UDC: 004.65 Info M str. 19-27

PREDLOG SKALABILNE SERVISNO-ORIJENTISANE ARHITEKTURE SISTEMA ZA AUTOMATIZOVANO PROJEKTOVANJE BAZE PODATAKA
PROPOSAL FOR SCALABALE SERVICE-ORIENTED ARCHITECTURE OF THE SYSTEM FOR AUTOMATED DATABASE DESIGN

Goran Banjac, Zoran Đurić, Dražen Brđanin, Danijela Banjac

REZIME: U ovom radu dat je pregled postojeće arhitekture sistema za automatizovano projektovanje baze podataka, koji je razvijen u istraživačkoj laboratoriji Mlab na Elektrotehničkom fakultetu Univerziteta u Banjoj Luci. Sistem se sastoji od četiri alata za podršku automatizovanom projektovanju konceptualnog modela baze podataka na osnovu ulaznih artefakata različitog tipa (modeli poslovnih procesa, tekst i govor). Svi Mlab alati podržavaju i ostale faze direktnog inženjeringa relacione baze podataka. U radu je detaljno predstavljeno svih 13 komponenata sistema, pri čemu svi Mlab alati dele pojedine zajedničke komponente. Pored pregleda arhitekture postojećeg sistema, dat je i predlog nove skalabilne servisno-orijentisane arhitekture sistema. Predloženo unapređenje sistema predstavlja osnov za nastavak istraživanja i naknadna unapređenja sistema.

KLJUČNE REČI: AMADEOS, DBomnia, projektovanje baze podataka, servisno-orijentisana arhitektura, SpeeD, TexToData

ABSTRACT: This article presents an overview of the architecture of the existing system for automated database design, developed at the Mlab research laboratory at the Faculty of Electrical Engineering, University of Banja Luka. The system consists of four tools that support the automated design of a conceptual database model based on various types of input artifacts (business process models, text, and speech). Also, all Mlab tools support other phases of forward engineering of relational databases. The article presents all 13 system components, where all Mlab tools share several common components. Apart from the overview of the architecture of the current system, the article presents a proposal for the new scalable service-oriented architecture of the system. The given proposal represents the basis for future research and further system improvements.

KEY WORDS: AMADEOS, database design, DBomnia, service-oriented architecture, SpeeD, TexToData

1. UVOD

Cilj projektovanja baze podataka je da se dođe do specifikacije strukture baze podataka (modela) za odgovarajući *sistem za upravljanje bazama podataka* (SUBP), a proces projektovanja baze podataka tipično se sastoji od nekoliko koraka [1]. Prvi i najvažniji korak je konceptualno modelovanje. Rezultat konceptualnog modelovanja je platformski nezavisan *konceptualni model baze podataka* (KMBP), koji opisuje entitete i veze između njih na visokom nivou apstrakcije. U svakom narednom koraku, rezultat/model (uključujući i konačan model za konkretan SUBP) može biti izveden na osnovu modela iz prethodnog koraka, pa to predstavlja glavni razlog dugogodišnjeg interesa istraživača za automatizaciju[1] projektovanja KMBP [2].

Početak istraživanja u cilju automatizacije projektovanja KMBP datira još iz 1980-ih [3], a od tada je objavljen veliki broj radova u kojima su predloženi različiti pristupi i alati koji omogućavaju (polu)automatsko generisanje KMBP ograničene kompletnosti, koji kao polazni osnov uzimaju različite tipove artefakata. U literaturi su najviše zastupljeni pristupi koji se zasnivaju na obradi teksta (npr. [4, 5, 6]) i pristupi koji kao polazni osnov uzimaju modele (npr. [7, 8, 9]). Međutim, svi postojeći alati imaju ograničenu efektivnost (omogućavaju automatsko generisanje KMBP koji nisu 100% kompletni niti 100% tačni) i ne postoji alat koji omogućava automatsko generisanje kompletnog ciljnog KMBP.

Domen rada istraživačke grupe Mlab[2] (*Laboratorija za softversko inženjerstvo vođeno modelima, Elektrotehnički fakultet, Univerzitet u Banjoj Luci*) jeste softversko inženjerstvo vođeno modelima, a trenutno je u fokusu automatizovano projektovanje KMBP na osnovu ulaznih artefakata različitog tipa, kao što su modeli, tekst, govor i dr. Najvažnija dostignuća grupe Mlab, u istraživanju u navedenoj oblasti, predstavljaju sledeći onlajn (engl. *online*) alati za podršku automatizovanom projektovanju KMBP: AMADEOS[3], TexToData[4], SpeeD[5] i DBomnia[6]. Navedeni alati postižu (relativno) visoku efektivnost automatskog generisanja KMBP, ali i dalje generisani modeli nisu niti 100% kompletni niti 100% tačni pa se i dalje radi na usavršavanju datih alata, te istraživanju mogućnosti povećanja kompletnosti i tačnosti generisanih KMBP. Pored automatizovanog projektovanja KMBP, pomenuti Mlab alati podržavaju i ostale faze procesa projektovanja baze podataka, koje uključuju: (i) transformaciju KMBP u *relacioni model baze podataka* (RMBP), (ii) transformaciju RMBP u DDL[7] skriptu za kreiranje šeme relacione baze podataka i (iii) kreiranje šeme relacione baze podataka, tj. izvršavanje DDL skripte na konkretnom SUBP. Svi nabrojani alati su javno dostupni i otvoreni su za upotrebu.

Svi Mlab alati dele pojedine zajedničke komponente, a u ovom radu dat je pregled trenutne arhitekture sistema, pri čemu je dat detaljan opis svake komponente sistema koji omogućava automatizovano projektovanje baze podataka na osnovu ulaznih artefakata različitog tipa. U radu je dat i predlog nove skalabilne servisno-orijentisane arhitekture sistema.

[1] Iako prvi korak [1] samog procesa projektovanja baze podataka predstavlja konceptualno modelovanje, u razvoju sistema sa bazom podataka fazi projektovanja prethode druge faze (npr. faza analize) koje rezultuju artefaktima koji mogu da se koriste za automatizaciju projektovanja KMBP.

[2] http://m-lab.etf.unibl.org

[3] http://m-lab.etf.unibl.org:8080/amadeos/

[4] http://m-lab.etf.unibl.org:8080/Textodata/

[5] http://m-lab.etf.unibl.org:8080/SpeeD/

[6] http://m-lab.etf.unibl.org:8080/dbomnia/

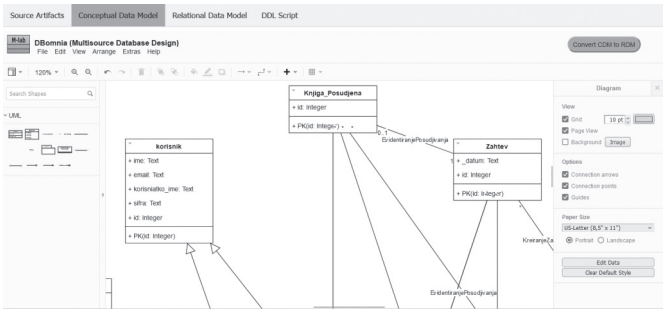
[7] *Data Definition Language*

Rad je organizovan na sledeći način. Nakon uvodnog dela, u drugom poglavlju dat je kratak pregled i opis Mlab alata za automatizovano projektovanje baze podataka. U trećem poglavlju prikazana je trenutna arhitektura sistema, dok je u četvrtom poglavlju dat predlog nove arhitekture sistema. Nakon zaključka pobrojana je citirana literatura.

2. PREGLED MLAB ALATA

Za svaki Mlab alat za automatizovano projektovanje baze podataka implementirana je klijentska veb aplikacija koja omogućava manipulaciju ulaznim artefaktima te generisanim artefaktima (KMBP, RMBP, DDL skripta), dok je serverska strana implementirana, tipično, kao orkestracija kolekcije veb servisa. Tipično, orkestratorski servis pojedinog alata upravlja procesom sinteze modela koji se odvija u nekoliko faza, dok se pojedine aktivnosti realizuju pojedinačnim servisima koji mogu da budu zajednički za više Mlab alata. Detaljan opis svakog servisa (serverska strana) dat je u sledećem poglavlju, dok je u ovom poglavlju dat kratak pregled Mlab alata sa fokusom na klijentsku veb aplikaciju.

