Sustav za praćenje darivanja krvi

Dubinjak, Mateo

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

Sustav za praćenje darivanja krvi

Diplomski rad

Mateo Dubinjak

Osijek, 2021.

SADRŽAJ

1. UVOD

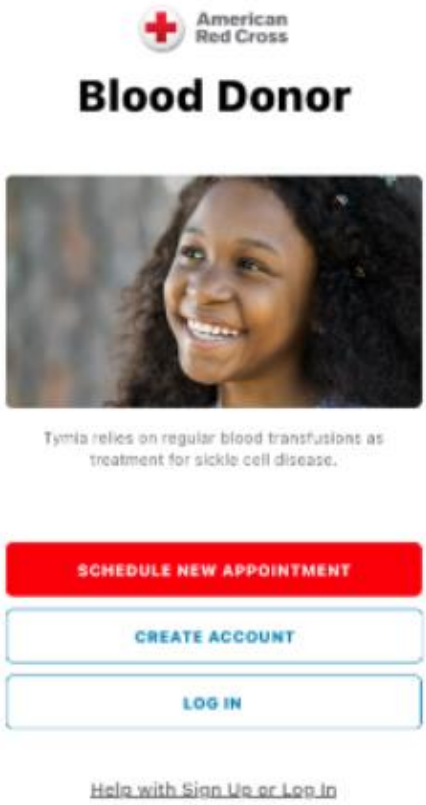
Dobrovoljno darivanje krvi je socijalni program država kojim se osiguravaju dovoljne zalihe krvi za brzo i sigurno liječenje bolesnika. Da bi se osigurale dovoljne količine krvi u pričuvi potrebno je imati dovoljan broj darivatelja krvi. Iako plemeniti čin koje dobrovoljci rado obnašaju, zalihe krvi su često u Republici Hrvatskoj premale. Razlog tome je neadekvatna i nedovoljna promidžba dobrovoljnog darivanja krvi. Kako bi se cijeli sustav darivanja krvi poboljšao i doveo na dobro organiziranu razinu potrebno je izmijeniti postojeći sustav s drugim, bolje koordiniranim sustavom. Aplikacija za darivanje krvi može riješiti dosta problema s kojima se Hrvatski zavod za transfuzijsku medicinu susreće te tako poboljšati rad, a time i povećati uspješnost zavoda. Praćenje darivanja krvi je bitno jer potiče, angažira i motivira potencijalne i postojeće darivatelje krvi. Darivatelji bi preko aplikacije imali uvid u detalje svake transakcije krvi koju su proveli. U to bi bili uključeni svi potrebni podaci o pojedincima za darivanje krvi; jesu li darivali krv, ako su odbijeni razlog tome, upitnik koji se ispunjava pri darivanju krvi o zdravstvenom stanju darivatelja i rezultati liječničkog pregleda za krvni tlak, ritam srca te razinu hemoglobina. Na isti način darivatelj nakon analize krvi može dobiti obavijest s povratnom informacijom za svoju krv, je li krv pogodna za darivanje i jesu li pronašli ikakve zdravstvene probleme kod samoga darivatelja. Još jedna stavka koja se može ukomponirati je praćenje lokacije krvi ako je ona pogodna za darivanje. Tako bi darivatelj znao gdje se njegova krv nalazi u svakom trenutku i je li ta ista krv iskorištena kako bi se nekome spasio život. Kada bi darivatelji imali uvid u sve te podatke bi imali i podsjetnik na to da njihovo plemenito djelo stvarno pomaže spasiti ljudima živote. Bili bi puno angažiraniji u cijelom procesu i samim time nastavili darivati krv. Aplikacija će koristiti vlastiti API kako bi sve te podatke prikazala na smislen i koncizan, a ujedno i u potpunosti informativan način.

2. PREGLED PODRUČJA TEME

Iako u Republici Hrvatskoj ne postoje slična rješenja problematike ovoga rada, jednu od najboljih raspoloživih aplikacija u SAD-u za praćenje darivanja krvi nudi Američki Crveni Križ (engl. *The American Red Cross*)[1].

Blood Donor App

Nadzornoj ploči aplikacije se pristupa preko stranice za prijavu, kao na slici 2.1.

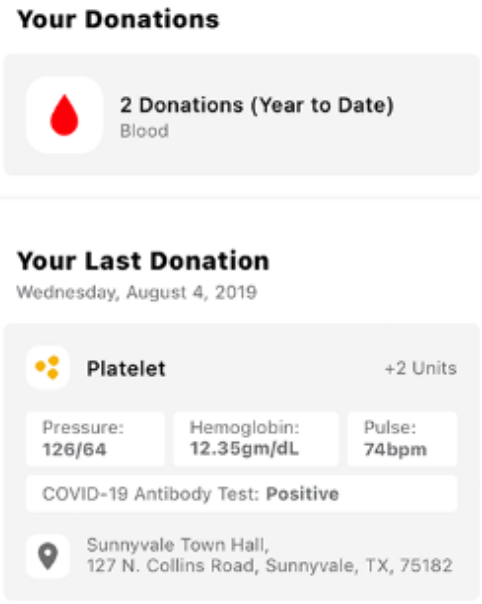


Slika 2.1. Stranica za prijavu u sustav

Isto tako aplikacija omogućava samostalnu registraciju u sustav. Kod sustava za praćenje darivanja krvi iz ovog rada ta opcija nije omogućena niti implementirana jer će registracija novih darivatelja biti prepuštena administratorima sustava. Razlog tome je lakše praćenje i validiranje podataka darivatelja koji bi se provjeravali već pri njihovoj prvoj donaciji krvi. Za

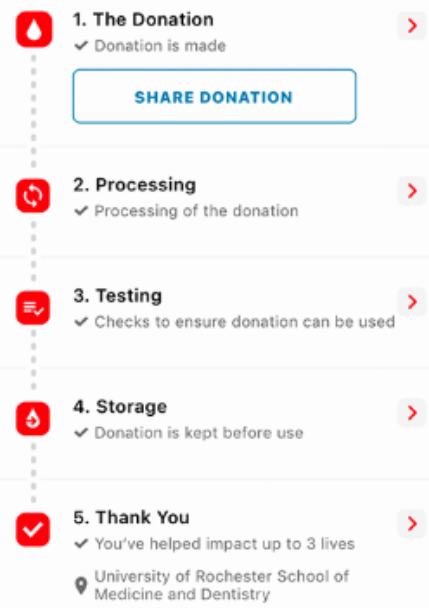
[1] American Red Cross aplikacija za praćenje darivanja krvi - https://www.redcrossblood.org/blood-donor-app.html

već postojeće darivatelje podatke o prijašnjim donacijama i registraciju administrator također može dodati na admin stranici.



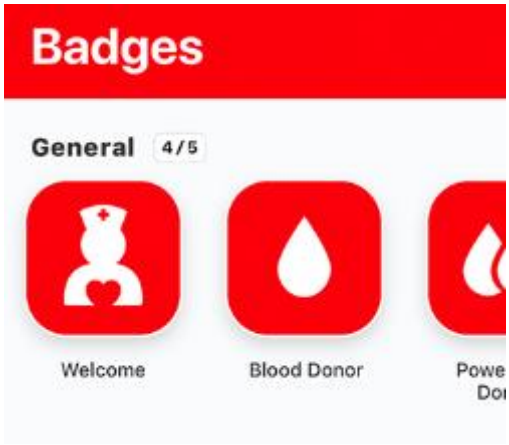
Slika 2.2. Povijest darivanja krvi

Kao i aplikacija američkog crvenog križa, ovaj sustav će sadržavati cijelu povijest darivanja krvi kako bi darivatelji imali uvid u prijašnje donacije. Tako mogu vidjeti kada su zadnji put darivali krv i pratiti zdravstveno stanje svoje krvi (Sl. 2.2.).



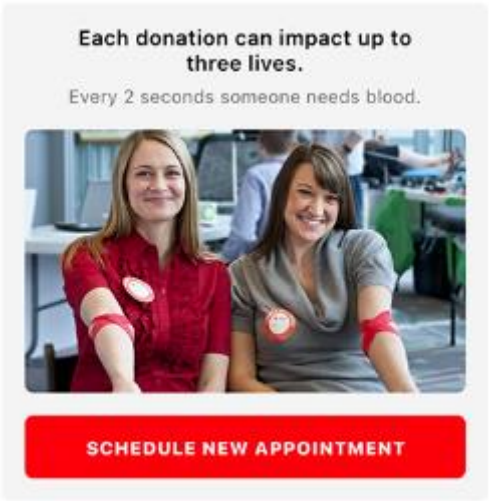
Slika 2.3. Praćenje darovane krvi

Sustav za praćenje darivanja krvi kao i aplikacija američkog crvenog križa prikazuje status darivane krvi (Sl. 2.3.). Jedna od brojnih značajki aplikacije američkog crvenog križa koja je informativnog sadržaja a ovaj sustav nema je praćenje dobivenih zahvalnica za darivanje krvi i društvenih priznanja poput povelje humanosti i odličja koje predsjednik predaje (Sl. 2.4.). Također bi bilo od informativnog značaja dodati i pregled pogodnosti koje su darivatelji krvi zaslužili svojim plemenitim činovima; to su u većini slučajeva oslobađanja od participacije u troškovima zdravstvene zaštite, besplatan gradski prijevoz i slično.



Slika 2.4. Društvena priznanja darivatelju

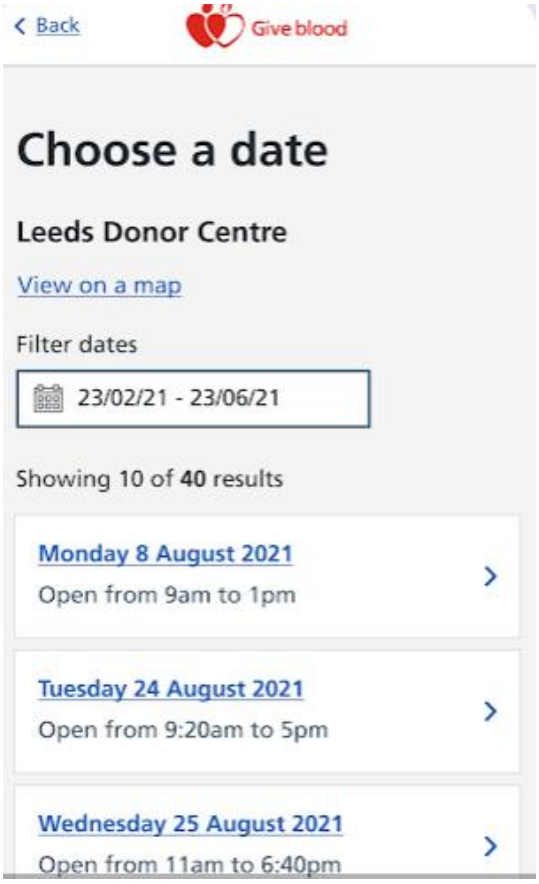
Isto tako aplikacija američkog crvenog križa pruža mogućnost dogovaranja termina darivanja krvi i mijenjanje istih (Sl. 2.5.).



