SISTEMSKI DIZAJN

SMART DENTAL SYSTEM

Članovi projekta:

- Mladen Grbić
- Bojan Jazić
- Aleksandar Tulić

SADRŽAJ

1. UVOD

1.1. Namjena sistema

Sistem je namijenjen da bude korišten od strane stomatoloških ordinacija. Sistem će zaposlenima u stomatološkoj ordinaciji olakšati organizaciju rada, lakše rukovođenje podacima I nalazima, a samim time i poboljšati kvalitet rada.

1.2. Projektni ciljevi

Glavni ciljevi ovog projekta su lakša, brza I pouzdana komunikacija unutar stomatološke ambulante. Projekat će omogućiti lakši I jednostavniji rad svih zaposlenih u okviru jedne stomatološke ordinacije. Zubar/stomatolog će imati direktan uvid u prethodne nalaze svakog pacijenta na jedan klik, dok će recepcionar moći odmah na osnovu podataka koje im proslijedi stomatolog da naplati dati pregled, pregleda termine itd.

1.3. Definicije i skraćenice

Definicija	Značenje
Uređaj	Personalni računar
Hardver	Fizička reprezentacija uređaja, “opipljivi” dio računara
Softver	Dio uređaja kojem ne možemo pristupiti osim uz korištenje odgovarajućeg hardvera, neopipljivi dio računara
Other korisnik(blagajnik, counter, recepcioner, …)	Osoba koja koristi aplikaciju namijenjenu za blagajnika
Sistem	Sveobuhvatna struktura, koja se sastoji od servera, baze podataka, aplikacije za administratora, aplikacije za zubare i aplikacije za blagajnike(recepcioner)

Tabela definicija

Skraćenica	Značenje
HW	Hardware
SW	Software

Tabela skraćenica

1.4. Referentni dokumenti

Specifikacija softverskih zahtjeva – Smart Dental System.

1.5. Kratak pregled dokumenta i predložena arhitektura

Ovaj dokument se sastoji iz tri cjeline: Uvoda, Arhitekture postojećeg sistema i predložene arhitekture. U ovom dokumentu je ukratko opisan pregled arhitekture predloženog sistema u projektu Smart Dental System.

Sistem se sastoji iz 3 cjeline tj. podsistema pri čemu svaki od njih predstavlja jednu posebnu aplikaciju. Razlikujemo dvije web aplikacije i jednu desktop aplikaciju. Web aplikacije se nalaze na serveru gdje vi onda njima pristupate koristeći vaše web browser-e. Dok desktop aplikacija se instalira na personalnom računaru svakog zubara(*desktop aplikacija je namijenjena samo za njih*) odgovarajuće ordinacije. Trenutna aplikacija se izvršava samo na Windows platformi. Dok web aplikacijama možete pristupati preko različiti platformi samo morate imati web browser.

2. ARHITEKTURA POSTOJEĆEG SISTEMA

Trenutno ne postoji sistem sa ovakvim dizajnom i strukturom, kao trenutna zamjena za funkcionalnost predloženog sistema koristi se više različitih platformi koji implementiraju određene segmente sistema. Upotreba više različitih platformi utiče na sam kvalitet rada, a pri padu kvaliteta dolazi do konfuzije u različitim segmentima sistema. Takođe problem trenutnih sistema je nepouzdanost i manjak kontrole pristupa, te I sama bezbjednost sistema.
Sve ove nedostatke kao I mnoge druge otkloniće predloženi sistem I dati nove mogućnosti za što kvalitetniji rad stomatološke ambulante.

3. PREDLOŽENA ARHITEKTURA

3.1. Kratak pregled arhitekture i funkcionalnosti podsistema

Razlikujemo 3 aplikacije:

- desktop aplikacija za tipa korisnika zubar
- admin web aplikacija
- other/counter web aplikacija

Za svaku od prethodnih aplikacija koristit ćemo varijaciju arhitekturnog stila **MVC**(*skr. Model - View - Controller*), u kojem možemo primjetiti da imamo 3 različita sloja i to:

- **Pogled** (eng. View)
- **Kontroler** (eng. Controller)
- **Model** (eng. Model)

U **podsistemu View**, odnosno sloju, sadržimo podsisteme koji se koriste za predstavljanje UI-a te odgovarajuce aplikacije. Pri čemu **Authentication** je jedini podsistem koji će svi korisnici morati da prođu jer je to prvi prozor koji će da vide. U ovom podsistemu kao što naziv ukazuje vrši se autentifikacija korisnika sistema (korištenjem korisničkog imena i lozinke). Postoji odgovarajuće variranje implementacije authentication-a kod admin web aplikacije u odnosu na druge dvije aplikacije u smislu da ovde radimo two-factor authentication tj. pored korisničkog imena i lozinke sada zahtijevamo od korisnika da unese i odgovarajuću vrijednost tokena. Što se tiče konkretne implementacije two-factor authentication-a možete više pročitati ovde:https://github.com/AleksandarTulic/Projektovanje-Softvera/blob/main/DentilAdmin/readme.md

Pri čemu o ostalim podsistemima možete više istražiti kod dijagrama komponenata. Što se tiče sloja Controller(kao i njegovog podsloja service) u njemu se implementira cijela aplikativna logika, tj. sva ona obrada podataka koja je specifična za konkretnu aplikaciju a ne za sami domen.

Pošto je ovo revidirana dokumentacija, navešćemo posebno da podsistem Language je odstranjen iz aplikacij tj. bolje rečeno odstranjen iz desktop aplikacije jer je bilo namijenjeno samo za to.

Theme podsistem zadužen je za dekoraciju gui-a pri čemu elementi koji mogu da promijene svoj izgled su:

- tema prozora(eng. theme window) - odnosi se na boje koje se koriste za reprezentaciju gui elemenata
- Theme podsistem, kao sto se moze vidjeti sa sljedećeg dijagrama dekompozicije, je namijenjen samo desktop aplikaciji.