Za ilustraciju, na sl. 1 prikazan je jedan od alata (DBomnia) *u akciji*. Klijentske veb aplikacije svih Mlab alata imaju sličan korisnički interfejs koji je, tipično, podeljen u nekoliko kartica (engl. *tab*) namenjenih za rad sa: (i) ulaznim artefaktima, (ii) KMBP, (iii) RMBP i (iv) DDL skriptom. U Mlab alatima i KMBP i RMBP reprezentuju se standardnim UML[8] dijagramom klasa (engl. *UML Class Diagram* – UML CD) [10].



Slika 1. DBomnia *u akciji*

2.1. *AMADEOS*

AMADEOS [11] je prvi onlajn veb-bazirani alat koji omogućava generisanje KMBP na osnovu kolekcije *modela poslovnih procesa* (MPP) koji mogu da budu reprezentovani korištenjem dve različite notacije, BPMN[9] i UML dijagram aktivnosti (engl. *UML Activity Diagram* – UML AD). AMADEOS klijentska veb aplikacija omogućava korisniku da učita kolekciju MPP (kolekcija XMI[10] serijalizovanih UML AD modela, kolekcija XMI serijalizovanih BPMN modela ili kolekcija XSD[11] serijalizovanih BPMN modela), na osnovu koje se na serverskoj strani vrši generisanje KMBP. Pored KMBP, generiše se i raspored (engl. *layout*) elemenata na dijagramu koji se po završetku procesa generisanja prikazuje u okviru veb čitača. Automatski generisani model i dijagram je moguće modifikovati, odnosno doraditi. AMADEOS pruža korisniku mogućnost da na osnovu KMBP generiše RMBP za jedan od podržanih SUBP (u trenutnoj verziji podržani su MySQL, Microsoft SQL Server, Oracle Database, PostgreSQL i IBM Db2). Dalje, korisnik ima mogućnost da na osnovu RMBP generiše korespondentnu DDL skriptu za generisanje šeme relacione baze podataka. Generisanu DDL skriptu moguće je preuzeti (kopirati) i/ili izvršiti na nekom javno dostupnom SUBP. Na serverskoj strani, AMADEOS orkestratorski servis upravlja procesom sinteze KMBP, kao i procesom generisanja RMBP, procesom generisanja DDL skripte te procesom izvršavanja DDL skripte na SUBP.

2.2. *TexToData*

TexToData [12] je onlajn višejezični veb-bazirani alat koji omogućava generisanje KMBP na osnovu tekstualnih specifikacija u prirodnom jeziku, a trenutno je podržano preko 100 jezika (engleski, nemački, italijanski, grčki, srpski i dr.). TexToData klijentska veb aplikacija omogućava korisniku da unese tekstualnu specifikaciju u prirodnom jeziku na osnovu koje se na serverskoj strani vrši generisanje KMBP, kao i generisanje rasporeda elemenata na dijagramu. Slično kao i AMADEOS, TexToData omogućava generisanje RMBP, kao i korespondentne DDL skripte za generisanje šeme relacione baze podataka. Na serverskoj strani, TexToData orkestratorski servis upravlja procesom sinteze KMBP, dok procesom generisanja RMBP, procesom generisanja DDL skripte i procesom izvršavanja DDL skripte na SUBP upravlja AMADEOS orkestratorski servis.

2.3. *SpeeD*

SpeeD [13] je prvi onlajn veb-bazirani alat koji omogućava generisanje KMBP na osnovu govora, a trenutno je podržano devet jezika, i to: engleski, nemački, kineski, hindski, italijanski, francuski, turski, grčki i švedski.

Generisanje KMBP izvršava se u dve faze. U prvoj fazi, SpeeD klijentska veb aplikacija omogućava korisniku da govor konvertuje u korespondentni tekst. Korisnik ima mogućnost da učita datoteku sa snimljenim govorom (podržani su različiti formati datoteke kao što su *m4a*, *mp3*, *ogg*, *wav*, *wma* i dr.) ili da pokrene snimanje govora sa mikrofona, pri čemu se podaci sa mikrofona periodično šalju na server. Nakon što se govor konvertuje u tekst na serverskoj strani, a rezultat prikaže u odgovarajućem polju na klijentskoj veb aplikaciji, korisnik ima mogućnost da, po potrebi, manuelno doradi prepoznati tekst, pri čemu je moguće koristiti alate za proveru gramatike. Trenutno postoji podrška za pet alata za proveru gramatike, i to: Sapling, TextGears, ProWritingAid, LanguageTool i Jazzy.

U drugoj fazi, aplikacija omogućava korisniku da generiše KMBP na osnovu teksta generisanog u prvoj fazi. Po završe-

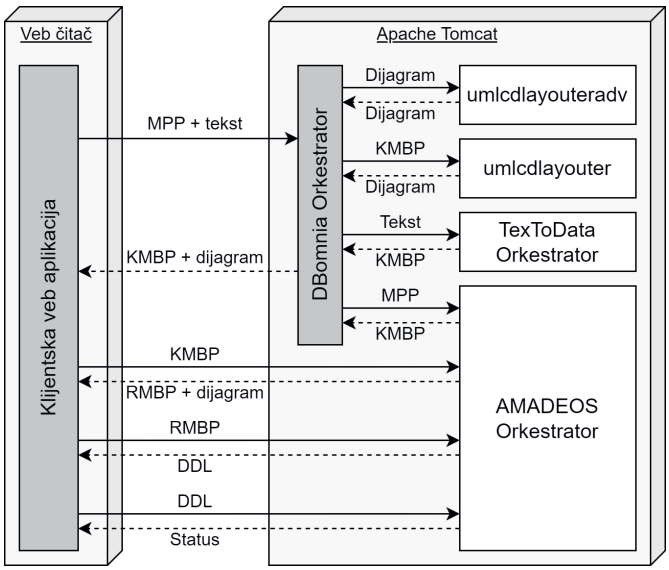
8 *Unified Modeling Language*
9 *Business Process Model and Notation*
10 *Extensible Markup Language Metadata Interchange*
11 *Extensible Markup Language Schema Definition*

nom generisanju, KMBP se vizuelizuje u veb čitaču. U prvoj fazi koristi se poseban servis za prepoznavanje govora, dok se u drugoj fazi koristi TexToData orkestratorski servis za generisanje KMBP. SpeeD, kao i AMADEOS i TexToData, podržava ostale faze direktnog inženjeringa relacione baze podataka, pri čemu se AMADEOS orkestratorski servis koristi za generisanje RMBP i DDL skripte te izvršavanje DDL skripte na SUBP.

2.4. *DBomnia*

DBomnia [14] je prvi onlajn veb-bazirani alat koji omogućava generisanje KMBP na osnovu skupa heterogenih ulaznih artefakata, a trenutno podržani tipovi artefakata su MPP i tekstualne specifikacije u prirodnom jeziku. DBomnia klijentska veb aplikacija omogućava korisniku da učita kolekciju MPP (BPMN ili UML AD) i unese tekstualnu specifikaciju u prirodnom jeziku, a zatim se, na serverskoj strani na osnovu unetih artefakata, vrši generisanje KMBP i rasporeda elemenata na dijagramu. Dijagram se vizuelizuje u veb čitaču. I DBomnia podržava ostale faze direktnog inženjeringa relacione baze podataka.