Slika 2.5. Omogućava dogovaranje termina darivanja

NHS Give Blood App[2]

Aplikacija koju nudi engleski nacionalni zdravstveni servis (engl. *National Health Service – NHS*). Pruža uvid u raspoloživost termina za darivanje krvi u stvarnom vremenu, kao što je prikazano na slici 2.6.



Slika 2.6. *NHS Give Blood* izbor termina

Šalje obavijesti svojim darivateljima ako postoje akcije darivanja krvi koje bi se ticale njih s obzirom na potrebnu krvnu grupu. Za razliku od sustava za praćenje darivanja krvi *NHS Give Blood* ne pruža uvid u povijest darivanja a time niti u detalje o prijašnjim darivanjima krvi.

Blood Donor Mobile[3]

Aplikacija koju nudi Yoko.co isključivo kao mobilnu aplikaciju. Šalje obavijesti i podsjetnike vezane za darivanje krvi. Sadrži *lokator* preko kojega se mogu locirati najbliži centri za darivanje krvi. Povijest darivanja krvi ne dodaje niti prikazuje krajnjem korisniku Yoko nego

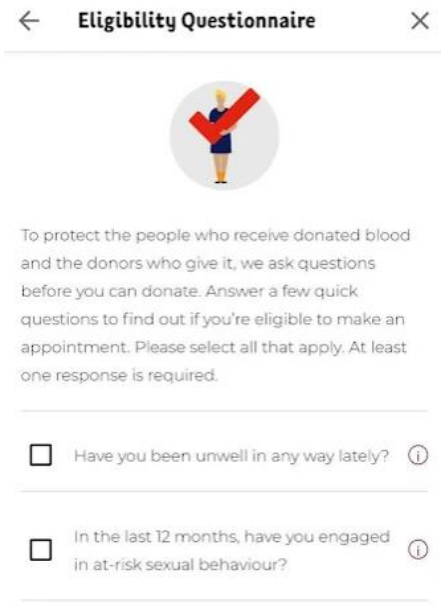
[2] NHS Give Blood App - https://www.blood.co.uk/the-donation-process/the-nhsgiveblood-app/
[3] Yoko.co Blood Donor Mobile - https://www.blooddonormobile.com/

nude opciju dodavanja povijesti darivanja krvi tako da darivatelji mogu sami brinuti o tome ako hoće (Sl. 2.7.).

Slika 2.7. Sučelje Blood Donor aplikacije

Donate Blood App[4]

Aplikacija koju nudi Australijski Crveni Križ (engl. *Australian Red Cross*). Nudi rezervacije za darivanje krvi i iste sinkronizira s kalendarom kako bi darivatelji lakše pamtili i pratili svoje termine. Nema mogućnost praćenja prijašnjih darivanja krvi i detalja vezanih uz njih. Sadrži mapu s popisom svih Lifeblood donacijskih centara s radnim vremenima istih. Pri rezervaciji je moguće ispuniti upitnik o zdravstvenom stanju tako da se taj korak može preskočiti u terminu darivanja krvi (Sl. 2.8.).



Slika 2.8. Zdravstveni upitnik aplikacije *Donate Blood*

[4] Donate Blood App - https://www.lifeblood.com.au/contact/donate-blood-app

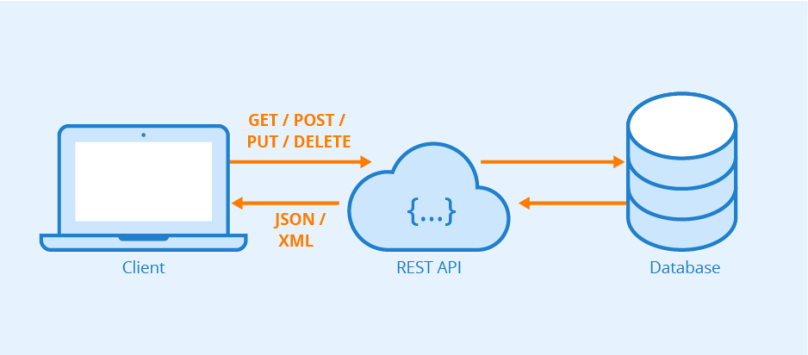
3. SUSTAV ZA PRAĆENJE DARIVANJA KRVI

U ovome poglavlju je opisan cijeli proces izrade sustava za praćenje darivanja krvi. Od korištenih tehnologija, arhitekture sustava, do programskog rješenja samoga sustava.

3.1. Arhitektura sustava

API

Primjensko korisničko sučelje (engl. *application programming interface - API*) je programski posrednik koji omogućuje komunikaciju između dvije ili više aplikacija. Pri korištenju aplikacije, aplikacija se spaja na internet i šalje podatke na server. Server te podatke dohvaća, interpretira, izvodi potrebne radnje i manipulacije nad njima te ih vraća na korisničko sučelje. Korisničko sučelje zatim interpretira te podatke i prikazuje krajnjem korisniku u informativnom i čitkom izdanju. Sve navedeno se odvija preko API sučelja, kao što je vidljivo na slici 3.1.



Slika 3.1. API komunikacija[5]

Web API

Sam po sebi nije tehnologija ali se koristi kao koncept. *Web* API je API na *webu* kojemu se može pristupiti koristeći HTTP (engl. *Hypertext Transfer Protocol*).

Protokoli

Protokoli definiraju skup pravila po kojima se podaci razmjenjuju unutar ili između računala. Prema [2], HTTP je protokol koji omogućava dohvaćanje resursa. Temelj je svake razmjene podataka na *webu* i obnaša ulogu klijent-server protokola jer se zahtjevi šalju od strane primatelja, često internetskog preglednika.

[5] Izvor: https://www.astera.com/type/blog/rest-api-definition/ (rujan 2021.)

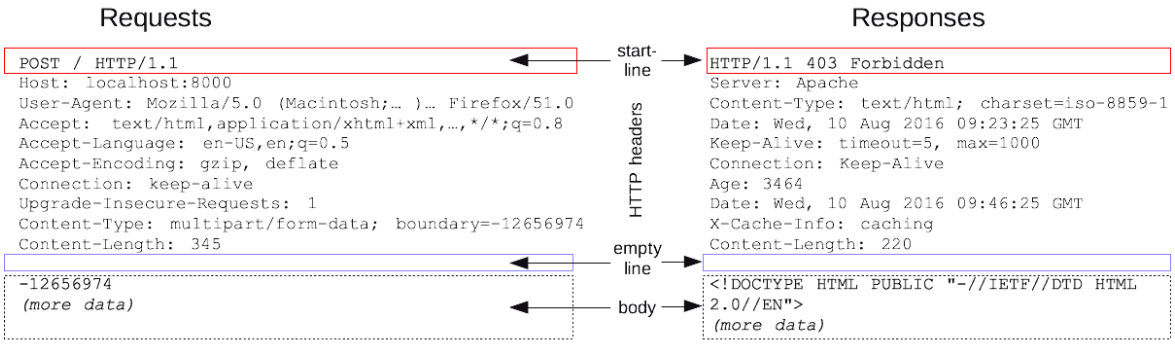
Najčešće HTTP metode za slanje zahtjeva su opisane u tablici 3.1.

Tablica 3.1. HTTP metode

Metoda	Funkcija	CRUD	Zahtjev ima tijelo	Odgovor ima tijelo
GET	Dohvatiti podatke	Read	Ne	Da
POST	Spremiti podatke	Create	Da	Da
PUT	Ažurirati podatke	Update	Da	Da
DELETE	Izbrisati podatke	Delete	Ne	Da

Svaka od tih metoda je ekvivalentna jednoj od CRUD metoda baze podataka jer ih se poziva upravo s namjerom da se iste te radnje obave i u bazi podataka.

HTTP zahtjevi se na API šalju u obliku URI-ja (engl. *Uniform Resource Identifier*) s HTTP porukom koja se sastoji od početne linije, zaglavlja i tijela (Sl. 3.2.).



Slika 3.2. HTTP poruka[6]

Primjer URI-ja za GET metodu:

https://localhost:4200/api/car/mazda

Prema [3], da bi se HTTP poruke mogle slati i dobivati odgovore potrebno je prije toga otvoriti TCP konekciju (engl. *Transmission Control Protocol*) koja omogućava razmjenu podataka

[6] Izvor: https://developer.mozilla.org/en-US/docs/Web/HTTP/Messages (rujan 2021.)

između klijenta i servera. Funkcionalno gledajući, TCP odrađuje samu razmjenu podataka, dok HTTP određuje i enkapsulira što točno želimo dohvatiti sa servera ili poslati na server.

3.2. Tehnički preduvjeti i tehnologije

JavaScript Object Notation (JSON)

Prema [4], JSON je format namijenjen razmjeni podataka. Lako se čita i piše, a strojevi ga bez ikakvih poteškoća raščlanjuju, odnosno *parsiraju*. U ovome projektu za razmjenu podataka preko tijela HTTP poruke JSON se koristi kao format.

Podijeljen je u dvije strukture:

- Zbirka ime/vrijednost parova koje programski jezici prepoznaju kao objekt
- Lista vrijednosti koja kod programskih jezika predstavlja listu ili niz

Primjer JSON-a koji se sastoji od zbirke imena/vrijednosti i liste vrijednosti prikazan je na slici 3.3.


```
{
  "business_id": "PK6aSizckHFWk8i0oxt5DA",
  "full_address": "400 Waterfront Dr E\nHomestead\nHomestead, PA 15120",
  "hours": {},
  "open": true,
  "categories": [
    "Burgers",
    "Fast Food",
    "Restaurants"
  ],
  "city": "Homestead",
  "review_count": 5,
  "name": "McDonald's",
  "neighborhoods": [
    "Homestead"
  ],
```


Slika 3.3. Primjer JSON-a

JSON *Web Token* (JWT)

Prema [5], JWT je žeton koji se šalje API-ju u zaglavlju HTTP poruke. API zatim ovlašćuje žeton i ako uspješno bude ovlašten odobrava pristup podacima i manipulaciji istih. Sastoji se od tri dijela odvojenih točkom.

Uzimajući u obzir strukturu žetona, izgled žetona je sljedeći:

xxxxxxx.yyyyyyy.zzzzzz

Zaglavlje

Sadrži podatak o vrsti žetona i algoritmu za digitalni potpis.

JSON zaglavlje (Sl. 3.4.) se radi sigurnosti i prijenosa enkodira Base64Url shemom iz binarnog oblika u ASCII format. Tako se izbjegavaju nedozvoljeni znakovi na određenim okruženjima i moguće izmjene podataka.

