Izrada aplikacije za izradu dnevnika rada koristeći programske tehnologije ASP.NET Core i React

Jerbić, Helena

VELEUČILIŠTE U VIROVITICI

Stručni prijediplomski studij Računarstvo

HELENA JERBIĆ

IZRADA APLIKACIJE ZA IZRADU DNEVNIKA RADA KORISTEĆI PROGRAMSKE TEHNOLOGIJE ASP.NET CORE I REACT

ZAVRŠNI RAD

VIROVITICA, 2024.

VELEUČILIŠTE U VIROVITICI

Stručni prijediplomski studij Računarstvo

IZRADA APLIKACIJE ZA IZRADU DNEVNIKA RADA KORISTEĆI PROGRAMSKE TEHNOLOGIJE ASP.NET CORE I REACT

ZAVRŠNI RAD

Predmet: PROJEKTIRANJE INFORMACIJSKIH SUSTAVA

Mentor: Student:

Ivan Heđi, dipl. ing., v. pred. Helena Jerbić

VIROVITICA, 2024.

Veleučilište u Virovitici

OBRAZAC 2

ZADATAK ZAVRŠNOG / DIPLOMSKOG RADA

Student/ica: **HELENA JERBIĆ** **JMBAG:** **0307017057**

Studij: Računarstvo **Modul:** Programsko inženjerstvo

Imenovani mentor: Ivan Heđi, dipl. ing., v. pred.

Imenovani komentor:

Naslov rada:

Izrada aplikacije za izradu dnevnika rada koristeći programske tehnologije ASP.NET Core i React

Puni tekst zadatka rada:

Koristeći tehnologije ASP.NET Core i React osmisliti i izraditi aplikaciju koja će služiti kao pomoć u vođenju dnevnika rada poludnevnih boravaka u organizaciji Centra za nestalu i zlostavljanu djecu. Vaš je zadatak i kreirati bazu podataka na platformi MSSQL server. Osim programskog rješenja, u pisanom dijelu završnog rada opišite ukratko korištene tehnologije, opišite detaljno arhitekturu sustava te opišite detaljno odabrane procese i/ili funkcije unutar same aplikacije. Prilikom opisivanja, osim neformalnih koristite i neke od formalnih metoda koje poznajete.

Datum uručenja zadatka studentu/ici: 29. 07. 2024.
Rok za predaju gotovog rada: 02. 09. 2024.

Mentor:

Ivan Heđi, dipl. ing., v. pred.

Dostaviti:
1. Studentu/ici
2. Povjerenstvu za završni i diplomski rad - tajniku

IZRADA APLIKACIJE ZA IZRADU DNEVNIKA RADA KORISTEĆI PROGRAMSKE TEHNOLOGIJE ASP.NET CORE I REACT

Sažetak

Ovaj završni rad govori o aplikaciji Dnevnik rada. Aplikacija je nastala kako bi olakšala i ubrzala administraciju u poludnevnim boravcima koji su jedna od usluga koju provodi Centar za nestalu i zlostavljanu djecu. Poludnevne boravke trenutno pohađa preko 100 djece u dobi od 7 do 18 godina. Ova djecu su manje uspješna u školi, obitelji ili među vršnjacima te im je zbog toga potrebna strukturirana pomoć kako bi ostvarili svoj puni potencijal. Automatizacija administracije omogućava odgajateljima da se više posvete djeci, a manje papirima. Aplikacija je stvarana prema postojećim poslovnim logikama i s ciljem da korisnicima bude što lakša za korištenje. U aplikaciji se svakodnevno može upisivati Dnevnik rada, a na kraju mjeseca u aplikaciji se u nekoliko klikova može izraditi i mjesečni izvještaj dolazaka. Do sada je ovaj mjesečni izvještaj trebalo pisati ručno. Pri stvaranju dnevnika rada unaprijed su upisane zadane i često korištene vrijednosti te se i tako pokušava uštedjeti na vremenu. Aplikacija se nalazi na serveru Veleučilišta u Virovitici i javno je dostupna za uporabu. Na kraju rada nalaze se i komentari koordinatorice poludnevnih boravaka i voditeljice socijalni usluga i jedne odgajateljice koja je u prošloj školskoj godini koristila aplikaciju. Za aplikaciju su korištene prvenstveno tehnologije ASP.NET Core i React, a u radu su osim njih opisane i tehnologije baze podataka, C# programsko jezika, .NET arhitektura, HTML, JavaScript i CSS.

Ključne riječi: *ASP.NET Core, Dnevnik rada, Mjesečni izvještaji, React, web aplikacija*

APPLICATION FOR CREATING A WORK LOG USING ASP.NET CORE AND REACT PROGRAMMING TECHNOLOGIES

Abstract

This final thesis talk about aplication Dnevnik rada (work log). The aplication was created to facilitate and speed up the administration of half-day-care, which are one of the services provided by the Center for Missing and Abused Children. Half-day-cares are currently attended by over 100 children aged 7 to 18. These children are less successful in school, family or among peers and therefore need structured help to reach their full potential. Automating administration allows educators to devote more time to children and less time to paperwork. The application was created according to existing business logic and with the aim of making it as easy as possible for users to use. In the application, you can enter a work log every day, and at the end of the month, in a few clicks, you can create a monthly arrival report in the application. Until now, this monthly report had to be written manually. When creating the work log, the default and frequently used values are written in advance, and thus an attempt is made to save time. The application is located on the server of the Virovitica University of Applied Sciences and is publicly available for use. At the end of the thesis are also comments from the coordinator of half-day-cares and the head of social services and one educator who used the application in the last school year. ASP.NET Core and React technologies were primarily used for the application, and the thesis also describes database technologies, C# programming language, .NET architecture, HTML, JavaScript and CSS.

Keywords: *ASP.NET Core, monthly report, React, web application, work log,*

Sadržaj

1. Uvod

Ideja za izradu aplikacije Dnevnik rada javila se prilikom volontiranja u Centru za nestalu i zlostavljanu djecu. Prilikom volontiranja primijećeno je kako odgajatelji iste podatke upisuju na više mjesta te time bespotrebno troše vrijeme koje bi mogli koristiti za rad s djecom, to jest korisnicima. Uviđen je veliki potencijal za automatizaciju i lakše izvođenje nekih administrativni poslova. Aplikacija je rađena tako da prati već postojeće poslovne procese. Aplikacija je dizajnirana s naglaskom na intuitivnost i korisničko iskustvo kako bi bila što jednostavnija za korištenje i prijelaz s prijašnjih načina vođenja evidencija.

Aplikacija se sastoji od dva glavna dijela, dio za odgajatelje, to jest administracija i dio koji koriste djeca. Dio za djecu je vrlo jednostavan, sastoji se od tablice gdje djeca svaki dan naznačuju jesu li ili nisu došli u boravak. Dio za odgajatelje sastoji se od tri dijela, prvi dio je za svakodnevno pisanje dnevnika rada, drugi za stvaranje i pregledavanje mjesečne evidencije, a treće su postavke.

U ovom završnom radu pisano je o korištenim tehnologijama, zatim detaljnije o samoj aplikaciji i na kraju su komentari korisnika aplikacije. Korištene tehnologije su ASP.NET Core za poslužiteljski dio i React za korisničko sučelje.

2. Korištene tehnologije

U ovom poglavlju će se opisati tehnologije korištene u razvoju projekta. Njih je moguće podijeliti u dva glavna sloja: poslužiteljska strana i klijentska strana. Poslužiteljska strana upravlja poslovnom logikom i pristupom podacima. Ovdje će biti opisani SQL server, C# programski jezik i ASP.NET Core. Klijentska strana služi za izgled aplikacije i komunikaciju s klijentom. Od tehnologija na klijentskoj strani opisane su HTML, JavaScript i CSS.

