniku se prikazuje meni usluga koje *Communication Bus* može da izvrši. Prikaz menija omogućava *client* funkcija. Zahtjev se formira na osnovu naredbi *GET*, *POST*, *PATCH*, *DELETE*. Odabirom jedne od opcija manevriše se sa podacima u bazi podataka, gdje je za ovu vrstu usluga korišćena *MySQL* baza. Nakon odabira usluge, zahtjev se šalje kroz komunikacioni kanal. Funkcija *print_response* prima podatak koji stiže kao odgovor od *Communication Bus* komponente, i ispisuje ga.

2. Communication-Bus

Za razliku od *Client* komponente, *Communication Bus* sadrži server koji sluša na određenom portu i prihvata klijentske zahtjeve. Glavna funkcija koja izvršava zahtjeve je execute_request. Funkcija primi zahtjev u *JSON* formatu, i pomoću *JsonXmlAdapter*-a, pretvara ga u *XML* format. *XML* format je neophodan za komunikaciju sa *XMLDataBaseAdapter*-om. On procesuira klijentski zahtjev dalje kroz sistem. Komunikacija između ove dvije komponente omogućena je preko *TCP* komunikacionog kanala. Odgovor sa *XMLDataBaseAdapter*-a se prima u *XML* formatu, pa je neophodno ponovo upotrijebiti *JsonXmlAdapter* za vraćanje u *JSON*

format (koji je jedini razumljiv *Client*-u). Uz to, urađene su i osnovne validacije kako se ne bi neispravan zahtjev slao dalje kroz sistem.

3. JsonXmlAdapter

Ima ulogu konverzije iz jednog formata u drugi. Kao što je u prethodnoj komponenti spomenuto, postoje konverzije iz *XML*-a u *JSON* format i obrnuto. Da bi se omogućila konverzija korišćena su dva paketa, *json* i *xmltodict*. Funkcija *json_to_xml* prima *JSON* objekat koji prepakuje u *dictionary*. Taj *dictionary* se pomoću *xmltodict* pretvori u *XML* format. Slična stvar je i sa *xml_to_json* funkcijom. Primi se *XML* podatak i onda se pretvori u *JSON dictionary*.

4. XmlDataBaseAdapter

Osnovna funkcionalnost je primanje zahtjeva od *Communication Bus* komponente i slanje zahtjeva ka *Repository* komponenti, kako bi se pristigli zahtjev mogao isporučiti. Klasa se sastoji od četiri osnovne funkcije: *get_method*, *delete_method*, *insert_method*, *update_method*. Svaka od navedenih funkcija prima *XML* objekat od kog pravi odgovarajući *SQL* upit. Funkcija *from_xml_to_sql* ima ulogu menija, gdje na osnovu izabrane opcije, poziva jedan od metoda. Kada se obavi konverzija, tako formiran zahtjev se dalje šalje *Repository* komponenti. Uspostavljanje komunikacije je, kao i do sada, preko *TCP* komunikacionog kanala. Odgovor od *Repository* komponente se dobija u obliku *dictionary*-ja, gdje je jasno naznačen *status code* (2000, 3000, 5000), *status* (*SUCCESS*, *REJECTED*) i *payload* (podaci koje baza vrati kao odgovor ili opis nemogućnosti izvršavanja zahtjeva u slučaju greške).

5. Repository

O funkcionalnostima ove klase se moglo upoznati i kroz opis prethodne komponente. To je komponenta koja dobavlja informacije iz baze i jedina koja komunicira sa bazom. Prima *SQL* upit od *XmlDataBaseAdapter*-a i izvršava taj upit preko *cursor* polja, a akcija koja prethodi izvršavanju upita nad tim poljem je uspostavljanje konekcije sa bazom. Kao što je već rečeno, *Repository* vraća

dictionary objekat sa *status code*-om, *status*-om i *payload* ključnim poljem. Sadrži i podignut server na kom se slušaju zahtjevi *XmlDataBaseAdapter*-a.

