UNIVERZITET „DŽEMAL BIJEDIĆ“ U MOSTARU
FAKULTET INFORMACIJSKIH TEHNOLOGIJA

Akademska 2024./2025. godina

Predmet: Softverske arhitekture

PAPER

Dokumentacija softverske arhitekture

Odobrio:
Prof. dr. sc. Dražena Gašpar

Autor:
Zaim Mehić, IB210011

Mostar, januar 2024.

Verzija: 2.0

SADRŽAJ

1. UVOD

PAPER (Patient Admission Prediction for an Emergency Room) je aplikacija za predikciju broja pacijenata u Službama hitne medicinske pomoći i upravljanje osobljem iste. Predviđanje se vrši na osnovu historijskih podataka o broju pacijenata po svakom danu pojedinačno vodeći računa o relevantnim podacima koji mogu imati utjecaj na navedeni broj (kao što su temperatura zraka, vlažnost, padavine, dan u sedmici, praznik, događaji, itd). Ovaj dokument opisuje strukturu softverske arhitekture koja predstavlja temelj adekvatnog funkcionisanja ove aplikacije.

1.1. Svrha

Posljednjih godina svjedočimo rastu potrebe za hitnim medicinskim uslugama u Službama hitne medicinske pomoći, a nažalost kapaciteti velikog broj Službi nisu dovoljni da odgovore na pomenuti rast. Navedeno je rezultovalo prevelikim brojem pacijenata što nadalje utječe ne samo na njihovo zadovoljstvo, nego i na kvalitet liječenja u cjelini.

Izgradnja kapaciteta adekvatnih da odgovore na skoro pa svako opterećenje sistema hitne medicinske pomoći zahtjevalo bi značajna financijska i vremenska ulaganja koja bi za cilj imala porast broja zaposlenih, usavršavanje postojećeg kadra, nadogradnju postojeće infrastrukture i nabavku medicinske opreme. Sve navedeno stvara prostor za dodatnu analizu problema i kreiranje inovativnih rješenja koji za cilj imaju da proizvedu slične efekte uz dosta manja ulaganja.

Svrha PAPER aplikacije je da koristeći jedan od statističkih modela za predikciju kreira pretpostavku o broju pacijenata za određeni dan na osnovu parametara koji imaju vidljiv utjecaj na navedeni broj, te na taj način omogući menadžmentu Službe za hitnu medicinsku pomoć da adekvatno rasporedi resurse koje ima na raspolaganju.

Ovakvi alati još uvijek se razvijaju diljem svijeta kako bi rasteretili sistem i osigurali kvalitetnu medicinsku pomoć krucijalnu za dugoročni opstanak navedenog sistema, ali komercijalizacija ovakvih alata još uvijek nije zaživjela ponajviše iz razloga što se radi o veoma osjetljivom

području koje daje minimalan prostor za pogreške. Uprkos tome samo je pitanje vremena kada će razvoj umjetne inteligencije i različitih modela za predikciju omogućiti primjenu ovakvih rješenja u realnom svijetu, odnosno donijeti prijeko potrebno olakšanje na veoma opterećeni medicinski sistem.

1.2. Obim

Sistem treba da osigura tri modula za tri različita tipa korisnika (šef Sužbe, zaposlenik Gradske uprave i administrator), jedan interni modul za obradu podataka, treniranje i testiranje modela, te kreiranje samih predviđanja.

Modul koji koristi šef Službe treba da im obezbijedi mogućnost upravljanja ljudskim i materijalnim resursima kojima raspolaže ta Služba konkretno.

To obuhvata pregled, dodavanje, brisanje i uređivanje svih informacija vezanih za medicinsku opremu koju imaju na raspolaganju. Upravljanje osobljem pored svih navedenih funkcionalnosti obuhvata i raspoređivanje po zadacima (smjenama) za naredni period. Potrebno je omogućiti šefu da u bilo kojem momentu dobije previđanje broja pacijenata za naredni period (npr. sedam dana), a od šefa se traži da vodi računa o unosu u već predviđeni sistem za evidenciju pacijenata, dok PAPER treba da vrši komunikaciju sa tim postojećim sistemom i preuzme vrijednost o ukupnom broju pacijenata za taj određeni dan.

Zaposlenik Gradske uprave u svom modulu treba imati mogućnost da u sistem evidentira već navedene parametre za buduće datume, te da evidentira sve događaje za područje koje pokriva Služba hitne medicinske pomoći koji će se održati u narednom periodu.

Modul za administraciju treba da omogući administratoru da u bilo kojem momentu može korigovati, dodati, obrisati ili čitati vrijednosti sa bilo kojeg dijela sistema kako bi se osigurao brz i efikasan oporavak od bilo kakvih nepredviđenih okolnosti.

Interni modul zadužen je da obrađuje ulazne informacije koje dostavljaju šef Službe preko postojećeg sistema za evidenciju pacijenata, podatke koje dobije interakcijom sa sistemom ustanove zadužene za hidrometeorološke aktivnosti, te podatke koje unese uposlenik Gradske

uprave, nadalje na osnovu datih informacija donosi tražena predviđanja i u adekvatnom formatu ih šalje na uvid.

1.3. Definicije i skraćenice

Služba za hitnu medicinsku pomoć – posebna oblast zdravstvene zaštite na primarnom nivou, koja se organizuje u cilju poduzimanja nephodne i neodložne medicinske intervencije;

Gradska uprava – dio izvršne vlasti zadužen za upravljanje i obavljanje djelatnosti od značaja za jedinicu lokalne samouprave;

ER – (*engl.* Emergency Room) Služba za hitnu medicinsku pomoć;

SARIMAX – (*engl.* Seasonal Auroregressive Integrated Moving Average with Exogenous Regressors) statistički model dizajniran da pronađe i predviđa skrivene uzorke, trendove i sezone u adekvatnim skupovima podataka;

TMS – (*engl.* Task Management System) sistem za upravljanje zadacima;

HMP – Hitna medicinska pomoć;

1.4. Zainteresirane strane

U zainteresirane strane ove aplikacije ubraja se zasigurno cijela Služba za hitnu medicinsku pomoć i ostatak ustanove koje je ista dio jer aplikacija kao takva za cilj ima da višestruku unaprijedi rad ove jedinice, a samim tim doprinese cjelokupnom zdravstvenom sistemu. Zaposlenici i menadžment će ravnomjernije rasporediti zadatke što će pridonijeti ugodnijem radnom okruženju i smanjenju stresa koji je izrazito opasan u osjetljivim oblastima kao što je zdravstvo.

Gradska uprava u spektru svojih djelatnosti uključuje i zdravstvo, pa bilo kakva poboljšanja iz ove oblasti direktno utječu na daljnji rad uprave. Osim toga, Gradska uprava kao direktni

korisnik ovog sistema pored navedenih direktnih poboljšanja, indirektno vodi registar svih događaja koji će se održati unutar grografskog područja koje obuhvata.

Treća, a vjerovatno i najbitnija, zainteresirana strana jesu sami pacijenti koji treba da uživaju poboljšanu uslugu, kraće vrijeme čekanja i sveukupno bolji tretman i dijagnoze u Službi za hitnu medicinsku pomoć. Oni, iako ne direktni korisnici, su glavna varijabla koja u fokusu cijelog ovog sistema, sve počinje i završava sa pacijentima.

2. PREDSTAVLJANJE SOFTVERSKE ARHITEKTURE

„Softverska arhitektura sistema je skup struktura potrebnih kako bi se promišljalo o sistemu, a koje čine softverski elementi, veze između njih i njihova svojstva (osobine).“[1]

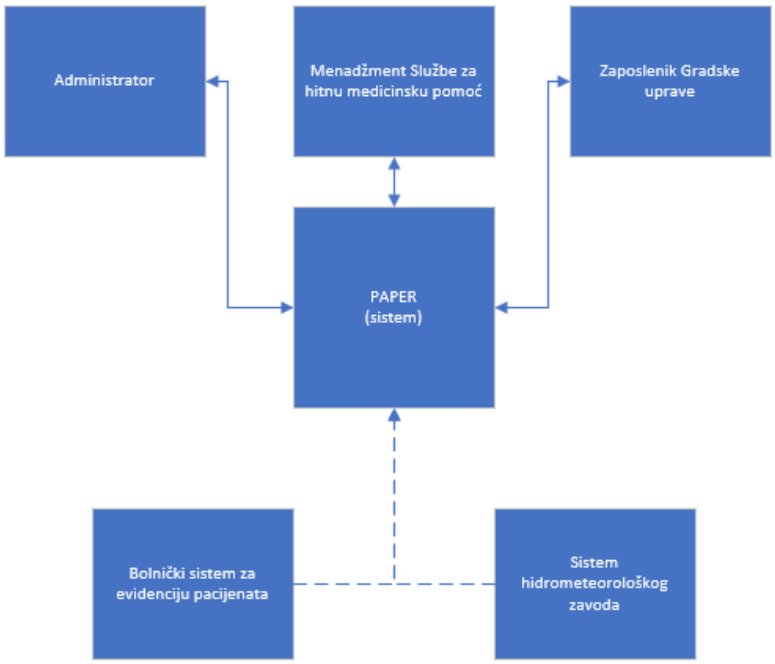
Arhitektura je most imeđu često apstraktnih poslovnih ciljeva organizacije i konačnog konkretnog softverskog sistema jer ona put od apstraktnih ciljeva do konkretnih sistema koji je najčešće veoma kompleksan može „ukrotiti“ i učiniti ga lakšim za kontrolu.