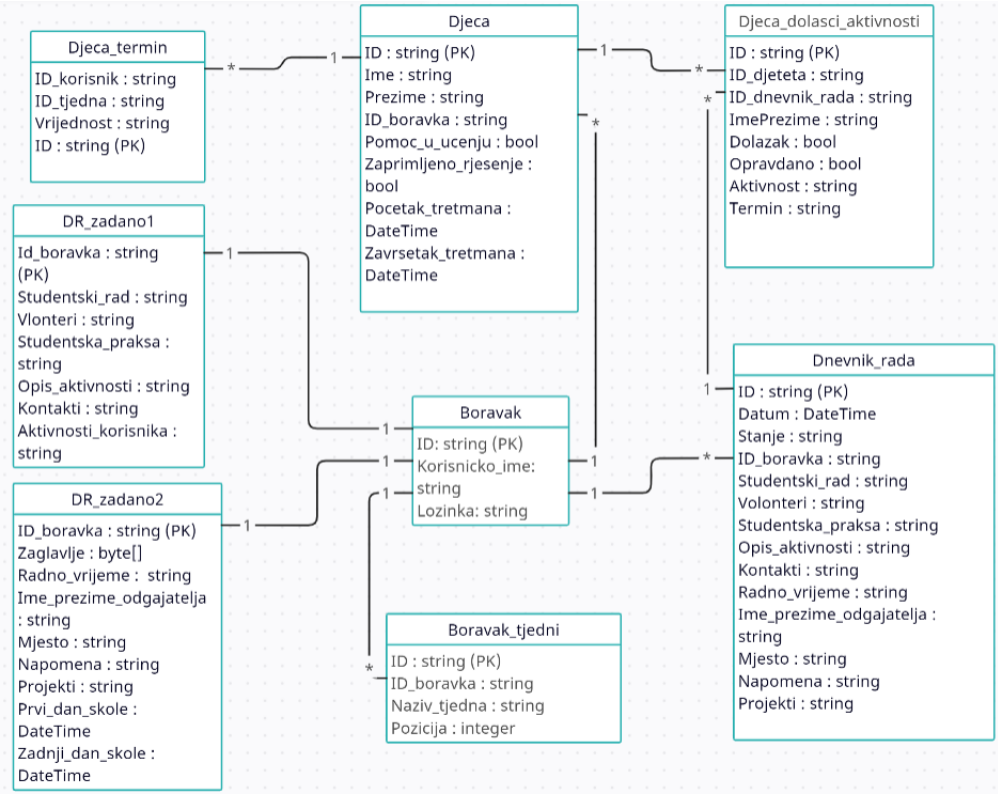
2.1. Baza podataka i SQL server

Baza podataka je računalno spremište podataka. Ona ima mogućnost opisivanja sadržaja, spremanje, pretraživanje i dohvaćanje podataka te održavanje integriteta[1], sadržaja i opisa podataka. Programski sustav koji omogućava upravljanje podataka u bazi podataka naziva se sustav za upravljanje bazom podataka ili DBMS (engl. *Database Management Systems*) [1]. Neke od prednosti korištenja sustava za upravljanje bazom podataka su: ne ponavljanje podataka, dijeljenje podataka između više aplikacija i programa, očuvanje integriteta podataka, sigurnost i fizička i logična neovisnost podataka. S druge strane nekoliko mana su: problemi koji dolaze s centralizacijom, cijena sklopovlja i programa za upravljanja bazom i kompleksnost sigurnosnog kopiranja i povrata podataka [2].

Način organiziranja podataka i odnosa među njima naziva se podatkovni model. U ovom projektu korišten je relacijski model, kojeg je opisao Edgar F. Codd 1970. godine. Podaci su organizirani u tablice koje imaju redove i stupce. Definiran je pojam primarnog ključa (engl. *primary key*) kao jedinstvenog identifikatora svakog reda u tablici. Također, Codd je zagovarao neovisnost podataka, to jest da promjene u strukturi baze podataka ne bi trebale utjecati na aplikacije koje pristupaju podacima [6]. Veze u relacijskim bazama podataka su jedan prema jedan, jedan prema više i više prema više. Primjer u aplikaciji Dnevnik rada za jedan prema jedan je veza Boravak – DR_Zadano_1 što znači da svaki DR_Zadano_1 ima samo jedan povezani boravak, isto kao što svaki boravak ima povezan samo jedan DR_Zadano_1. Primjer za vezu jedan prema više je veza Boravak – Dijete što znači da svako dijete može imati samo jedan boravak, ali boravak može imati više djece. U aplikaciji nije korištena veza više prema više koja bi označavala da je redak u jednoj tablici

[1] Integritet podataka – sigurnost da su podaci u bazi točni, cjeloviti, pouzdani i dosljedni tijekom cijelog životnog ciklusa podatka

povezan s više redova u drugoj tablici i da je jedan redak u drugoj tablici povezan s više redova u prvoj tablici. Na slici (slika 1) može se vidjeti primjer navedenih veza. Ovo je također i shema baze korištene u aplikaciji.



Slika 1. Dijagram baze podataka

Jedna od najpoznatijih implementacija sustava za upravljanje relacijskim bazama podataka je SQL Server. Njegova temeljna komponenta je SQL Server Database Engine koja upravljanje podatkovnom memorijom, obradom i sigurnošću. Ona se sastoji od dvije komponente: relational engine i storage engine. Relational engine služi za obradu upita i naredbi. Storage engine služi za upravljanje tablicama, indeksima, stranicama, datotekama, međuspremnicima i prometom baze podataka te kreira i izvršava spremljene postupke, okidače, poglede i drugo [7].

2.2. C# programski jezik

C# je moderni programski jezik otvorenog koda[2]. Razvijen je od strane Microsoft-a, a glavni dizajner je Anders Hejlsberg [8]. Jedan je od najpopularnijih programskih jezika pa

[2] Otvoreni kod (engl.*open source*) označava da je izvorni kod slobodno dostupan te omogućuje korisnicima pregledavanje, modificiranje i dijeljenje programa

se tako našao kao jedan od pet najpopularnijih jezika na GitHub-u[3] [9]. Zbog svoje jednostavnosti, sigurnosti i snažne podrške C# se koristi za mnoge desktop, web i mobilne aplikacije [9].

C# je statički tipiziran što znači da se tipovi varijabli određuju tijekom kompilacije, a ne tijekom izvođenja. Osnovni tipovi podataka u C# programskom jeziku su integer (engl. *cijeli brojevi*) (sbyte, byte, short, ushort, int, uint, long i ulong), pravi brojevi s pomičnim zarezom (float, double), brojevi s decimalnom preciznošu (decimal), logički tip (bool), znak (char), niz znakova (string) i objekt (object) [3]

Ovaj jezik omogućuje mnoge pristupe razvoju softvera, kao na primjer objektno-orijentirano programiranje, strukturalno programiranje, funkcijsko programiranje, generičko programiranje i asinkrono programiranje. Ovi pristupi ili paradigme mogu zajedno postojati unutar jednog projekta što se tada naziva multiparadigmatsko programiranje, a jezik koji podržava više paradigmi nazivamo multiparadigmatski jezik.

C# posjeduje mnoge biblioteka i alata poput Entity Frameworka za jednostavno upravljanje bazom podataka, Data Annotations za učinkovitu provjeru ispravnosti podataka, ASP.NET Core MVC (engl. *Model View Controller*) za razvoj web aplikacija temeljenih na MVC (engl. *Model View Controller*) arhitekturi, Security Cryptography za implementaciju kriptografskih algoritama[4] i Identity Model koja upravlja tokenima[5] kod autentifikacije. Sve navedene biblioteke korištene su pri razvoju aplikacije Dnevnik rada, a neke će biti još dodatno objašnjene u ostatku rada.

2.3. .NET arhitektura

.NET je platforma za razvoj softvera koju je razvio Microsoft. Ona pruža infrastrukturu koja nadograđuje operativni sustav kako bi olakšala razvoj različitih aplikacija, od web aplikacija, preko servisa i desktop aplikacija do mobilnih aplikacija. .NET aplikacije mogu biti napisane u više jezika, a neki od popularnijih su C#, F# i Visual Basic. Postoji nekoliko inačica .NET-a koje omogućavaju izvršavanje koda na različitim platformama. .NET Framework je originalna implementacija .NET-a koja omogućava pokretanje aplikacija na operativnom sustavu Windows, .NET je implementacija koja podržava pokretanje na sustavima Windows, Linux i macOS. Ova implementacija se prije

[3] GitHub je internetska platforma za verzioniranje i zajednički razvoj programa koja omogućava programerima praćenje promjena u kodu, upravljanje projekata, surađivanje na projektima i slično
[4] Kriptografski algoritmi su koraci ili pravila koji služe za šifriranje i dešifriranje podataka
[5] Token je digitalna oznaka koje potvrđuju identitet korisnika

nazivala .NET Core, te je program otvorenog koda na GitHub-u. Zadnja implementacija je Xamarin/Mono koja služi za pokretanje aplikacija na mobilnim operativnim sustavima poput iOS i Android. Kako bi se isti kod i knjižnice mogle izvršavati na različitim implementacijama postoji .NET Standard, to jest formalna specifikacija API[6]-ja koji su zajednički svim implementacijama.[10]

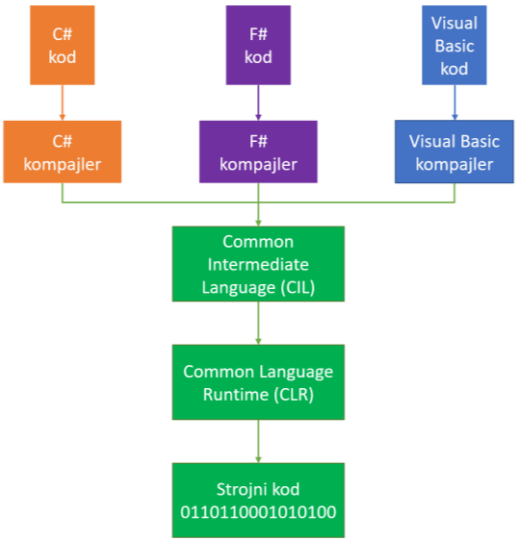
Dvije najvažnije komponente .NET Frameworka su Common Language Runtime (CLR) i Class Library. Common Language Runtime je softverski sustav u kojem se kod izvršava. On pruža različite usluge kao što je upravljanje nitima (engl.*threads*), čišćenje smeća (engl. *garbage collection*), sigurnost tipova i rukovanje iznimkama (engl. *Exception handling*). Čišćenje smeća znači da CLR automatski upravlja memorijom, uklanjajući objekte koji više nisu potrebni ili dostupni. Sigurnost tipova sprječava korištenje neželjenih operacija nad podacima. Na primjer, ako postoji varijabla koja je integer, u nju se ne može spremiti podatak tipa string. CLR omogućuje strukturirano upravljanje iznimkama, što omogućuje programerima identificiranje, praćenje i rukovanje pogreškama ili neočekivanim situacijama tijekom izvođenja programa.

