4. Arhitektura i dizajn sustava

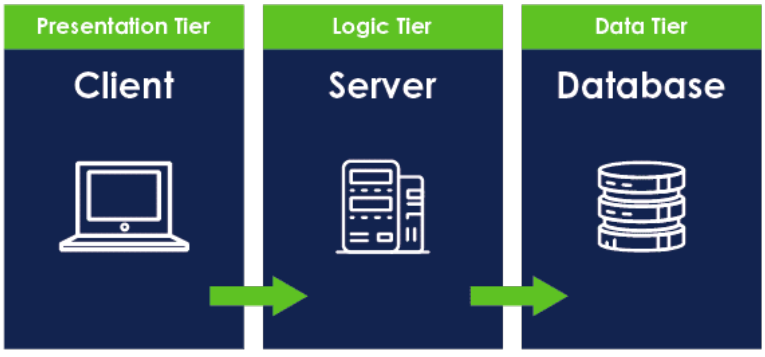
• Proces odabira arhitekture

U našem je zadatku cilj web aplikacije olakšati darivateljima krvi te Crvenom križu cijeli proces darivanja krvi: od organiziranja akcija, do pronalaženja istih i rezerviranja termina. Specifikacijom dionika i definirianjem funkcionalnih zahtjeva zaključili smo kako naša aplikacija mora imati tri razine: razinu baze podataka, razinu web-aplikacije te razinu klijenta. Tako navedenim razinama korespondiraju tri sloja: sloj pristupa podatcima, sloj aplikacijske logike te sloj korisničkog sučelja. Razmatrajući potrebna svojstva, poput smanjivanja nepotrebnih međuovisnosti između modula radi jednostavnosti pri budućim promjenama, hijerarhijske organizacije razina koja omogućava pristup viših razinama nižima preko programskog sučelja te grupiranje dijelova programa koji pristupaju i mijenjaju iste podatke i procedura koje se izvode slijedno radi lakšeg snalaženja, odabrali smo višeslojnu arhitekturu temeljenu na klijent-poslužitelj odnosu, najsličniju MVC stilu arhitekture.

• Višeslojna arhitektura i MVC stil

Odnos klijent-poslužitelj opisuje međusobni odnos programa koji sudjeluju u aplikaciji, to jest njihovo funkcioniranje prije, tijekom i po završetku komunikacije klijenta i poslužitelja. Aplikacija sadrži tri osnovna sloja:

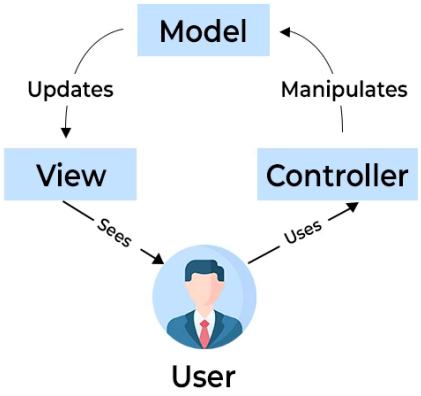
1. Klijentsku komponentu - sloj s korisničkim sučeljem, predstavljena web preglednikom, služi kao sučelje između klijenta i poslužitelja, omogućava klijentu slanje HTTP zahtjeva i primanje HTTP odgovora, prevoditelj koji web stranicu pisanu u kodu prikazuje u klijentu razumljivom obliku
2. Poslužiteljsku komponentu - sloj aplikacijske logike, predstavlja web poslužitelj, centar za razmjenu informacija i pružanje usluga, služi za pohranu, obradu i dostavu web stranica klijentu, pokreće web aplikaciju i prosljeđuje joj klijentove zahtjeve, po potrebi pristupa bazi podataka te vraća klijentu odgovor vidljiv u web pregledniku
3. Baza podataka - sloj za pohranu podataka



Prednosti klijent-server arhitekture je mogućnost odvojenog oblikovanja slojeva, što također omogućava “separation of concerns”, to jest svaki sloj brine samo za svoju zadaću i ne brine o funkcionalnostima drugih slojeva, a njihova se međuovisnost ostvaruje komunikacijom pomoću sučelja.

Ovako razrađenoj arhitekturi, najsličniji je Model-View-Controler stil. Osnovna je karakteristika ovog stila razdvajanje briga nezavisnim razvojem pojedinih dijelova aplikacije, što omogućuje jednostavnije i učinkovitije testiranje i izmjene dijelova sustava radi njihove dorade te skalabilnost i fleksibilnost cijele aplikacije razvijene na ovakvom principu. Korisničko sučelje je odvojeno od ostatka sustava, a povezanost elemenata se ostvaruje kroz tri sloja: sloja “View” na klijentskoj strani, te slojeva “Model” i “Controller” na poslužiteljskoj strani.

1. Model (model) - predstavlja podatke i logiku vezanu za njih, služi za oblikovanje podataka i prilagodbu komunikaciji s bazom
2. View (pogled) - komunikacija s korisnikom i prikaz podataka koje je dobio od upravitelja
3. Controller (upravitelj) - povezuje modele i poglede, na temelju primljenih korisničkih zahtjeva i podataka izvršava potrebne funkcije i formira odgovore



Web aplikacija je podijeljena na front-end i back-end.
Front-end čini prezentacijski dio aplikacije, to jest ono što korisnik vidi u web pregledniku. Za izradu aplikacije na klijentskoj strani smo odabrali radni okvir React, koji je zasnovan na JavaScriptu te pruža učinkovit razvoj korisničkog sučelja koristeći već gotove web komponente.

Back-end čini dio sustava u kojem se obrađuju korisnički zahtjevi i vrše potrebne akcije, to jest poslužiteljska strana klijent-server arhitekture. Za njegovu smo izradu odabrali programski jezik Python te radni okvir FastAPI.
I za razvoj klijentske i za razvoj poslužiteljske strane odabrali smo Visual Studio Code kao radno okruženje.