Ona nam je bitna iz razloga što predstavlja temelj softverskog sistema, a kako se na temelju gradi ostatak našeg softverskog sistema, on predstavlja i njegov najbitniji dio.

2.1. Kontekst sistema

Sam sistem kao jednu cjelinu, te eksterne aktere koji komuniciraju sa sistemom prikazujemo koristeći kontekstijalni dijagram. Sistem će da obuhvati sve korisničke module te će oni biti „interfejsi“ za komunikaciju između sistema i eksternih aktera. Sistem također enkapsulira cijeli proces obrade podataka, treniranja i testiranja modela, te kreiranja predikcija.

[1] Bass, L..; Clements, P.; Kazman, R. (2013). Software Architecture in Practice, 3rd Ed.

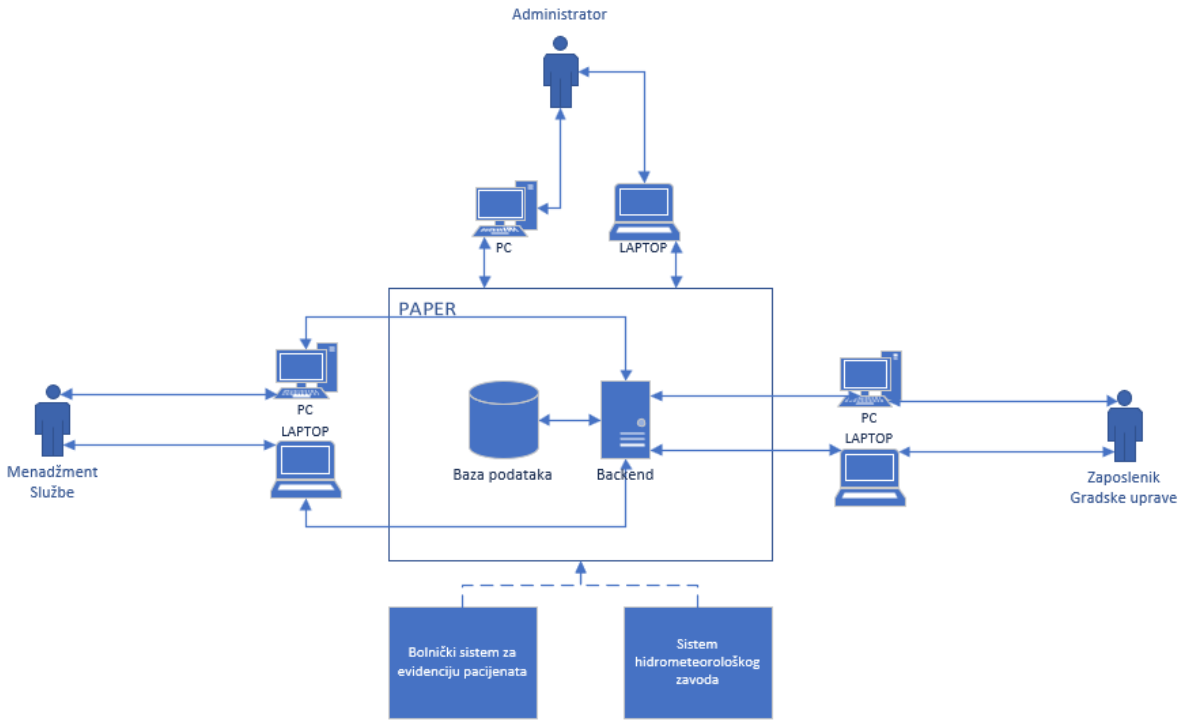


Slika 2.1. Konteksutalni dijagram sistema

2.2. Interakcija korisnika

Kako sistem radi se povjetljivim podacima vezanim za zdravstvene usluge i evidencije koje vodi Gradska uprava potrebno je korisničku interakciju modelirati na način da se maksimalno smanji mogućnost za neautorizovan pristup ili gubitak podataka. U skladu sa navedenim pristup sistemu omogućen je isključivo putem desktop aplikacije instalirane na računar ili laptop koji se nalazi u vlasništvu Gradske uprave ili Službe za hitnu medicinsku pomoć za korisnike sistema koji nisu administrator.

Administrator pristupa sistemu putem desktop aplikacije instalirane na računar ili laptop čija je jedina namjena da služi administraciji navedenog sistema. On kao takav ima pristup aplikaciji, ali može i direktno pristupiti bazi podataka kako bi vršio administraciju iste.



Slika 2.2. Dijagram interakcije korisnika sa sistemom

2.3. Ciljevi arhitekture

Najvažniji zadatak sistema je da kreira pretpostavku o broju pacijenata za neki naredni period, a to radi koristeći historijske podatke o tom broju u Službi hitne medicinske pomoći za koju želimo kreirati pretpostavku. Nadalje, jedan od ciljeva sistema je da se kontinuriano usavršava ponavljajući proces treniranja modela za predviđanje koristeći nove, stvarne podatke o broju pacijenata u Službi. Shodno tome za sistem je od izuzetne važnosti da podaci koje isti obrađuje budu trajno spašeni u njegovu bazu podataka kako bi sistem uvijek dostavljao što tačnije predikcije.

Pošto sistem podatke o broju pacijenata u Službi uzima sa već postojećeg sistema za evidenciju pacijenata, potrebno je osigurati da se takva komunikacija i razmjena informacija izvršava neometano uz minimalno ili bez pogrešaka u cjelini. Pored ovih podataka, podaci o

vremenskim prilikama trebaju da se preuzmu od nekog od postojećih sistema koji vode evidenciju o njima.

Sistem koristi podatke iz jedne od najosjetljivijih oblasti u kontekstu privatnosti podataka, zdravstvu. Potrebno je osigurati da komunikacija sa postojećim sistemom ne naruši sigurnost niti jednog sistema kako nebi došlo do neautorizovanog pristupa povjerljivim podacima kao što su kartoni pacijenata ili zadaci doktora. Nadalje kako sistem na osnovu podataka pohranjenih u bazi radi predviđanja i može imati direktan utjecaj na kapacitete Službe u realnom svijetu, zaključujemo da ukoliko se naruši integritet podataka na bilo koji način, rezultati mogu biti izrazito loši, a u najgorem slučaju za samog pacijenta.

3. KARAKTERISTIKE SOFTVERSKE ARHITEKTURE

Arhitekti razdvajaju karakteristike softverske arhitekture u širi spektar kategorija u zavisnosti od operacije, rijetko viđenih zahtjeva, strukture, itd. Karakteristike softverske arhitekture ispunjavaju tri kriterija, i to:

- Specificiraju ne-domenska razmatranja dizajna;
- Utječu na neke strukturalne aspekte dizajna;
- Krucijalne su i važne za uspjeh softvera;

Karakteristike softverske arhitekture mogu se klasificirati u tri kategorije, i to: operativne, strukturne i mješovite

3.1. Sposobnost arhiviranja

Praksa čuvanja podataka na duži vremenski period, često iz legalnih, historijskih ili analitičkih razloga naziva se arhiviranje. To je proces čuvanja podataka koji su izuzetno važni, ali se ne koriste u redovnoj upotrebi. U suštini to je dugotrajno skladište podataka za podatke kojima bismo trebali pristupiti u budućnosti, pa ih ne trebamo imati uvijek spremne za upotrebu. Pošto PAPER u nekim momentima obrađuje velike količine podataka, dok se u nekim ponaša kao jednostavan TMS, zaključujemo da je potrebno adekvatno čuvati ove podatke kako ne bismo narušili performanse sistema, a usput smanjili troškove i podigli nivo sigurnosti. Arhiviranje se primarno odnosi na podatke potrebne servisu za mašinsko učenje, preciznije podatke o vremenskoj prognozi i pacijentima, a može se odnositi i na arhiviranje iz drugih modula.

3.2. Kompatibilnost

Kompatibilnost predstavlja mogućnost različitih softvera (ili hardvera) koji potječu od različitih proizvođača ili nisu dio istog projekta da međusobno komuniciraju bez potrebe za

naknadnom nadogradnjom. Može se smatrati da je kompatibilnost stepen u kojem produkt, sistem ili dio može razmjenjivati informacije sa drugim produktima, sistemima ili dijelovima i/ili izvršavati svoje zadaće dok dijeli isti hardver ili softversko okruženje. Pošto PAPER komunicira sa postojećim sistemima za evidenciju pacijenata koji imaju vlastitu softversku arhitekturu, potrebno je prilagoditi zahtjeve koje PAPER šalje, te način na koji obrađuje odgovore od postojećeg sistema kako bi se osigurala nesmetana i (skoro pa) nepogrješiva transmisija podataka. Nadalje potrebno je prilagoditi sistem na način da uspješno šalje zahtjev i prima odgovor od sistema koji vode evidenciju o parametrima za vremensku prognozu što su u suštini jedni od ključnih parametara koji utječu na broj pacijenata u Službi za hitnu medicinsku pomoć.