Class Library sadrži skup API-ja i tipova za često korištene funkcionalnosti. API-ji koje uključuje su namijenjeni za čitanje i pisanje datoteka, povezivanje s bazom podatak i još mnogo toga. Tipovi koje Class Library sadrži su tipovi za rad s nizovima, datumima, brojevima i tako dalje.

Na Slici 2 prikazan je put od koda napisanog u nekom programskom jeziku do strojnog koda. Aplikacije napisane u jezicima poput C#, F# i Visual Basic se prvo prevode u CIL (engl. *Common Intermediate Language*). Tako kompilirani kod se pohranjuje u skupove podataka, to jest datoteke s nastavcima .dll ili .exe. Kada se aplikacija pokrene CLR (engl. *Common Language Runtime*) uzima skup podataka i koristi JIT (engl. *Just in time*) kompajler[7] kako bi ga pretvorio u strojni kod koji se zatim izvodi na računalu.

[6] API (engl. *Application Programming Interface*) – skup definiranih pravila i protokola koji omogućavaju komunikaciju između različitih softverskih komoponenti

[7] Kompajler – program koji prevodi kod napisan u neko programskom jeziku u strojni kod koji je razumljiv računalu



Slika 2. Dijagram .NET kompajlera [10]

2.3.1. ASP.NET MVC arhitektura

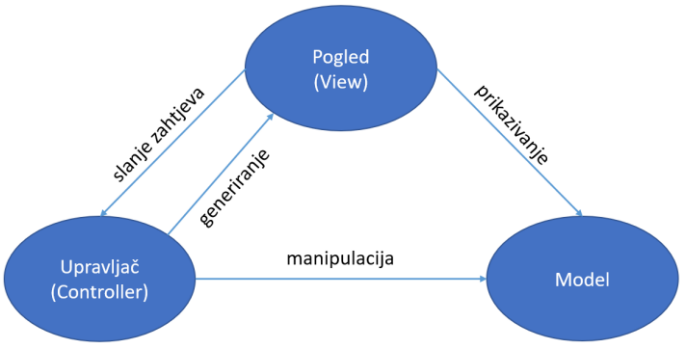
MVC (engl. *Model View Controller*) arhitektura je obrazac arhitekture koji odvaja pojedine dijelove aplikacije u tri komponente: model (engl. *model*), pogled (engl. *view)* i upravljač (engl. *controller*) [11].

Model (engl. *model*), sadrži, to jest predstavlja, podatke. U kodu je prikazan kao klasa. Podatke preuzima iz baze podataka. U ovoj komponenti sprema se poslovna logika.

Pogled (engl. *view)* je u suštini korisničko sučelje. Dakle on prikazuje podatke u obliku obrazaca, tablica i dijagrama. Pisan je u HTML-u, CSS-u i još jednoj posebnoj sintaksi koja olakšava komunikaciju s modelom i kontrolerom.

Upravljač (engl. *controller*) upravlja korisničkim zahtjevima. Obično korisnik koristi pogled i preko njega šalje HTTP (engl. *HyperText Transfer Protocol*) zahtjev kojim zatim upravlja upravljač. U ovoj komponenti sprema se ulazna logika (engl. *input logic*).

Ovakva podjela poslovne logike, korisničkog sučelja i upravljanja interakcijama omogućava nezavisan razvoj svake komponente te je zato vrlo pogodna za korištenje u radu na projektima u kojima je uključen veći broj programera. Olakšava i sam razvoj aplikacije jer omogućuje da se izrađuje i testira svaka komponenta zasebno. Na slici (Slika 3) je vidljivo kako su komponente međusobno povezane.



Slika 3. Prikaz MVC arhitekture [12]

Kao što je i prije spomenuto u kodu je korištena MVC arhitektura. Slijede primjeri za navedene komponente iz samog koda aplikacije. Korišten je primjer za dnevnik rada. U modelu, to jest klasi je vidljivo koji su podaci spremljeni u dnevnik rada. U upravljaču je vidljivo što se događa kada se dobije HTTP get zahtjev za prikaz dnevnika. U pogledu je vidljivo kako se taj zahtjev šalje i kako se odgovor postavlja u varijablu dnevnik (pomoću setDnevnik). U pogledu je također vidljiv i prikaz tablice dnevnika rada.

Isječak programskog koda 1: Model – DnevnikRadaModel

```
1. public class DnevnikRadaModel
2.     {
3.         [JsonPropertyName("ID")]
4.         public string ID { get; set; }

5.         [JsonPropertyName("Boravak")]
6.         public string Boravak { get; set; }

7.         [JsonPropertyName("Datum")]
8.         public DateTime Datum { get; set; }
9.
                ...

25.        [JsonPropertyName("Projekti")]
26.        public string Projekti { get; set; }

27.        [JsonPropertyName("Napomena")]
28.        public string Napomena { get; set; }
29.
30.        public MyDBContext dbContext;
31.    }
```

Isječak programskog koda 2: Upravljač – DnevnikRadaController

```
1. [Route("api/[controller]/[action]")]
2.     [ApiController]
3.     public class DnevnikRadaController : Controller
4.     {
5.         private readonly MyDBContext dbContext;
6.
7.         public DnevnikRadaController(MyDBContext dbContext)
8.         {
9.             this.dbContext = dbContext;
10.        }
11.
12.        [HttpGet]
13.        public IActionResult DohvatiDnevnik(string IDboravka, DateTime Datum)
14.        {
15.            DnevnikRadaModel Dnevnik = new DnevnikRadaModel(dbContext, IDboravka, Datum);
16.            if (Dnevnik != null)
17.            {
18.                return Ok(Dnevnik);
19.            }
20.            return BadRequest("Dnevnik rada je null");
21.        }
22.    }
23.
```

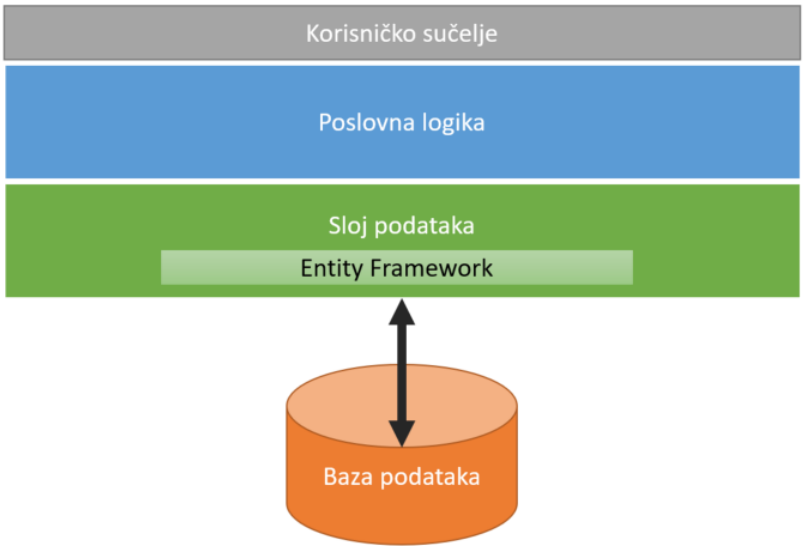
Isječak programskog koda 3: Pogled – prikaz vrijednosti dnevnika

```
useEffect(() => {
    axios({
            method: "get",
            url: "http://193.198.57.183:7078/api/DnevnikRada/DohvatiDnevnik",
            headers: { 'Content-Type': 'application/json' },
            params: {
                IDboravka: IDboravka,
                Datum : datum
            }
        }).then(function (response) {
            setDnevnik(response.data);
            console.log(response.data);
            setLoading(false);
        }).catch(function () {
            setLoading(false);
        });
    },[datum]);

    <>
    <table className="table-admin table-aktivnost">
        <tbody>
            <tr>
                <td>Opis aktivnosti: </td>
                <td>
                    <textarea className="form-control" id="opis_aktivnosti"
                            value={dnevnik.OpisAktivnosti ? dnevnik.OpisAktivnosti : ""}/>
                </td>
            </tr>
            <tr>
                <td>Kontakti: </td>
                <td><textarea className="form-control" id="kontakti"
                            value={dnevnik.Kontakti ? dnevnik.Kontakti : ""}/></td>
            </tr>
            <tr>
                <td>Studentski rad: </td>
                <td><textarea className="form-control" id="studentski_rad"
                            value={dnevnik.StudentskiRad ? dnevnik.StudentskiRad : ""}
                           /></td>
            </tr>
            <tr>
                <td>Volonteri: </td>
                <td><textarea className="form-control" id="volonteri"
                            value={dnevnik.Volonteri ? dnevnik.Volonteri : ""} /></td>
            </tr>
            <tr>
                <td>Studentska praksa: </td>
                <td><textarea className="form-control" id="studentska_praksa"
                            value={dnevnik.StudentskaPraksa ? dnevnik.StudentskaPraksa : ""}
                           /></td>
            </tr>
            <tr>
                <td>Napomena: </td>
                <td><textarea className="form-control" id="napomena"
                            value={dnevnik.Napomena ? dnevnik.Napomena : ""} /></td>
            </tr>
            <tr>
                <td>Projekti: </td>
                <td><textarea className="form-control" id="projekti"
                            value={dnevnik.Projekti ? dnevnik.Projekti : ""} /></td>
            </tr>
        </tbody>
    </table>
    </>
```