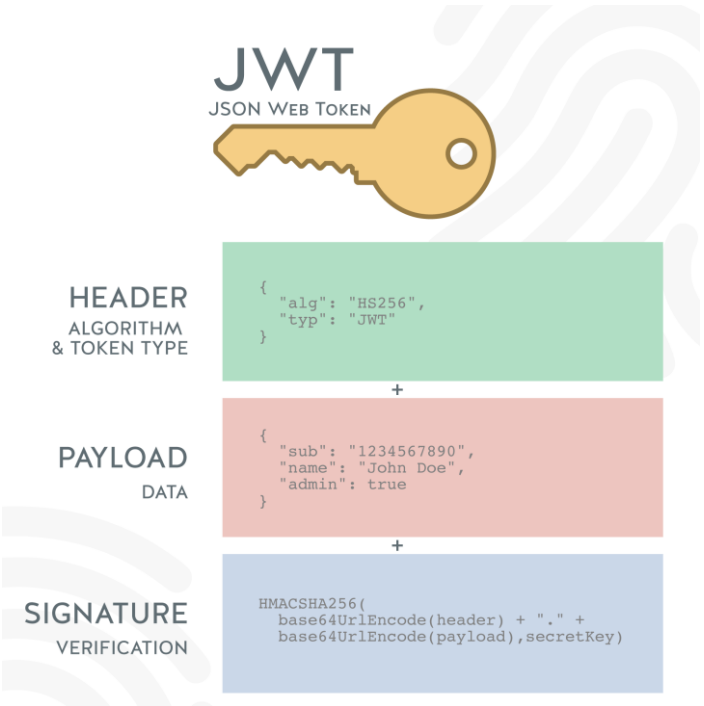
Teret

Sastoji se od takozvanih *claims-a* koji sadrže osnovne podatke o osobi i dodatne podatke po potrebi.

JSON tereta (Sl. 3.4.) se također enkodira Base64Url shemom.

Potpis

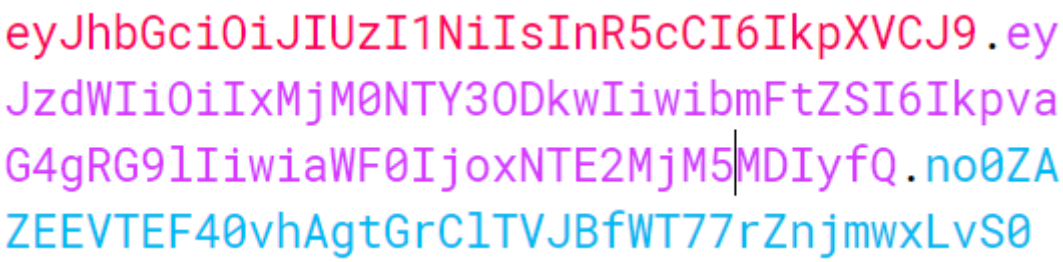
Koristi algoritam kako bi napravio potpis koji se kasnije koristi za potvrđivanje poruke. Tako se osigurava da poruka nije izmijenjena negdje na putu. Na slici 3.4. se koristio HMAC SHA256 algoritam koji uzima zaglavlje, teret te zadanu tajnu i s njima enkodira potpis.



Slika 3.4. Primjer JWT-a prije enkodiranja JSON-a[7]

[7] Izvor: https://coldbox-security.ortusbooks.com/jwt/jwt-services (rujan 2021.)

Rezultat su tri Base64Url enkodirana string-a odvojeni točkama koji se lako prenose u HTML okruženju (Sl. 3.5.).

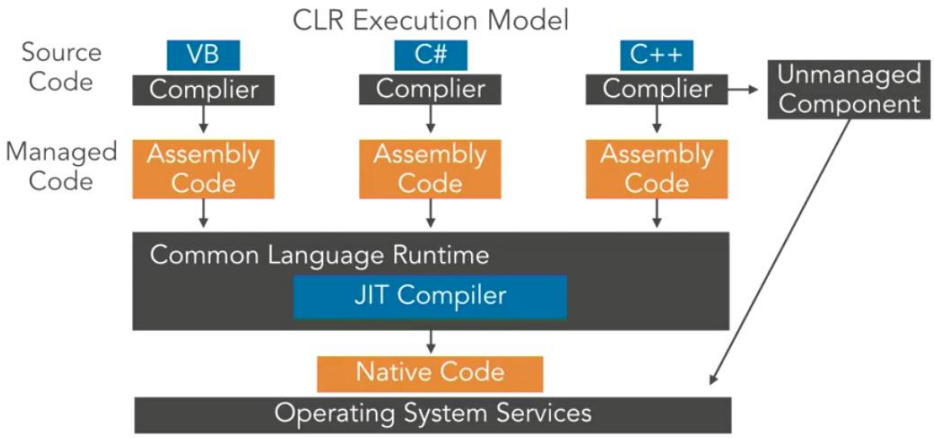


Slika 3.5. Primjer JWT-a

C# i .NET *framework*

Već ranije, u potpoglavlju 3.1. je spomenut i opisan *Web* API. Za realizaciju istoga u ovome projektu se koristio .NET *framework* uz programski jezik C#.

Prema [6], C# je objektno-orijentirani programski jezik. Definira tipove podataka i njihova ponašanja. Programi pisani u C# se prvo šalju u *Common Language Runtime* koji je zadužen za upravljanje kodom, memorijom, sigurnosnim granicama i aspektima te pretvorbu koda u strojni kod. Iz tog razloga se C# naziva *managed* kodom. Za razliku od primjerice C++ programskog jezika kojeg nazivaju *unamanaged* kodom jer sve te značajke programer mora ukomponirati samostalno. Krajnji program odnosno kod je već u suštini binaran i operacijski sustav ga može direktno staviti u memoriju i pokrenuti. Usporedba je prikazana na slici 3.6.



Slika 3.6. Usporedba Managed i Unmanaged komponenti[8]

[8] Izvor: https://imgur.com/uuWbVAb (rujan 2021.)

.NET je platforma sastavljena od alata, programskih jezika i biblioteka za izradu različitih vrsta aplikacija. Neke od najbitnijih značajki su:

- *Code Editor* - pisanje i uređivanje koda
- *Compiler* – pretvara kod koji je razumljiv čovjeku u oblik koji je razumljiv računalu
- *Debugger* – alat koji se koristi tijekom testiranja kako bi se otklonile pogrješke
- Hijerarhijski dijagram klasa – vizualizacija strukture objektno orijentiranog koda

Language-Integrated Query (LINQ)

Skup tehnologija koje integriraju mogućnosti *query-a* direktno u C#. *Query* se inače piše u obliku *string-a* koji zna biti poprilično nepregledan te iz tog razloga rezultira lakšim i češćim pogrješkama u pisanju istih. Primjer string *query-a*:

```
"SELECT SCORE FROM SCORES WHERE SCORE > 80"
```

Isti *query* napisan u LINQ-u:

```
var scoreQuery = from score in scores
                 where score > 80
                 select score;
```

Vidljivo je da LINQ pisanje *query-a* čini čitljivijim i manje sklonije pogrješkama. Povrh toga, LINQ automatski odrađuje pretvorbu vlastitog koda u objekte, listu objekata, SQL te XML, ovisno o tome kakva manipulacija i radnja nad podacima se želi obaviti.

Password-Based Key Derivation Function 2 (PBKDF2)

Algoritam koji primjenjuje pseudoslučajnu funkciju, primjerice HMAC-SHA1, na lozinku i njezinu sol te iterira cijeli proces mnogo puta kako bi se stvorio izvedeni ključ. Metoda se još naziva i rastezanjem ključa jer se tako dodaje dodatan procesorski napor kako bi se potvrdilo je li lozinka točna; to čini probijanje lozinke težim i dugotrajnijim procesom, pogotovo metodom uzastopnog pokušavanja.

```
IK = PBKDF2(PSF, Lozinka, Sol, i, ikDuljina)
```

Gdje vrijedi:

IK – izvedeni ključ

PSF – pseudoslučajna funkcija, primjerice HMAC-SHA1

i – broj željenih iteracija

ikDuljina – željena duljina bitova izvedenog ključa

Sol

Nasumičan podatak koji se miješa s lozinkom kako bi se probijanje lozinke otežalo.

HMAC (Hash-based Message Authentication Code)

Prema [7], HMAC je komad informacije kojim se provjerava integritet i autentičnost poruke.

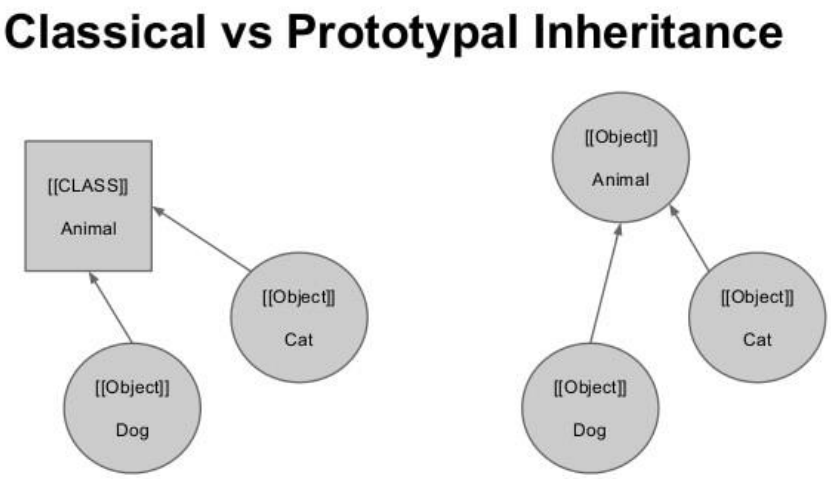
SHA-1

Hash funkcija koja stvara HMAC. *Hash funkcije* za razliku od *enkripcije* stvaraju izmiješan podatak koji je nepovratan, odnosno ne može se *dekriptirati*. Ulazni podaci su joj lozinka, tj. privatni ključ i sol .

Rezultat je izvedeni ključ koji se ne može *dekriptirati* i sprema se u bazu. Međutim, uz pomoć privatnog ključa moguće ga je rekreirati. Rekreirani izvedeni ključ se zatim uspoređuje s izvedenim ključem u bazi i potvrđuje autentičnost poruke, u našem slučaju lozinke. Tako ako se baza kompromitira lozinke će biti u obliku *hash-a* koji sam po sebi nema značenje.

JavaScript (JS)

Prema [8] JS je skriptni programski jezik koji se prevodi u stvarnom vremenu za razliku od programskih jezika koji se prevode prije samoga pokretanja. Najčešće se koristi za pisanje *front-end* dijela *web* stranica. Koristi paradigmu koja se temelji na prototipovima. Objektno orijentirano programiranje koje se temelji na klasama stvara hijerarhiju nekog objekta nasljeđivanjem klase. To znači da izvedena klasa mora definirati sva svojstva osnovne klase i ona su dio cjelokupne hijerarhije objekta. Kod OOP-a koje se temelji na prototipovima nema nasljeđivanja u klasičnom smislu, ali postoji poveznica između objekata. Tako svaka klasa predstavlja objekt sama po sebi i nije potrebno definirati svojstva baznih klasa, no ako se odluči dodati poveznica na neku baznu klasu, objekt će ga klonirati i samo nadodati svoja nova svojstva (Sl. 3.7.).