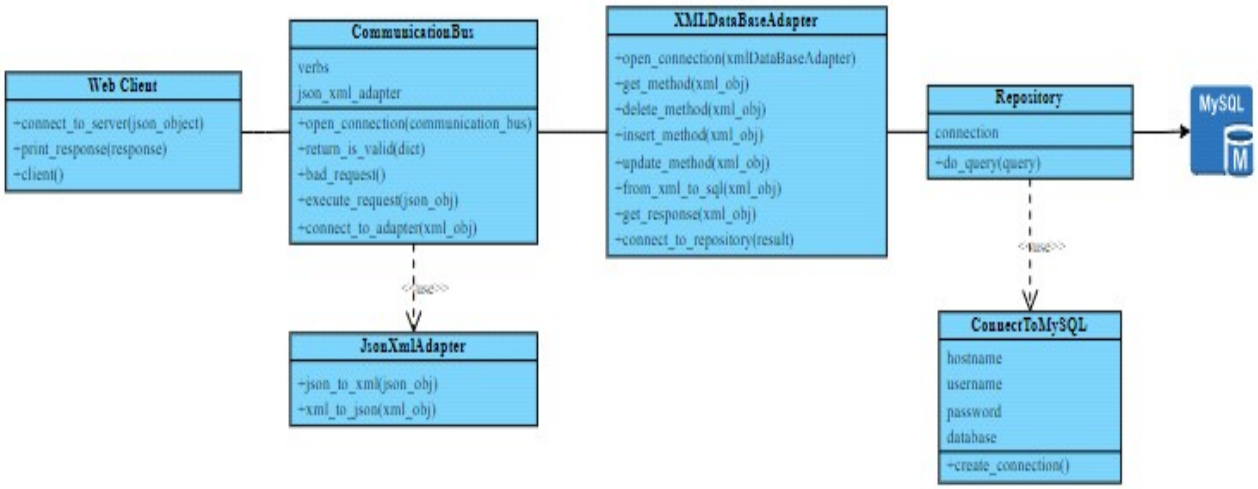
6. ConnectToMySQL

Klasa u sebi sadrži otvaranje konekcije ka *MySQL* bazi podataka, koje koristi *Repository* komponenta. Funkcija koja uspostavlja vezu je *create_connection* kojoj se prosleđuju osnovna polja koja svaka baza ima: *hostname*, *username*, *password*, *database*.

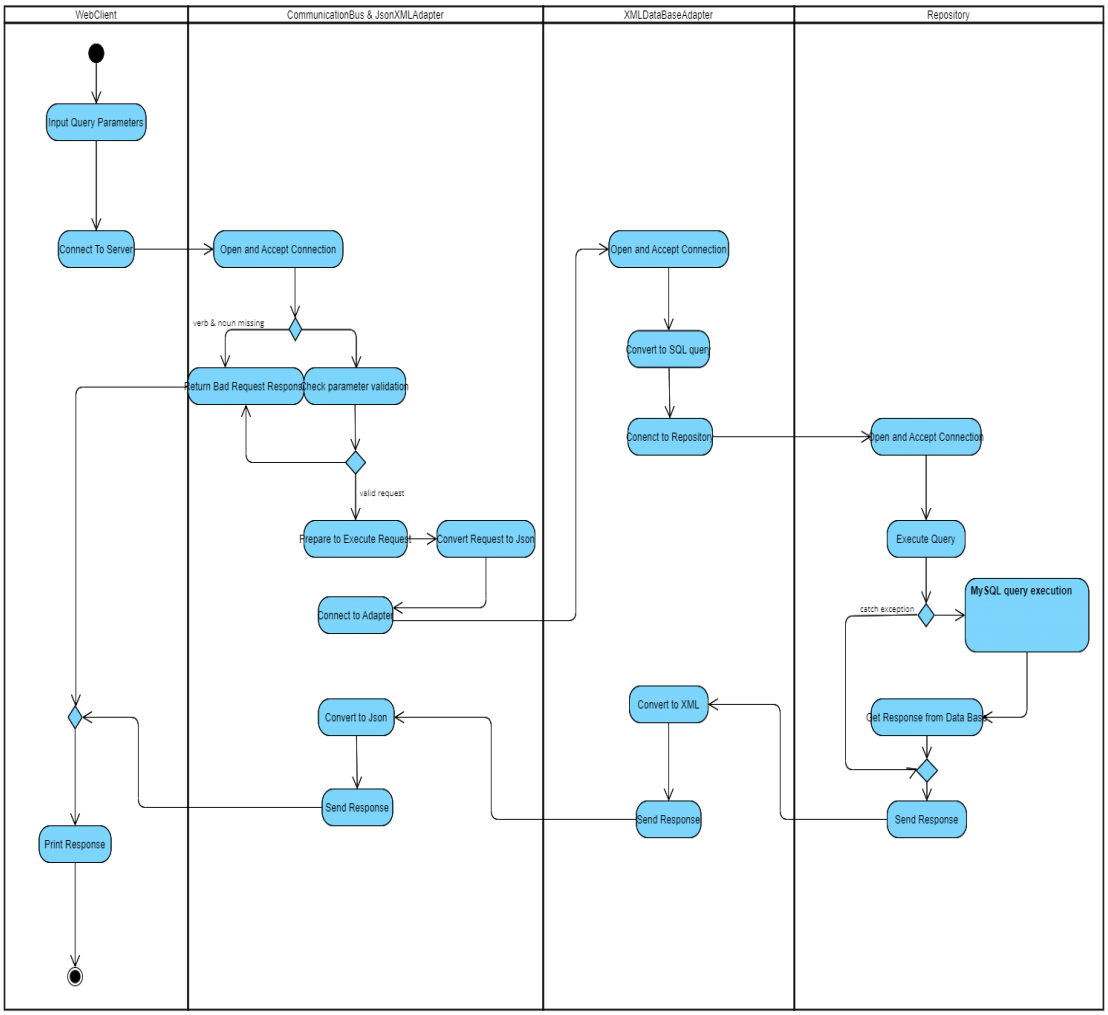
7. Test klase

Projekat sadrži četiri test klase: *test_CommunicationBus*, *test_JsonXmlAdapter*, *test_Repository*, *test_WebClient*. U njima su testirane funkcionalnosti svake od funkcija, podijeljene po klasama.

Za bolje razumijevanje procesa dobijanja klijentskih zahtjeva i dobavljanja traženih podataka iz baze, formiran je *UML* dijagram (**slika 1**.) kao i *Activity* dijagram (**slika 2**.).



Slika 1. *UML dijagram za prikaz dobavljanja klijentskih zahtjeva sa MySQL baze*



Slika 2. *Activity dijagram za prikaz dobavljanja klijentskih zahtjeva sa MySQL baze*