2.3.2. Entity Framework

Entity Framework je ORM (engl. *Object-relational Mapping*)[8] alat koji omogućava rad s podacima u relacijskoj bazi podataka koristeći objekte umjesto SQL upita[13]. To omogućuje programerima da se posvete logici same aplikacije, a ne na detalje oko baze podataka. Svaka aplikacija sadrži slojeve: korisničko sučelje, poslovna logika i sloj podataka koji je povezan s bazom podataka. Na slici (Slika 4) vidljivo je da se Entity framework nalazi u sloju podataka, između poslovne logike i baze podatka. Glavne komponente Entity frameworka su: modeli podataka, DbContext, upiti i migracije.



Slika 4. Pozicija entity frameworka u slojevima aplikacije [14]

Modeli podataka su klase koje predstavljaju strukturu baze podataka. Nazivamo ih još i entiteti. Model predstavlja tablicu u bazi, a svojstva modela predstavljaju stupce. Isječak programskog koda (Isječak programskog koda 4) i slika (Slika 5) pokazuju ovu povezanost. Isječak programskog koda je klasa Boravak koja sadrži svojstva ID, Korisnicko_ime i Lozinka, a slika je snimka zaslona tablice Boravak koja sadrži stupce ID, Korisnicko_ime i Lozinka.

[8] Tehnika programiranja koja omogućava transformiranje podataka iz baze podataka u objekte i obrnuto bez korištenja SQL upita

Isječak programskog koda 4: Model podataka – Boravak

```
1. public class Boravak
2.     {
3.         public string ID { get; set; }
4.         public string Korisnicko_ime { get; set; }
5.         public string Lozinka { get; set; }
6.
7.     }
```

DBSERVER.Evidenci...NZD - dbo.Boravak

	ID	Korisnicko_ime	Lozinka
	ID00	pb_00	0873f1bb86...
	ime.prezime133467706291281987	ime.prezime	ff8c3628658...

Slika 5. Snimka zaslona - tablica Boravak u bazi podataka

DbContext je ključna komponenta koja predstavlja kontekst podataka. To je klasa koja sadrži setove podataka koji predstavljaju entitete. Ona omogućava izvršavanje upita prema tim entitetima te tako omogućava komunikaciju između aplikacije i baze podataka. U ovoj aplikaciji klasa se zove MyDBContext, a može se vidjeti u isječku programskog koda (isječak programskog koda 5).

Isječak programskog koda 5: MyDBContext komponenta

```
 1. public class MyDBContext : DbContext
 2.     {
 3.         public MyDBContext(DbContextOptions<MyDBContext> options) : base(options)
 4.         {
 5.         }
 6.         public DbSet<Boravak> Boravak { get; set; }
 7.         public DbSet<Dnevnik_rada> Dnevnik_rada { get; set; }
 8.         public DbSet<Djeca> Djeca { get; set; }
 9.         public DbSet<Djeca_dolasci_aktivnosti> Djeca_dolasci_aktivnosti { get; set; }
10.         public DbSet<Djeca_termini> Djeca_termini { get; set; }
11.         public DbSet<Boravak_tjedni> Boravak_tjedni { get; set; }
12.         public DbSet<DR_zadano1> DR_zadano1 { get; set; }
13.         public DbSet<DR_zadano2> DR_zadano2 { get; set; }
14.     }
```

Upiti su važni jer zbog njih nije potrebno pisati SQL upite, što je i cijela ideja Entity frameworka. Upiti se pišu koristeći LINQ (engl. *Language-Integrated Query*) i metode upita. LINQ se može pojednostavljeno zamisliti kao SQL (engl. *Structured query language*) upite koji se piše unutar aplikacije. Ipak postoje velike razlike između to dvoje. LINQ se piše unutar C# programskog jezika dok je SQL zaseban jezik, LINQ radi s objektima i kolekcijama, a SQL s relacijskim tablicama i podacima u bazi podataka. Metode upita su funkcije u kojima je konkretna implementacija LINQ, na primjer Where(), OrderBy(). Select(). Ovako napisani upiti prevode se u SQL upite koji se zatim izvršavaju nad bazom. U isječku programskog koda (isječak programskog koda 6) vidljiva je kombinaciju LINQ i

metoda upita, a u isječku programskog koda (isječak programskog koda 7) vidljivo je korištenje samo metode upita Add().

Isječak programskog koda 6: LINQ i metode upita

```
var result = from d in DjecaDb
                         join dt in DjecaTerminiDb on d.ID equals dt.ID_korisnik into d_dt
                         from dt in d_dt.DefaultIfEmpty()
                         join t in TjedniDb on dt.ID_tjedna equals t.ID into dt_t
                         from t in dt_t.DefaultIfEmpty()
                         group new { dt, t } by new { d.ID, d.Ime, d.Prezime,
                         d.Pocetak_tretmana, d.Zavrsetak_tretmana, d.ID_boravka,
                         d.Pomoc_u_ucenju, d.Zaprimljeno_rjesenje } into g
                         select new
                         {
                             id = g.Key.ID,
                             ime = g.Key.Ime,
                             prezime = g.Key.Prezime,
                             pocetak_tretmana = g.Key.Pocetak_tretmana,
                             pomoc_u_ucenju = g.Key.Pomoc_u_ucenju,
                             zaprimljeno_rjesenje = g.Key.Zaprimljeno_rjesenje,
                             zavrsetak_tretmana = g.Key.Zavrsetak_tretmana,
                             raspored = g.Where(x => x.t != null).ToDictionary(x =>
                                   x.t.ID, x => x.dt.Vrijednost)
                         };
```

Isječak programskog koda 7: Metoda upita

```
dBContext.Djeca_termini.Add(termin);
```

Migracije su mehanizam koji omogućava automatsko ažuriranje strukture baze podataka kako se model podataka mijenja. Entity framework prati promjene u modelu i kada pokrenemo migraciju ona generira SQL skriptu za stvaranje istih promjena u bazi podataka. Primjer promjene modela bi bilo dodavanje tablice, a u programskom isječku (programski isječak 8) vidljivo je kako izgleda migracija AddTableDjeca_dolasci_aktivnost u kojoj je dodana jedna tablica.

Isječak programskog koda 8: Primjer migracije

```
public partial class AddTableDjeca_dolasci_aktivnost1 : Migration
    {
        /// <inheritdoc />
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.CreateTable(
                name: "Djeca_dolasci_aktivnosti",
                columns: table => new
                {
                    ID = table.Column<string>(type: "nvarchar(450)", nullable: false),
                    ID_djeteta = table.Column<string>(type: "nvarchar(max)", nullable:
                false),
                    ID_dnevnik_rada = table.Column<string>(type: "nvarchar(max)",
                nullable: false),
                    Dolazak = table.Column<bool>(type: "bit", nullable: false),
                    Opravdano = table.Column<bool>(type: "bit", nullable: true),
                    Aktivnost = table.Column<string>(type: "nvarchar(max)", nullable:
                false),
                    Termin = table.Column<string>(type: "nvarchar(max)", nullable: false)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_Djeca_dolasci_aktivnosti", x => x.ID);
                });
        }

        /// <inheritdoc />
        protected override void Down(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.DropTable(
                name: "Djeca_dolasci_aktivnosti");
        }
    }
```

2.4. HTML

HTML (engl. *HyperText Markup Language*) je osnovni jezik koji upotrebljavamo za kreiranje web stranica. To je jezik pomoću kojeg komuniciramo s web preglednicima, to jest web preglednici prikazuju podatke pisane u HTML-u. Iako je ovo jezik za pisanje hipertekstualnih dokumenata, to nije programski jezik jer njime nije moguće izvršiti nikakav zadatak, čak niti zbrojiti dva broja.[15] Može se reći da je HTML temelj weba jer bez njega ne bismo imali web stranice. Zbog mogućnosti dodavanja poveznica (engl. *hyperlink*) moguće je povezivati različite stranice, resurse i informacije.