Slika 3.7. Usporedba OOP-a s klasama i prototipovima

React

Prema [9], React je JavaScript biblioteka za izradu korisničkog sučelja i njezinih komponenti. Održavana je od strane Facebook-a i zajednice programera. Iako nije potreban, često se za izradu HTML komponenti koristi JSX.

JSX (JavaScript XML)

Produžetak JS sintakse koja pomaže vizualizirati komponente korisničkog sučelja.

Primjer:

```
const element = <h1>Hello, world!</h1>;
```

HTML (HyperText Markup Language)

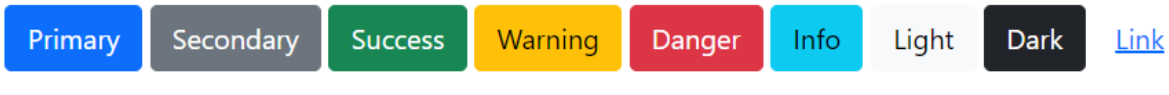
Opisni jezik koji definira kako će internetski preglednik prikazivati određene elemente. Potpomognut je CSS-om, a često i JavaScript-om koji u cijelu priču uvodi dinamičnost i *renderiranje* HTML-a u stvarnom vremenu.

CSS (Cascading Style Sheets)

HTML prikazuje sadržaj, dok CSS daje prezentaciju istome; ona uključuje izgled stranice, boje i fontove.

React Bootstrap

Framework koji sadrži unaprijed napisane i dizajnirane UI komponente. Kako bi postignuo isto, koristi CSS i React. Samo jedan on mnogih primjera je dugme, prikazano na slici 3.8.




```
<>
  <Button variant="primary">Primary</Button>{' '}
  <Button variant="secondary">Secondary</Button>{' '}
  <Button variant="success">Success</Button>{' '}
  <Button variant="warning">Warning</Button>{' '}
  <Button variant="danger">Danger</Button> <Button variant="info">Info</Button>{' '}
  <Button variant="light">Light</Button> <Button variant="dark">Dark</Button>{' '}
  <Button variant="link">Link</Button>
</>
```

Slika 3.8. React Bootstrap komponenta dugmeta

Važno je napomenuti da se, iako su komponente već unaprijed dizajnirane, mogu dodatno uređivati tzv. *override* css klasama.

Node.js

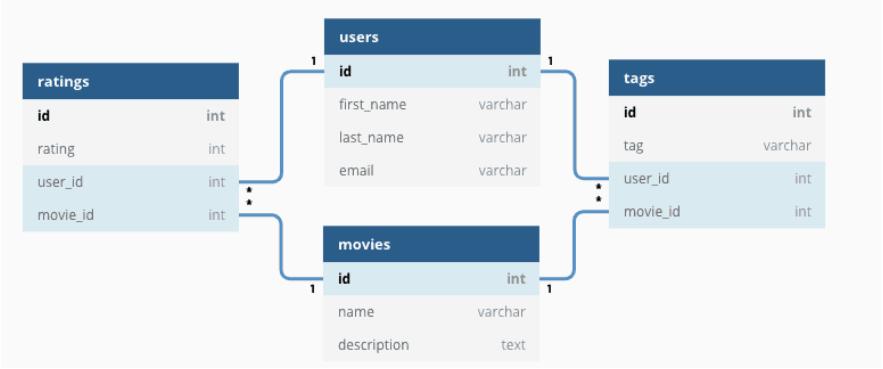
Open-source serversko okruženje koje omogućava pokretanje JavaScript koda na serveru. Uz Node.js se često koristi i njegov upravitelj paketima NPM (Node.js *package manager*) koji instalira sve ovisnosti potrebne aplikaciji da bi funkcionirala. Također ima *komandno* sučelje za dodavanje novih paketa, nadogradnju postojećih paketa ili pak brisanje istih.

3.3. Baza podataka

Baza podataka je spremište podataka u kojem su podaci organizirani tako da im je olakšan pristup, njihovo upravljanje i ažuriranje. U svrhu ovoga projekta se koristio *Microsoft SQL Server*, organizacijski sustav relacijske baze podataka. Glavni zadatak mu je spremati i dohvaćati podatke poštujući zahtjeve aplikacija koje ga koriste a nalaze se na istom računalu ili na drugoj mreži. Relacijska baza podataka zadržava sve podatke u tablicama koje su povezane relacijama.

Za upravljanje relacijskom bazom podataka MSSQL koristi se T-SQL (engl. *Structured Query Language*). Slanjem query-a moguće je iz baze podataka čitati podatke, dodavati nove tablice,

stupce, retke, mijenjati sadržaj tablica te brisati tablice i njezine elemente. Primjer relacijske baze podataka s tablicama i njenim relacijama prikazan je na slici 3.9.



Slika 3.9. Primjer relacijske baze podataka i njezinih tablica[9]

Sve moguće relacije i njezini opisi dani su tablicom 3.2.

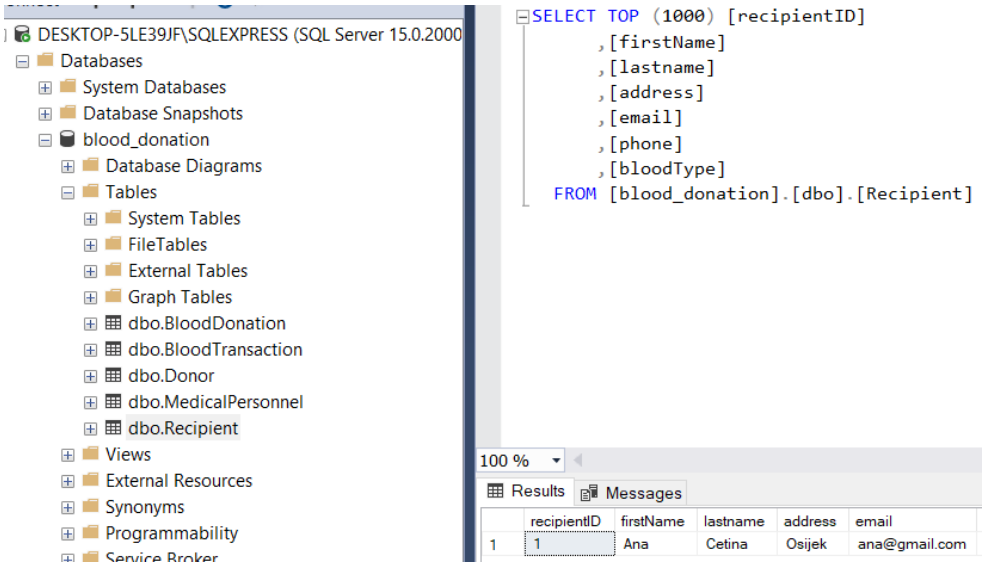
Tablica 3.2. Veze između tipova entiteta E1 i E2, prema [1, str. 25]

OZNAKA	OPIS
1:1	Jedan primjerak od E1 može biti povezan najviše s jednim primjerkom od E2. Također, jedan primjerak od E2 može biti povezan najviše s jednim primjerkom od E1.
1:M	Jedan primjerak od E1 može biti povezan s više primjeraka od E2. Istovremeno, jedan primjerak od E2 može biti povezan najviše s jednim primjerkom od E1
M:1	Jedan primjerak od E1 može biti povezan najviše s jednim primjerkom od E2. Istovremeno, jedan primjerak od E2 može biti povezan s više primjeraka od E1.
M:M	Jedan primjerak od E1 može biti povezan s više primjeraka od E2. Također, jedan primjerak od E2 može biti povezan s više primjeraka od E1.

Kako bi se *Microsoft SQL Server* uspostavio potrebno je koristiti *Microsoft SQL Server Management Studio* (SSMS), aplikaciju koja konfigurira i upravlja svim komponentama unutar

[9] Izvor: https://www.omnisci.com/technical-glossary/relational-database (rujan 2021.)

SQL servera. Glavna značajka ovoga programa je *Object Explorer* koji nudi pregled i označavanje svih objekata u bazi te dozvoljava radnje nad njima (Sl. 3.10.).



Slika 3.10. SQL Management Studio

3.4. Programsko rješenje

Aplikacija Sustav za praćenje darivanja krvi se sastoji od dva dijela: *back-enda* koji komunicira s bazom podataka, i *front-enda* koji šalje zahtjeve na back-end kako bi povratne informacije prikazao na *web* stranici, odnosno korisničkom sučelju.

Baza podataka

Za potrebe sustava stvorena je baza naziva blood_donation. Prije stvaranja baze potrebno je uspostaviti SQL server pomoću SSMS-a. Za daljnje potrebe stvaranja tablica i pisanje *query-a* se koristio *Object Explorer*.

Tablice se kreiraju koristeći sintaksu sa slike 3.11.

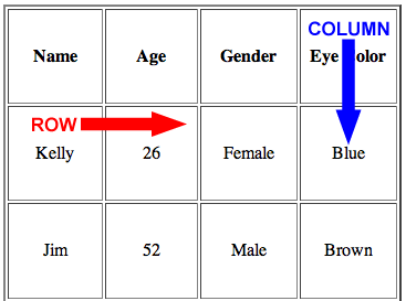
```
CREATE TABLE Blood_Donation(
    bloodID uniqueidentifier NOT NULL,
    donorID uniqueidentifier NOT NULL,
    dateDonated datetime NOT NULL,
    PRIMARY KEY (bloodID),
    FOREIGN KEY (donorID) REFERENCES Donor(donorID)
);
```

Slika 3.11. Primjer SQL query-a za kreiranje tablice

Gdje Blood_Donation predstavlja ime tablice. Svaka tablica se sastoji od redaka i stupaca, kao što je predočeno na slici 3.12.

Stupci tablice se definiraju na sljedeći način:

[ime stupca] [tip podatka] [smije li podatak biti prazan]



Slika 3.12. Predočenje stupaca i redaka tablice[10]

[10] Izvor: https://www.thoughtco.com/understanding-how-sql-databases-work-2693878 (rujan 2021.)

Primarni ključ je jedinstveni ključ svakog retka u tablici po kojemu se redci identificiraju i pretražuju.

Strani ključ služi kao referenca na redak iz druge tablice. Dani primjer sa slike 3.11. sadrži referencu na redak iz tablice Donor. Time je omogućena relacija M:1 jer jedan darivatelj može više puta darivati krv te srodno tome imati više darivanja.

Sustav za praćenje darivanja krvi se sastoji od sljedećih tablica:

Donor

Sadrži osnovne podatke o pojedincu darivatelju. Prikazivat će se na kontrolnoj ploči za svakoga korisnika nakon prijave u sustav (Sl. 3.13.).