3.3. Zaštita

Zaštita je stepen u kojem softver štiti informacije i podatke tako da ljudi ili drugi produkti ili sistemi imaju pristup podacima u skladu sa njihovom vrstom ili razinom autorizacije. Ovo podrazumjeva povjerljivost, integritet, neopovrgavanje, odgovornost i autentičnosti. Pošto sistem radi sa povjerljivim podacima vezanim za tok pacijenata kroz Službu hitne medicinske pomoći, a uz to komunicira sa sistemom koji sadrži detaljne podatke o svakom pacijentu pojedinačno, potrebno je zaštititi navedene sisteme. Time osiguravamo da ne bi došlo do bilo kakve zloupotrebe podataka ili sistema, što bi neupitno dovelo do nepovoljnih situacija kako za osoblje, tako i za pacijente.

4. ZNAČAJNI USE-CASE-OVI

PAPER sistem kao takav obuhvata funkcionalnosti za svaki od tri navedena tipa korisnika i funkcionalnosti koje sistem izvršava sa drugim sistemima, pa tako razlikujemo:

- Funkcionalnosti šefa Službe za hitnu medicinsku pomoć;
- Funkcionalnosti uposlenika Gradske uprave;
- Funkcionalnosti adminstratora;
- Interne funkcionalnosti sistema;

4.1. Funkcionalnosti šefa Službe za hitnu medicinsku pomoć

Ako u obzir uzmemo korisnike sistema koji nisu zaduženi za održavanje istog, onda možemo zaključiti da su članovi menadžmenta Službe, odnosno šef službe, oni koji će vršiti najveću interakciju sa sistemom.

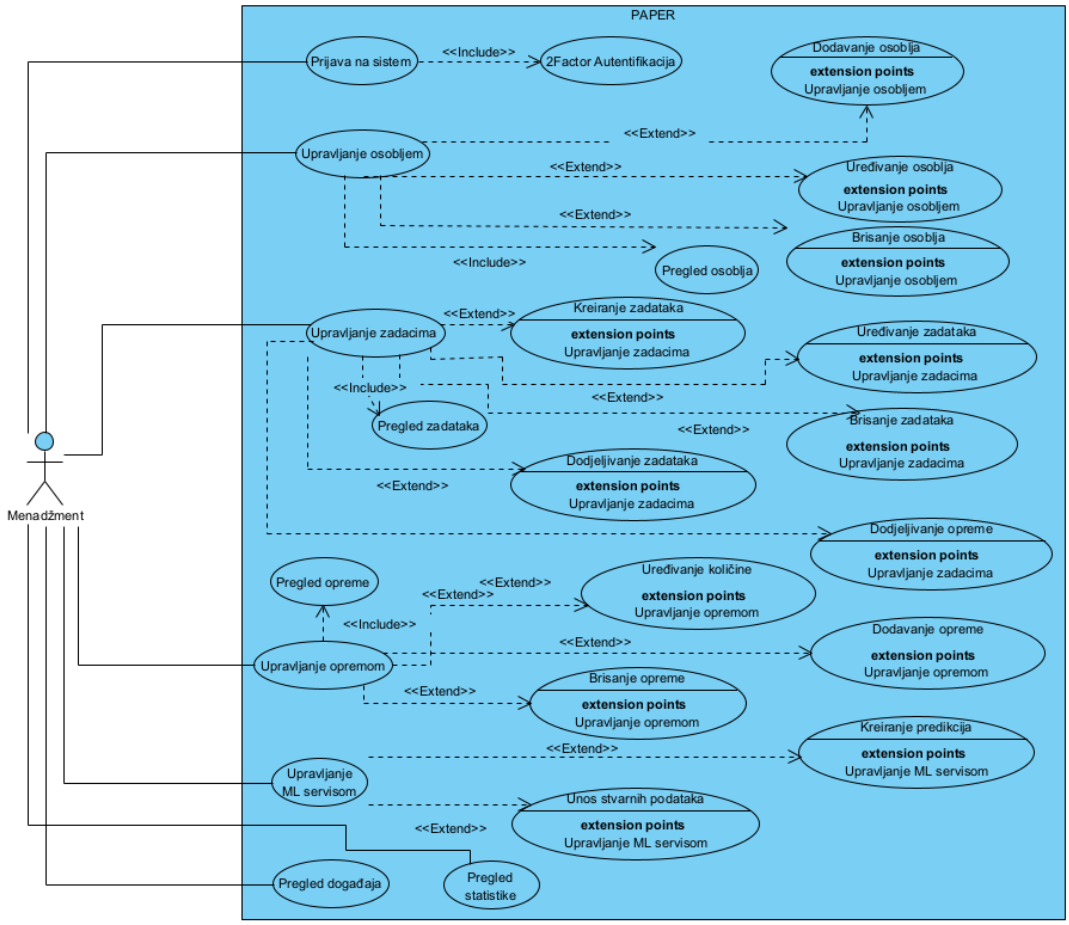
Od šefa službe se prilikom svakog pokretanja sistema traži da uz važeće kredencijale izvrši prijavu na sistem. Pošto šef Službe ima poseban modul za pristupanje sistemu, unosom bilo kojeg od postojećih kredencijala koji pripadaju nekom od drugih tipova korisnika neće se odobriti pristup sistemu kroz ovaj modul, a korisnik će biti adekvatno obaviješten. Unos takvih ili netačnih kredencijala unosi se u adekvatno skladište za logiranje kako bi se taj zapis mogao koristiti za kasnije forenziku u slučaju neautorizovanog pristupa sistemu.

Nakon uspješne prijave prikazuje se početni ekran sa osnovnim informacijama, kao što se predviđeni broj pacijenata za taj dan, broj do tada primljenih pacijenata, podaci o doktoru koji je trenutno na smjeni, itd. Šef Službe putem navigacije može da vrši upravljanje zadacima na jednom ekranu, na drugom analizu podataka i upravljanje predikcijama, na trećem da upravlja medicinskom opremom (lijekovi, rukavice, maske, igle, itd.), a na četvrtom ima uvid u događaje koje je unio uposlenik Gradske uprave kako bi dobio ideju o tome šta može očekivati u datom periodu;

Upravljanje zadacima omogućava šefu da ima mjesečni uvid u raspored smjena uposlenika Službe, da dodjeljuje smjene zaposlenicima, ima uvid u pretpostavljeni i stvarni broj pacijenata za određeni dan i smjenu, te mu na taj način omogućava adekvatno upravljanje kadrom.

Na ekranu za analizu podataka i upravljanje predikcijama šef može da ima uvid u duže historijske podatke o broju pacijenata u Službi, radi određena filtriranje podataka potrebna za analizu, vrši analizu unesenih parametara za predikciju, te na kraju pokreće proces kreiranja predikcije za određeni period koji odredi.

Kroz upravljanje medicinskom opremom šefu je omogućeno da za svaki dan rasporedi istu kako bi osigurao adekvatnu potrošnju opreme, a također imao uvid u trenutno kvantativno stanje što mu omogućava pravovremenu nabavku opreme koja nedostaje.



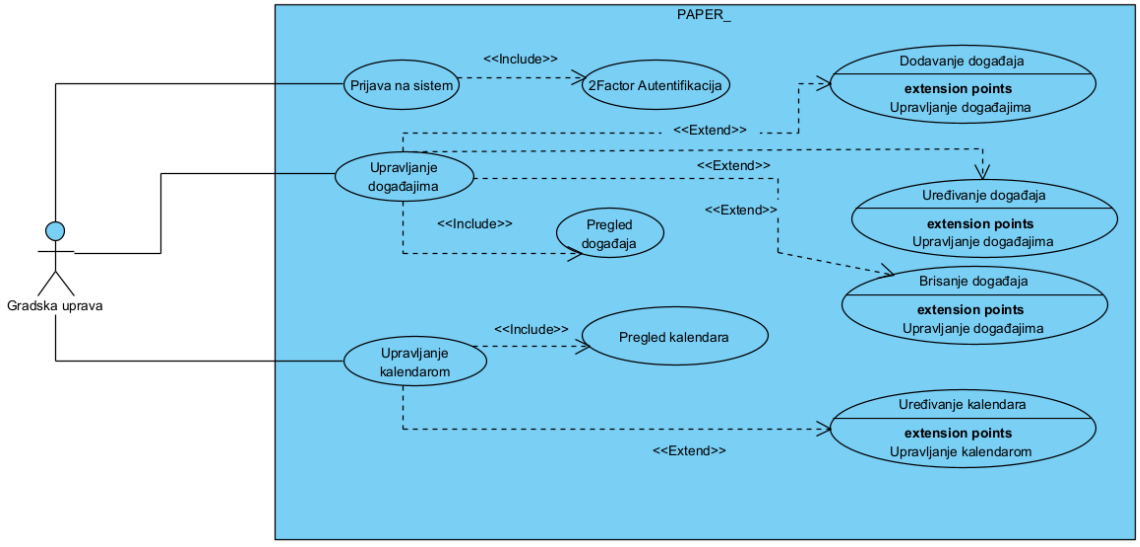
Slika 4.2.1. Use-case dijagram - Služba za hitnu medicinsku pomoć

4.2. Funkcionalnosti uposlenika Gradske uprave

Uposleniku se prilikom pokretanja sistema otvara ekran za prijavu na njegov i samo njegov modul. Ukolio uposlenik unese kredencijale nekog drugog tipa korisnika, ili unese netačne kredencijale, neće mu biti odobren pristup i ispisat će mu se adekvatna poruka. Ukoliko unese važeće kredencijale otvara mu se početni ekran za njegov tip korisinika.

