SVEUČILIŠTE U SPLITU

FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

**WEB SUSTAV ZA AUTOMATSKU DODJELU MENTORA NA ZAVRŠNIM I DIPLOMSKIM RADOVIMA.**

KRISTIJAN GLUMAC

# Uvod

Trenutno se na Fakultetu elektrotehnike, strojarstva i brodogradnje Split pri dodjeli mentora za završne i diplomske radove koristi već postojeći sustav koji na temelju različitih parametara određuje točno jednog mentora svakom studentu. Studenti koji završavaju treću odnosno petu godinu studija, dužni su obraniti završni odnosno diplomski rad prije stjecanja željene diplome.

Sadašnji sustav zahtjeva od svakog studenta da odabere minimalno pet, a najviše deset profesora kao potencijalne mentore. Studentu su na izbor ponuđeni isključivo oni profesori koji su bili nositelji kolegija na njegovom studijskom smjeru. Nakon odabira ponuđenih mentora, njegov zapis se pohranjuje u bazu podataka. Završetkom roka prijave za odabir mentora, sustav na temelju vlastitog algoritma, obavi dodjelu mentora pojedinom studentu. Kao što je već rečeno, svaki student može imati samo jednoga mentora, a svaki mentor može imati više studenata. Također, profesor može imati konačan broj studenata, a taj broj također o više različitih parametara.

Nakon uspješne dodjele mentora, svakom studentu se dodjeljuje jedan od odabranih mentora. Tu u principu prestaje rad sustava, idući korak je interni dogovor studenta i mentora putem email, sastanka i sl.

Cilj ovog završnog rada je napraviti novi sustav za dodjelu mentora na FESB-u koristeći pritom nove tehnologije, unaprijeđene algoritme, prihvatljiviji UI/UX za korisnika, nova razvojna okruženja, itd.

Sustav će se dijeliti na na *Frontend* i *Backend* dio. *Backend* se sastoji od baze podataka, realizirana pomoću *SQL Server 2019*, kao IDE se koristio *Visual Studia 2019 Enterprise Edition*. Koristili su se dodatni razvojni alati kao što su *Entity Framework Core, AutoMapper, ASP.NET Core, ServiceCollections,* itd. Vizualni dio sustava, odnosno *Frontend* se sastoji od standardnih tehnologija poput *HTML-a, CSS-a, Javascripta* te nekih dodatnih biblioteka i alata kao što su *jQuery library, AJAX-a* i *CSHTML-a.* Detaljni opis korištenih tehnologija i njihovih funkcija biti će objašnjene u nastavku ovoga rada.

Princip rada sustava biti će sličan postojećem, ali biti će bolji utoliko što će za izbor dodjele mentora uzimati u obzir neke dodatne parametre te neće *a priori* polaziti od pretpostavke da svaki student mora dobiti jednog od mentora sa njegove lista odabira mentora. U slučaju da student ne dobije nijednog od željenih mentora, njegovu situaciju će nadalje rješavati administrator. Sustav će imati tri vrste korisnika, student, profesor (mentor) te administrator. Svaki od korisnika ima svoju ulogu unutar sustava.

Administrator u određenom trenutku, poštujući pravilnike i statute fakulteta, pokreće sustav koji zatim započinje dodjelu. Nakon završetka obrade, studentu se određuje ili jedan od željenih mentora ili, ukoliko nije uspio dobiti nikoga sa liste željenih mentora, odluka o dodjeli mentora prepušta se administratoru koji na temelju liste slobodnih profesora sa studentovog smjera, ručno spaja studente i profesore/mentore. Prilikom spajanja, administrator uzima u obzir želje studenta i na temelju područja rada profesora, dodjeljuje studentu mentora.

Važno je napomenuti da se unutar sustava trenutno koriste *dummy* (lažni) podaci odnosno svi zapisi unutar baze podataka koje sustav koristi su slučajno-generirani podaci.

# Potreba za sustavom

## 2.1 Opis sustava

Kao što je već navedeno u Uvodu, zadatak sustava je da u određenom trenutku, svakom studentu dodijeli jednog od profesora kao mentora iz njegove liste željenih mentora. Svaki student mora imati minimalno dva, a maksimalno pet profesora na svojoj listi željenih mentora. Broj ponuđenih mentora ovisi o smjeru studija koji student pohađa. Popis svih studenata nalazi se u bazi podataka, gdje se uz osnovne informacije o studentu, nalazi broj godina studiranja te težinski prosjek svih predmeta zaokružen na pet decimala.