Prvu verziju HTML-a stvorio je Tim Berners-Lee krajem 1991. godine, a službeno je objavljen 1993. Verzija HTML 1.2. koja je u početku bila jako jednostavna. 1999. izašla je vrlo uspješna i često korištena verzija HTML 4.0. HTML 5.0 objavljen je 2014. godine te je proširena verzija HTML 4.01. Ova verzija se trenutno koristi u cijelom svijetu. U tablici (Tablica 1) su vidljive značajke dodane u određenoj verziji HTML-a

HTML naravno ima određena pravila, ali su većinski ta pravila slaba jer većina Internet preglednika dopušta da se gotovo sve prikaže. Ipak, trebalo bi se držati smjernica i pravila jer njihovo nepoštivanje dolazi do izražaja kada se doda CSS ili JavaScript. [4]

Svaki HTML dokument se sastoji od elemenata koji se koriste za označavanje dijelova sadržaja web stranica. Svaki element se sastoji od dvije oznake (engl. *tag*). Jedna se postavlja na početak elementa, a druga na kraj. Oznaka koja se stavlja na početak izgleda <*ime_elementa*>, a oznaka koja se stavlja na kraj </*ime_elementa*>. Postoje elementi, kao na primjer element *img* koji se koristi za postavljanje slike, koji se mogu sastojati od samo jedne oznake. U tom slučaju ta oznaka izgleda <*ime_elementa/*>. Unutar prve oznake postavljaju se i atributi. Atributi pružaju dodatne informacije o elementima. Oznake se mogu postavljati jedna u drugu. U tom slučaju bitno je da se prvo zatvore unutarnje oznake, a zatim vanjske, to jest da vanjske oznake obgrle unutarnje. U isječku programskog koda (*Isječak programskog koda 9*) prikazan je primjer točno napisano HTML koda. Može se vidjeti element *div* koji se sastoji od dvije oznake. Unutar tog elementa prikazan je element *p* koji sadrži atribut *id*. Ovaj programski isječak uzet je iz koda aplikacije, dijela za odjavu korisnika.

Isječak programskog koda 9*: Primjer HTML koda*

```
1. <div className="odjava_div">
2.        <p id="p_odjava">Hvala vam na vašem trudu.</p>
3. </div>
```

Svaki HTML dokument ima isti kostur koji je vidljiv u programskom isječku (*programski isječak 10*). U projektu Dnevnik rada ovaj kostur nalazi se u datoteci index.html.

Isječak programskog koda 10*: HTML kostur*

```
 1. <!DOCTYPE html>
 2. <html lang="en">
 3.       <head>
 4.                <meta charset="utf-8" />
 5.                <title>Dnevnik rada</title>
 6.       </head>
 7.       <body>
 8.                <div id="root"></div>
 9.       </body>
10. </html>
```

2.5. JavaScript

JavaScript je razvijen u svibnju 1995. u samo 10 dana. Njegov stvoritelj je Brendan Eich, ujedno i suosnivač Mozilla projekta. Eich je radio u Netscape-u te je implementirao JavaScript u njihov internetski preglednik Netscape Navigator. JavaScript je mijenjao ime nekoliko puta, prvo kodno ime je bilo Mocha, zatim LiveScript da bi u prosincu 1995. dobio današnji naziv. [17]

JavaScript primarno služi za stvaranje interaktivnih i dinamičnih elemenata web stranica, a ključna značajka ovog jezika je što se izvodi na klijentskoj strani što omogućuje brzo izvršenje i stvaranje korisničko iskustva bez čekanja [5]. JavaScript standardi su ECMAScript Language Specification (ECMA-262) i ECMAScript Internationalization API specification (ECMA-402) koji se redovito ažuriraju [18].

JavaScript je dinamički, slabo tipizirani programski jezik. Dinamički znači da se tipovi varijabli mogu promijeniti tijekom izvođenja, objekti mogu biti prošireni novim svojstvima i metodama, te se novi kod može dinamički generirati i izvršavati. Slabo tipizirani znači da je tip varijabli fleksibilan i da se pretvorba između različitih tipova podataka automatski događa kada je potrebna. Oba svojstva doprinose fleksibilnosti i brzini pisanja koda, ali nedostaci su da se program može nepredvidivo ponašati te je teško pronaći greške.

Pomoću JavaScripta je moguće raditi interaktivne i dinamične stranice jer ima sposobnost upravljanja DOM-om (engl. *Document Object Model*) u stvarnom vremenu. Do velike primjene Javascript-a došlo je jer podržava različite paradigme programiranja, uključujući objektno-orijentirano, funkcionalno, imperativno i tako dalje. Tijekom godina JavaScript se razvio i u platforme poput Node.js što znači da se izvodi na strani poslužitelja, to jest moguće je napraviti cijelu aplikaciju koristeći samo jedan jezik. Osim toga ima i brojne framework-ove i biblioteke kao što su React, Angular, Vue.js koje olakšavaju razvoj kompleksnih aplikacija i poboljšavaju produktivnost programera.

Može se reći da je JavaScript jedan on najvažnijih programski jezika te da je zbog svoje svestranosti, fleksibilnosti, jednostavnosti upotrebe i kontinuiranom razvoju nezamjenjiv u današnjem svijetu.

2.5.1. React

React je popularna JavaScript biblioteka razvijena od strane Facebook-a, a objavljena 29. svibnja 2013. od kada se i neprestano razvija. Prvenstveno je bio korišten za izradu

korisničkih sučelja, posebno aplikacija koje se sastoje od jedne stranice, a danas se koristi za izradu raznovrsni aplikacija, od jednostavnih do vrlo složenih. Neke od najpoznatijih aplikacije napravljene pomoću React-a su: Facebook, Instagram, WhatsApp, Airbnb, Pinterest, Netflix, Wix, Uber Eats, Myntra – Fashion Shopping i Discord.

React koristi virtualni DOM (engl. *Document Object Model*) kako bi što efikasnije ažurirao korisničko sučelje. Umjesto da direktno mijenja stvarni DOM, React stvara virtualnu reprezentaciju i uspoređuje s prethodnom verzijom. Zatim se samo promjene primjenjuju na stvarni DOM što poboljšava performanse. U programskom isječku (*programski isječak 11*) vidljivo je stvaranje virtualnog DOM-a. To se događa u datoteci index.js.

Isječak programskog koda 11*: React virtualni dom*

```
1. const root = ReactDOM.createRoot(document.getElementById('root'));
2. root.render(
3.   <React.StrictMode>
4.     <App />
5.   </React.StrictMode>
6. );
```

Komponente su jedna od ključni značajki React-a. Komponenta je dio korisničkog sučelja i može sadržavati HTML, CSS i JavaScript dijelove. Bitno je da se mogu koristiti više puta i organizirati u hijerarhijsku strukturu. To znači da komponente mogu biti u dijete-roditelj odnosu. To je važno jer React koristi jednosmjerno vezivanje podataka, to jest podaci se prenose od roditeljske komponente prema djetetu. Kada se podaci promijene, React automatski ažurira korisničko sučelje.

Jedna od prednosti React-a je i velika zajednica koja ga koristi. Kada postoji mnogo ljudi koji koriste istu biblioteku, za tu biblioteku postoji obilje resursa i podrške, kao i dokumentacije, edukativnih videa i članaka i slično. Velika zajednica olakšava učenje i razvoj React-a. Svejedno React može biti kompleksan za početnike zbog svojih složenih koncepata. Potrebno je uložiti vrijeme da se svladaju potrebni elementi za stvaranje prve aplikacije.

React se lako integrira s raznim alatima i bibliotekama, kao što su Redux za upravljanje stanjem, React Router za rutiranje i brojni drugi alati za razvoj i testiranje aplikacija.

Osim za stvaranje web aplikacija, React Native omogućuje izgradnju mobilnih aplikacija za iOS i Android koristeći istu biblioteku. Zato se može reći da je React vrlo univeralan.

Naravno, React ima i neke mane. Tako na primjer brza evolucija, to jest česta mijenjanja i ažuriranja mogu predstavljati izazov za razvojne programere da budu u toku s najnovijim funkcijama i promjenama. Budući da je React prvenstveno klijentski rendirana biblioteka, to može dovesti do problema sa SEO (*engl. Search Engine Optimization*) u nekim slučajevima. To znači da stranica napravljena pomoću React-a u tom slučaju neće pojavljivati među prvim rezultatima pri pretraživanju jer joj je potrebno više vremena da se učita. Ovaj problem se najčešće rješava korištenjem SSR-a (engl. *server-side rendering*) s alatima kao što je Node.js. SSR je tehnika rendiranja stranice na strani servera, a ne na klijentskoj strani. U tom slučaju server generira cijelu HTML stranicu s dinamičkim podacima i šalje je klijentu.