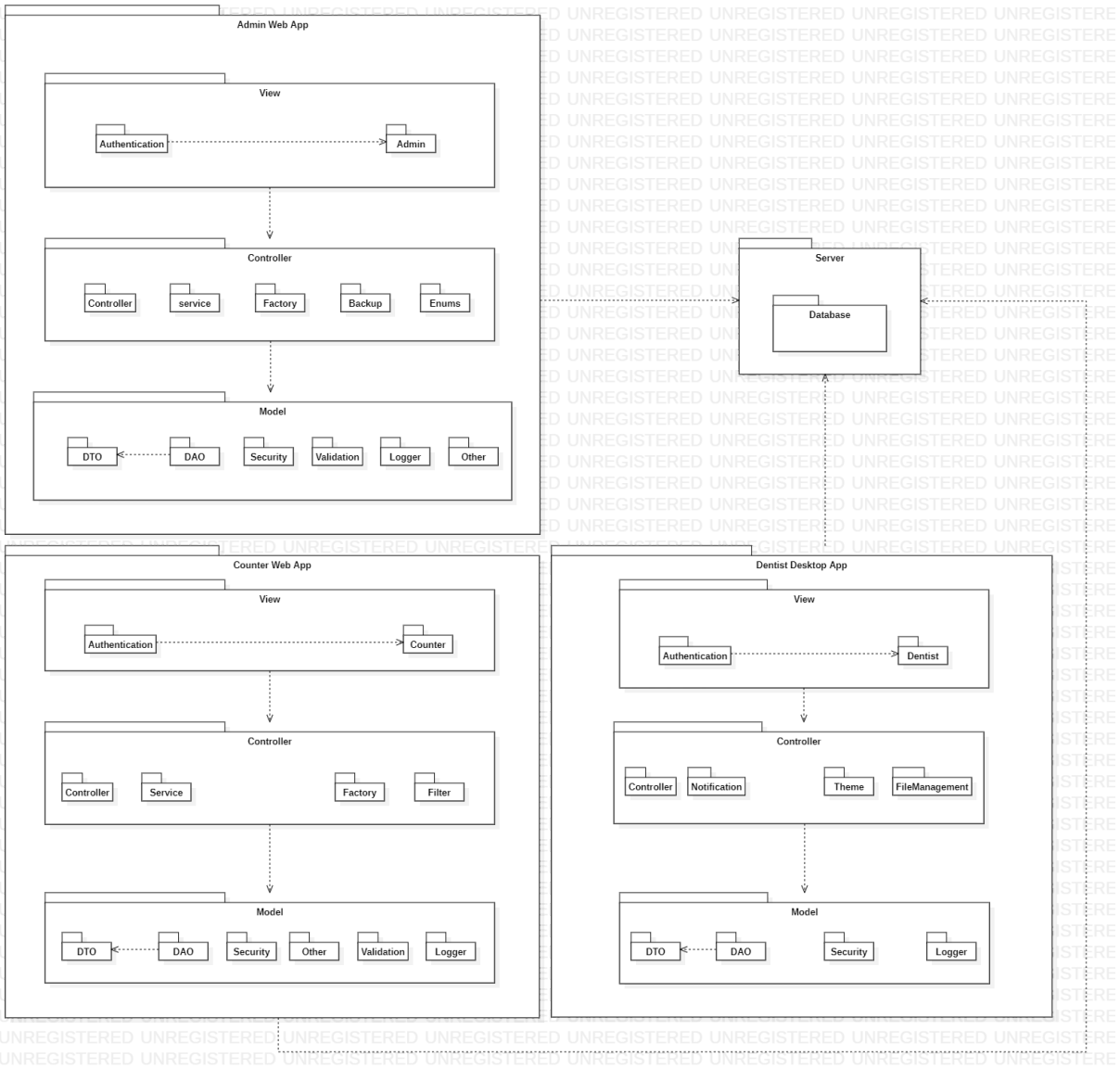
Sami **podsistem Controller** u sloju Controller je mnogo više opisan, istražen na dijagramu komponenata, tako da ovde nećemo nista posebno opisivati o njemu. **Sloj Model** je zadužen za interakciju sa perzistentnim podacima, pri čemu pod interakcijom podrazumijevamo CRUD operacije: kreiranje (eng. create), ažurirati (eng. update),brisanje (eng. delete), dohvatanje (eng. retrieve). Naravno pored interakcije sa perzistentnim podacima, mi u ovom sloju vodimo računa i o domenskoj logici(voditi računa o razlici domenske i aplikativne logike).

U **podsistemu File Management** implementiramo odgovarajuce funkcionalnosti za čitanje podataka iz odgovarajućih fajlova koje koriste svi podsistemi iz Controller sloja (ali nece svi podsistemi podsistema Controller). Preko **podsistema DTO** vršimo predstavljanje samih domenskih objekata. **Podsistem DAO** implementira komunikaciju sa bazom podataka. **Podsistem Security** obezbjeđuje sigurno kreiranje korisničkih naloga, kao i njihovu autentifikaciju. **Podsistem Validation** obezbjeđuje validaciju vrijednosti atributa(polja) domenskih objekata. **Podsistem Logger** obezbjeđuje perzistenciju exception-a koji se dese tokom rada aplikacije. **Podsistem Factory** koristimo za kreiranje objekata odgovarajućih klasa(factory design pattern). **Podsistem Backup** namijenjen je za perzistenciju cijele baze podataka u fajl. Kada se admin aplikacija prvi put instancira u tom trenutku pravi se kopija(eng. snapshot) trenutnog stanja baze podataka. I potrebno je napomenuti da se pravljenje kopija i dalje nastavlja na sedmicnom nivou, tj. ako smo napravili kopiju 2.5.2022 sljedeca kopija se pravi 9.5.2022. **Notification/Other**(*notification = other*) podsistem koristimo za(zavisi od konkretne aplikacije):

- dobijanje odgovarajuce poruke ili
- za prikaz odgovarajuce poruke

Sa sljedećeg dijagrama(dijagram dekompozicije) možemo da vidimo da je ovo zatvorena arhitektura tj. nijedan podsistem iz jednog sloja neće komunicirati sa podsistemom iz sloja koji nije njemu susjedan. Npr. podsistem Dentist neće direktno uzimati vrijednosti iz baze podataka i predstavljati ih sa odgovarajućom gui implementacijom već imamo posrednika Controller koji će da dohvati podatke, na odgovarajući način transformiše i vrati podsistemu Dentist. Možda se pitate pa zašto to radimo? Pa zato što vjerujemo da sami domenski objekti će se mnogo rjeđe mijenjati nego što će se sama aplikativna logika mijenjati ili sami način njihovog predstavljanja na gui-u.