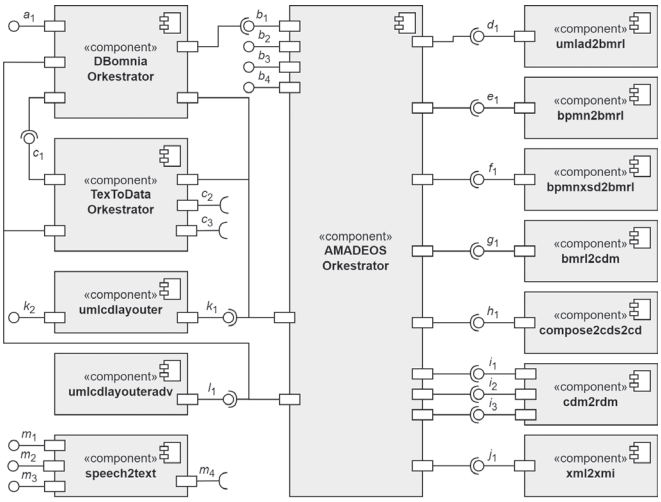
Na sl. 2 prikazana je arhitektura alata DBomnia. DBomnia orkestratorski servis koristi AMADEOS orkestratorski servis za generisanje KMBP na osnovu kolekcije MPP, a TexToData orkestratorski servis za generisanje KMBP na osnovu tekstualne specifikacije. Za generisanje dijagrama koriste se servisi koji su zajednički za sve orkestratorske servise (*umlcdlayouter* i *umlcdlayouteradv*). AMADEOS orkestratorski servis koristi se i za generisanje RMBP i DDL skripte te izvršavanje DDL skripte na SUBP.



Slika 2. Arhitektura alata DBomnia

3. PREGLED TRENUTNE ARHITEKTURE SISTEMA

U ovom poglavlju dat je pregled trenutne arhitekture sistema. UML dijagram komponenata (engl. *UML Component Diagram*) trenutnog sistema prikazan je na sl. 3. Svaka prikazana komponenta sistema razvijena je kao dinamička Java/Jakarta Enterprise Edition aplikacija, koja obezbeđuje odgovarajući REST[12] ili WebSocket programski interfejs (engl. *Application Programming Interface* – API). Pri tome, svaka krajnja tačka (engl. *endpoint*) programskog interfejsa odgovara jednom portu (tačka interakcije komponente) i korespondentnom obezbeđenom interfejsu (engl. *provided interface*) komponente prikazane na dijagramu. Na dijagramu nisu prikazane klijentske veb aplikacije (koje su opisane u prethodnom poglavlju). Klijentske veb aplikacije su statičke veb aplikacije, čija je implementacija bazirana na jeziku JavaScript i biblioteci za rad sa dijagramima mxGraph[13]. Sve prikazane komponente sistema izvršavaju se u okviru istog Apache Tomcat Servlet kontejnera, koji servira i klijentske veb aplikacije. U nastavku poglavlja dati su detaljniji opisi komponenata sistema.



Slika 3. UML dijagram komponenata trenutnog sistema

3.1. *Komponente za ekstrakciju specifičnih koncepata iz MPP*

Komponente *umlad2bmrl* [11], *bpmn2bmrl* [11] i *bpmnxsd-2bmrl* [15] omogućavaju ekstrakciju karakterističnih koncepata iz MPP, koji imaju semantički potencijal za generisanje KMBP. Ekstrahovani koncepti reprezentuju se u formi BMRL[14] koda [16]. Komponenta *umlad2bmrl* omogućava ekstrakciju specifičnih koncepata iz MPP reprezentovanog UML AD notacijom, dok komponente *bpmn2bmrl* i *bpmnxsd2bmrl* omogućavaju ekstrakciju specifičnih koncepata iz MPP reprezentovanog BPMN notacijom.

Svaka prethodno navedena komponenta ima po jednu krajnju tačku (na sl. 3 označene sa d_1, e_1 i f_1), koja prihvata HTTP[15] POST zahteve tipa *multipart/form-data*. Telo zahteva treba da sadrži polje *input* koje predstavlja MPP. Relativne putanje krajnjih tačaka d_1, e_1 i f_1 u odnosu na korespondentne komponente (*umlad2bmrl*, *bpmn2bmrl* i *bpmnxsd2bmrl*, respektivno) su iste, a to je putanja */services/convert*. Polje *input* zahteva koji se šalje na krajnje tačke d_1, e_1 i f_1 treba da bude XMI serijalizovani UML AD model, XMI

[12] *Representational State Transfer*
[13] https://jgraph.github.io/mxgraph
[14] *Business Model Representation Language*
[15] *Hypertext Transfer Protocol*

serijalizovani BPMN model i XSD serijalizovani BPMN model, respektivno. Za sve tri krajnje tačke, odgovor tipa *application/octet-stream* sa statusnim kodom 200 je korespondentni BMRL kôd. U slučaju da BMRL kôd ne može da se generiše, tada je statusni kôd odgovora 204 za krajnje tačke d_1 i e_1.

3.2. Komponenta *bmrl2cdm*

Komponenta *bmrl2cdm* [11] omogućava generisanje KMBP na osnovu BMRL koda. Jedina krajnja tačka g_1 (čija je relativna putanja */services/generate/cdm*) ove komponente prihvata HTTP POST zahteve čije telo tipa *multipart/form-data* treba da sadrži polje *input* koje predstavlja ulazni BMRL kôd, a kao rezultat sa statusnim kodom 200 vraća XMI serijalizovani UML CD koji reprezentuje KMBP. U slučaju greške pri generisanju KMBP, statusni kôd odgovara na zahtev je 204.

3.3. Komponenta *compose2cds2cd*

Komponenta *compose2cds2cd* [11] omogućava spajanje dva KMBP u jedan KMBP primenom odgovarajućeg skupa pravila. Ova komponenta ima, takođe, jednu krajnju tačku, i to h_1 (relativna putanja je */services/compose*) koja prihvata HTTP POST zahteve. Telo zahteva tipa *multipart/form-data* treba da sadrži tri polja. Polja *first_model* i *second_model* treba da budu dva ulazna KMBP (XMI serijalizovani UML CD), a polje *use_advanced_composition* treba da bude logička vrednost. Ako je vrednost polja *use_advanced_composition* vrednost *tačno*, tada se pri upoređivanju modela (tj. naziva koncepata u modelu) koristi: (i) neosetljivost na veličinu slova i (ii) Levenštajnova distanca [17] da bi se smanjio negativan uticaj tipografskih grešaka i različitog imenovanja koncepata na efektivnost procesa spajanja modela. Odgovor na zahtev sa statusnim kodom 200 je jedan (objedinjeni) KMBP (XMI serijalizovani UML CD), a ako dođe do greške u procesu spajanja modela tada je statusni kôd odgovara 204.

3.4. *Komponenta za direktni inženjering baze podataka*

Kako su u početku Mlab alati podržavali samo automatizovano generisanje KMBP, naknadno je razvijena komponenta *cdm2rdm* [10] u cilju podrške ostalih faza procesa projektovanja baze podataka. Komponenta *cdm2rdm* ima tri krajnje tačke, i to: i_1 (relativna putanja je */services/convert*) koja omogućava generisanje RMBP na osnovu KMBP, i_2 (relativna putanja je */services/convert/acceleo*) koja omogućava generisanje DDL skripte za konkretan SUBP na osnovu RMBP, i i_3 (relativna putanja je */services/convert/database*) koja omogućava izvršavanje DDL skripte na nekom dostupnom SUBP. Sve tri krajnje tačke prihvataju HTTP POST zahteve čije je telo tipa *multipart/form-data*.

Telo HTTP POST zahteva na krajnju tačku i_1 treba da sadrži polje *input* koje predstavlja ulazni KMBP (XMI serijalizovani UML CD), kao i tekstualna polja *integer_mapping*, *float_mapping*, *string_mapping*, *date_mapping*, *blob_mapping* i *logical_mapping* te polje *generate_index* logičkog tipa. Tekstualna polja treba da sadrže nazive korespondentnih tipova u RMBP, dok polje *generate_index* nosi informaciju o tome da li pri transformaciji KMBP u RMBP treba generisati i odgovarajuće indekse. Ako je RMBP uspešno generisan, odgovor sa statusnim kodom 200 sadrži XMI serijalizovani UML CD (RMBP). U suprotnom, u slučaju neuspešne transformacije KMBP u RMBP, statusni kôd odgovora na zahtev je 204.