HTML je vrlo povezan s React-om. React nije zamjena za HTML, već se koriste zajedno. React koristi JSX (engl. *JavaScript eXtensible Markup Language*) sintaksu koja omogućuje pisanje HTML koda unutar JavaScripta što olakšava izradu korisnički sučelja i njihovu interakciju s ostalim dijelovima aplikacije. Potrebno je napomenuti da neki razvojni programeri ne vole JSX sintaksu koja koristi HTML i JavaScript u istoj datoteci jer smatraju da to krši princip razdvajanja odgovornosti (*engl. Separation of Concerns*). Ovaj princip predlaže da različiti dijelovi sustava trebaju imati jasno definirane uloge i odgovornosti, a cilj je podjela složenih problema na manje, nezavisne komponente koje je lakše razumjeti, testirati i održavati.

U programskom isječku (*programski isječak 12*) vidljiv je cijela komponenta Odjava. Ona se nalazi na većini stranica u projektu. HTML dio smo već vidjeli u programskom isječku 9, a ovdje je vidljivo kako je taj dio HTML koda dio React-a.

Isječak programskog koda 12: Komponenta Odjava

```
export default function Odjava() {
    const Token = localStorage.getItem('token');
    const [username, setUsername] = useState("");

    useEffect(()=>{
      setUsername(jwt_decode(Token).Korisnicko_ime);
    },[]);

    const Navigate = useNavigate();
    const promjeniStranicu = () => {
        Navigate('/prijava');

        localStorage.clear();
      };

    return (
    <>
    <div className="odjava_div">
        <p id="p_odjava">Hvala vam na vašem trudu, {username}.</p>
        <button className="btn btn-outline-danger" id="btn_odjava" onClick={event =>
                promjeniStranicu()}>Odjava</button>
    </div>
    </>
    );
}
```

2.6. CSS

CSS (engl. *Cascading Style Sheet*) je jednostavan mehanizam za dodavanje stila web dokumentu. Razvio ga je W3C (engl. *World Wide Web Consortium*). Ideja je da se odvoji sadržaj stranice od prezentacije, to jest stila kako bi kod bio čišći i kako bi ga bilo lakše za održavati. CSS kod se piše izvan HTML-a te tako poboljšava performanse i omogućava brže učitavanje stranica. Danas većina Internet preglednika i mnoge druge aplikacije podržavaju CSS. Za pisanje CSS koda je potreban samo uređivač teksta, ali postoje alati koji pisanje koda čine jednostavnijim [19].

CSS omogućava precizan izgled elemenata na stranici, uključujući boje, fontove, pozadine, razmak, raspored te čak i animacije i tranzicije. Budući da se stilovi definiraju u zasebnim modulima moguće je jednostavno ažurirati izgled cijele stranice jednostavnim promjenama u CSS datotekama. CSS također omogućava i respozivni dizajn što je ključna komponenta modernog web razvoja, jer osigurava optimalno korisničko iskustvo na različitim uređajima i ekranima.

CSS kod sastoji se od selektora, svojstva i vrijednost. Selektor označava na koje se elemente odnosi dio koda. U programskom isječku (*programski isječak 13*) vidljiv je primjer CSS koda. U isječku smo elementu html i body postavili visinu na 100%, a svim elementima s klasom btn-rozo smo postavili boju pozadine roza, a boju slova bijelu.

Isječak programskog koda 13*: Primjer CSS koda*

```
1. html, body {
2.     height: 100%;
3.   }
4. .btn-rozo{
5.
6.     background-color: #b41e52;
7.     color:white;
8. }
```

2.6.1. Bootstrap

Bootstrap je snažan set alata za jednostavno stvaranje responzivnih web stranica [20]. Razvili su ga Mark Otto i Jacob Torntona u Twitteru, a kasnije je postao projekt otvorenog koda. Usmjeren je na mobilne uređaje i koristi mobile-first pristup, što znači da se prvo razvija za mobilne uređaje, a zatim prilagođava većim ekranima.

Najvažniji koncept u Bootstrapu je korištenje predefinirani klasa koje unaprijed definiraju stilove i funkcionalnosti. Primjer takve klase su klasa *btn* i *btn-outline-danger* čija je upotreba vidljiva u programskom isječku (*programski isječak 14*). Klasa *btn* je predefinirana klasa za izgled gumba, a klasa *btn-outline-danger* će gumbu dati crveni obris i kada se prijeđe pokazivačem preko gumba , gumb će postat cijeli crven. Prednost korištenja Bootstrapa je što se ovaj efekt dobije pisanjem imena samo dvije klase, dok da Bootstrap nije korišten, bilo bi potrebno pisati cijeli CSS kod.

Isječak programskog koda 14*: Korištenje Bootsrtap klasa*

```
1. <button className="btn btn-outline-danger">Odjava</button>
```

Bootstrap koristi sustav mreže (engl. *grid system*) pomoću kojeg se elementi na stranici jednostavno postavljaju u redove i stupce. To se postiže klasama *row* za red i *col* (engl. *column*) za stupce.

Jedna od najvažnijih prednosti Bootstrapa je automatska prilagodba izgleda stranice različitim veličinama ekrana.

Bootstrap sadrži mnoge predefinirane komponente kao što su obrasci, navigacijske trake, iskačući prozori i tako dalje. Svaka komponenta ima svoje klase koje se postavljaju na HTML elemente kako bismo dobili željeni izgled i funkcionalnost.

Uvođenjem Bootstrap framework-a, web developeri mogu postići konzistentnost u dizajnu te uštedjeti vrijeme koje bi inače bilo potrošeno na pisanje osnovnog CSS koda.

Njegova velika popularnost i široka podrška u zajednici dodatno potvrđuju njegovu vrijednost u modernom web razvoju.

3. Aplikacija za izradu dnevnika rada

U ovom poglavlju bit će opisano kako se aplikacija koristi. U aplikaciju se potrebno prvo prijaviti, a nakon toga aplikacija se dijeli na administrativni dio koji je vidljiv samo odgajateljima, i na upise, dio koji je vidljiv i djeci.

3.1. Prijava i registracija

Prvi pogled na aplikaciju je vidljiv na slici 6., a radi se o prijavi u sustav. Pri pokretanju aplikacije potrebno se prijaviti ili registrirati u aplikaciju. Kada se korisnik prijavi/registrira, u memoriju se sprema token koji sadrži ID boravka (ID korisnika). Ako se posjeti neka druga stranica, a token nije zapisan u memoriji, korisnika se preusmjerava nazad na prijavu.

Slika 6. Stranica za prijavu u sustav

Ako korisnik/boravak nema račun u sustavu, potrebno ga je registrirati, a stranica za registraciju vidljiva je na slici 7.

Pri registraciji provjerava se postoji li već netko sa istim korisničkim imenom. Pri stvaranju lozinke moraju biti ispunjeni određeni sigurnosni uvjeti, a tek kada se oni ispune stranica omogućava registraciju.

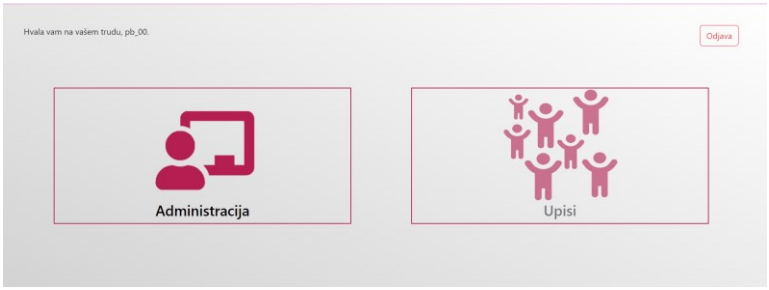
Slika 7. Stranica za registraciju u susta

3.2. Administracija/upisi odabir

Nakon prijave u sustav u gornjem desnom dijelu ekrana vidljiv je gumb koji služi za odjavu korisnika iz sustava. Gornji lijevi dio sadrži poruku i korisničko ime korisnika prijavljenog u sustav. Ovaj red i mogućnost odjave prikazuju se i na stranicama administracije, ali na dnu.

Na slici 8. se može vidjeti kako izgleda početna stranica nakon prijave. Gumb Administracija vodi korisnika na stranice administracije. Ovaj dio je predviđeno imati upaljeno na računalu koji koristi samo odgajatelj/odgajateljica.

Gumb Upisi vodi korisnika na stranice koje su predviđene da djeca pri dolasku označe svoj dolazaka, te je zbog toga predviđen da se pokrene na računalu kojem djeca imaju pristup. Predviđeno je da zamjeni potpisne liste korisnika poludnevnih boravaka.