3.2. Dekompozicija sistema



Dekompozicija sistema

3.3. HW/SW mapiranje

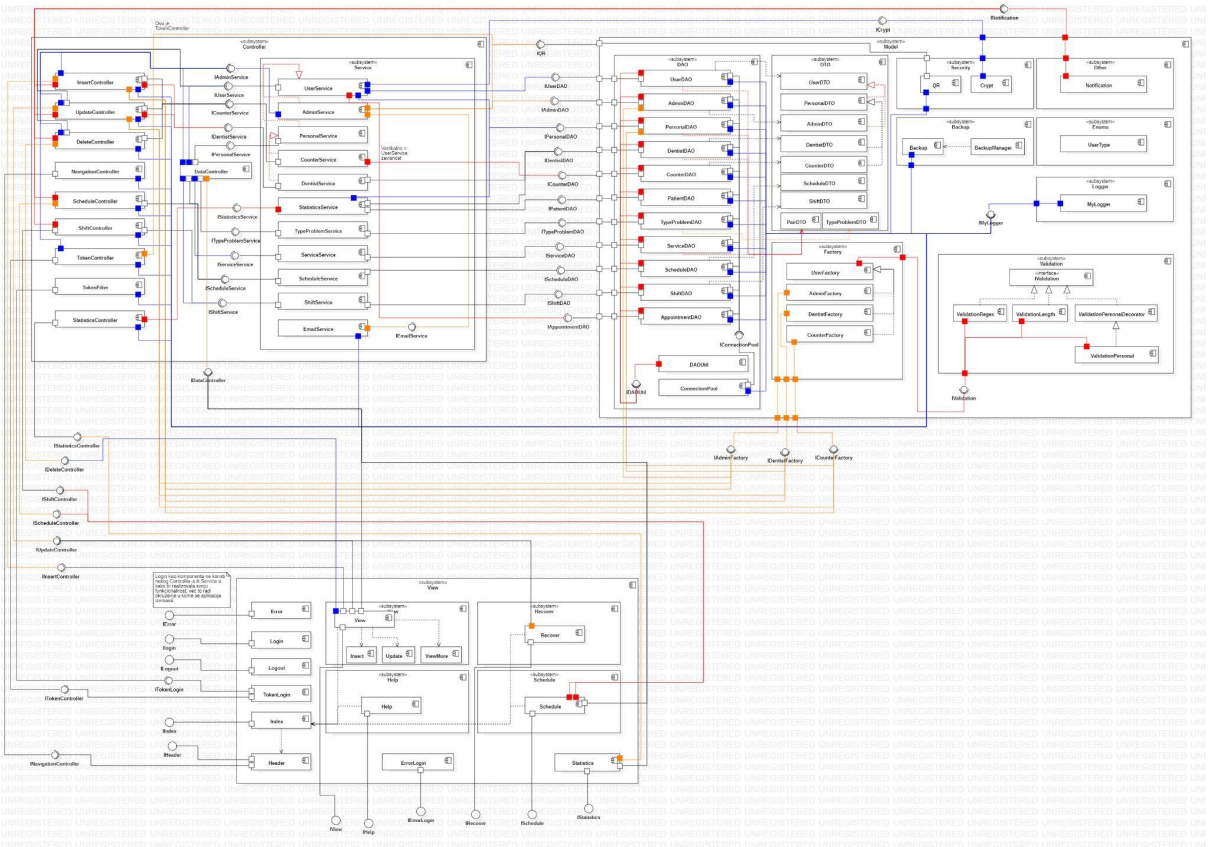
3.3.1. Dijagram komponenti

Zbog kompleksnosti samih dijagrama pojedini dijelovi su označeni bojom kako bi se omogućilo korisniku koji razgleda(razmatra) dijagram koja komponenta ima "vezu" sa kojom komponentom. Kad kazemo "veza" tada mislimo na:

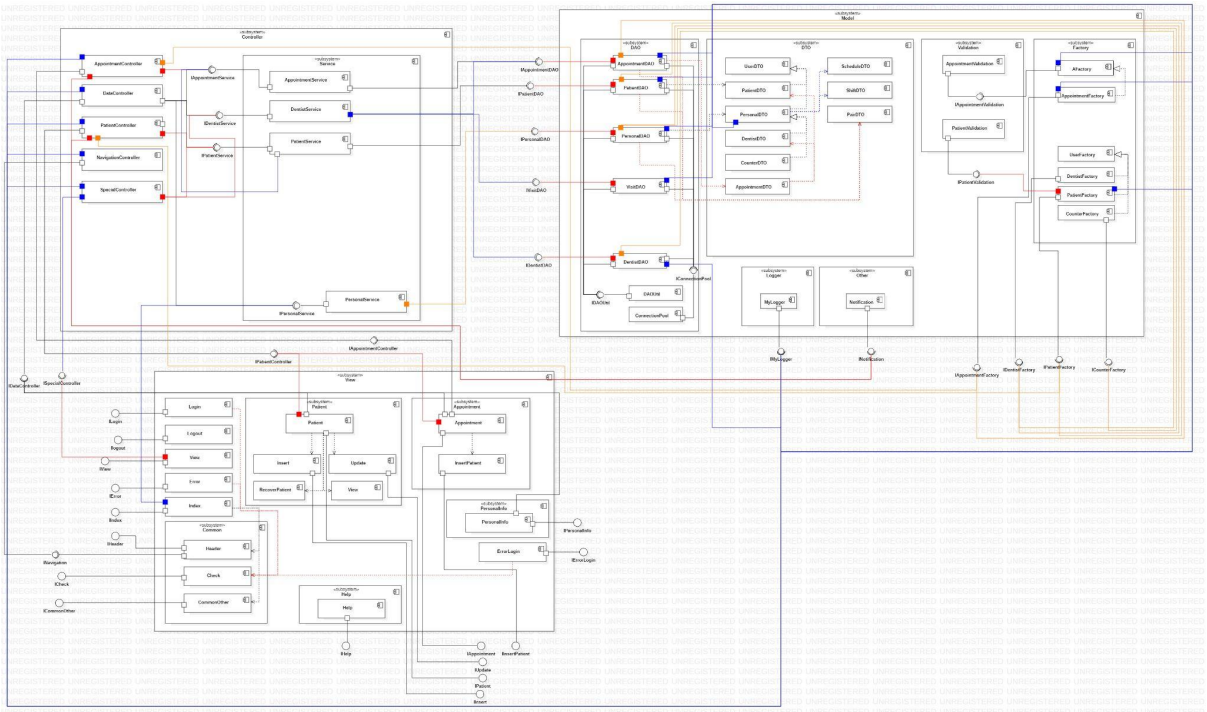
- Interface Realization
- Component Realization
- Dependency
- Association

- Aggregation
- itd.

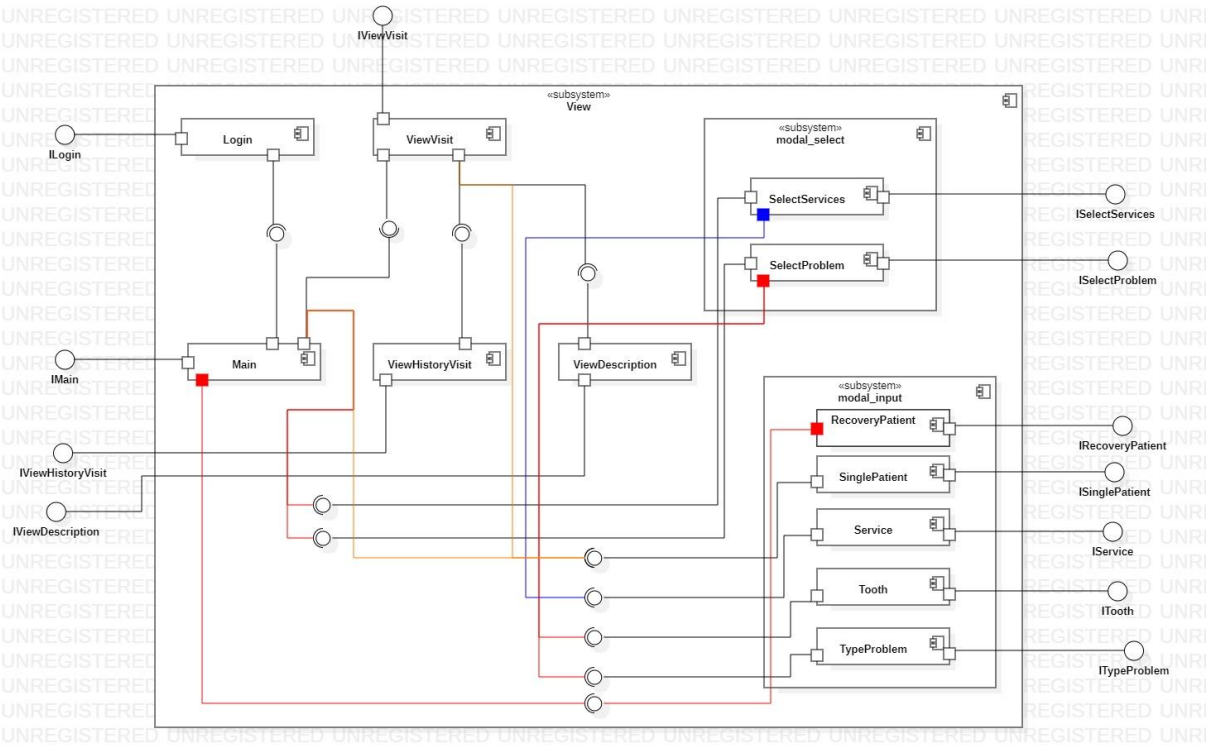
Tj. za razliku od prošle verzije ove dokumentacije, kada su boje oznacavale grupu sličnih aktivnosti sada boje pomažu Vama samo u razmatranju dijagrama.