MDUBINJAK.blood_...ation - dbo.Donor

Column Name	Data Type	Allow Nulls
donorID	uniqueidentifier	☐
firstName	varchar(50)	☐
lastname	varchar(50)	☐
gender	int	☐
address	varchar(60)	☐
email	varchar(100)	☐
phone	varchar(20)	☐
bloodType	varchar(3)	☐
age	int	☐
		☐

Slika 3.13. Tablica donora

BloodDonation – donacija krvi

Sadrži referencu na donora i datum darivanja krvi (Sl. 3.14.).

MDUBINJAK.blood_...bo.BloodDonation

Column Name	Data Type	Allow Nulls
bloodID	uniqueidentifier	☐
donorID	uniqueidentifier	☐
dateDonated	datetime	☐
		☐

Slika 3.14. Tablica donacije krvi

BloodTransaction – detalji o obavljenoj transakciji krvi

Sadrži referencu na redak iz tablice za donaciju krvi. Također sadrži reference na medicinsku osobu koje je obavila donaciju krvi (empID) i osobu ili ustanovu koja je primila krv darivatelja (recipientID). U *success* stupac se sprema podatak o uspješnosti donacije krvi; može biti *false*, odnosno laž, ili *true*, odnosno istina. Tablica je prikazana slikom 3.15.

MDUBINJAK.blood_....BloodTransaction

Column Name	Data Type	Allow Nulls
transactID	uniqueidentifier	☐
empID	uniqueidentifier	☐
dateOut	datetime	☐
quantity	int	☐
hemoglobin	int	☐
bloodPressure	varchar(50)	☐
notes	varchar(500)	☐
recipientID	uniqueidentifier	☐
bloodID	uniqueidentifier	☐
success	bit	☐

Slika 3.15. Tablica detalja transakcije krvi

MedicalPersonnel

Sadrži osnovne podatke o službenoj medicinskoj osobi koja je sprovela donaciju krvi (Sl. 3.16.).

MDUBINJAK.blood_....MedicalPersonnel

Column Name	Data Type	Allow Nulls
empID	uniqueidentifier	☐
firstName	varchar(50)	☐
lastname	varchar(50)	☐
address	varchar(60)	☐
email	varchar(100)	☐
phone	varchar(20)	☐

Slika 3.16. Tablica medicinskog osoblja

Recipient

Sadrži osnovne podatke o primatelju darovane krvi. Također sprema podatak o tome želi li primatelj ostati anoniman (Sl. 3.17.).

MDUBINJAK.blood_d...n - dbo.Recipient

Column Name	Data Type	Allow Nulls
recipientID	uniqueidentifier	☐
name	varchar(500)	☐
address	varchar(60)	☐
email	varchar(100)	☐
phone	varchar(20)	☐
bloodType	varchar(3)	☐
anon	int	☐

Slika 3.17. Tablica primatelja darovane krvi

BloodStock

Sadrži podatak o zalihama svake postojeće krvne grupe (Sl. 3.18.).

MDUBINJAK.blood_d...- dbo.BloodStock

Column Name	Data Type	Allow Nulls
zeroMinus	int	☐
zeroPlus	int	☐
aMinus	int	☐
aPlus	int	☐
bMinus	int	☐
bPlus	int	☐
abMinus	int	☐
abPlus	int	☐

Slika 3.18. Tablica zaliha krvi

LoginData

Sadrži korisničko ime svakoga korisnika, hash njegove lozinke i podatak o tome je li korisnik administrator ili darivatelj (Sl. 3.19).

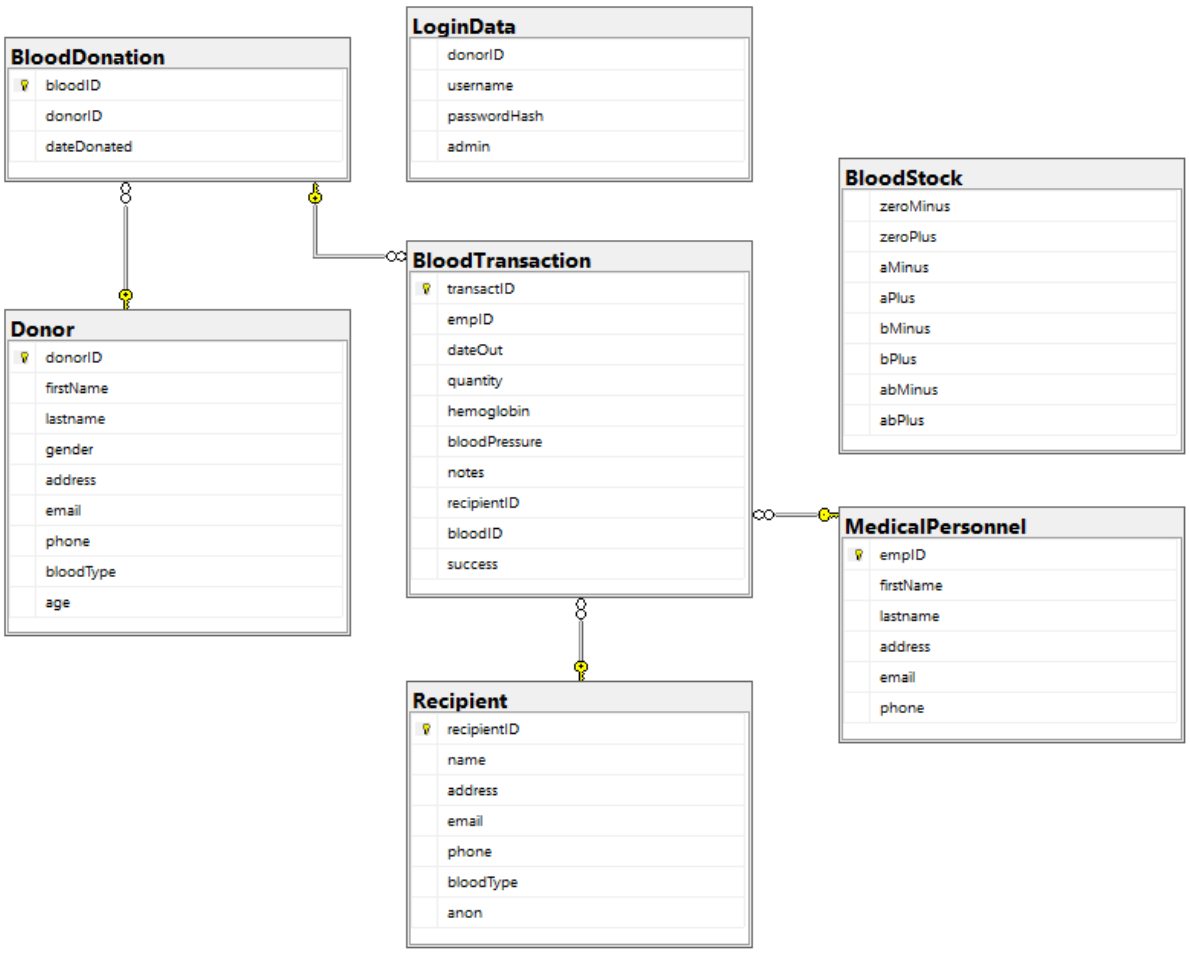
MDUBINJAK.blood_...n - dbo.LoginData

Column Name	Data Type	Allow Nulls
donorID	uniqueidentifier	☐
username	varchar(50)	☐
passwordHash	varchar(500)	☐
admin	bit	☐
		☐

Slika 3.19. Tablica za prijavu

Sve relacije su zamišljene kao 1:1 osim tablice za donaciju krvi koja ima M:1 odnos s Donor tablicom.

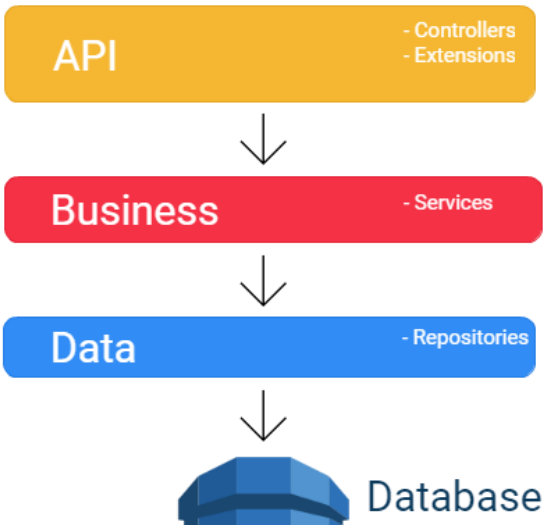
Krajnji izgled relacijski povezanih tablica baze podataka za sustav za praćenje darivanja krvi prikazan je na slici 3.20.



Slika 3.20. Relacijska baza podataka sustava

3.4.1. API

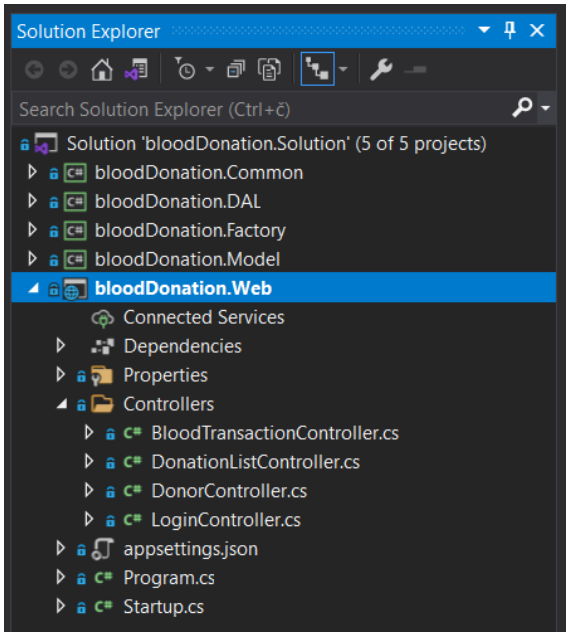
Struktura API-ja sustava za praćenje darivanja krvi se temelji na višeslojnoj arhitekturi. Podijeljena je u tri različita sloja. Tako se odgovornosti aplikacije razdvajaju u više odjeljaka što čini razvijanje i održavanje aplikacije lakšim. API sustava je podijeljen na tri odjeljka: API, *Bussiness* i *Data* (Sl. 3.21.).



Slika 3.21. Višeslojna arhitektura API-ja[11]

API (Web) sloj

API sloj primarno služi kao *endpoint* za HTTP zahtjeve. Tu dužnost endpointa obavljaju *kontroleri* kojima se pristupa s *front-enda* preko HTTP poruka. S obzirom na rutu i metodu HTTP poruke, poziva se odgovarajuća metoda kontrolera. *Kontroleri* sustava za praćenje darivanja krvi su prikazani na slici 3.22.