Slika 8. Odabir između registracije i upisa

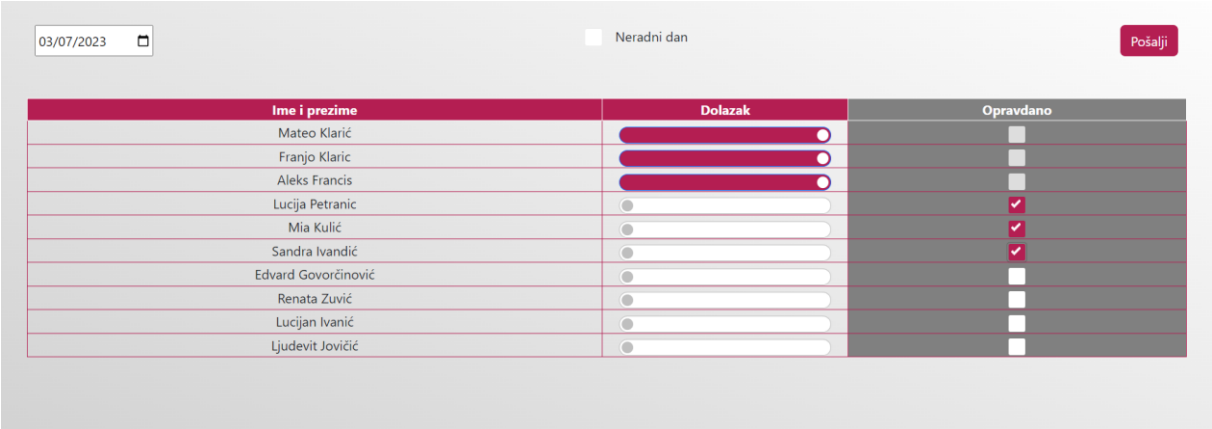
3.3. Upisi

Na slici 9. prikazana je stranica Upisi. U ovom poglavlju bit će objašnjeno kako se koristi ova stranica.

Kada se pokrene stranica Upisi, prvo je potrebno postaviti datum. Trebao bi biti današnji datum, ali program omogućava mijenjanje bilo kojeg dana. Također je moguće označiti da je neki dan neradni, označavanjem check opcije koja se nalazi na vrhu stranice u sredini.

Glavni dio stranice zauzima tablica s popisom korisnika poludnevnog boravka. Pored svakog imena korisnika nalazi se element kojim se može označiti je li korisnik došao u boravak. U trećem stupcu tablice prikazan je "check" koji označava je li nedolazak korisnika opravdan ili neopravdan. Ako je korisnik poludnevnog boravka došao, ovaj "check" je nemoguće označiti što je prikazano sivkastom bojom.

Pritiskom na gumb Pošalji, podaci iz tablice šalju se na server. Potrebno je poslati podatke prije pritiska gumba Osvježi na stranici Dnevnik u administracijskom dijelu.



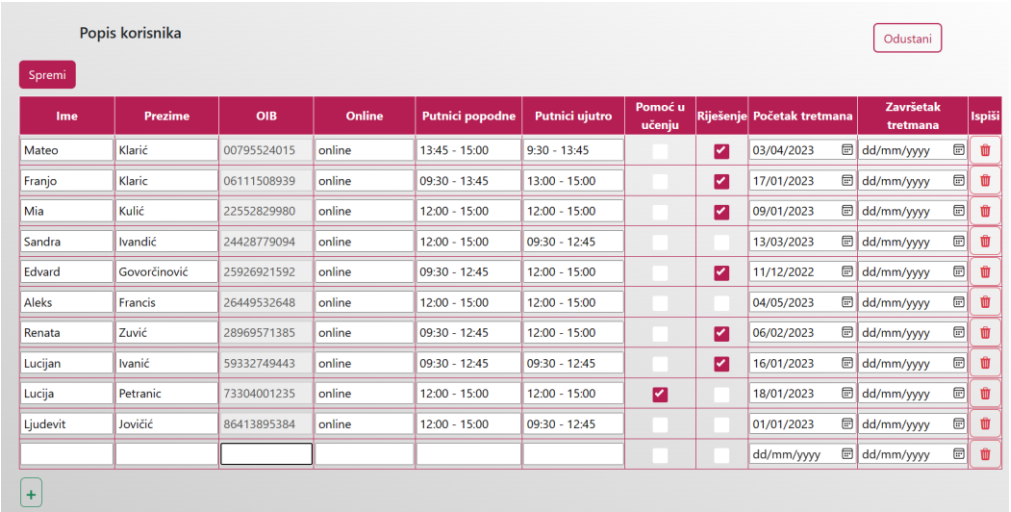
Slika 9. Stranica upisi

3.4. Administracija

3.4.1. Postavke

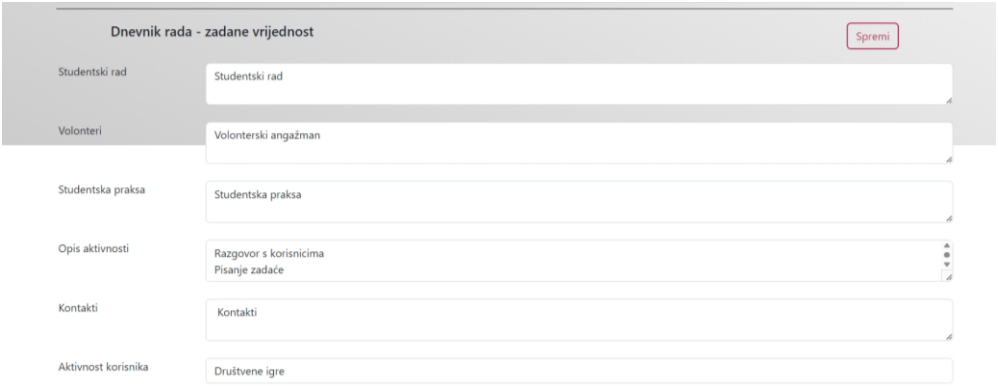
Stranica postavke sastoji se od nekoliko odjeljaka. U ovom poglavlju opisat će se svaki odjeljak zasebno po redu kako se nalaze na stranici.

Prvi odjeljak vidljiv je na slici 10., a to je tablica djece/korisnika poludnevnog boravka. Ovdje se mogu pregledati djeca, postavite zadane vrijednosti za termine u određenom tjednu, označiti je li dijete u boravku kao pomoć u učenju (označiti "check" u stupac pomoć u učenju), označiti je li zaprimljeno rješenje, napisati početak tretmana i završetak tretmana. Moguće je izbrisati djecu i dodati novu. Pritiskom na tipku Spremi spremaju se sve promjene, a pritiskom na tipku Odustani, odustaje se od promjena.



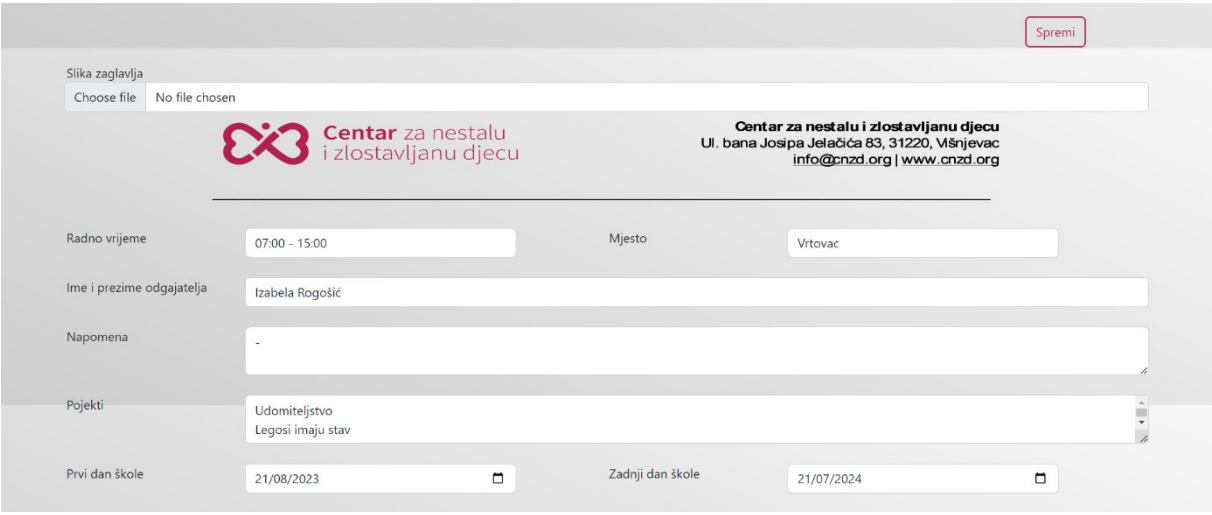
Slika 10. Prikaz tablice - popis korisnika

Idući odjeljak su Zadane vrijednosti dnevnika rada, prvi pododjeljak su zadane vrijednosti koje se mijenjaju na svakom dnevniku rada, a uključuju studentski rad, volonterski angažman, studentska praksa, opis aktivnosti, kontakti i aktivnosti korisnika. Ovaj pododjeljak je prikazan na slici 11. U ovom dijelu se postavljaju zadane vrijednosti, to jest vrijednosti koje će se postaviti prilikom kreiranja svakog dnevnika rada. Njih je moguće mijenjati kod svakog zasebnog dnevnika rada. Kada se promjene ove postavke i nakon što se pritisne gumb Spremi, nove vrijednosti će se postavljati na svaki novostvoreni dnevnik rada, a u starim dnevnicima rada ostat će vrijednosti zadane tom dnevniku rada.