Na početnom ekranu dat mu je prikaz događaja i njihovih detalja za kraći vremenski period, skupa sa informacijama o vremenskoj prognozi koje sistem povlači sa postojećeg sistema za tu namjenu. Korisnik ima mogućnost da prilagodi taj uži pregled događaja i da ima uvid u kratak opis svakog od njih.

Navigacija mu ostavlja mogućnost da pređe na drugi ekran koji mu omogućava da ima širi pregled na događaje, ima uvid u njihove detalje, pojedinačno ih uređuje i briše. Nadalje korisnik može dodati određeni događaj i spasiti ga u kalendar. Uposlenik je zadužen da vodi evidenciju o praznicima, vikendima i drugim neradnim danima kako bi sistem vodio računa o njima prilikom kreiranja predviđanja.

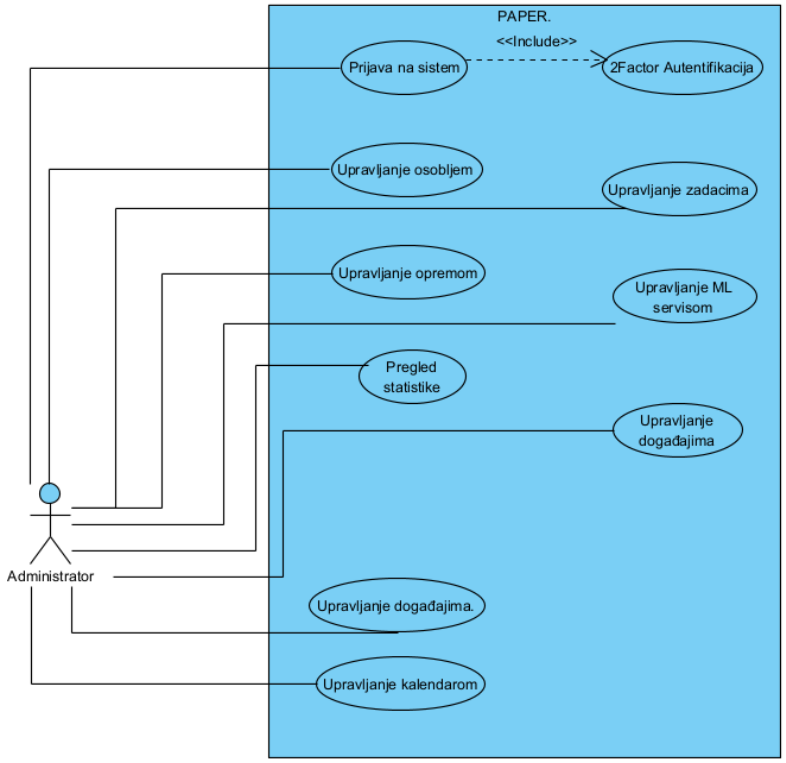


Slika 4.2.1. Use-case dijagram – Uposlenik gradske uprave

4.3. Funkcionalnosti administratora

Administrator, kao i u drugim sistemima, je osoba koja ima najveći nivo pristupa sistemu. Njegovi zadaci su akcentirani na obezbjeđivanje ispravnog rada sistema, te kao takav ne koristi klasične funkcionalnosti u redovnoj upotrebi.

U svakom slučaju, prilikom pokretanja sistema, prikazuje mu se ekran za prijavu na njegov modul gdje on, kao i kod drugih korisnika, mora unijeti kredencijale kako bi dobio pristup sistemu. Ukoliko kredencijali iz bilo kojeg razloga nisu validni, ne daje mu se pristup, pokušaj se bilježi u bazu podataka i ispisuje mu se adekvatna poruka.Ukoliko se prijavi na sistem, na početnom ekranu mu se ispisuje lista određenog broja posljednje dodanih događaja, zadataka, medicinske opreme i kreiranih predviđanja. Za svaku od ovih stavki ima poseban ekran gdje su mu dozvoljene funkcije kreiranja, brisanja, uređivanja i čitanja podataka.Administrator može dopustiti ili zabraniti ostalim tipovima korisnika da imaju uvid u određene dijelove sistema kako bi se zaštitio sistem u slučaju neželjenih ulazaka u isti.

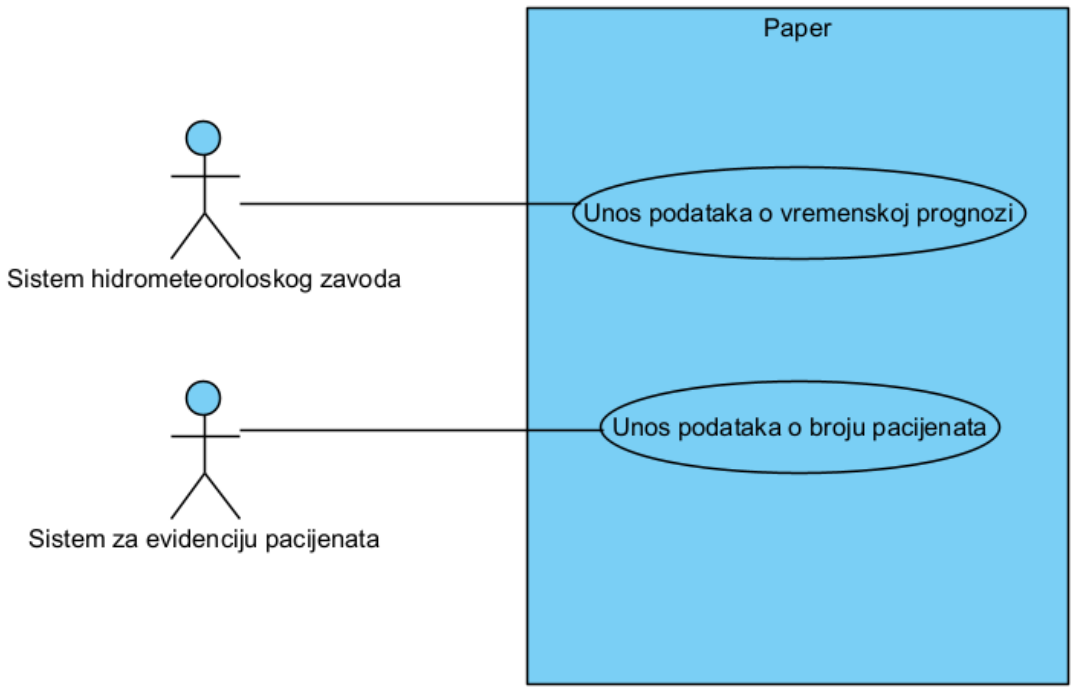


Slika 4.3.1. Use-case dijagram – administrator

4.4. Interne funkcionalnosti sistema

Iako se većina rada sistema zasniva na interakciji korisnika sa istim, već smo spomenuli da sistem ima zadatak da samostalno komunicira sa drugim sistemima i na taj način prikuplja za sistem važne podatke.

Prema tome, sistem ima mogućnost da šalje upite prema postojećem sistemu za evidenciju pacijenata kako bi dobio relevantne informacije, te da komunicira sa sistemom za evidenciju parametara vremenske prognoze.



Slika 4.4.1. Use-case dijagram – interakcija sa drugim sistemima

5. VRSTE (PATERNI) SOFTVERSKE ARHITEKTURE

U kontekstu ovog sistema analzirane su **mikroservisna**, **service-oriented arhitektura** i **mircokernel/plug-in arhitektura**. Svaku od ovih arhitektura karakterišu određene prednosti i nedostaci koji će biti detaljno analizirani u nastavku, nakon čega se vrši odabir najadekvatnije arhitekture u odnosu na uočene ključne karakteristike koje arhitektura ovog sistema treba da posjeduje.

5.1. Paterni i karakteristike SA

Mikroservisna arhitektura je jedna od distribuiranih arhitektura i zasniva se na podjeli sistema na manje, nezavisne servise koji međusobno, ili sa korisnikom, komuniciraju putem protokola (najčešće HTTP/REST).

Svaki od kreiranih mikroservisa može imati vlastiti mehanizam za **arhiviranje podataka**, što omogućava efikasnije i fleksibilnije upravljanje podacima i prilagođavanje različitim potrebama. Na primjer, arhiva podataka o broju pacijenata i parametrima koji utječu na predikciju može biti odvojena od arhive podataka koja sadrži informacije o resursima.

Što se tiče **kompatibilnosti** i integracije sa drugim sistemima, ona je u ovoj arhitekturi poprilično trivijalna ukoliko imamo ili kreiramo dobro definisane API-je za međusobnu komunikaciju između sistema, a možemo istu vršiti i direktno putem baze podataka.