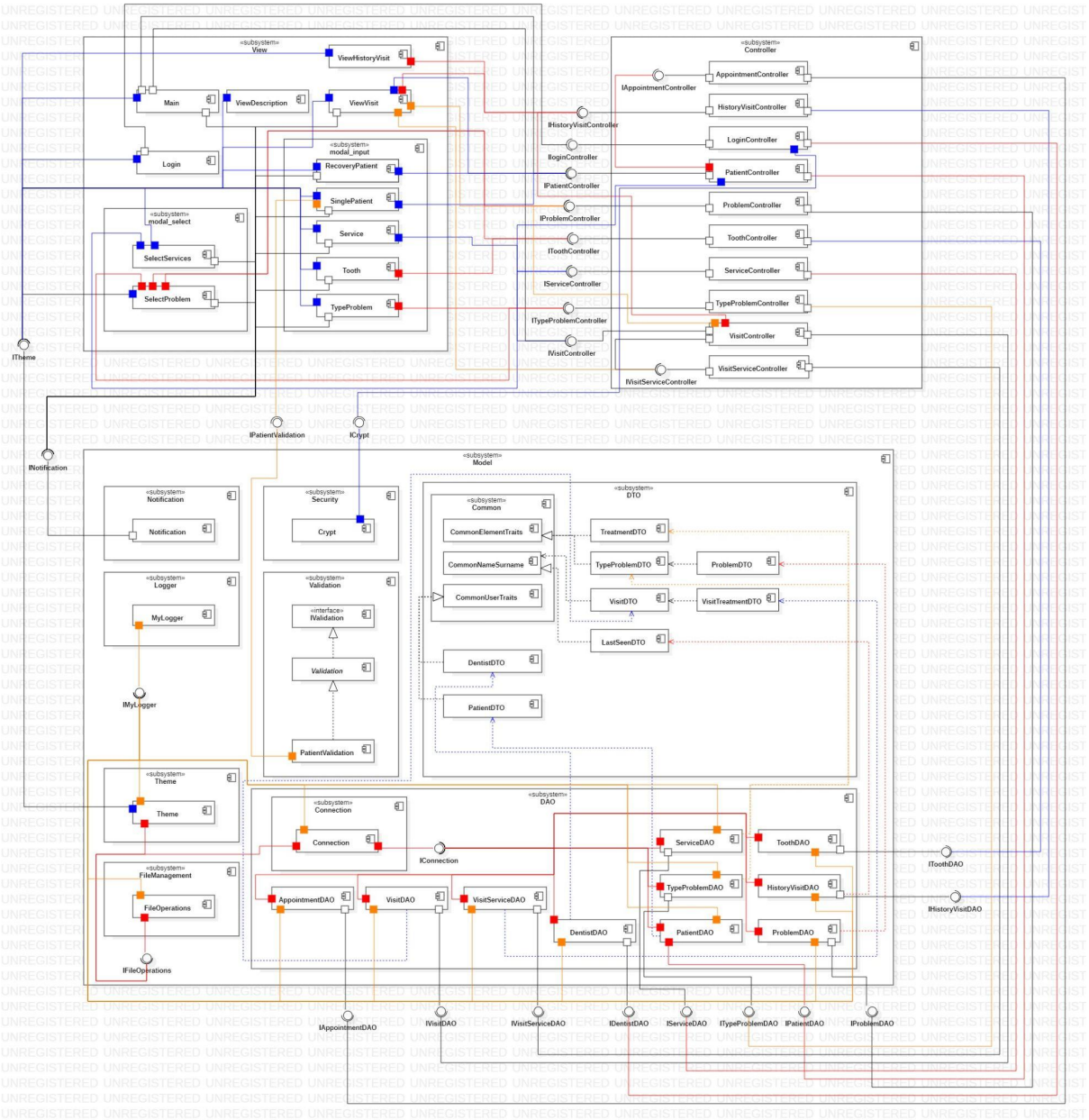
Dijagram komponenata - Admin



Dijagram komponenata - Counter(recepcioner, blagajnik)