Slika 3.22. *Web*, odnosno API sloj

[11] Izvor: https://medium.com/@hamzaak/a-multi-layer-back-end-application-architecture-in-net-core-c08898f2427e (rujan 2021.)

Osim kontrolera, API sloj sadrži i dvije vrlo bitne klase: Program.cs koja pokreće aplikaciju, i Startup.cs gdje se *aplikacija* konfigurira. Iako je moguće ova dva zadatka smjestiti u istu klasu, koriste se dvije klase kako bi se ispoštovao SRP (*Single Responsibility Principle*). Prednost korištenja SRP-a je ta da klase postaju čitljivije i imaju samo jednu odgovornost o kojoj brinu. Klase koje su natrpane s više odgovornosti teže je ponovno koristiti jer lako dolazi do situacije gdje je drugoj klasi potreban samo jedan dio te klase što često vodi do potrebe za izmjenjivanjem. Stoga klase koje poštuju SRP nije potrebno toliko mijenjati, lakše ih je održavati i mogu se češće koristiti na drugim mjestima.

Osim SRP-a, kroz API se još pretežito primjenjuje i oblikovni obrazac *Dependency Injection* (DI) koji obavlja *instanciranje* objekata tek po potrebi te tako štedi računalne resurse. Korišten je Autofac paket kojemu je potrebno jedino predati module s registriranim klasama kako bi mogao podesiti kontejner da omogući DI za iste. To se radi na način prikazan slikom 3.23.

```
public void ConfigureContainer(ContainerBuilder builder)
{
    builder.RegisterModule(new DALModule());
    builder.RegisterModule(new FactoryModule());
}
```

Slika 3.23. Registracija modula za Autofac DI kontejner

Osim podešavanja DI kontejnera, u Startup.cs klasi potrebno je još dodati CORS i konfigurirati *endpoint* rute.

***Cross-Origin Resource Sharing* (CORS)**

Front-end i *back-end* se ne nalaze na istome serveru. HTTP sa svojim osnovnim postavkama ne dopušta slanje zahtjeva s jednog na drugi server. Stoga da bi se ta značajka omogućila se koristi CORS koji prije svakog HTTP zahtjeva provjerava prima li server zahtjev (Sl. 3.24.).

```
app.UseCors(builder => builder
    .AllowAnyOrigin()
    .AllowAnyMethod()
    .AllowAnyHeader());
```

Slika 3.24. Dodavanje CORS postavki

Konfiguriranje *endpoint* rute se odrađuje na način prikazanim na slici 3.25.

```
app.UseEndpoints(endpoints =>
{
    endpoints.MapControllerRoute(
        name: "default",
        pattern: "{controller=Home}/{action=Index}/{id?}");
```

Slika 3.25. Endpoint konfiguracija

Endpoint je u suštini lokacija metode nekog kontrolera. Ako primjerice želimo dobaviti podatke o darivatelju, s *front-enda* će se preko TCP veze poslati HTTP poruka na GET metodu *kontrolera*. Metode u *kontroleru* će raspoznati preko atributa koji se dodjeljuju svakoj metodi u kontroleru (Sl. 3.26.).

```
[HttpGet("{id}")]
0 references
public async Task<IActionResult> GetDonor(Guid id, [FromHeader] string token)
```

Slika 3.26. HttpGet atribut

Znajući sve ovo, ovako će izgledati URI koji vodi do ove metode:

http://localhost:4200/api/donor/id_donora

Ako u istom *kontroleru* postoji više GET metoda potrebno je definirati i *akciju*, odnosno ime metode iz kontrolera kako bi ih se moglo razlikovati (Sl. 3.25.):

http://localhost:4200/api/donor/GetDonor/id_donora

Na slici 3.26. u parametrima koji se predaju metodi može se vidjeti [FromHeader] atribut. Tako metoda zna gdje u HTTP poruci tražiti određene podatke. Iz toga se da zaključiti da će metoda GetDonor vrijednost *tokena* tražiti u zaglavlju HTTP poruke. Osim FromHeader-a postoje i atributi FromUri koji vrijednosti vuku iz samog URI-ja te FromBody koji podatke traži u tijelu HTTP poruke. U tijelu se šalje JSON kojeg je moguće pretvoriti u C# objekt *mapiranjem* ime/vrijednost stavki iz JSON-a u istoimene *varijable* zadanog objekta.

Dakle, uzimajući sve to u obzir se može zaključiti da glavna svrha kontrolera leži upravo u zadaći da primaju HTTP poruke od klijenta i prosljeđuju podatke sljedećim slojevima na obradu. Nakon obrade podaci se istim putem vraćaju na kontrolere koji rezultat obrade šalju nazad klijentu koji je prvotno i poslao HTTP poruku. Uzmimo za primjer *kontroler* za donora i njegovu GET metodu (Sl. 3.27). Preko HTTP poruke ova metoda dobiva korisničko ime donora i *token* kojim se provjerava je li korisnik prijavljen u sustav. Oba zadatka, validaciju *tokena* i dohvaćanje donora po njegovom korisničkom imenu će se delegirati na daljnje slojeve. Ako validacija prođe, *instancira* se novi model kojemu se prije vraćanja klijentu dodjeljuju vrijednosti vraćene od GetDonorByUsername metode; u protivnom se klijentu vraća prazan model.

```
public async Task<IActionResult> GetDonorByUsername(string username, [FromHeader] string token)
{
    try
    {
        var validationResult = JWTAuth.ValidateCurrentToken(token);

        var claim = String.Empty;
        var modelDto = new DonorModelDto();

        if (!validationResult) return Ok(modelDto);

        claim = JWTAuth.GetClaim(token, "UserRole");

        if (claim.Equals("Admin"))
        {
            var model = await _donorFactory.GetDonorByUsername(username);

            modelDto = new DonorModelDto()
            {
                DonorID = model.DonorID,
                FirstName = model.FirstName,
                LastName = model.LastName,
                Address = model.Address,
                Email = model.Email,
                Phone = model.Phone,
                BloodType = model.BloodType,
                Gender = model.Gender.ToString(),
                Age = model.Age
            };
        }

        return Ok(modelDto);
```

Slika 3.27. Primjer GET metode donor kontrolera

Modeli koji se prave u *kontroleru* su drukčiji od modela za bazu podataka. Razlog tome je upravo obrada podataka koja se odvija između. Moguć scenarij bi bio da se neki podaci iz baze žele izostaviti i ne slati klijentu nazad ili se određeni podaci ne žele spremati u bazu nego se koriste isključivo pri obradi.

Data (DAL) sloj

Zadnji od tri sloja višeslojne arhitekture sustava za praćenje darivanja krvi. Svaka tablica iz baze podataka ima svoj repozitorij koji upravlja osnovnim CRUD operacijama te tablice. Primjer jedne metode iz repozitorija prikazan je na slici 3.28.

```
public async Task<bool> EditBloodDonation(BloodDonationModel model)
{
    var success = false;
    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        string queryString = @"Update BloodDonation set donorID = @donorID,
                                                dateDonated = @dateDonated
                                                where bloodID = @id;";

        SqlCommand command = new SqlCommand(queryString, connection);
        command.Parameters.AddWithValue("@dateDonated", model.DateDonated);
        command.Parameters.AddWithValue("@donorID", model.DonorID);
        command.Parameters.AddWithValue("@id", model.BloodID);

        connection.Open();

        if ((await command.ExecuteNonQueryAsync()) > 0) success = true;
    }
    return success;
```

Slika 3.28. Metoda za uređivanje tablice za donaciju krvi

C# prije svega uspostavlja vezu sa sql bazom. Za to su mu potrebni podaci o serveru i imenu baze, te korisničko ime i lozinka kojima se spaja na bazu:

```
connectionString =
@"Server=mdubinjak;Database=blood_donation;Trusted_Connection=True;
```

Sql naredba tek tada može izvršiti napisani *query* koristeći metodu ExecuteNonQueryAsync koja vraća istinu ili laž kao indikator uspjeha odnosno neuspjeha (Sl. 3.28).

Za čitanje podataka iz baze koristi se SqlReader koji dohvaća sve retke koji odgovaraju zadanom *query-u*. Čitač onda sadrži listu redaka koju je moguće iterirati koristeći petlju i pristupiti podacima iste.

Business (Factory) sloj

Sloj koji obrađuje podatke prije nego ih pošalje u sloj koji rukovodi bazom ili *kontroleru*. Ima pristup svakom *repozitoriju* i stoga može pozivati metode svakoga. Sustav za praćenje darivanja krvi prikazuje sve podatke o donaciji krvi na jednom mjestu. Iz tog razloga *kontroleru* je dovoljan jedan model sa svim potrebnim podacima kojeg će on slati nazad klijentu. Međutim kako je baza relacijski povezana, nije moguće doći do podataka o primjerice medicinskom osoblju i primatelju darovane krvi prije nego smo došli do podataka o samoj transakciji krvi, kao što je prikazano na slici 3.29.

```
var allBloodDonations = await _bloodDonationDAL.GetBloodDonationsAsync(id);

foreach(var x in allBloodDonations)
{
    var bloodTransaction = await _bloodTransactionDAL.GetBloodTransaction(x.BloodID);
    var recipient = await _recipientDAL.GetRecipient(bloodTransaction.RecipientID);
    var personnel = await _medicalPersonnelDAL.GetMedicalPersonnel(bloodTransaction.EmpID);

    donations.Donations.Add(
        new DonationDto()
        {
            DateDonated = x.DateDonated.ToString(),
            Quantity = bloodTransaction.Quantity,
            Hemoglobin = bloodTransaction.Hemoglobin,
            BloodPressure = bloodTransaction.BloodPressure,
            Notes = bloodTransaction.Notes,
            Success = bloodTransaction.Success,
            DateOut = bloodTransaction.DateOut.ToString(),
            PersonnelName = $"{personnel.FirstName} {personnel.LastName}",
            PersonnelWorkPhone = personnel.Phone,
            RecipientName = $"{recipient.Name}",
            RecipientBloodType = recipient.BloodType,
            Anon = recipient.Anon
        }
    );
}

var listCount = donations.Donations.Count;
var skipCount = pageSize * (page - 1);

donations.ListCount = listCount;
donations.SuccessCount = donations.Donations.Where(x => x.Success == true).ToList().Count;
donations.Donations = donations.Donations.OrderByDescending(x => x.DateDonated).ToList();
donations.Donations = donations.Donations.Skip(skipCount).Take(pageSize).ToList();
return donations;
```

Slika 3.29. Metoda koja obavlja cijelu logiku i priprema podatke spremne za slanje klijentu

Cijela lista se naposljetku koristeći LINQ *sortira* po datumu, najbliže prema najstarijem. Osim *sortiranja* na listu je primijenjen i *pagination*, metoda kojom se lista secira i vraća točno određen broj objekata. Algoritam za sljedeće bi bio:

Lista.Skip (broj stranice * veličina stranice).Take (veličina stranice)

Ako lista sadrži sveukupno sto objekata i po stranici se želi prikazati samo deset objekata, postojat će deset stranica. Stoga se metodi još predaje i broj stranice kako bi znala koje objekte vratiti. Tako će primjerice za broj stranice tri, vratiti objekte s indeksima 20-29 iz cjelokupne liste.