Telo HTTP POST zahteva na krajnju tačku i_2 treba da sadrži dva polja, *input* i *dbms*. Polje *input* predstavlja ulazni RMBP (XMI serijalizovani UML CD), dok tekstualno polje *dbms* predstavlja identifikator tipa ciljnog SUBP. Dozvoljene vrednosti su: *MYSQL* za MySQL, *MICROSOFT_SQL* za Microsoft SQL Server, *ORACLE* za Oracle Database, *POSTGREE* za PostgreSQL i *IBM_D2* za IBM Db2. Odgovor sa statusnim kodom 200 je generisana DDL skripta. U slučaju greške, statusni kôd odgovora na zahtev je 204.

Telo HTTP POST zahteva na krajnju tačku i_3 treba da sadrži: tekstualno polje *dbms* koje predstavlja identifikator tipa ciljnog SUBP (skup dozvoljenih vrednosti naveden je u prethodnom pasusu), tekstualno polje *ddl* koje predstavlja DDL skriptu koju treba izvršiti, tekstualno polje *server* koje predstavlja lokaciju na kojoj se nalazi ciljni SUBP (npr. FQDN[16] ili IP[17] adresa), celobrojno polje *port*, tekstualno polje *username* (korisničko ime), tekstualno polje *password* (lozinka) i tekstualno polje *databaseName* (naziv baze podataka). Odgovor na zahtev sa statusnim kodom 200 je tipa *application/json*, a telo odgovora sadrži JSON[18] tekst. Ako je DDL skripta uspešno izvršena, tada celobrojno polje *status* JSON odgovora sadrži vrednost 0. U suprotnom, polje *status* sadrži vrednost 1, a polje *notification* sadrži tekstualnu poruku o grešci.

3.5. *Komponente za raspoređivanje elemenata na dijagramu*

U cilju vizuelizacije automatski generisanih modela (u početku KMBP, a kasnije i RMBP), razvijene su komponente koje omogućavaju kreiranje dijagrama (vizuelne reprezentacije modela), a to podrazumeva i definisanje rasporeda elemenata na dijagramu. Inicijalno je razvijena komponenta *umlcdlayouter* [18] koja omogućava kreiranje rasporeda elemenata na dijagramu koje je bazirano na jednostavnom raspoređivanju elemenata grafa (dijagram klasa može da se posmatra kao graf u kojem klase predstavljaju čvorove, a asocijacije predstavljaju grane). Kasnije, sa ciljem generisanja boljeg rasporeda elemenata na dijagramu, razvijena je komponenta *umlcdlayouteradv* [19], koja koristi modifikovani Sugiyama algoritam [20]. Dakle, komponenta *umlcdlayouter* omogućava generisanje inicijalnog/osnovnog rasporeda (izlaz je dijagram) na osnovu modela (ulaz), a komponenta *umlcdlayouteradv* omogućava poboljšavanje rasporeda (dijagram je i ulaz i izlaz). Za reprezentaciju dijagrama koristi se notacija EGMFN[19], koju koristi i alat Papyrus[20]. EGMFN reprezentacija bazirana je na principima OMG[21] *Diagram Interchange* specifikacije.

[16] *Fully Qualified Domain Name*
[17] *Internet Protocol*
[18] *JavaScript Object Notation*
[19] *Eclipse Graphical Modeling Framework Notation*
[20] https://eclipse.dev/papyrus/
[21] *Object Management Group*

Komponenta *umlcdlayouter* ima dve krajnje tačke, k_1 (relativna putanja je */services/layout*) i k_2 (relativna putanja je */services/layout/json*). Obe krajnje tačke prihvataju HTTP POST zahteve tipa *multipart/form-data* i omogućavaju generisanje dijagrama za zadati model, a razlika je jedino u formatu odgovora. Telo zahteva (za obe krajnje tačke) treba da sadrži polje *input* koje predstavlja XMI serijalizovani UML CD. Odgovor na zahtev na krajnju tačku k_1 sa statusnim kodom 200 je tipa *application/octet-stream* i predstavlja generisani dijagram (XMI serijalizovani EGMFN). U slučaju greške, statusni kôd odgovora je 204. Odgovor na zahtev na krajnju tačku k_2 sa statusnim kodom 200 je tipa *application/json* i sadrži JSON tekst. Ako je generisanje dijagrama uspešno, tada celobrojno polje *status* JSON odgovora sadrži vrednost 0, polje *uml* sadrži originalni XMI serijalizovani UML CD, a polje *notation* sadrži generisani dijagram (XMI serijalizovani EGMFN). U suprotnom, ako generisanje dijagrama nije uspešno izvršeno, tada polje *status* sadrži vrednost 1.

Komponenta *umlcdlayouteradv* ima jednu krajnju tačku l_1 (relativna putanja je */layout*), a HTTP POST zahtevi na datu krajnju tačku tipa *multipart/form-data* treba da sadrže dijagram (XMI serijalizovani EGMFN). Nakon poboljšavanja rasporeda elemenata na dijagramu, odgovor na zahtev sa statusnim kodom 200 je tipa *application/octet-stream* i sadrži modifikovani dijagram (XMI serijalizovani EGMFN).

3.6. Komponenta *xml2xmi*

Komponenta *xml2xmi* omogućava generisanje UML CD modela na osnovu reprezentacije dijagrama klasa koja se koristi u JavaScript biblioteci mxGraph. HTTP POST zahtev na krajnju tačku j_1 (relativna putanja je */services/xml2xmi/convert*) treba da sadrži XML reprezentaciju mxGraph UML CD, a odgovor na zahtev je XMI reprezentacija UML CD sa statusnim kodom 200.

3.7. *Orkestratori*

AMADEOS Orkestrator. Komponenta *AMADEOS Orkestrator* [11] upravlja procesom generisanja KMBP na osnovu kolekcije MPP, kao i ostalim fazama procesa projektovanja baze podataka. Krajnja tačka b_1 (relativna putanja je */service/generate/composed/cdmd*) omogućava generisanje KMBP na osnovu kolekcije MPP pa telo HTTP POST zahteva tipa *multipart/form-data* treba da sadrži polja: *inputs* koje predstavlja kolekciju ulaznih MPP, tekstualno polje *source_model_type* koje određuje tip ulazne kolekcije, i logičko polje *use_advanced_composition*. Dozvoljene vrednosti za polje *source_model_type* su: *AD* za XMI serijalizovani UML AD model, *BPMN* za XMI serijalizovani BPMN model i *BPMNXSD* za XSD serijalizovani BPMN model. Za svaki ulazni MPP koristi se odgovarajuća krajnja tačka (d_1, e_1 i f_1) komponente za generisanje BMRL koda (*umlad2bmrl*, *bpmn2bmrl* i *bpmnxsd2bmrl*, respektivno), a za svaki generisani BMRL kôd koristi se krajnja tačka g_1 komponente *bmrl2cdm* za generisanje KMBP. Za inkrementalnu kompoziciju generisanih KMBP u jedan KMBP koristi se krajnja tačka h_1 komponente *compose2cds2cd*, kojoj se prosleđuje i polje *use_advanced_composition*. Na kraju, za generisani KMBP generiše se i dijagram. Prvo se koristi krajnja tačka k_1 komponente *umlcdlayouter* za generisanje inicijalnog dijagrama, a zatim se koristi krajnja tačka l_1 komponente *umlcdlayouteradv* za unapređenje rasporeda elemenata na dijagramu. Odgovor na zahtev sa statusnim kodom 200 je tipa *application/json*, a telo odgovora sadrži JSON tekst. Ako je proces generisanja uspešno izvršen, tada celobrojno polje *status* JSON odgovora sadrži vrednost 0, polje *uml* sadrži KMBP (XMI serijalizovani UML CD), a polje *notation* sadrži dijagram (XMI serijalizovani EGMFN). U suprotnom, ako proces generisanja nije uspešno izvršen, polje *status* sadrži vrednost 1.