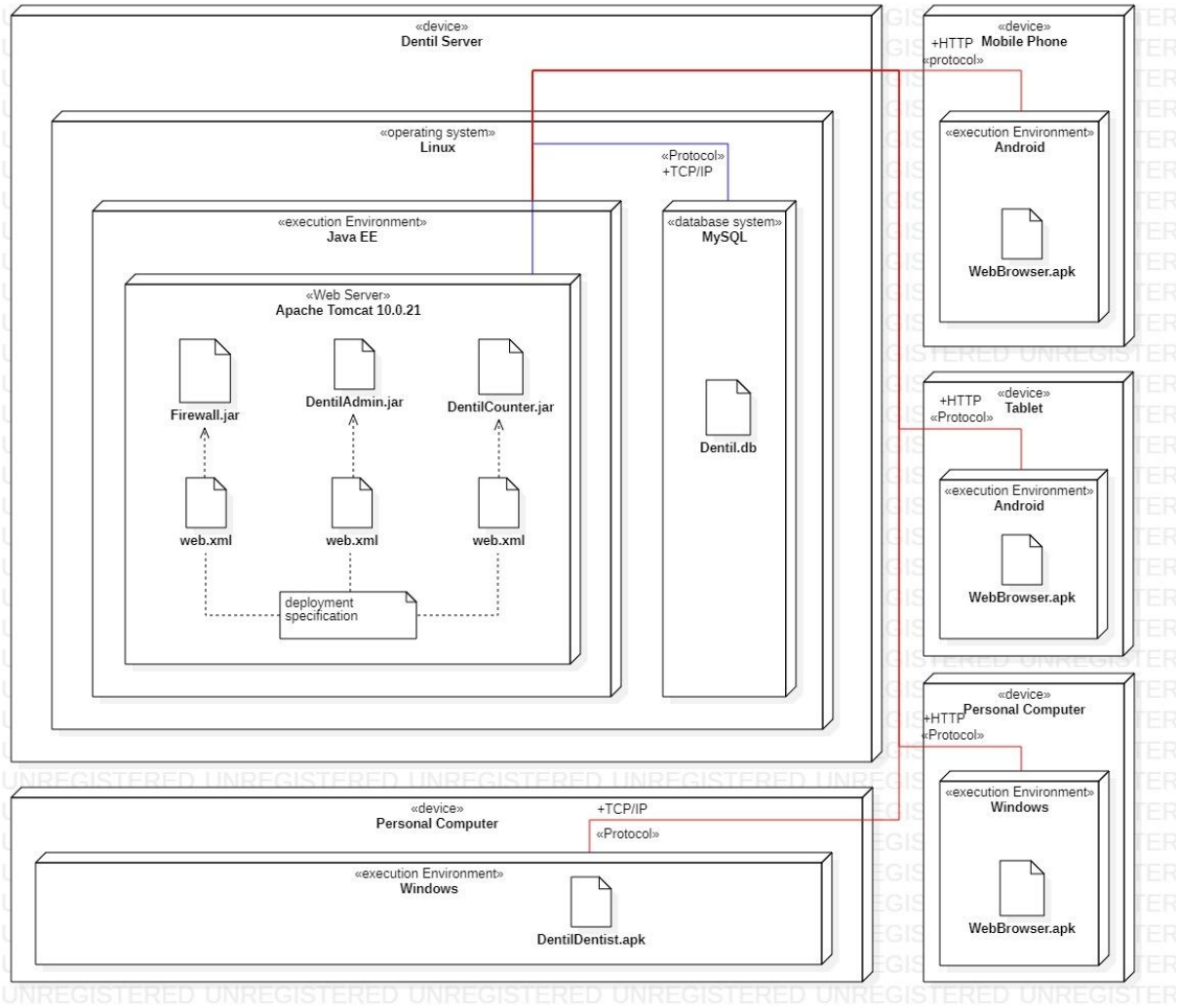


Dijagram komponenata Dentist - Detaljnije View subsystem



Dijagram komponenata Dentist

3.3.2. Dijagram razmještaja



Dijagram razmještaja

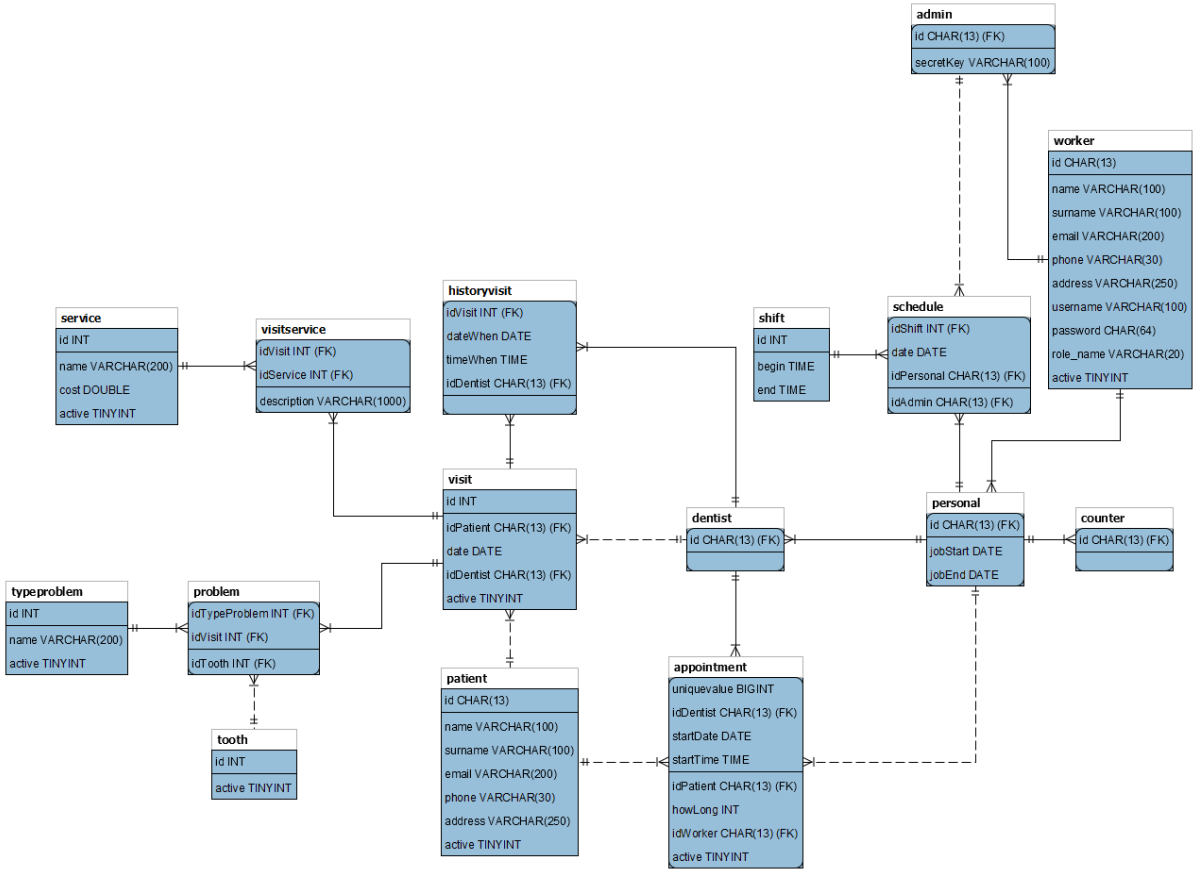
3.4. Perzistentni sloj

Za perzistenciju domenskih objekata aplikacije koristit ćemo MySql bazu podataka. Potrebno je napomenuti da koristimo i fajlove za čuvanje perzistentnih podataka. Kod Admin web aplikacije postoji two - factor authentication pri čemu smo to implementirali pomoću QR tokena. Da bi se korisnik tipa Admin mogao kreirati potrebno je da kreiramo prvo njegov file tipa .png pri čemu taj file predstavlja qr token jedinstveno generisan za novog korisnika. Tako da jedan vid perzistencije kod ove aplikacije jeste skladištenje QR tokena. Takođe obadvije web aplikacije skladište Exception-e koji se dese na file system-u servera. Takođe vršimo skladištenje cijele baze podataka sedmično, tako da, ako i dođe do nekog otkaza gubimo vrlo malo podataka.