Broj godina studiranja i prosjek potrebni su radi određivanja bodova pojedinom studentu. Kada se svim studentima odrede bodovi, algoritam za dodjelu mentora započinje sa radom. Važno je napomenuti, da svaki profesor ima konačan broj studenata kojima može biti mentor. U ovom sustavu, pretpostavka je da svaki profesor ima 10 bodova. Svaki student koji mu je dodijeljen, umanjuje taj broj bodova za jedan ili dva, ovisno je li završni odnosno diplomski rad u pitanju. Tako naprimjer, mentor može maksimalno imati pet studenata koji pišu diplomski rad odnosno najviše 10 ukoliko pišu završni rad.

Po završetku dodjele, cilj je da velika većina studenata (preko 95% sa svakog studijskog smjera) dobije jednog mentora sa liste željenih mentora. Studenti koji su dobili svog mentora, unutar sustava mogu vidjeti osnovne informacije o njemu te mu se preko interakcije sa sustavom, direktno javiti putem emaila. Sustav unutar sebe nema razvijenu podršku za internu komunikaciju studenata i profesora zato što mislimo da taj zahtjev nadilazi potrebe samoga sustava. Nadalje, studenti koji nisu dobili nijednoga mentora, njihov odabir prepušten je administratoru. Administrator je vanjski entitet, odnosno aktivni korisnik sustava, no međutim on ne mora poznavati niti profesora niti studenta. Upravo iz tog razloga, svaki profesor ima tri tzv. *taga* koji pobliže opisuju područje rada i interesa samog profesora, a svaki student ima komentar čija je funkcija upravo u slučaju da ne dobije nijednog mentora, olakšati administratoru ručnu dodjelu mentora.

## 2.2 Zahtjevi

### 2.2.1 Funkcionalni zahtjevi

U ovom poglavlju ćemo se dotaknuti glavnih funkcionalnih zahtjeva koji se nalaze ispred ovoga rada. Prvi zahtjev koji se postavlja je sama prijava u sustav. Korisnik da bi se logirao, mora unijeti svoju email adresu i svoju zaporku koja mu je dodijeljena prilikom upisa na fakultet. Sam sustav trenutno nema interni mehanizam prijave, već se korisnik identificira isključivo putem emaila. Kasnije u radu ćemo više objasniti zašto je prijava realizirana na upravo ovaj način. Nadalje, nakon prijave, ovisno o tipu korisnika, koji su navedeni **[ZAMIJENITI SA KORISNICIMA KADA IH NAPISES],** korisnik ulazi u sustav. Student mora imati mogućnost odabira najmanje dva, a najviše pet mentora, ovisno o smjeru studija koji studira. Nakon što sustav završi dodjelu mentora, student mora moći vidjeti mentora kojeg je dobio odnosno ukoliko nije dobio nijednoga, također mora imati tu informaciju. Profesor u ovoj verziji sustava, ima na uvid listu svih studenata kojima je on mentor zajedno sa nekim njihovim općenitim informacijama. Administrator ima najveću ulogu u sustavu, on je odgovoran za pokretanje sustava dodjele mentora, pregled svakog pojedinačnog studenta te u slučaju da neki studenti nisu dobili mentora, mogućnost ručnog spajanja mentora i studenta.

2.2.2 Nefunkcionalni zahtjevi

Sustava je napravljen na način da odziv bilo kojeg dijela sustava ne smije trajati dulje od šest sekundi. Dostupnost mora biti neprestana i nesmetana odnosno sustav mora biti na raspolaganju 24/7h. Sustav je također prilagođen svih vrstama ekrana i rezolucija (monitorima, laptopima, tabletima, mobitelima). Sigurnost sustava osigurava autentifikaciju i autorizaciju korisnika, integritet u bazi podataka te zaštićenost privatnih podataka. Osigurana je robusnost, odnosno sustav je otporan na moguće greške i smetnje u radu, gdje se korisnika pravovremeno i ispravno informira o nastalim problemima. Posebno treba naglasiti skalabilnost odnosno ukoliko se unutar sustava nalazi veliki broj podataka to neće utjecati na nijednu od prethodno navedenih karakteristika sustava.

# Korištene tehnologije

## 3.1 Backend dio sustava

Kao što je spomenuto u uvodu, sustav se dijeli na *Frontend* i *Backend.* Prvo ćemo opisati tehnologije koje su se koristile na *Backend*dijelu sustava. Taj dio sustava je odgovaran za dohvaćanje podataka, obradu i analizu podataka te prosljeđivanje tih podataka na *Frontend* dio sustava. Kod većina projekata, tako i kod ovog, upravo je *Backend* dio aplikacije najzahtjevniji zato što iziskuje razumijevanje problematike sustava i njeno razvojno rješenje.