Validacija korisnika i prijava u sustav

Kod prijave u sustav poziva se metoda koja pristupa bazi i dohvaća *hash* lozinke određenog korisnika. U isto vrijeme se generira *hash* lozinke koja je predana u sustav PBKDF2 algoritmom. Ako se *hash-ovi* poklapaju generira se JWT (Sl. 3.30):

```
var testHash = GetPbkdf2Bytes(password, salt, iterations, hash.Length);

if (SlowEquals(hash, testHash))
{
    return (donor.Admin ? JWTAuth.GenerateToken("Admin") : JWTAuth.GenerateToken("User"), donor.DonorID, donor.Admin);
}
```

Slika 3.30. Generiranje tokena za admina ili donora

Ako je prijavljeni korisnik administrator *claim* će biti *Admin*, u protivnom *User*. *Administratori* imaju pristup većem broju *kontrolera* i metoda: za dodavanje podataka, uređivanje i brisanje. Donori imaju pristup isključivo GET metodama preko kojih im se vraćaju njihovi osnovni podaci te podaci o donacijama krvi.

3.4.2. Korisničko sučelje sustava

Front-end dio sustava je u cijelosti izrađen koristeći React i bootstrap komponente. Metode su asinkrone kao i na *back-end* strani jer je potrebno određeno vrijeme da bi se odgovor vratio. Ako se to čekanje iz nekog razloga produži aplikacija će blokirati dok ne dobije odgovor nazad. Koristeći asinkronost ostatak aplikacije može raditi neometano a podaci vezani za nju će se prikazati tek kada se odgovor vrati (Sl. 3.31.).

```
async fetchDonationList(page) {
    let donorID = sessionStorage.getItem('donorID');
    await fetch(`https://localhost:44336/api/donationlist?page=${page}&pageSize=${3}&id=${donorID}`,
        {
            method: 'GET',
            headers: {
                "Access-Control-Allow-Origin": "*",
                "token": sessionStorage.getItem('loginToken')
            }
        })
        .then(async res => await res.json())
        .then(json => {
            this.setState({
                isLoaded: true,
                currentPage: json
            });
        });
```

Slika 3.31. GET metoda za listu donacija

Fetch metodi se predaje URI koji vodi do metode iz *kontrolera* za donacijsku listu. Vrijednosti parametara se šalju preko URI-ja, dok se kod POST metoda mogu slati i preko tijela HTTP poruke. Bitno je naznačiti koja HTTP metoda se koristi i zaglavlju predati *token* kojim će se validirati korisnik koji šalje zahtjev te provjeriti ima li ovlasti za korištenje određenih metoda s *back-enda*. Sve podatke koje želimo prikazati ili zadržati u kodu zbog dodatne manipulacije nad njima potrebno je spremati u takozvane *state* varijable.

```
return (
    <div className="box">
        <h2>Sustav za praćenje doniranja krvi</h2>
        <div class="container" id="container">
            <div class="form-container sign-in-container">
                <form onSubmit={handleSubmit}>
                    <h1>Prijava</h1>
                    <input onChange={e => setUserName(e.target.value)} type="username" placeholder="Username" />
                    <input onChange={e => setPassword(e.target.value)} type="password" placeholder="Password" />
                    <button type="submit">Prijavi se</button>
                </form>
```

Slika 4.22. Primjer JSX koda za prijavu

Na slici 3.32. *state* varijable se postavljaju metodama setUsername i setPassword. Pritiskom dugmeta za prijavu okida se *fetch* metoda koja navedene podatke šalje na *back-end* server gdje će se provjeriti jesu li uneseni podaci točni. Ako jesu generirat će se token koji će se kao odgovor vratiti *front-endu* koji će taj token spremiti u *sesiju* internet preglednika pod imenom loginToken sljedećom naredbom:

```
sessionStorage.setItem('loginToken', donorData.Token !== 'null' ? donorData.Token : ' ');
```

Tako spremljen *token* bit će u sesiji preglednika sve dok se preglednik ne zatvori ili ručno ne isprazni. U tom slučaju prijava će biti ponovno zatražena.

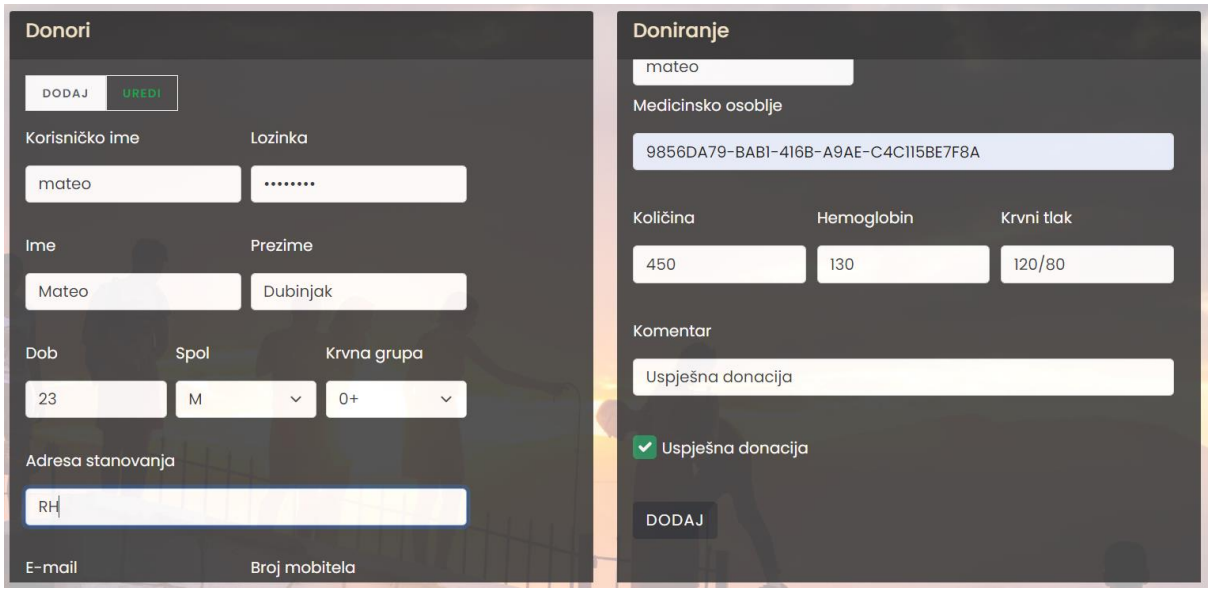
4. DEMONSTRACIJA I ISPITIVANJE FUNKCIONALNOSTI

Stranica za prijavu u sustav. Ovdje se prijavljuju i administratori i darivatelji. S obzirom na *claim* korisnika postoje dvije stranice na koje ga prijava može odvesti; to su admin stranica te stranica za darivatelja, odnosno sustav za praćenje darivanja krvi (Sl. 4.1.).

Slika 4.1. Stranica za prijavu korisnika

Admin stranica

Administrator sustava za praćenje darivanja krvi ima ovlasti dodavati, uređivati i brisati podatke iz baze. Da bi imao pristup metodama na *back-endu* koje vrše manipulaciju podacima potrebno je generirati JWT koristeći *admin claim*. U bazu podataka se sprema podatak o tome je li korisnik administrator. Ako je ta stavka istinita kod je podešen tako da generira JWT koristeći taj *claim*.



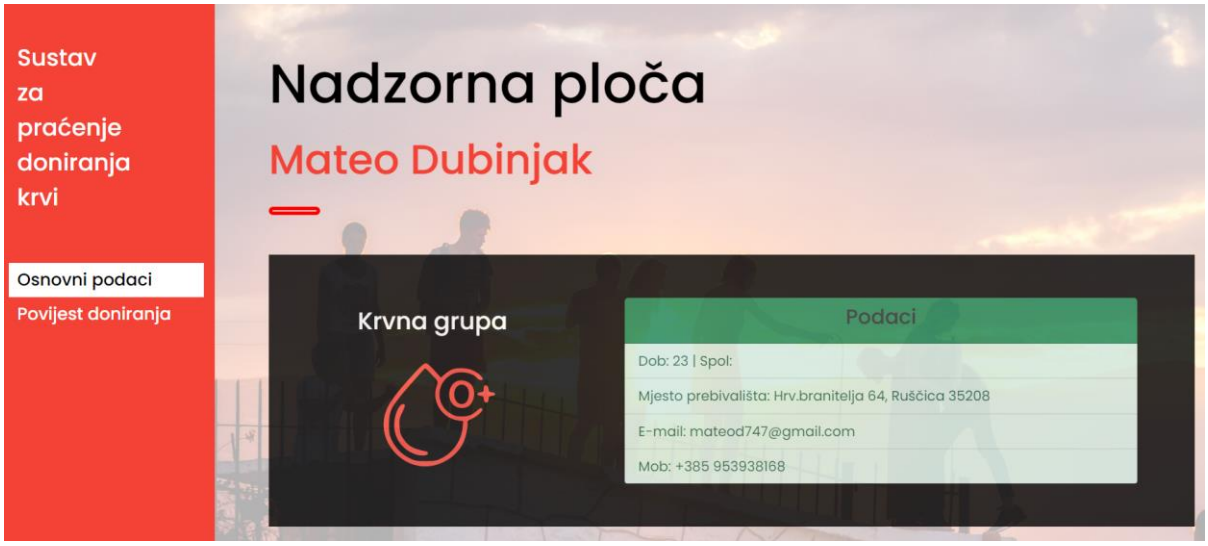
Slika 4.2. Izgled admin stranice

Na slici 4.2. prikazan je *admin panel*. Lijevi *panel* služi za dodavanje, uređivanje i brisanje donora iz baze, dok se desni koristi za sve ostale podatke vezane uz samu donaciju krvi, što uključuje i medicinsko osoblje i primatelje.