Distribuirani mikroservisi uveliko ograničavaju kretnju potencijalnog napadača na razinu jednog ugroženog mikroservisa i ne dopušta njegovo širenje. Ovakav način organizacije doprinosi **zaštiti** sistema što neautorizovan pristup osjetljivim podacima čini izazovnijim.

Service-oriented arhitektura je arhitektura koja omogućava kreiranje i korištenje međusobno povezanih usluga koje vrše međusobnu interakciju koristeći neki od standardizovanih protokola (npr. SOAP, ponekad i REST).

Ova arhitektura može vršiti nešto „centralizovaniju“ evidenciju i **arhiviranje podataka** sa različitih servisa. Ovaj pristup može pojednostaviti upravljanje podacima, ali u isto vrijeme stvara jednu tačku neuspjeha i smanjuje fleksibilnost arhiviranja jer svi sistemi zavise od centralizovanog mehanizma arhiviranja.

Pošto je standardni protokol za komunikaciju SOAP, **kompatibilnost** sa sistemima koji također koriste ovaj protokol se podrazumjeva, ali može doći do problema u komunikaciji sa drugim sistemima, što se može prevazići koristeći određene transformacije podataka.

Zaštita se često oslanja na centralizovanu kontrolu pristupa, što pojednostavljuje upravljanje sigurnosnim politikama. Standardne prakse za autentifikaciju, autorizaciju i siguran rad sa podacima glavne su komponente zaštite ovakvih sistema.

Mikrokernel arhitektura zasniva se na centralnom jezgru koje je zaduženo za sve bazne operacije, te ekstenzijama koje služe za proširenje sistema.

Slično kao i kod service-oriented arhitekture upravljanje podacima je dosta centralizovanije i tu ulogu ima upravo centralno jezgro, te su i **arhiviranje** i pristup podacima uveliko centralizovaniji u odnosu čak i na service-oriented arhitekturu.

Proširenje, prilagođavanje novih tehnologija i povezivanje sa drugim sistemima ograničeno je centralnim jezgrom i svaka adaptacija sistema koja bi omogućila **kompatibilnost** sa nekim drugim sistemom zahtjeva kreiranje zasebnih modula.

Mikrokernel arhitektura je veoma dobro **zaštićena**, pošto se ključne funkcionalnosti ili servisi izvršavaju u kernelu. Pošto je ovo arhitektura koja se bazira na jednom jezgru, teško je napadačima iskoristiti nedostatke da dobiju neautorizovan pristup podacima.

5.2. Ograničenja paterna

Kod **mikroservisne arhitekture** kontekst **arhiviranja podataka** poprilično je jasan, svaki od mikroservisa ima mogućnost arhiviranja podataka koji pripadaju tom mikroservisu. Kod određenih sistema ova osobina bi bila ograničenje, ali u PAPER sistemu ona predstavlja

ključnu vrlinu koja omogućava adekvatno arhiviranje i jednostavan pristup arhiviranim podacima.

Ako posmatramo **kompatibilnost** u kontekstu kompatibilnosti sa drugim sistemima, tu ova arhitektura ne prepoznaje značajne probleme. Problemi mogu nastati u kompatibilnosti između samih servisa unutar istog sistema. Naime, različiti timovi programera rade na različitim servisima što može dovesti do određenih problema u njihovoj međusobnoj komunikaciji.

U distribuiranim arhitekturama uvijek postoji određena stopa rizika po sistem u kontekstu njegove **zaštite** jer podaci dosta vremena provode u mreži. Uz to, koliko je dobra činjenica da su servisi odvojeni i djelimično nezavisni, što može izolirati napadača na samo taj jedan servis, toliko je ona loša iz razloga što veliki broj servisa ostavlja više mogućnosti za grešku koju napadač može iskoristiti.

Service-oriented arhitektura ima sličan pristup **arhiviranju** kao i mikroservisna arhitektura, ali pošto zajednička sabirnica međusobno povezuje servise sam proces arhiviranja i dohvatanja podataka je nešto komplikovaniji.

Standardno koristi SOAP protokol za komunikaciju koji je idealan za prenos podataka između različitih servisa unutar samog sistema. Problem nastaje prilikom pokušaja povezivanja sa drugim sistemima koji dosta rijeđe koriste ovaj protokol što rezultuje slabom **kompatibilnosti** sa drugim sistemima.

Za kontekst **zaštite** važno je spomenuti da su svi servisi povezani putem zajedničke sabirnice što podatke može ostaviti ranjivim ukoliko dođe do neovlaštenog pristupa na glavnu sabirnicu direktno ili preko nekog od servisa. Prema tome ova arhitektura može smanjiti broj ranjivih tačaka, ali ukoliko se i jedna od njih probije skoro svi podaci bivaju dostupni napadaču.

Arhiviranje se kod **mikrokernel arhitekture** vrši preko centralnog jezgra, pa iako postoje određene sličnosti sa service-oriented arhitekturom, kod mikrokernel arhitekture je arhiviranje višestruko centralizovanije. Samim bilo kakav pristup samo određenom dijelu arhiviranih podataka zahtjeva učitavanje cijelog skladišta, a isto važi i za upisivanje u arhivu.

Bilo kakva komunikacija sa drugim sistemima i sa centralnim jezgrom zahtjeva kreiranje novih specijalnih modula na koje se drugi sistemi „spajaju“. Mikrokernel operativni sistem najčešće nije **kompatibilan** i ne podržava standarnde API-je i sistemske pozive, te ne podržava komunikaciju sa nekim hardverskim uređajima. To zahtjeva različite adaptere i prepravke na drugim sistemima.

Slično kao i kod service-oriented arhitekture, mnogo je lakše **zaštititi** manji broj servisa, ali bilo kakav propust u programskom kodu narušava ne samo ovaj sistem, nego i sisteme koji su povezani sa njim.

5.3. Odluka i obrazloženja

Uzimajući u obzir sve prednosti i nedostatke svake od posmatranih softverskih arhitektura možemo zaključiti da se **mikroservisna arhitektura** najbolje uklapa u tražene karakteristike i kao takva daje temelj za daljnji razvoj sistema.

Razdvajanje različitih potreba u zasebne servise koji međusobno komuniciraju i nemaju zajedničku sabirnicu ili centralno jezgro osigurava da se potencijalni napadi mnogo brže, jednostavnije i efektivnije stave pod kontrolu u okviru ugroženog servisa. Na taj način se osigurava visok nivo **zaštite**, koja je i bila jedna od karakteristika sistema.

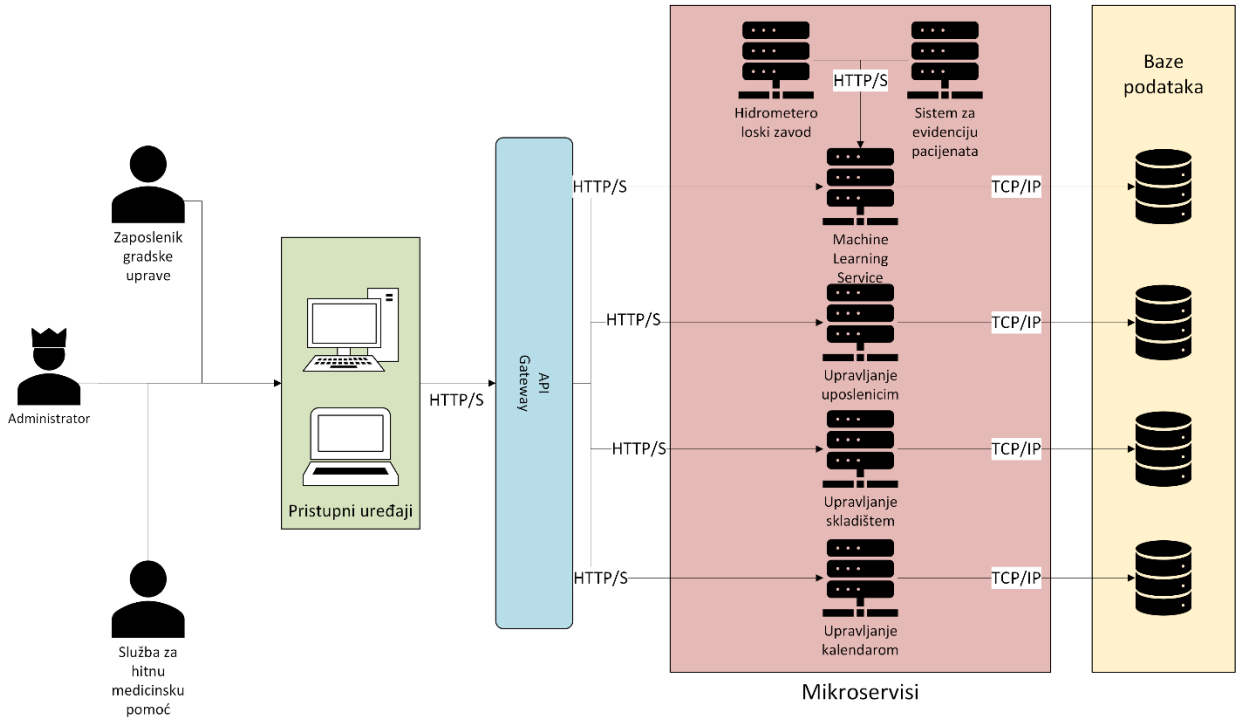
Spomenuta osobina omogućava i **arhiviranje podataka** za svaki servis pojedinačno što mnogo utječe na rad sistema prilikom analize i korištenja velike količine podataka, dok bi se kod ostalih arhitektura arhiviranje vršilo u centralnom jezgru ili putem centralne sabirnice.