Pored opisane krajnje tačke, postoje još tri krajnje tačke koje omogućavaju orkestriranje procesima: (i) transformacije KMBP u RMBP (krajnja tačka b_2 čija je relativna putanja */services/generate/composed/cmd2rdm*), (ii) transformacije RMBP u DDL skriptu (krajnja tačka b_3 čija je relativna putanja*/services/generate/composed/rdm2ddl*) te (iii) izvršavanja DDL skripte na SUBP (krajnja tačka b_4 čija je relativna putanja*/services/generate/composed/ddl2database*). Sve tri krajnje tačke prihvataju HTTP POST zahteve tipa *multipart/form-data*, a odgovor na svaki zahtev sa statusnim kodom 200 je tipa *application/json* i sadrži JSON tekst.

Telo zahteva na krajnju tačku b_2 treba da sadrži polje *input* koje predstavlja ulazni KMBP u formi XML reprezentacije mxGraph UML CD, kao i tekstualna polja *integer_mapping*, *float_mapping*, *string_mapping*, *date_mapping*, *blob_mapping* i *logical_mapping* te polje *generate_index* logičkog tipa. Za konverziju XML reprezentacije mxGraph UML CD u XMI serijalizovani UML CD koristi se krajnja tačka j_1 komponente *xml2xmi*, a zatim se XMI serijalizovani UML CD, zajedno sa ostalim prethodno navedenim poljima zahteva, prosleđuje na krajnju tačku i_1 komponente *cdm2rdm* koja generiše RMBP (XMI serijalizovani UML CD). Potom se za generisani RMBP generiše i dijagram, a za to se koriste krajnja tačka k_1 komponente *umlcdlayouter* i krajnja tačka l_1 komponente *umlcdlayouteradv*. Ako je kompletan proces uspešno izvršen, tada celobrojno polje *status* JSON odgovora sadrži vrednost 0, polje *uml* sadrži RMBP (XMI serijalizovani UML CD), a polje *notation* sadrži dijagram (XMI serijalizovani EGMFN). U suprotnom, ako proces nije uspešno izvršen, polje *status* sadrži vrednost 1, a polje *notification* poruku o grešci.

Telo zahteva na krajnju tačku b_3 treba da sadrži polje *input* koje predstavlja ulazni RMBP u formi XML reprezentacije mxGraph UML CD, kao i tekstualno polje *dbms* koje predstavlja identifikator tipa ciljnog SUBP (dozvoljene vrednosti navedene su u odeljku 3.4). XML reprezentacija mxGraph UML CD kovertuje se u XMI serijalizovani UML CD korištenjem krajnje tačke j_1 komponente *xml2xmi*, a zatim se XMI serijalizovani UML CD i polje *dbms* prosleđuju na krajnju tačku i_2 komponente *cdm2rdm* koja generiše korespondentnu DDL skriptu. Ako je kompletan proces uspešno izvršen, tada celobrojno polje *status* JSON odgovora sadrži vrednost 0, a polje *script* sadrži generisanu DDL skriptu. U suprotnom, ako proces nije uspešno izvršen, polje *status* sadrži vrednost 1, a polje *notification* poruku o grešci.

Telo zahteva na krajnju tačku b_4 treba da sadrži sledeća tekstualna polja: *dbms* (identifikator tipa ciljnog SUBP), *server* (lokacija), *port*, *username* (korisničko ime), *password* (lozinka), *databaseName* (naziv baze podataka) i *input* (DDL skripta). Sva polja iz zahteva se prosleđuju na krajnju tačku i_3 komponente *cdm2rdm* koja omogućava izvršavanje specifikovane DDL skripte na specifikovanom SUBP. Ako je proces uspešno izvršen, tada celobrojno polje *status* JSON odgovora sadrži vrednost 0. U suprotnom, ako proces nije uspešno izvršen, polje *status* sadrži vrednost 1, a polje *notification* poruku o grešci.

TexToData Orkestrator. Komponenta *TexToData Orkestrator* [12] upravlja procesom generisanja KMBP na osnovu tekstualne specifikacije u prirodnom jeziku. Telo HTTP POST zahteva tipa *application/json* na krajnju tačku c_1 (relativna putanja je */rest/modeler*) treba da sadrži JSON tekst sa poljima: *text* koje sadrži tekstualnu specifikaciju u prirodnom jeziku, *translationService* koje sadrži identifikator servisa za prevođenje tekstualne specifikacije na engleski jezik, i *fromLang* koje sadrži identifikator jezika tekstualne specifikacije. Dozvoljene vrednosti za polje *translationService* su: *yandex* za Yandex servis, *mymemory* za MyMemory servis, *microsofttext* za Microsoft Translator Text servis, *deep* za DeepTranslate servis, *open* za OpenTranslator servis i *text* za Text Translator servis. Ako identifikator servisa za prevođenje nije specifikovan ili nije pravilno specifikovan, tada se koristi podrazumevani servis (Yandex). Ako nije specifikovan identifikator jezika, tada se prvo korištenjem specifikovanog servisa identifikuje jezik ulaznog teksta (zahtevani interfejs c_2). Dalje, ako tekstualna specifikacija nije na engleskom jeziku, korištenjem specifikovanog servisa (zahtevani interfejs c_2) vrši se prevođenje tekstualne specifikacije na engleski jezik. Pored eksternog servisa za identifikovanje jezika i prevođenja teksta, komponenta *TexToData Orkestrator* koristi još jedan eksterni servis (zahtevani interfejs c_3), i to TextRazor za obradu teksta u prirodnom jeziku (engl. *Natural Language Processing* – NLP) kojem se šalje (prevedena) tekstualna specifikacija na engleskom jeziku. Ako originalna tekstualna specifikacija nije na engleskom jeziku, tada se nakon generisanja KMBP, ponovo koristi servis za prevođenje (zahtevani interfejs c_2), pri čemu se nazivi koncepata generisanog KMBP prevode na jezik originalne tekstualne specifikacije. Na kraju, generiše se i dijagram, pri čemu se prvo koristi krajnja tačka k_1 komponente *umlcdlayouter*, a zatim krajnja tačka l_1 komponente *umlcdlayouteradv*. Dodatno, za KMBP kreira se i XML reprezentacija mxGraph UML CD. Odgovor na zahtev sa statusnim kodom 200 je tipa *application/json*, a telo odgovora sadrži JSON tekst. Ako je kompletan proces uspešno izvršen, tada polje *englishText* JSON odgovora sadrži tekstualnu specifikaciju prevedenu na engleski jezik, polje *originalText* sadrži originalnu tekstualnu specifikaciju, polje *originalTextLang* sadrži identifikator jezika originalne tekstualne specifikacije, polje *uml* sadrži KMBP (XMI serijalizovani UML CD), dok polja *layout* i *mxGraphUml* sadrže dijagram (XMI serijalizovani EGMFN i XML reprezentacija mxGraph UML CD, respektivno). U suprotnom, ako proces nije uspešno izvršen, polje *error* JSON odgovora sadrži tekst poruke o grešci.