### 3.1.1 Visual Studio 2019 Enterprise Edition

Kao razvojno okruženje (IDE) koristio se Visual Studio 2019 sa Enterprise licencom razvijen od strane Microsofta. VS2019 koristi se u razvoju web aplikacija, web stranica, desktop aplikacija, te mobilnih aplikacija. VS2019 podržava 36 različitih programskih jezika od kojih su neki C,C++, C#, Visual Basic, itd. Sastoji se od tekstualnog/kodnog editora sa omogućenim *Intellisensom,* integriranog *debuggera*, predefiniranih GUI sučelja za web aplikacije/stranice, dizajn klasa, shema bazi podataka odnosno tablica. VS2019 omogućava i pisanje u drugim jezicima kao što su Pyhton, Ruby, Node.js, ali za njih je potrebno instalirati dodatne pakete.

### 3.1.2 Microsoft SQL Server 2019

Microsoft SQL Server je relacijska baza podataka. To je ujedno i sustav za upravljanje tim podacima. Prva inačica se pojavila davne 1989.godine. Od 2017. moguće je korištenje MSSQL Servera na *Linux* platformi. Temeljna funkcija je spremanje i dohvaćanje podataka prema zahtjevima različitih aplikacija odnosno korisnika koje se mogu pokretati ili lokalno ili preko mreže. Jezik koji se koristi unutar MSSQL Servera zove se *Transact SQL* (TSQL)

### 3.1.3 SQL Server Management Studio

SSMS omogućava konfiguriranje, upravljanje i administriranje bazama podataka unutar MSSQL Servera. Centralni dio SSMS-a je *Object explorer* koji omogućava manipulaciju nad objektima (bazama, tablicama, procedurama, kolonama, ograničenjima, itd.) unutar MSSQL servera. Također, SSMS se omogućuje kreiranje izvještaja, upravljanje rolama i operacijama nad bazama podataka. Trenutno, SSMS je moguće koristi isključivo na Windows platformi gdje su proizvođači procesora ili *Intel* ili *AMD*.

### 3.1.4 ASP.NET Core

ASP.NET Core je besplatni web *framework* otvorenog tipa (*open-source*), razvijen od strane Microsofta, namijenjen razvoju web aplikacija na .NET platformi. Važno je napomenuti da se ASP.NET Core može pokrenuti na *macOS-u, Linuxu* i *Windows* platformi. Prema **TechEmpower benchamerks**, ASP.NET Core najbrži je od svih popularnih web *frameworka*. Za razvoj ovog sustava korištena je 3.1 verzija, koja je ujedno i zadnja verzija. Također, web aplikacije razvijene u ovom *frameworku* mogu biti postavljenje na različitim platformama kao što su IIS, Apache, itd. Primjer arhitekture sustava koji koristi ASP.NET Core *framework* možete vidjeti na Slika 1

A screenshot of a cell phone

Description automatically generated

Slika 1- ASP:NET Core Arhitektura

### 3.1.5 Entity Framework Core

EF Core je objektno relacijski okvir (ORM) otvorenog tipa za manipulacija podacima i pristup bazi podataka, razvijen od strane Microsofta. EF Core omogućava izvršavanje CRUD (*Create-Read-Update-Delete*)operacija bez ručnog pisanja SQL upita. EF Core ima više vrsta pristupa odnosno omogućava aplikacijama da budu *database-first, code-first, model-first*, omogućava korištenje SQL procedura, itd. Važno je istaknuti i automatsko migracije koje uvelike olakšavaju razvijanje sustava. EF Core se najčešće kombinira sa C# sintaksom, naročito sa LINQ.

## 3.2 Frontend dio sustava

Ovaj dio sustava odgovoran je većinom za prikaz informacija koji dolaze sa *Backend* dijela. Cilj *Frontend* dijela je učiniti korisničko sučelje jednostavnim za korištenje, a u isto vrijeme omogućava korisniku sustava obavljanje željenih radnji sustava uz minimalni napor i vrijeme. Kod kreiranja korisničkog sučelja bitno je poštovati neke od principa kao što su razmještaj, svijest o sadržaju, minimiziranje korisnikovog truda, konzistentnost, itd. Tehnologije korištene u ovom dijelu sustava su standardne tehnologije za razvoj *Frontend* dijela aplikacija. HTML, CSS i Javascript kao neizostavni dijelovi te još dodatne biblioteke kao što je jQeury te korištenje CSHTML-a.