Table Name	Field Name	Type Of Field	Null	Unsigned	Auto Increment	Unique	References	Primary Key	Condition
Worker	id	char(13)	no					yes	length(id)=13
	name	varchar(100)	no						length(name)>=2
	surname	varchar(100)	no						length(surname)>=2
	email	varchar(200)	no						length(email)>=3
	phone	varchar(30)	no						length(phone)>=2
	address	varchar(250)	no						length(address)>=2
	username	varchar(100)	no			yes			length(username)>=2
	password	char(64)	no						length(password)=64
	active	tinyint	no						
	role_name	varchar(20)	no						length(role_name)>=2
Patient	id	char(13)	no					yes	length(id)=13
	name	varchar(100)	no						length(name)>=2
	surname	varchar(100)	no						length(surname)>=2
	email	varchar(200)	yes						
	phone	varchar(30)	yes						
	active	tinyint	no						
	address	varchar(250)	yes						
Admin	id	char(13)	no				Worker(id)	yes	
	secretKey	varchar(100)	no						length(secretKey)>=2
Personal	id	char(13)	no				Worker(id)	yes	
	jobStart	date	no						
	jobEnd	date	yes						
Counter	id	char(13)	no				Personal(id)	yes	
Dentist	id	char(13)	no				Personal(id)	yes	
Shift	id	int	no		yes			yes	
	begin	time	no						
	end	time	no						
Schedule	idAdmin	char(13)	yes				Admin(id)		
	idPersonal	char(13)	no				Personal(id)	yes	
	date	date	no					yes	
	idShift	int	no				Shift(id)	yes	
Tooth	id	int	no	yes				yes	
	active	tinyint	no						
Service	id	int	no		yes			yes	
	cost	double	no						
	active	tinyint	no						
	name	varchar(200)	no						length(name)>=2
TypeProblem	id	int	no		yes			yes	
	active	tinyint	no						
	name	varchar(200)	no						length(name)>=2
Visit	id	int	no		yes			yes	
	date	date	no						
	idPatient	char(13)	no				Patient(id)		
	active	tinyint	no						
	idDentist	char(13)	no				Dentist(id)		
VisitService	idVisit	int	no				Visit(id)	yes	
	idService	int	no				Service(id)	yes	
	description	varchar(1000)	yes						
Problem	idTypeProblem	int	no				TypeProblem(id)	yes	
	idVisit	int	no				Visit(id)	yes	
	idTooth	int	yes	yes			Tooth(id)		
HistoryVisit	idVisit	int	no				Visit(id)	yes	
	idDentist	char(13)	no				Dentist(id)	yes	
	dateWhen	date	no					yes	
	timeWhen	time	no					yes	
Appointment	uniquevalue	bigint	no		yes	yes		yes	
	startDate	date	no					yes	
	startTime	time	no					yes	
	howLong	int	no	yes					howLong>0
	idWorker	char(13)	no				Personal(id)		
	active	tinyint	no						
	idPatient	char(13)	no				Patient(id)		
	idDentist	char(13)	no				Dentist(id)	yes	

Detaljnije informacije o bazi podataka

Potrebno je napomenuti da jedan atribut sa prethodne tabele(*slike*) ima drugačiju vrijednost i to je kod tabele **Schedule**. Misli se na vrijednost idShift u smislu da ono neće biti primarni ključ te tabele već samo vrijednosti date i idPersonal.



Model Baze Podataka

3.4.1. Perzistencija elemenata povezana sa Podsistemom Theme

Način na koji se pamti neka tema je preko nekoliko fajlova:

- colorPalette.txt - čuva boju na kojoj se tema bazira
- theme.txt - čuva vrstu teme primjene prethodno navedene boje

Kako se čuvaju ove vrijednosti i koje vrijednosti mogu da se nalaze u fajlovima?

- vrijednosti se čuvaju u jednoj liniji fajla
- vrijednost colorPalette.txt je npr. Blue, pri čemu moguce vrijednosti ovog fajla su: BlueGrey, Green, Blue. Tj. ograničen je broj boja koje korisnik može da koristi.
- vrijednost theme.txt je npr. DARK, pri čemu moguce vrijednosti ovog fajla su: LIGHT, DARK.

Ali šta to zapravo znači “čuva vrstu teme primjene prethodno navedene boje”? Pod tima zapravo mislimo ako se npr. u colorPalette.txt nalazi Green(zelena boja) onda ce vam se prikazati odgovarajuci elementi sa tom bojom najčešće kako bi se naglasio neki odjeljak, npr. dodajete posjet nekog pacijenta u sistem, tada u toj formi bi imali naglašeno sa zelenom bojom naslov gdje pise “Add Visit”. Dok ostatak aplikacije bi imao zamracenu pozadinu(ukoliko se unutar theme.txt nalazi DARK) sa nekom varijacijom bijelih slova. Ovo

smo implementirali zato što veliki broj današnjih, makar desktop aplikacija, podržava i cak mozemo reci da je i standard danas, jednostavno receno ono povećava user experience.

3.4.2. Perzistencija elemenata povezana sa Podsistemom Logger

Ranije je već ukratko pricano o perzistenciji korištenjem fajlova i jedan od primjera jeste zapravo za skladištene logova. Kada govorimo o web aplikacijama tada neophodno je postojanje sljedećeg fajl sistema:

- Dentil
- qr
- Admin
- Counter
- Dentist

Pri čemu ovaj fajl sistem treba da se nalazi unutar sljedećeg foldera: CATALINA_HOME. Tj. pri konfigurisanju sistema neophodno je da **definišete environment varijablu CATALINA_HOME** i unutar toga kreirate prethodno navedeni fajl sistem(*kreiranje je opcionalno ali definisanje CATALINA_HOME varijable je neophodno*). Naravno sve ovo sto mi pričamo jeste zapravo šta se radi na serveru gdje ove aplikacije se izvršavaju. Dok desktop aplikacija skladišti log fajlove unutar sljedece putanje: current_working_dir/../../model/history. Pošto je aplikacija implementirana pomoću c# onda current_working_dir ima sljedeće značenje: “Directory.GetCurrentDirectory()”

3.5. Kontrola prava pristupa i sigurnost

3.5.1. Kontrola prava pristupa

ADMIN	
Table	**Operations**
Worker	unlimited access
Personal	unlimited access
Admin	unlimited access
Dentist	unlimited access
Counter	unlimited access

Shift	unlimited access
Schedule	unlimited access
Appointment	select, update and delete
HistoryVisit	select and delete
Problem	select and delete
VisitService	select and delete
Visit	select, update and delete
Patient	select(only)
Service	select(only)
TypeProblem	select(only)

Tabela prava pristupa admina

COUNTER(Recepcioner**)**	
Table	**Operations**
Patient	unlimited access
Appointment	unlimited access
Worker	select(only)
Personal	select(only)
Counter	select(only)
Dentist	select(only)
Schedule	select(only)
Shift	select(only)
HistoryVisit	select and delete
Problem	select and delete
VisitService	select and delete
Visit	select, update and delete

Tabela prava pristupa recepcionera

Dentist	
Table	**Operations**
Worker	unlimited access
VisitService	unlimited access
Visit	unlimited access
TypeProblem	unlimited access
Tooth	unlimited access
Shift	unlimited access
Service	unlimited access
Schedule	unlimited access
Problem	unlimited access
Personal	unlimited access
Patient	unlimited access
HistoryVisit	unlimited access
Dentist	unlimited access
Counter	unlimited access
Appointment	unlimited access
Admin	unlimited access