DBomnia Orkestrator. Komponenta *DBomnia Orkestrator* [14] upravlja procesom generisanja KMBP na osnovu skupa heterogenih ulaznih artefakata. Komponenta ima krajnju tačku a_1 (relativna putanja je */services/cdm/json*) koja prihvata HTTP POST zahteve čije telo tipa *multipart/form-data* treba da sadrži: tekstualno polje *input_text* koje predstavlja tekstualnu specifikaciju u prirodnom jeziku, tekstualno polje *source_model_type* koje određuje tip ulazne kolekcije MPP, logičko polje *use_advanced_composition* i polje *inputs* koje predstavlja kolekciju MPP. Za generisanje KMBP na osnovu tekstualne specifikacije u prirodnom jeziku koristi se krajnja tačka c_1 komponente *TexToData Orkestrator* kojoj se prosleđuje ulazna tekstualna specifikacija (polje *input_text*) i identifikator servisa za provođenje. Trenutno, uvek se koristi isti servis (Yandex). Za generisanje KMBP na osnovu kolekcije MPP, koristi se krajnja tačka b_1 komponente *AMADEOS Orkestrator* kojoj se prosleđuju polja *source_model_type*, *use_advanced_composition* i *inputs*. Nakon generisanja pojedinačnih KMBP, komponenta *DBomnia Orkestrator* integriše generisane KMBP u jedan KMBP, a na kraju, korištenjem krajnje tačke k_1 komponente *umlcdlayouter* i krajnje tačke l_1 komponente *umlcdlayouteradv* generiše se dijagram za integrisani KMBP. Odgovor na zahtev sa statusnim kodom 200 je tipa *application/json*, a telo odgovora sadrži JSON tekst. Ako je proces generisanja uspešno izvršen, tada celobrojno polje *status* JSON odgovora sadrži vrednost 0, polje *uml* sadrži integrisani KMBP (XMI serijalizovani UML CD), a polje *notation* sadrži dijagram (XMI serijalizovani EGMFN). U suprotnom, ako proces generisanja nije uspešno izvršen, polje *status* sadrži vrednost 1.

3.8. Komponenta *speech2text*

Komponenta *speech2text* omogućava konverziju govora u tekst, kao i proveru gramatike teksta. Komponenta ima tri krajnje tačke, pri čemu dve krajnje tačke (m_1 i m_2) čine REST API, a jedna (m_3) predstavlja WebSocket API serversku krajnju tačku.

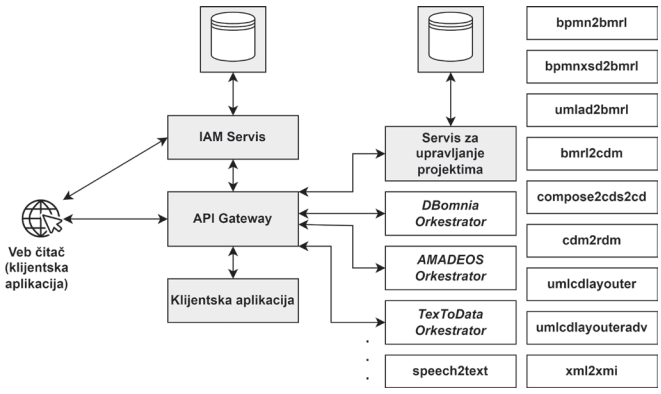
Krajnja tačka m_1 (relativna putanja je */rest/languages*) omogućava dohvatanje podržanih jezika, dok krajnja tačka m_2 (relativna putanja je */rest/analyzetext*) omogućava proveru gramatike teksta, pri čemu se koriste eksterni servisi (zahtevani interfejs m_4). Krajnja tačka m_1 prihvata HTTP GET zahteve (bez parametara). Odgovor na zahtev sa statusnim kodom 200 je tipa *application/json* i sadrži JSON niz stringova, pri čemu svaki string predstavlja naziv podržanog jezika. Krajnja tačka m_2 prihvata HTTP POST zahteve tipa *application/json*, čije telo treba da sadrži JSON tekst sa poljima: *text* koje sadrži tekstualnu specifikaciju u prirodnom jeziku, *tool* koje sadrži identifikator servisa za proveru gramatike i *language* koje sadrži identifikator jezika tekstualne specifikacije. Dozvoljene vrednosti za polje *tool* odgovaraju nazivima dostupnih servisa za proveru gramatike: Sapling, TextGears, ProWritingAid, LanguageTool i Jazzy. Odgovor na zahtev sa statusnim kodom 200 je tipa *application/json*, a telo odgovora sadrži JSON tekst koji predstavlja niz sugestija. Pri tome, svaka sugestija je objekat koji sadrži polja: *original* koje sadrži označeni deo originalnog teksta za koji postoje sugestije, *start* koje sadrži početnu poziciju označenog dela teksta, *end* koje sadrži krajnju poziciju označenog dela teksta, i *suggestions* koje predstavlja niz stringova, a svaki string predstavlja jednu sugestiju.

WebSocket API krajnja tačka m_3 (relativna putanja je */ws*) omogućava prijem audio zapisa, a vraća prepoznati tekst. Neposredno pre slanja audio zapisa, na krajnju tačku m_3, potrebno je poslati dve tekstualne poruke, i to prvo tekstualnu poruku koja sadrži identifikator jezika audio zapisa, a zatim tekstualnu poruku koja sadrži informaciju o tome da li se šalje datoteka sa snimljenim govorom (poruka *file*) ili se radi o periodičnom slanju audio zapisa sa mikrofona (poruka *start*). Ako se radi o periodičnom slanju audio zapisa, tada je pri završetku slanja audio zapisa na krajnju tačku m_3 potrebno poslati poruku *end*. Server vraća odgovor rečenicu po rečenicu, tj. rečenica se vraća odmah nakon prepoznavanja.

4. PREDLOG NOVE ARHITEKTURE SISTEMA

Razvoj alata i komponenata opisanih u prethodnim poglavljima karakteriše: dug period razvoja (preko 15 godina), relativno velik broj povremenih učesnika u razvoju (osim četiri stalna člana, ostali saradnici Mlab grupe učestovali su u razvoju kroz projektne zadatke i završne radove prvog i drugog ciklusa studija), korištenje različitih tehnologija i verzija biblioteka pri razvoju (delom uzrokovano i periodom razvoja) itd. Navedene karakteristike razvoja predstavljaju i izazove zbog kojih je održavanje trenutne verzije sistema otežano, jer se sve komponente izvršavaju u okviru istog Apache Tomcat Servlet kontejnera.

U skladu sa dugoročnim ciljem istraživanja, planiran je razvoj novih alata i komponenata, kao i unapređenje svih postojećih alata i odgovarajućih komponenata. Zbog toga je prvo neophodno prilagoditi postojeći sistem tako da olakšava ugrađivanje novih komponenata, kao i zamenu postojećih komponenata unapređenim verzijama. Pri tome, unapređene i nove komponente mogu da koriste različite (novije) verzije pojedinih biblioteka i tehnologija, odnosno potpuno različit skup tehnologija. S obzirom na to da je sistem već dekomponovan, upotreba virtuelizacije i kontejnerske tehnologije nameće se kao moguće rešenje koje pored lakšeg skaliranja sistema, omogućava i tehnološku nezavisnost, veću otpornost, visoku dostupnost, lakšu nadogradnju i kontinuiranu isporuku [21]. Na sl. 4 prikazana je blok šema predloga novog sistema, pri čemu svaki pravougaonik predstavlja jedan kontejner. Kontejneri koji sadrže postojeće komponente predstavljeni su pravougaonikom sa belom pozadinom, dok su pravougaonikom sa sivom pozadinom predstavljeni kontejneri sa novim servisima neophodnim u novoj verziji sistema.



Slika 4. Blok šema predložene arhitekture sistema

Važan aspekat koji treba rešiti u narednoj iteraciji je poboljšanje sigurnosti sistema. U trenutnoj verziji sistema svi servisi su javno dostupni. U cilju povećanja sigurnosti planirano je da se u narednoj iteraciji sistema obezbedi pristup samo registrovanim korisnicima. Zbog toga se predlaže implementacija i uvođenje nekoliko novih komponenata.