Mikroservisna arhitektura je jedna od najpopularnijih arhitektura upravo iz razloga što podrazumjeva kreiranje API-ja kojima je moguće pristupiti sa više aplikacija i uređaja. Samim tim **kompatibilnost** neće predstavljati problem kod ove softverske arhitekture.

6. LOGIČKI POGLED

Funkcionalnosti koje sistem treba pružiti krajnjim korisnicima predstavljene su kroz logički pogled na sistem.

Korisnici sistema istom mogu pristupiti isključivo putem računara ili laptopa smještenih na radnom mjestu. Na ovim uređajima instalirana je odgovarajuća aplikacija koja koristi API Gateway za komunikaciju sa servisima. Svaki od servisa pristupa specijalno određenom dijelu baze podataka koji pripada tom servisu.



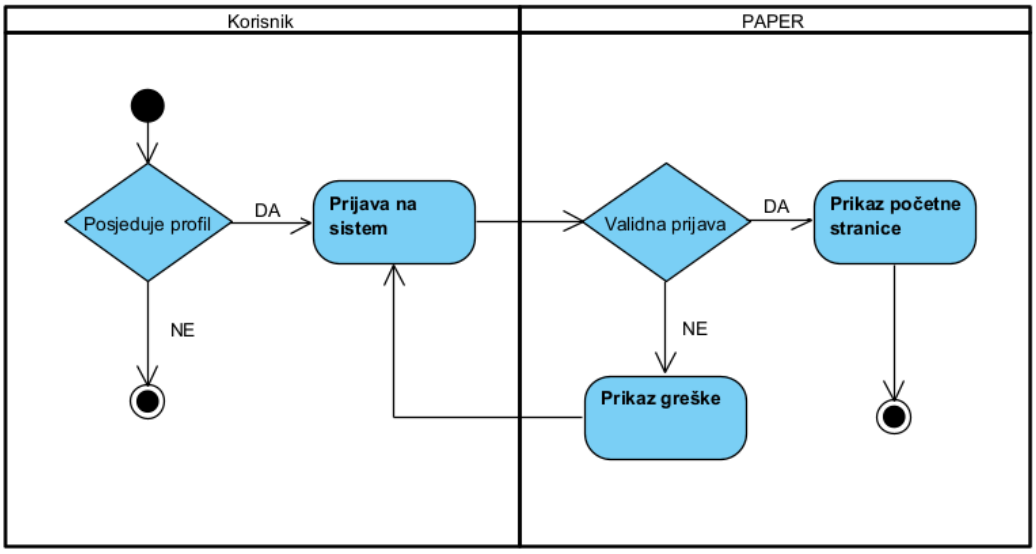
Slika 6.1. Logički pogled

7. PROCESNI POGLED

Dijagram aktivnosti se koristi za prikaz dinamičkih aspekata sistema i procesa koji su od važnosti za određeni sistem. Aktivnosti koje će biti prikazane su: prijava na sistem, dodavanje novog zaposlenika, dodavanje novog zadatka, dodjeljivanje zadataka zaposleniku, kreiranje predikcija. Za svaku od navedenih prikazan je dijagram aktivnosti, te kratko obrazloženje o aktivnostima koje su prikazane na navedenom grafu.

7.1. Prijava

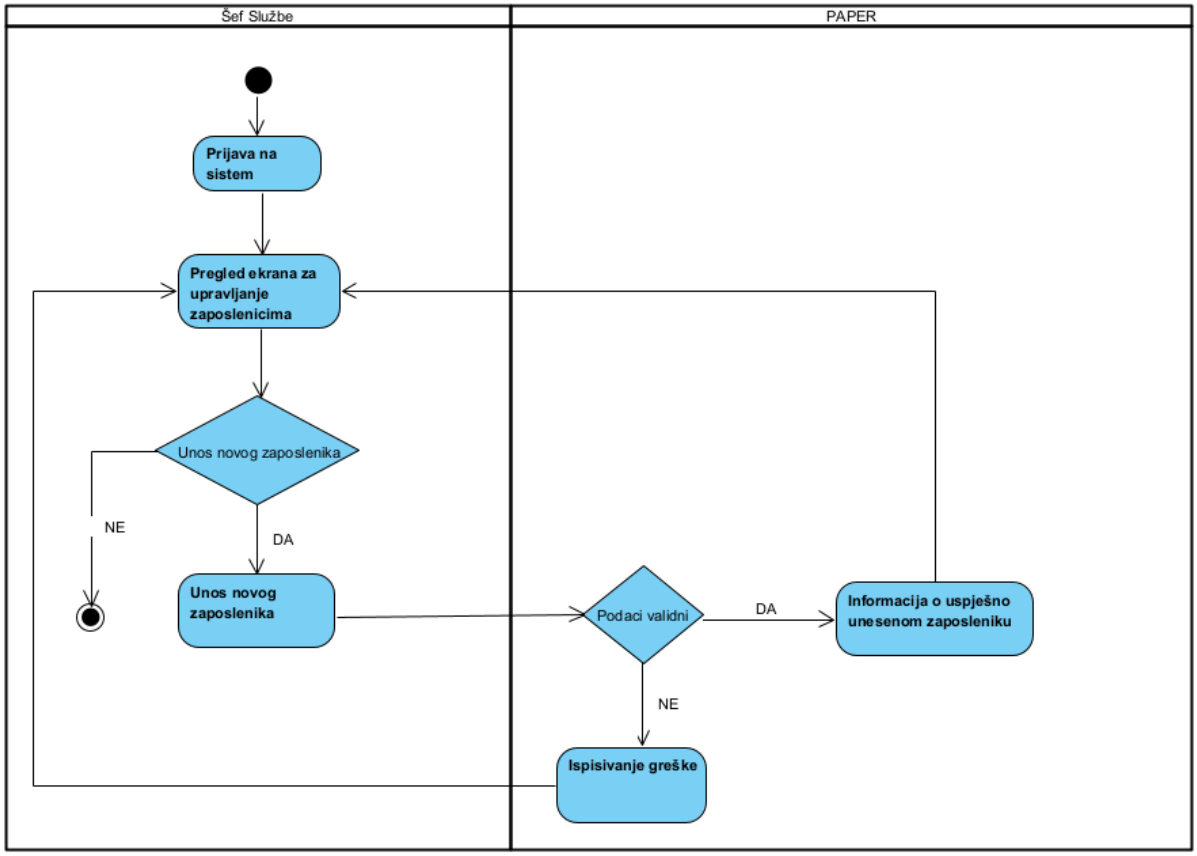
Pošto smo rekli da je sigurnost jedna od osnovnih karakteristika ovog sistema, važno je kreirati adekvatan program provjere autentičnosti i autorizacije. Cjelokupan sadržaj aplikacije sakriven je od neprijavljenog korisnika i ne postoji API, osim API-ja za prijavu na sistem kojem neautentifikovana i neautorizovana osoba može pristupiti. Korisničko ime i lozinka moraju biti uneseni tačno onako kako su i spašeni prilikom kreiranja korisničkih računa, a korisnička imena se ne smiju ponavljati za korisnike. Ukoliko kredencijali ne budu ispravni korisnik dobije poruku o grešci i vraća se ponovo na proces prijave.