### 3.2.1 HTML, CSS, Javascript

HTML zapravo i nije programski jezik, već prezentacijski jezik za izradu web stranica. Jedan od glavnih razloga njegove opće prihvaćenosti je njegova jednostavnost u korištenju i brzom učenju. HTML kreira „kostur“ stranice odnosno označava elemente koji će se nalaziti na samoj stranici. Izgled i oblikovanje samog HTML-a odnosno web stranice odvija se preko CSS-a. CSS je stilski jezik, koji kao što je već rečeno, omogućava uređivanje HTML elemenata dodavanjem različitih boja, veličine slova, slika, okvira, itd. Konačno, interakciju tih elemanta unutar web stranice omogućuje Javascript. Javascript je skriptni jezik kojeg podržavaju svi popularni web preglednici. Ono što se nekad smatralo najvećom prednošću, a danas se nekako to pokušava izbjeći korištenjem drugih biblioteka, to što Javascript nema tipove podataka te vrlo lako može doći do greške ukoliko niste sasvim sigurno što radite. Javascript je interpreter odnosno kod se izvršava liniju po liniju bez prethodnog prevođenja.

Standarde HTML-a, CSS-a i Javascripta postavlja *World Wide Web Consortium* poznatiji kao W3C. U sustavu su korištene najnovije verzije ovih tehnologija, odnosno HTML5, CSS3 i ECMAScript 2015.

### 3.2.2 jQuery , Bootstrap i CSHTML

Bootstrap, jQuery i CSHTML se baziraju na već postojećim tehnologijama uz neka poboljšanja. jQuery je biblioteka koja proširuje mogućnosti odnosno olakšava manipulaciju DOM-om. Iako je u samim začecima, jQeury bio jako popularan, bio je „*easy-to-use“*, njegovo zlatno doba je brzo prošlo. Danas su ga zamijenile nove tehnologije i proširenja kao što su Angular, React i sl. U ovom sustavu jQuery se koristio isključivo radi AJAX poziva, odnosno radi asinkronih poziva prema *backend* dijelu aplikacije. Bootstrap omogućuje korištenje već predefiniranih HTML elemenata i njihovih stilova. Vrlo je koristan upravo zbog brzine razvoja ovog dijela (*frontend)* aplikacije. CSHTML omogućuje korištenje C# sintakse unutar HTML-a. Najlakše ćemo shvatili razliku između HTML-a i CSHTML-a ako kažemo da se HTML izvršava isključivo na klijentskom dijelu, npr. unutar browsera, a CSHTML omogućuje da se dio podataka koji se prikazuju na klijentu, prvo dohvate iz *backend* dijela aplikacije koristeći C# sintaksu te zatim renderiraju unutar HTML-a.

# Razvoj web sustava za automatsku dodjelu mentora na završnim i diplomskim radovima

## 4.1 **Arhitektura sustava**

Za razvoj ovog sustava koristila se jedna od inačica *n-tier* arhitektura. Princip rada ove arhitekture je odvojiti slojeve sustava u zasebne cjeline gdje svaki sloj ima svoju ulogu unutar sustava. Cilj ove arhitekture je omogućiti bolju testabilnost, održivost i neovisnost slojeva. Također, bitno je napomenuti da u ovoj arhitekturi komunikacija se odvija isključivo između susjednih slojeva. Puno ime ove arhitekture naziva se *Onion* arhitektura odnosno slojevita arhitektura. Ova arhitektura se uvelike oslanja na *Dependency Inversion Principle*.

Slika 2 prikazuje podjelu ove arhitekture po slojevima, a u nastavku teksta ćemo objasniti koja je funkcija svakog pojedinog sloja.

A close up of a logo

Description automatically generated

Slika 2 - Prikaz Onion arhitekture

Sloj *Domain Entities* je centralni dio arhitekture. Unutar njega se nalaze svi domenski entiteti. Entiteti unutar ovoga sloja nemaju nikakve *dependencies* tj. ovisnosti.

Repozitorij sloj predstavlja apstrakcijski sloj između poslovnog (*service*) i domenskoj sloja. Preko repozitorija se dohvaćaju i spremaju svi podaci u bazu podataka. Unutar njega kreiramo generičke metode za sve CRUD operacije te unutar njega mapiramo DTO-ove u entitete.

Servis sloj (poznatiji i kao poslovni sloj), sadrži svu logiku aplikacije. Dakle, osim što je on vezivni sloj između repozitorija i domenskog sloja, tu se nalaze svi poslovi procesi, *business* logika aplikacije. Npr. unutar ovoga sloja, nalazi se naš algoritam za izračun bodova svakom studentu i dodjelu mentora.

UI sloj odnosno korisničko sučelje je ono što korisnik vidi. Ovaj sloj je za običnog korisnika najbitniji upravo zato što korisnik ima interakciju samo sa tim slojem.

Da bi upotpunili sliku kako ova arhitektura funkcionira, prikazat ćemo na primjeru prijave u sustav.