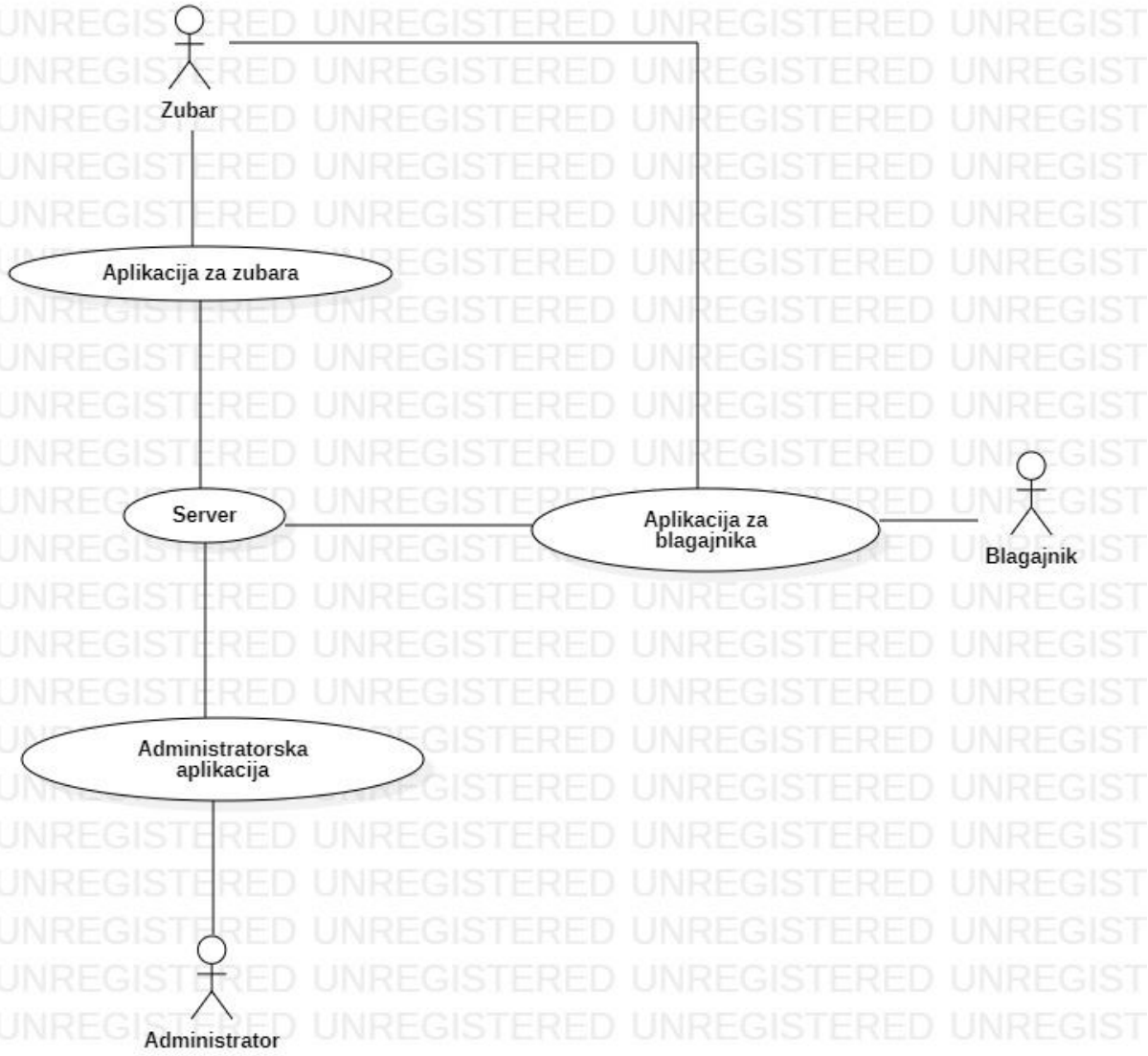
Tabela prava pristupa zubara

3.5.2. Sigurnost

Sigurnost sistema od malicioznih napada i neovlaštenog pristupa pojedinim dijelovima sistema biće ostvareno korištenjem korisničkog imena i šifre. Pri čemu način na koji ćemo prethodno implementirati je preko Hash + Salt vrijednosti. Naravno, pored ove stavke sigurnosti poduzet će se i neke druge, koje bi trebale da prevaziđu sljedeće maliciozne napade(uzeti u obzir da imamo web aplikaciju za administratora):

- SQL injection
- Session Hijacking
- DDOS attacks

3.6. Kontrola toka



Kontrola toka

3.7. Granična stanja sistema

3.7.1. Server

Pokretanje

Prilikom prvog pokretanja dolazi do inicijalizacije svih podsistema i kreiranja perzistentnih objekata poput baze podataka. Kada svi sistemi budu operativni, pokreće se analiza rada svih podsistema i provjere sigurnosnog statusa cjelokupnog servera.

Ukoliko dođe, iz nekog razloga, do ponovnog pokretanja servera ponavlja se isti postupak, osim kreiranja perzistentnih objekata i dodatno se na zahtjev izvršava proces rehabilitacije sistema.

Gašenje

Nakon prvog pokretanja nije predviđeno da se server gasi. Ukoliko dođe usljed neke greške ili prekida napajanja, izvršiće se čuvanje trenutnog stanja sistema i memorisanje informacija o uzroku gašenja.

Prekid konekcije

Ukoliko dođe do prekida konekcije sa nekom od eksternih aplikacija, pokušaće se ponovna uspostava konekcije, ukoliko to nije moguće konekcija se terminira, jedini izizetak je ,,Administratorska aplikacija'', na kojoj je prijavljen Alfa administrator. Konekcija sa ,,Administratorskom aplikacijom'' će biti otvorena jedan vremenski period. Ukoliko se konekcija ne uspostavi do određenog vremenskog perioda, konekcija će biti terminirana.

Pad podsistema

Ukoliko dođe do pada podsistema, server započinje proces oporavka datog podsistema. Dok proces traje, svi zahtjevi upućeni ka tom podsistemu se odbacuju i pošiljalac zahtjeva dobija obavještenje o nedostupnosti tražene operacije. Sve informacije vezane za pad podsistema i proces oporavka se čuvaju u arhivi.

Pojava greške

Ukoliko dođe do pojave greške, vrši se čuvanje informacija o grešci u arhivi i u zavisnosti od vrste greške i mjesta gdje se greška desila preduzimaju se određene akcije. Ako je greška manjeg intenziteta preduzimaju se mjere brzog oporavka i otklanjanja greške uz minimalne gubitke vremena i podataka. Ako je greška većeg intenziteta, čitav podsistem u kome se dogodila greška ide u blokadu, sve dok se greška ne otkloni. Prilikom pojave greške, procesu koji zahtjeva rezultat, iz procesa u kome se desila greška, dostavlja se obavještenje o ozbiljnosti problema.