Komponenta *API Gateway* [22] predstavlja jedinstvenu ulaznu tačku (engl. *single entry point*) u unapređeni sistem. Primarna uloga ove komponente u predloženom sistemu je upravljanje komunikacijom između eksternog klijenta i pozadinskih komponenata. *API Gateway* obavlja niz funkcionalnosti koje uključuju: usmeravanje zahteva prema odgovarajućim komponentama na osnovu definisanih pravila, autentifikaciju korisnika i autorizaciju korisnika. Osim toga, *API Gateway* omogućava implementaciju različitih politika upravljanja saobraćajem kao što su: kontrola broja zahteva (engl. *rate limiting*), keširanje, balansiranje opterećenja (engl. *load balancing*) i transformacija zahteva (npr. promena formata poruke ili zaglavlja). *API Gateway* omogućava centralizovanu kontrolu nad pristupom komponentama, a tipično omogućava logovanje informacija te pruža uvid u zahteve i odgovore, što je ključno za praćenje komunikacije, otklanjanje grešaka i rešavanje problema.

Dok komponenta *API Gateway* obavlja funkciju spoljašnjeg kontrolnog sloja koji obezbeđuje autentifikaciju, autorizaciju, kontrolu i usmeravanje saobraćaja, interne orkestratorske komponente upravljaju poslovnim tokovima, tj. pozivaju druge interne servise, prikupljaju odgovore te formiraju i vraćaju odgovor na zahtev. Korištenjem orkestratorskih komponenata apstrahuje se kompleksnost sistema i pojednostavljuje interakcija spoljašnjih klijenata sa internim komponentama sistema. Zbog svega navedenog, tri postojeće orkestratorske komponente (*DBomnia Orkestrator*, *AMADEOS Orkestrator* i *TexToData Orkestrator*) treba prilagoditi novom sistemu. Iako osnovna svrha ovih servisa ostaje ista i u novom sistemu, potrebno je modifikovati implementaciju tako da odgovara distribuiranoj arhitekturi. Kontejneri koji sadrže ove komponente na sl. 4 predstavljeni su pravougaonikom sa belom pozadinom, pri čemu je naziv komponente napisan u kurzivu.

Opisana podela odgovornosti između komponente *API Gateway* i orkestratorskih komponenata doprinosi boljoj modularnosti, fleksibilnosti i lakšem održavanju sistema, ali donosi i odgovarajuća ograničenja, kao i izazove pri implementaciji takvog sistema. Najznačajniji nedostatak jeste uvećana složenost infrastrukture. Uvođenjem dodatnih slojeva između klijenta i pojedinih servisa, sistem dobija nove komponente koje je potrebno konfigurisati i održavati. Otkazivanje komponente *API Gateway* i/ili orkestratorskih komponenata može dovesti do prekida u komunikaciji između klijenta i sistema. S obzirom na to da su ove komponente potencijalne tačke otkaza (engl. *single point of failure*), nameće se potreba za redundantnim instancama, implementacijom mehanizma oporavka i stalnim nadzorom. Takođe, drugi izazov odnosi se na performanse i kašnjenja (engl. *latency*). Svaki zahtev koji prolazi kroz *API Gateway* prolazi kroz dodatni sloj obrade (u odnosu na trenutnu implementaciju) što može povećati vreme odziva sistema, naročito u situacijama sa velikim brojem istovremenih zahteva ili složenim pravilima obrade.

Pošto je u novoj verziji sistema potrebno obezbediti pristup samo registrovanim korisnicima, dolazni zahtevi treba da budu autorizovani. Za identifikaciju i upravljanje pristupom (engl. *Identity and Access Management* – IAM) koristi se posebna komponenta, *IAM Servis*, koja treba da podržava standardne IAM protokole kao što je OAuth 2.0[22]. Umesto razvoja sopstvenog sistema, u praksi se sve češće koriste specijalizovana rešenja otvorenog koda (npr. Keycloak[23]), koja omogućavaju standardizovan, skalabilan i siguran pristup za upravljanje korisničkim identitetima. Jedna od najvećih prednosti takvih sistema je njihova potpuna usklađenost sa savremenim sigurnosnim standardima, kao što su OAuth 2.0 i OpenID Connect[24]. Sopstvena rešenja često implementiraju pojednostavljene ili nestandardne mehanizme prijave, što može otežati integraciju sa spoljnim sistemima i povećati rizik od sigurnosnih propusta. Takođe, IAM servisi najčešće podržavaju integraciju sa različitim sistemima autentifikacije (npr. društvene mreže). Komponenta *IAM Servis* može da koristi zaseban SUBP (u zasebnom kontejneru) za čuvanje informacija o korisnicima.

Važno je pomenuti da sva komunikacija treba da bude zaštićena korištenjem HTTPS[25] protokola, uz onemogućavanje korištenja zastarelih protokola poput TLS[26] 1.0 i 1.1, te onemogućavanje korištenja slabih kombinacija kriptografskih algoritama (engl. *cipher suites*).

Radi jednostavnosti, na sl. 4 nisu prikazane veze između komponente *API Gateway* i svih postojećih komponenata opisanih u prethodnom poglavlju, a izostavljene su i veze između pojedinih postojećih komponenata. S obzirom na to da je komunikacija između postojećih komponenata već implementirana i ugrađena u same komponente, u narednoj iteraciji neće biti razmatrano uvođenje posebne (namenske) infrastrukture za komunikaciju između komponenata, ali će ovo otvoreno pitanje biti deo budućeg rada.

Prethodno opisani cilj (poboljšanje sigurnosti sistema) u skladu je sa planiranom nadogradnjom sistema kojom je predviđeno da se registrovanim korisnicima omogući rad sa *projektima*, pri čemu registrovani korisnik treba da ima mogućnost da: (i) kreira projekte, (ii) dodaje (učitava) ulazne artefakte u projekat, (iii) generiše KMBP na osnovu ulaznih artefakata, (iv) manuelno projektuje KMBP *od nule*, (v) generiše RMBP na osnovu KMBP, (vi) sačuva KMBP i RMBP i naknadno nastavi sa radom, (vii) preuzima KMBP kao i RMBP itd. Zbog svega navedenog, planiran je razvoj nove klijentske aplikacije koja treba da omogućava registraciju korisnika kao i (registrovanim korisnicima) rad sa projektima i pripadajućim artefaktima. U novu klijentsku aplikaciju treba da budu ugrađene klijentske aplikacije prikazane u poglavlju 2, koje omogućavaju automatizovano projektovanje KMBP kao i ostale faze direktnog inženjeringa relacione baze podataka. Nova klijentska aplikacija može da bude razvijena korištenjem modernog JavaScript radnog okvira i/ili biblioteke kao što su Angular[27], Vue.js[28], React[29] i sl. Komponenta *Klijentska aplikacija*, koja je na sl. 4 predstavljena pravougaonikom sa sivom pozadinom, treba da servira klijentsku aplikaciju. Pored razvoja nove klijentske aplikacije, neophodno je razviti i servis za upravljanje projektima, koji koristi SUBP (koji se nalazi u zasebnom kontejneru) za čuvanje informacija o projektima i pripadajućim artefaktima.

[22] https://oauth.net/2/

[23] https://www.keycloak.org

[24] https://openid.net

[25] *Hypertext Transfer Protocol Secure*

[26] *Transport Layer Security*

5. ZAKLJUČAK

U ovom radu dat je pregled sistema za automatizovano projektovanje baze podataka, koji je razvijen u istraživačkoj laboratoriji Mlab na Elektrotehničkom fakultetu Univerziteta u Banjoj Luci. Sistem se sastoji od četiri onlajn alata (AMADEOS, TexToData, SpeeD i DBomnia) te ukupno 13 komponenata, pri čemu svi Mlab alati dele pojedine zajedničke komponente. AMADEOS je prvi onlajn veb-bazirani alat koji omogućava generisanje KMBP na osnovu kolekcije MPP. TexToData je onlajn višejezični (podržano preko 100 jezika) veb-bazirani alat za generisanje KMBP na osnovu teksta. SpeeD je prvi onlajn veb-bazirani alat koji omogućava generisanje KMBP na osnovu govora. DBomnia je prvi onlajn veb-bazirani alat koji omogućava generisanje KMBP na osnovu skupa heterogenih ulaznih artefakata (trenutno na osnovu kolekcije MPP i teksta u prirodnom jeziku). Pored generisanja KMBP, svi alati podržavaju i ostale faze direktnog inženjeringa relacione baze podataka.