Slika 7.1.1. Dijagram aktivnosti za prijavu

7.2. Dodavanje novog zaposlenika

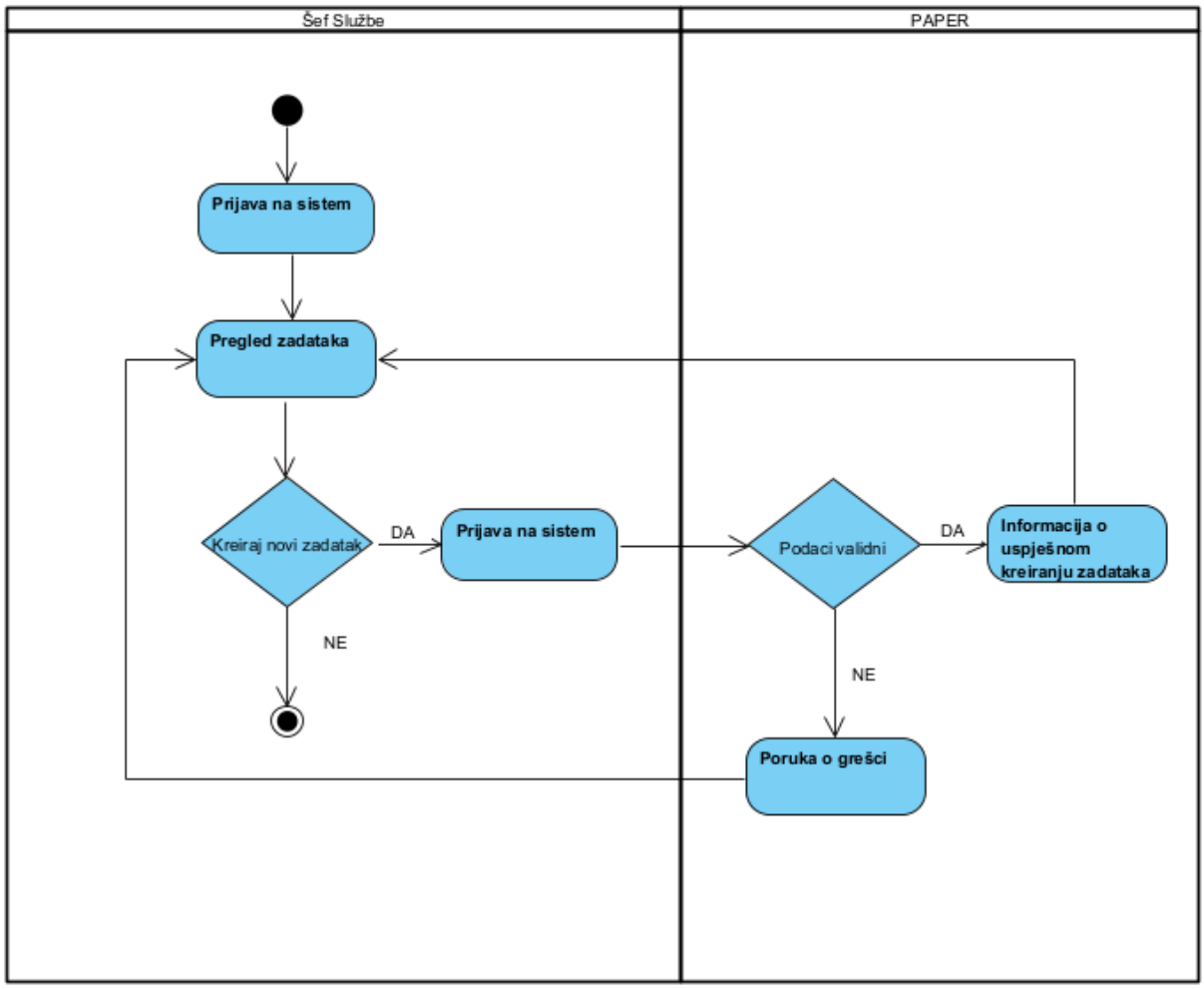
Da bi se omogućilo dodavanje novog zaposlenika, korisnik treba da se prijavi na modul Šefa Službe, te navigira na ekran za pregled svih zaposlenika. Nakon toga može otvoriti formu za dodavanje novog zaposlenika. Od korisnika se traži da unese podatke kao što su: ime, prezime, radno mjesto, obrazovanje, broj licence i specijalnost (ukoliko se radi o doktoru). Nakon što unese sve navedeno vrši se validacija unesenih podatak nakon koje se novi zaposlenik evidentira u bazu podataka, a korisnik biva obaviješten o uspješnosti ove radnje.



Slika 7.2.1. Dijagram aktivnosti za dodavanje novog zaposlenika

7.3. Dodavanje novog zadatka

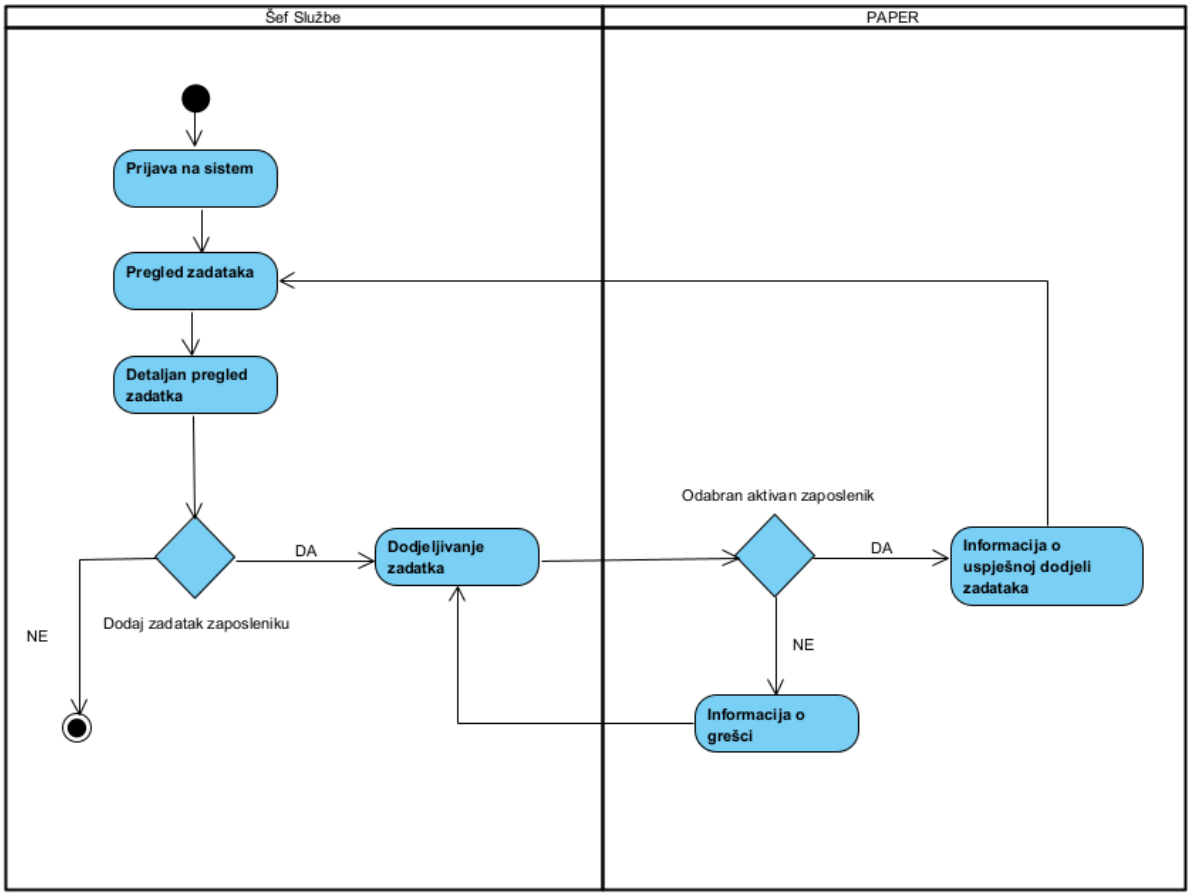
Da bi se omogućilo dodavanje novog zadatka, kao i u prethodnom primjeru, korisnik treba da se prijavi na mogul Šefa Službe, te navigira na ekran za pregled svih zadataka. Nakon toga može otvoriti formu za dodavanje novog zadatka. Od korisnika se traži da unese podatke kao što su: naziv zadatka, vrsta zadatka, datum početka i datum završetka zadatka i napomena. Nakon što unese sve navedeno, vrši se validacija nakon koje se zadatak evidentira u bazu podataka. Korisnik biva obaviješten o uspješnosti ili neuspješnosti ove radnje.