Sustav za praćenje darivanja krvi

Sustav ima dvije stranice kojima korisnik može pristupiti:

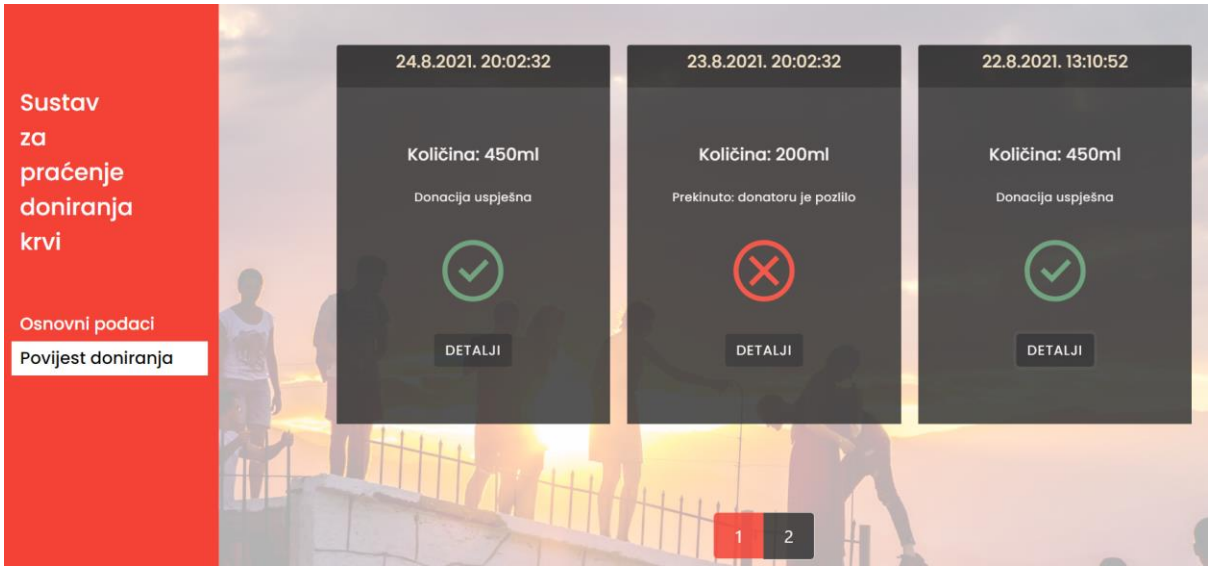
Osnovni podaci – stranica na kojoj svaki darivatelj može vidjeti svoje osnovne podatke, uključujući krvnu grupu. Izgled nadzorne ploče prikazan je na slici 4.3.



Slika 4.3. Nadzorna ploča sustava

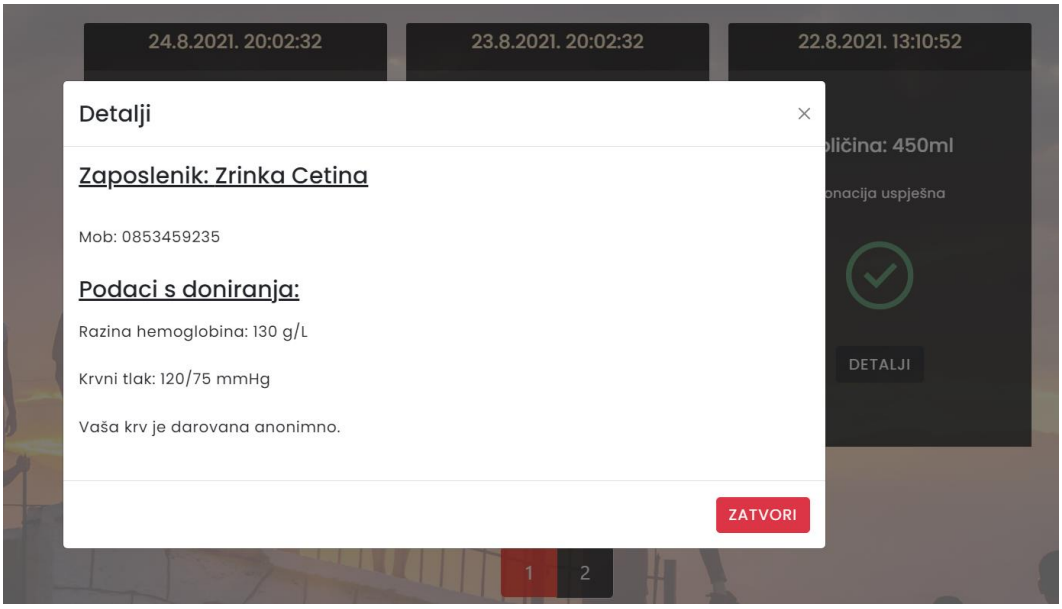
Prelaskom strelice miša preko ikone krvne grupe prikazuje se zaliha krvi. Znajući taj podatak, darivatelj može zaključiti je li potrebno darivati krv.

Druga raspoloživa stranica prikazuje povijest svih donacija krvi (Sl. 4.4).



Slika 4.4. Povijest donacija krvi

Uspješna donacija krvi označena je kvačicom, neuspješna simbolom iksa. Svaka kartica predstavlja jedno darivanje krvi i na njoj se nalaze osnovni podaci donacije krvi poput datuma darivanja i uzete količine krvi. Svaka kartica također sadrži dugme za detalje koje otvara iskočni prozor s detaljima o ciljanoj donaciji krvi (Sl. 4.5.).



Slika 4.5. Iskočni prozor s detaljima

5. ZAKLJUČAK

Aplikacija je koncipirana tako da darivateljima prikazuje detalje s prijašnjih darivanja krvi. Podijeljena je u dva dijela; jedan prikazuje informacije i zove se *front-end*, dok drugi dio naziva *back-end* komunicira s bazom i obavlja manipulaciju podataka koja će se kasnije prikazati krajnjem korisniku. Iako je moguće cijelu logiku API-ja odraditi na *front-endu*, odvajanje *back-end* dijela odnosno API-ja na drugi server je u današnje vrijeme skoro obvezno. Ista se *aplikacija* koristi na više uređaja, poput računala i mobitela, i svaki od njih traži specifičnu prilagodbu koda kako bi *aplikacija* pravilno funkcionirala. Odvajanjem *back-enda* u API manipulacija podataka se enkapsulira na jednom mjestu i tako povećava modularnost aplikacije. Izmjene na jednoj strani neće nužno zahtijevati izmjene na drugoj i mogu se skalirati neovisno jedna o drugoj. U Republici Hrvatskoj povijest darivanja krvi se bilježi ali detalji ostaju u internom sustavu crvenog križa. Darivatelji dobivaju knjižicu s popisom prošlih darivanja, međutim uvida u detalje nema. Knjižice su u papirnatom obliku što često vodi gubljenju ili lomljenju iste. Sustav za praćenje darivanja krvi rješava sve navedene probleme vezanu uz evidenciju darivanja krvi. Međutim, mišljenja sam da registraciju novih i postojećih darivatelja treba obavljati administrator jer se na taj način najbrže i najefikasnije mogu validirati osnovni podaci darivatelja. Iako sustav za praćenje darivanja krvi pruža dosta detalja vezanih uz svako darivanje krvi moguće je dodatno razviti *aplikaciju* kako bi uključila i prikaz zahvalnica i pogodnosti koje su darivatelji primili. Također je moguće sustav učiniti informativnijim i napraviti sustav koji će omogućiti dogovaranje termina za buduća darivanja krvi te slati bitne obavijesti darivateljima.

U Republici Hrvatskoj donacija krvi je anonimna što znači da podaci o primatelju darovane krvi ostaju anonimni. Treba li taj podatak ostati anoniman je debata o kojoj se i dalje raspravlja, no ovaj sustav je dizajniran tako da je te podatke moguće prikazati. Ako bi obje osobe uključene u cijeli proces, darivatelj i primatelj krvi dale privolu koja dopušta pokazivanje tih podataka sustav će darivatelju pokazati kada je krv darovana i kome. Osim toga postoji i anonimna opcija koja ako je uključena prikazuje jedino podatak o tome da je krv darovana, ali anonimno. Tako darivatelj ima uvid u detalje svih svojih donacija što povećava angažiranost i osjećaj samoispunjenja kod pojedinaca.

LITERATURA

[1] R. Manger, Osnove projektiranja baza podataka – https://www.srce.unizg.hr/files/srce/docs/edu/osnovni-tecajevi/d310_polaznik.pdf, Sveučilište u Zagrebu

[2] Mozilla Developer Network, https://developer.mozilla.org/en-US/docs/Web/HTTP [rujan 2021.]

[3] Mozilla Developer Network, https://developer.mozilla.org/en-US/docs/Glossary/TCP [rujan 2021.]

[4] Ecma International, https://www.json.org/json-en.html [rujan 2021.]

[5] Internet Engineering Task Force, https://jwt.io/introduction [rujan 2021.]

[6] Microsoft, 2021, https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/ [rujan 2021.]

[7] Network Working Group, 1997, https://datatracker.ietf.org/doc/html/rfc2104 [rujan 2021.]

[8] Mozilla Developer Network, https://developer.mozilla.org/en-US/docs/Web/JavaScript [rujan 2021.]

[9] Facebook, https://reactjs.org/tutorial/tutorial.html [rujan 2021.]

SAŽETAK

Aplikacija je koncipirana tako da darivateljima krvi pruži što informativniji i pregledniji prikaz prijašnjih darivanja krvi. Rješava problem nedostatka uvida u detalje transakcija krvi te praćenje iste. Darivateljima se prikazuju osnovni podaci sa svih darivanja krvi: uspješnost darivanja, razlozi odbijanja, razina hemoglobina, tlak te zdravstveno stanje krvi. Aplikacija nudi pregled svih obavljenih darivanja krvi, bile one uspješne ili neuspješne. Korisnik darivatelj ima pristup sustavu za praćenje darivanja krvi, a ako je korisnik administrator prikazuje mu se admin panel gdje ima pristup i ovlasti manipulirati podacima sustava za praćenje darivanja krvi, unositi nove podatke u bazu, uređivati ih ili brisati iz baze. Administrator također dodaje nove darivatelje i medicinsko osoblje u bazu.

Ključne riječi: blood donation, blood tracking, API, .NET

ABSTRACT

Title: Blood donation tracking system

Application is designed to provide more informative and neater look at the past blood donations to all blood donators. It solves the problem with missing insight into the details of blood transactions and its tracking: success of the blood transaction, failed transaction reasons, hemoglobin level, blood pressure and health of the donator's blood. Application provides overview of every blood donation, whether they were successful or not. User blood donator has access to the blood donation tracking system while administrators have access to the admin panel which they can and alse have permission to use and manipulate data related to blood donations, add new data into the database, edit the data and even delete it. Administrator also registers new blood donators and medical personnel.

Key words: blood donation, blood tracking, API, .NET

ŽIVOTOPIS

Mateo Dubinjak je rođen 22.09.1997. u Slavonskom Brodu. Nakon završene osnovne škole upisuje Tehničku Školu Slavonski Brod. Sudjeluje u Erasmus programu pohađajući stručnu praksu u Bray-u, Republika Irska. Po završetku srednje škole upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija u Osijeku, smjer Računarstvo. Nakon uspješnog završetka preddiplomskog studija nastavlja školovanje i upisuje diplomski studij računarstva, smjer Informacijske i podatkovne znanosti na istom fakultetu. Odradio je studentsku praksu u tvrtki Mono d.o.o. gdje se nakon stručne prakse i zapošljava kao full-stack programer.

_____

PRILOZI

CD priložen uz diplomski rad sadrži:

DiplomskiRadMateoDubinjak.pdf

DiplomskiRadMateoDubinjak.docx

BloodDonationApp.zip