Pored pregleda svih alata i detaljnog pregleda svih komponenata, u ovom radu dat je i predlog nove arhitekture sistema. Nova skalabilna servisno-orijentisana arhitektura sistema treba da olakša i rešavanje važnog aspekta, a to je poboljšanje sigurnosti u odnosu na trenutnu verziju sistema. U radu je dat okvirni predlog nove arhitekture, a detalji će biti razmotreni kada budu poznati detalji dostupnih resursa za implementaciju sistema.

U skladu sa dugoročnim ciljem istraživanja, nakon implementacije predložene arhitekture sistema, u nastavku istraživanja planirana su unapređenja pristupa i alata, odnosno rad na otvorenim pitanjima, kao što je primena veštačke inteligencije i velikih jezičkih modela u sistemu za automatsko projektovanje ciljne baze podataka.

LITERATURA

[1] Date, C.: An Introduction to Database Systems, 8th ed. AddisonWesley (2003)
[2] Brđanin, D.: Automatizovano projektovanje relacionih baza podataka. Univerzitet u Banjoj Luci, Elektrotehnički fakultet (2024)
[3] Chen, P.: English sentence structure and entity-relationship diagrams. Information Sciences 29(2-3), 127-149 (1983)
[4] Omar, N., Hanna, P., McKevitt, P.: Heuristics-based entity-relationship modelling through natural language processing. In: Proceedings of the Irish Conference on Artificial Intelligence and Cognitive Science. AICS 2004, 302-313 (2004)
[5] Harmain, H., Gaizauskas, R.: CM-Builder: a natural language-based CASE tool for object-oriented analysis. Automated Software Engineering 10(2), 157–181 (2003)

[27] https://angular.dev

[28] https://vuejs.org

[29] https://react.dev

[6] Overmyer, S., Benoit, L., Owen, R.: Conceptual modeling through linguistic analysis using LIDA. In: Proceedings of the 23rd International Conference on Software Engineering. ICSE 2001, IEEE, 401-410 (2001)
[7] Nikiforova, O., Gusarovs, K., Gorbiks, O., Pavlova, N.: BrainTool: a tool for generation of the UML class diagrams. In: Proceedings of the 7th International Conference on Software Engineering Advances. ICSEA 2012, IARIA, 60-69 (2012)
[8] Rodriguez, A., Garcia-Rodriguez de Guzman, I., Fernandez-Medina, E., Piattini, M.: Semi-formal transformation of secure business processes into analysis class and use case models: an MDA approach. Information and Software Technology 52(9), 945–971 (2010)
[9] Kriouile, A., Addamssiri, N., Gadi, T.: An MDA method for automatic transformation of models from CIM to PIM. American Journal of Software Engineering and Applications 4(1), 1–14 (2015)
[10] Spasic, Z., Vukotic, A., Brdjanin, D., Banjac, D., Banjac, G.: UML-based Forward Database Engineering. 22nd Int. Symposium INFOTEH-JAHORINA, 1-6 (2023)
[11] Brdjanin, D., Vukotic, A., Banjac, D., Banjac, G., Maric, S.: Automatic Derivation of the Initial Conceptual Database Model from a Set of Business Process Models. Computer Science and Information Systems 19(1), 455-493 (2022)
[12] Banjac, D., Matic, M., Cvijanovic, N., Brdjanin, D., Banjac, G., Stojisavljevic, D.: Employing Multiple Online Translation Services in a Multilingual Database Design Tool. In: Tekli, J., et al. New Trends in Database and Information Systems. ADBIS 2024. Communications in Computer and Information Science, Vol. 2186, 238-249. Springer Nature Switzerland, Cham (2025)
[13] Brdjanin, D., Banjac, G., Keserovic, D., Babic, N., Golubovic, N.: Combining Speech Processing and Text Processing in Conceptual Database Design. TELFOR Journal 16(1), 8-13 (2024)
[14] Banjac, G., Brdjanin, D., Banjac, D.: Automatic Conceptual Database Design based on Heterogeneous Source Artifacts. Computer Science and Information Systems 21(4), 1913-1961 (2024)
[15] Brdjanin, D., Ilic, S., Banjac, G., Banjac, D., Maric, S.: Automatic derivation of conceptual database models from differently serialized business process models. Software and Systems Modeling 20(1), 89-115 (2021)
[16] Brdjanin, D., Banjac, D., Banjac, G., Maric, S.: Automated Two-phase Business Model-driven Synthesis of Conceptual Database Models. Computer Science and Information Systems 16(2), 657-688 (2019)
[17] Levenshtein, I.V.: Binary codes capable of correcting deletions, insertions and reversals. Soviet Physics Doklady 10(8), 707–710 (1966)
[18] Dujlovic, I., Obradovic, N., Kelec, A., Brdjanin, D., Banjac, G., Banjac, D.: An Approach to Web-based Visualization of Automatically Generated Data Models. IEEE EUROCON 2019 – 18th International Conference on Smart Technologies, Novi Sad, Serbia, 1-6 (2019)
[19] Miscevic, M., Brdjanin, D.: An Approach to Automatic Layout of Eclipse Papyrus Class Diagrams. 2019 27th Telecommunications Forum (TELFOR), Belgrade, Serbia, 1-4 (2019)
[20] Seemann, J.: Extending the Sugiyama algorithm for drawing UML class diagrams: Towards automatic layout of object-oriented software diagrams. In: DiBattista, G. (eds) Graph Drawing. GD. Lecture Notes in Computer Science, Vol. 1353, 415-424. Springer, Berlin, Heidelberg (1997)
[21] Amaral, M., Polo, J., Carrera, D., Mohomed, I., Unuvar, M., Steinder, M.: Performance Evaluation of Microservices Architectures Using Containers. 2015 IEEE 14th International Symposium on Network Computing and Applications, Cambridge, MA, USA, 27-34 (2015)
[22] Zhao, J.T., Jing, S.Y., Jiang, L.Z.: Management of API gateway based on micro-service architecture. Journal of Physics: Conference Series 1087(3), 032032 (2018)

Goran Banjac je viši asistent i PhD student na Elektrotehničkom fakultetu Univeziteta u Banjoj Luci (Bosna i Hercegovina). Član je istraživačke grupe M-lab
Kontakt: goran.banjac@etf.unibl.org
Oblasti interesovanja: Razvoj softvera vođen modelom, modelovanje poslovnih procesa, baze podataka, UML.

Zoran Đurić je redovni profesor na Elektrotehničkom fakultetu Univeziteta u Banjoj Luci (Bosna i Hercegovina).
Kontakt: zoran.djuric@etf.unibl.org
Oblasti interesovanja: Sigurnost, kriptografija, PKI, platni sistemi i protokoli, formalna verifikacija, mašinsko učenje, data science, objektno-orijentisano modelovanje i programiranje, Internet programiranje, razvoj mobilnih aplikacija, XML-bazirana međuoperativnost, Web servisi, računarske mreže, penetration testing, sistem integracija.

Dražen Brđanin je redovni profesor na Elektrotehničkom fakultetu Univeziteta u Banjoj Luci (Bosna i Hercegovina). Rukovodilac je istraživačke grupe M-lab. Argo Informacione Tehnologije, Beograd
Kontakt: drazen.brdjanin@etf.unibl.org
Oblasti interesovanja: Informacioni sistemi, baze podataka, softversko inženjerstvo.

Danijela Banjac je viša asistentkinja i PhD studentkinja na Elektrotehničkom fakultetu Univeziteta u Banjoj Luci (Bosna i Hercegovina). Član je istraživačke grupe M-lab.
Kontakt: danijela.banjac@etf.unibl.org
Oblasti interesovanja: Razvoj softvera vođen modelom, modelovanje poslovnih procesa, objektno-orijentisani informacioni sistemi, UML.