Slika 7.3.1. Dijagram aktivnosti za dodavanje novog zadatka

7.4. Dodjeljivanje zadatka zaposleniku

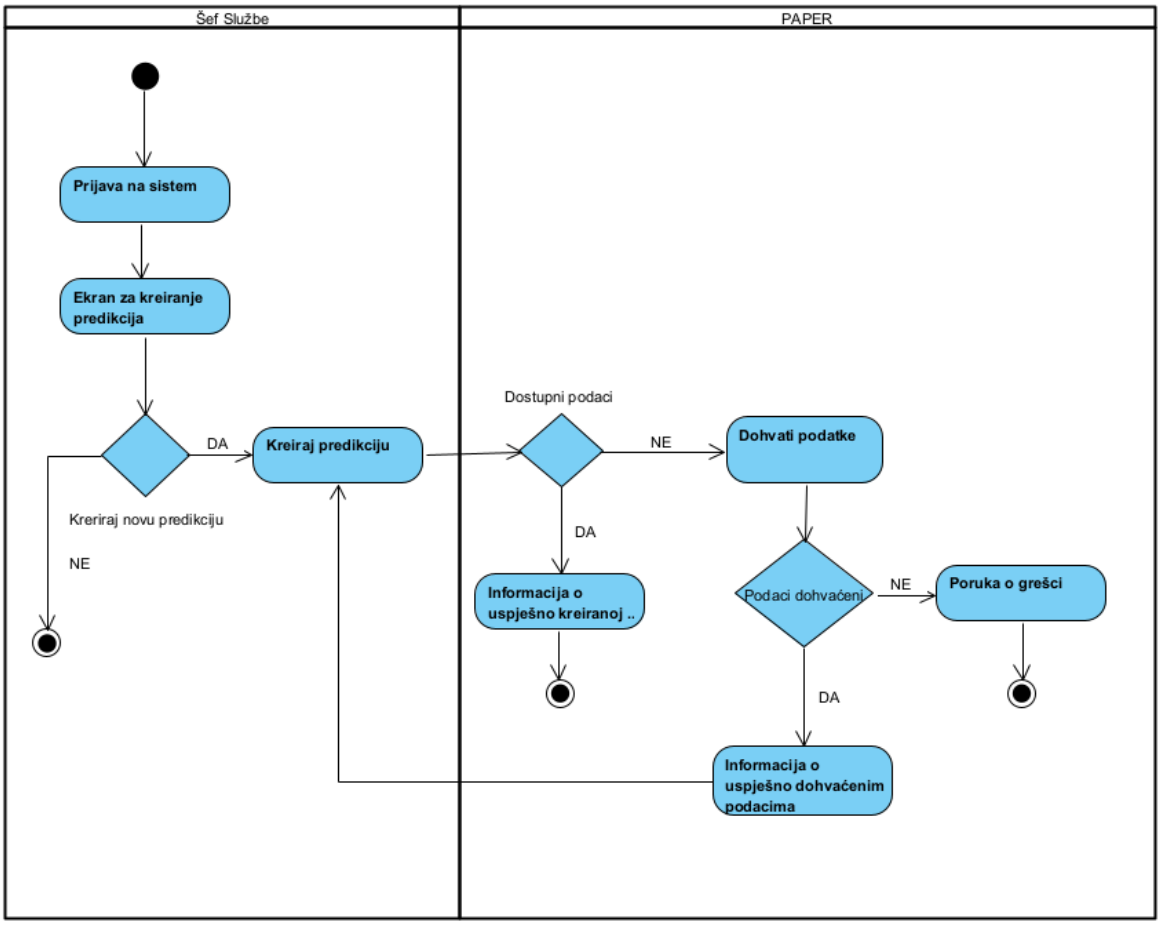
Da bi korisnik, odnosno šef Službe dodijelio zadatak nekom od zaposlenika najprije mora da se prijavi na svoj modul, te navigira na ekran za prikaz svih zadataka. Nakon toga ulazi u detaljan pregled nekog od zadataka, gdje ima mogućnost da iz padajućeg menija odabere zaposlenika kojem će dodijeliti navedeni zadatak. Nakon što završi sa dodjelom, prima informaciju o uspješnosti ili o grešci i biva navigiran na odgovarajući ekran. U slučaju uspješne dodjele vraća se na pregeld zadataka, a u slučaju greške vraća se na ekran za dodjelu.



Slika 7.4.1. Dijagram aktivnosti za dodjeljivanje zadataka zaposleniku

7.5. Kreiranje predikcija

Za kreiranje predikcija korisnik najprije treba da se prijavi na modul Šefa Službe i da se navigira na ekran za kreiranje predikcija. U slučaju da želi kreirati novu predikciju za određeni period u budućnosti, vrši se provjera dostupnosti varijabli od kojih zavisi predikcija, te ako su one dostupne predikcija se uspješno kreira. Ukoliko te informacije nisu dostupne, šalje se zahtjev eksternom sistemu za evidenciju pacijenata i eksternom sistemu za vremenske parametre da popune nedostajuće podatke. Ukoliko se podaci uspješno popune, korisnik treba ponovo da pokuša kreirati predikciju, a ukoliko PAPER ne može dohvatiti navedene podatke korisniku se ispisuje poruka o grešci.



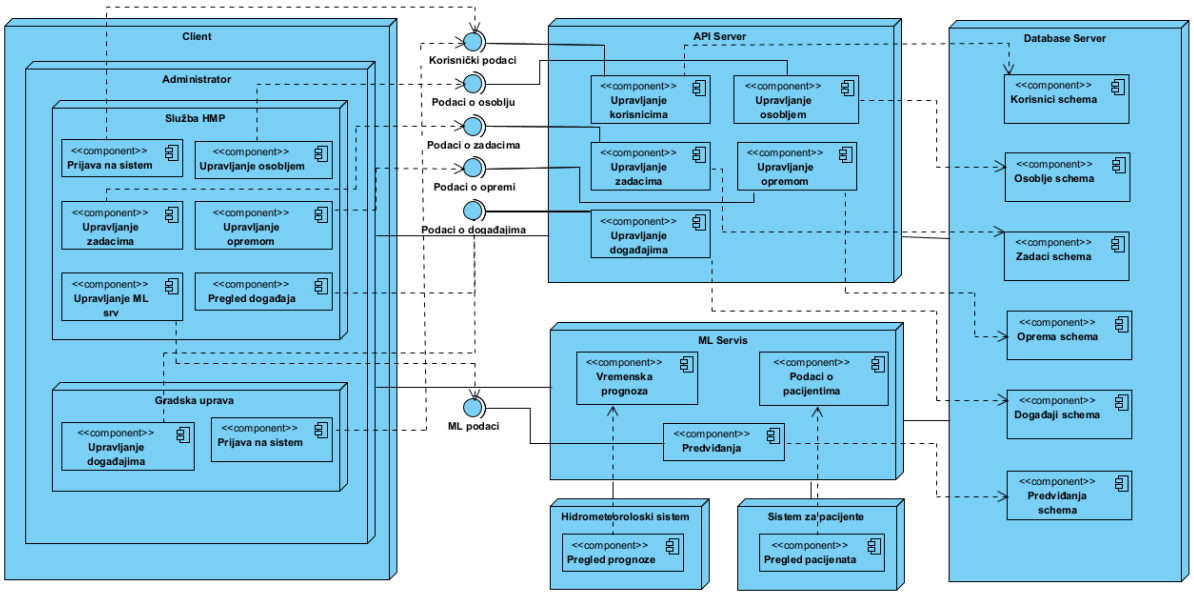
Slika 7.5.1. Dijagram aktivnosti za kreiranje predikcija

8. IMPLEMENTACIJSKI POGLED

Arhitekturalne osobine vezane za implementaciju se prikazuju putem implementacijskog pogleda.

U dijagramu su prikazani artefakti koji sudjeluju u gradnji kompletnog sistema, a to su:

1. Baza podataka;
2. Infrastruktura;
3. Servisi;
4. Korisničko sučelje;
5. API-evi eksternih sistema;



Slika 8.1. Implementacijski pogled

9. VELIČINA I PERFORMANSE

Sistem u do sada razrađenom obliku, sa primarno 3 tipa korisnika koja su već spomenuta, nema elemente značajnije kompleksnosti. U skladu sa navedenim zaključujemo da sistem nema potrebu za obradom veće količine transakcija što znači da dovoljna brzina i pristojne performanse sistema neće biti teško ostvariti bez značajnijeg napora. Ostavljena je mogućnost proširenja sistema sa još jednim tipom korisnika, a to je zaposlenik. Uvođenjem zaposlenika bi se kompleksnost povećala za određeni stepen, ali to opet ne bi zahtjevalo značajan broj transakcija kao što je slučaj u sistemima za online kupovinu, ili real-time sistemima. Pošto bi uvođenjem zaposlenika u sistem u manjoj mjeri porasla kompleksnost, samim tim bi se u manjoj mjeri smanjila sigurnost. Ova posljedica može se ublažiti adekvatnim odabirom funkcionalnosti zaposlenika kako bi se smanjio dodir za povjerljivim podacima.

10. KVALITETA

Razdvojenost servisa u mikroservisnoj arhitekturi omogućava svakom od njih da vrši upravljanje smo onim podacima kojima ima pristup, te nadalje njihovo arhiviranje. Taj način rada sa podacima omogućava dohvatanje samo onih podataka koji su nam potrebni bez potrebe za pristupanjem nepotrebnim podacima, što će u suštini poboljšati performanse i sigurnost podataka.

Osim što izolacija samih podataka poboljšava sigurnost, ona je povećana i izolacijom rada sa podacima pošto određeni mikroservis obrađuje samo podatke koji su relevantni za njegovu funkciju. Nadalje specificiranje uloga, odnosno tipova korisnika koji mogu pristupiti sistemu, stvara dodatni sloj zaštite. Razdvajanje servisa omogućava svakom servisu da nametne vlastita pravila i procedure, kao što su enkripcija, autentifikacija i autorizacija i slično.

Mikroservisi komuniciraju koristeći standardne API-je (npr. REST, GraphQL, gRPC) što olakšava integraciju sa drugim sistemima, a mogućnost da se svaki od mikroservisa razvija u različitim programskim jezicima omogućava da se za svaki servis koriste najbolji i najkompatibilniji alati. Na samom kraju razvoj jednog mikroservisa ne utječe na rad drugog, što neće dovesti do prestanka rada sistema.

11. REFERENCE

1. Predavanja, materijali i prezentacije, prof.dr. Dražena Gašpar, 2024;
2. Software Architecture in Practice (3rd Edition), Len Bass, Paul Clements, Rick Kazman, Parson, 2015;
3. Software Architecture Patterns, Mark Richards, O'REILLY, 2015;
4. Materijali sa predavanja, Modeliranje poslovnih procesa, prof.dr. Emina Junuz, 2023;
5. Materijali sa predavanja, Analiza i dizajn softvera, prof. dr. Emina Junuz, 2023;
6. Unified Modeling Language User Guide, Grady Booch, James Rumbaugh, and Ivar Jacobson;
7. Modern Systems Analysis and Design; Hoffer J.A., George J.F., Valacich J.S.; 2003;