3.7.2. Administratorska aplikacija

Pokretanje

Prilikom svakog pokretanja aplikacije prikazuje se prozor za prijavu administratora. Nakon uspješne prijave učitavaju se podešavanja i aplikacija se inicijalizuje u početno stanje.

Zatvaranje

Prilikom zatvaranja aplikacije provjerava se da li je administrator ostavio neki proces nedovršen. Ukoliko se otkrije postojanje nedovršenog procesa, administratoru se daje naznaka u vidu iskačućeg prozora u kome se zahtjeva da se dati proces završi ili da se izvrši resetovanje izmjena koje je dati proces napravio. Prilikom zatvaranja aplikacije od administratora se zahtjeva da se odjavi sa date aplikacije, ukoliko se to ne dogodi, aplikacija vrši prisilnu odjavu administratora, a informacija o tome se dostavlja serveru.

Prekid konekcije sa bazom podataka

Ukoliko dođe do prekida konekcije sa serverom, svi zahtjevi koji su bili upućeni ka serveru se obustavljaju i stavljaju u red, a slanje novih zahtjeva se onemogućava sve do ponovnog uspostavljanja konekcije. Administrator dobija obavještenje o gubitku konekcije. Prilikom ponovnog uspostavljanja konekcije, administrator dobija obavještenje o uspostavi konekcije i dobija dijalog prozore za ponovno slanje neposlanih zahtjeva, za svaki zahtjev pojedinačno uz objašnjenje zahtjeva.

Pad servera

Ukoliko dođe do pada servera, svi zahtjevi koje korisnik upućuje ka serveru se odbacuju. Ukoliko korisnik ne upućuje nikakav zahtjev ka serveru trenutno učitana stranica(ukoliko ste učitali prije pada servera) ce biti vidljiva korisniku, pri čemu on moze da radi statičke operacije bez gubljenja sadržaja stranice. U zavisnosti koji web pretraživač je korišten, korisnik će dobiti odgovarajuću poruku.

Pojava greške

Ukoliko dođe do pojave greške, vrši se čuvanje informacija o grešci u arhivi. Korisniku koji je zahtijevao od sistema da se izvrsi operacija kojom je prouzrokovana greška, prikazuje se poruka da je izvršavanje operacije neuspješno bez ikakvih detalja jer to je namijenjeno za arhive. Ali sad se možda pitate šta ako neku grešku niste predvidjeli tj. niste je obradili? Ukoliko se desi greška tokom izvršavanja neke operacije koja nije programerski obrađena, onda se korisniku prikazuje posebna stranica sa generičkim tekstom, kao što je npr. Ooops something happened …

3.7.3. Aplikacija za zubara

Pokretanje

Prilikom prvog pokretanja aplikacije korisniku se prikazuje prozor za unos kredencijala. Taj isti prozor se prikazuje svaki put kada se aplikacija ponovo pokreće, a korisnik se prilikom posljednjeg gašenja aplikacije odjavio.

Pri svakom pokretanju aplikacije, dolazi do provjere da li je korisnik već prijavljen. Potom se od korisnika traži da unese ispravnu lozinku, nakon čega dolazi do učitavanja podešavanja i inicijalizacije aplikacije u početno stanje.

Zatvaranje

Prilikom zatvaranja aplikacije se ne vrši provjera da li je neki proces ostao nedovršen te ga odbacuje.

Prekid konekcije sa bazom podataka

Ukoliko dođe do prekida konekcije sa serverom, svi zahtjevi koji bivaju upućeni ka serveru se obustavljaju i dolazi do njihove klasifikacije. Jedan dio zahtjeva, koji je za to predviđen, stavlja se u red i čeka na ponovno uspostavljanje konekcije sa serverom kako bi mogli biti poslati, dok se ostali zahtjevi odbacuju. U oba slučaja korisnik dobija vizuelno obavještenje o stanju njegovog zahtjeva.

Pad servera

Ukoliko dođe do pada servera, svi zahtjevi koje korisnik upućuje ka serveru se odbacuju. Ukoliko korisnik ne upućuje nikakav zahtjev ka serveru trenutno učitana stranica(ukoliko ste učitali prije pada servera) ce biti vidljiva korisniku, pri čemu on moze da radi statičke operacije bez gubljenja sadržaja stranice. U zavisnosti koji web pretraživač je korišten, korisnik će dobiti odgovarajuću poruku.

Pojava greške

Ukoliko dođe do pojave greške, vrši se čuvanje informacija o grešci u arhivi. Korisniku koji je zahtijevao od sistema da se izvrsi operacija kojom je prouzrokovana greška, prikazuje se poruka da je izvršavanje operacije neuspješno bez ikakvih detalja jer to je namijenjeno za arhive.

3.7.4. Aplikacija za blagajnika

Pokretanje

Prilikom prvog pokretanja aplikacije, korisniku se prikazuje prozor za unos kredencijala. Taj isti prozor se prikazuje svaki put kad se aplikacija ponovo pokreće, a korisnik se prilikom poslednjeg gašenja aplikacije odjavio.

Pri svakom pokretanju aplikacije dolazi do provjere da li je korisnik već prijavljen. Aplikacija učitava podešavanja i inicijalizuje se u početno stanje.

Prekid konekcije sa bazom podataka

Ukoliko dođe do prekida konekcije sa serverom, svi zahtjevi koji su bili upućeni ka serveru se obustavljaju i stavljaju u red, a slanje novih zahtjeva se onemogućava sve do ponovnog uspostavljanja konekcije. Administrator dobija obavještenje o gubitku konekcije. Prilikom ponovnog uspostavljanja konekcije, administrator dobija obavještenje o uspostavi konekcije i dobija dijalog prozore za ponovno slanje neposlanih zahtjeva, za svaki zahtjev pojedinačno uz objašnjenje zahtjeva.

Pad servera

Ukoliko dođe do pada servera, svi zahtjevi koje korisnik upućuje ka serveru se odbacuju. Ukoliko korisnik ne upućuje nikakav zahtjev ka serveru trenutno učitana stranica(ukoliko ste učitali prije pada servera) ce biti vidljiva korisniku, pri čemu on moze da radi statičke operacije bez gubljenja sadržaja stranice. U zavisnosti koji web pretraživač je korišten, korisnik će dobiti odgovarajuću poruku.

Pojava greške

Ukoliko dođe do pojave greške, vrši se čuvanje informacija o grešci u arhivi. Korisniku koji je zahtijevao od sistema da se izvrsi operacija kojom je prouzrokovana greška, prikazuje se poruka da je izvršavanje operacije neuspješno bez ikakvih detalja jer to je namijenjeno za arhive. Ali sad se možda pitate šta ako neku grešku niste predvidjeli tj. niste je obradili? Ukoliko se desi greška tokom izvršavanja neke operacije koja nije programerski obrađena, onda se korisniku prikazuje posebna stranica sa generičkim tekstom, kao što je npr. Ooops something happened …