Slika 11. Zadane vrijednosti dnevnika rada

Nakon toga prikazane su također zadane vrijednosti, ali ove su zamišljene da se ne mijenjaju tako često. One su vidljive na slici 12., a uključuju zaglavlje, radno vrijeme boravka, mjesto, ime i prezime odgajatelja, napomenu, projekte, prvi dan škole i zadnji dan škole. Napomenu i projekte je moguće mijenjati za svaki novonastali dnevnik rada.



Slika 12. ostale zadane vrijednosti

Zadnji odjeljak postavki su tjedni, a vidljivi su na slici 13. Ovdje se dodaju nazivi tjedna, moguće je brisati tjedne i dodavati nove. Ovim tjednima se zatim u gornjoj tablici za

svako dijete posebno dodaje termin. Potrebno je postaviti jedan glavni tjedan. U slučaju da prilikom izrade određenog dnevnika rada nije zadan tjedan, uzimaju se termini tjedna koji je označen kao Glavni tjedan.

Slika 13. Prikaz tablice tjedana

3.4.2. Dnevnik rada

Glavni dio aplikacije je stranica Dnevnik rada. Kada se posjeti stranica Dnevnik rada, prvo je potrebno označiti datum.

U prvom redu osim polja za datum vidljiv je link za Pogledaj PDF (engl. *portable document format*), o kojem će biti rečeno nešto više na kraju ovog poglavlja, i stanje.

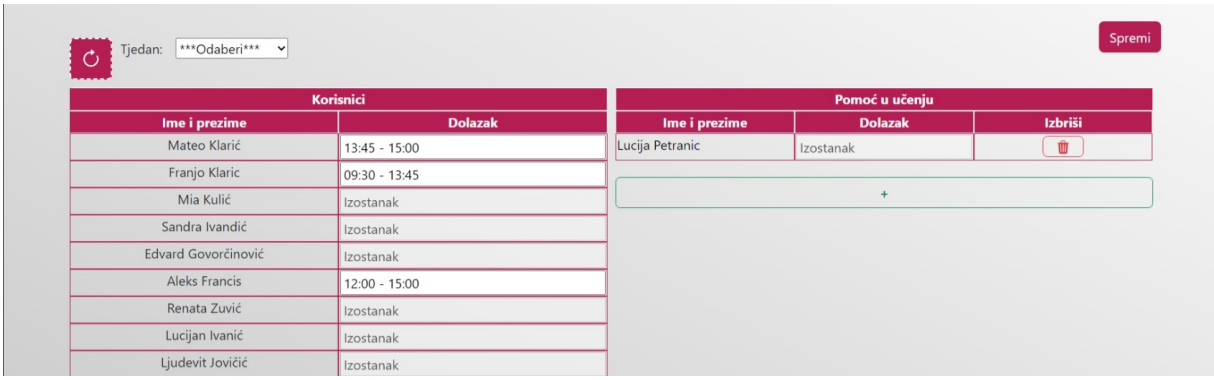
Stanje označava u kojem je stanju dnevnik rada. Ono može biti:

- Novo → novi dnevnik rada, postavlja program
- Upisi → zadnje što se dogodilo su povućeni novi upisi djece, postavlja program
- Uređivanje → dnevnik je u fazi uređivanja, postavlja korisnik
- Gotovo → dnevnik je završen, postavlja program nakon što je isprintano/spremljeno kao pdf
- Neradni → označava neradni dan, postavlja korisnik ili ovdje ili na stranici upisi.

Stanje se uvijek može mijenjati.

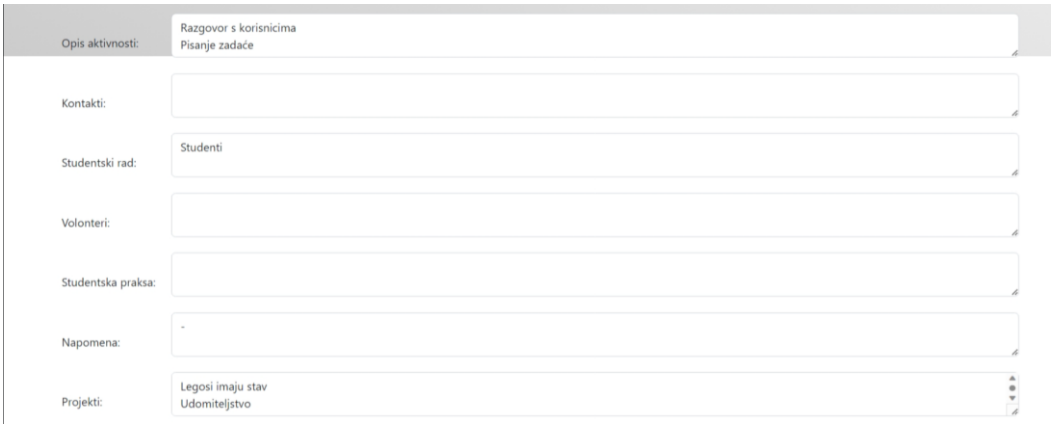
Na slici 14. je vidljiva tablica s popisom djece. Djeca su podijeljena na korisnike i pomoć u učenju. Sva djeca upisana u sustav, a čiji datum početka i završetka tretmana odgovara datumu dnevnika rada su vidljiva u tablici. Ako je došlo još neko dijete u boravak samo taj dan, to jest nije upisano u sustav, moguće je dodati to dijete pritiskom na zeleni gumb sa znakom plus (+) te se to dijete dodaje pod pomoć u učenju. Zatim se dodaje ime i prezime te termin (u stupac dolazak). Zbog tog slučaja moguće je djecu iz tablice pomoć u učenju izbrisati. Kao što je vidljivo na slici, neka djeca u stupcu Dolazak imaju Izostanak, što označava da djeca nisu došla u boravak. Djeci koja su došla u boravak se uzima termin zadan u postavkama, ali je moguće za svaki pojedinačni dnevnik rada urediti termin za svako

dijete. Promjenom tjedna, svi termini se mijenjaju u termine za taj tjedan i novi dolasci također uzimaju termin za novoizabrani tjedan. Pritiskom gornjeg lijevog gumba s ikonom strelice se osvježavaju dolasci djece, to jest provjerava se na serveru jesu li još neka djeca došla u boravak. Gumb Spremi pojavljuje se dva puta na stranici, gore desno i skroz dolje lijevo. Razlog ponavljanja ovog gumba je što se preporuča spremanje dnevnika nakon svake promjene, pogotovo ako se mijenjaju termini ili aktivnosti djece.



Slika 14. Tablica dolazaka djece s terminima

Idući odjeljak je vidljiv na slici 15, a to su vrijednosti dnevnika kao što su: opis aktivnosti, kontakti, studentski rad, volonterski angažman, studentska praksa, napomena i projekti. Unaprijed pišu vrijednosti koje su zadane u postavkama, a ovdje se one mogu mijenjati za svaki poseban dnevnik rada.



Slika 15. Vrijednosti dnevnika rada

Zadnji odjeljak je tablica aktivnosti korisnika vidljiva na slici 16. Sva djeca koja su upisana u sustav i djeca koja su dodatno došla su upisana u ovu tablicu. Svoj djeci koja su došla u boravak prvo se zapisuje aktivnost koja je definirana u postavkama pod Aktivnost korisnika. Ta vrijednost je zadana vrijednost te je nakošena zbog bolje preglednosti. Aktivnosti se mogu mijenjati kod djece koja su došla u boravak. Djeca koja nisu došla u boravak pod aktivnost imaju " – " i njihova imena su malo izbijeljena. Toj djeci nije moguće

dodati aktivnost. Ovdje je vidljiv i drugi Spremi gumb koji šalje dnevnik na server, to jest sprema ga u bazu.

Ime i prezime	Aktivnosti
Mateo Klarić	*Društvene igre*
Franjo Klaric	MAT - pisano dijeljenje
Mia Kulić	-
Sandra Ivandić	-
Edvard Govorčinović	-
Aleks Francis	*Društvene igre*
Renata Zuvić	-
Lucijan Ivanić	-
Lucija Petranic	-
Ljudevit Jovičić	-

Spremi

Slika 16. Tablica aktivnosti korisnika

Na početku poglavlja spomenut je link pogledaj PDF. Pritiskom na taj link dolazi se na stranicu na kojoj se može pregledati dnevnik rada, a primjer je vidljiv na slici 17.

Isprintaj

DATUM: 03. 07. 2023.
ponedjeljak

RADNO VRIJEME: 07:00 - 15:00
Vrtovac

IME I PREZIME: Izabela Rogošić

NAPOMENA: -

POPIS DJECE:

				Pomoć u učenju	
Mia Kulić	Izostanak	Ljudevit Jovičić	Izostanak	Lucija Petranic	Izostanak
Franjo Klaric	09:30 - 13:45	Sandra Ivandić	Izostanak		
Lucijan Ivanić	Izostanak	Mateo Klarić	13:45 - 15:00		
Aleks Francis	12:00 - 15:00	Edvard Govorčinović	Izostanak		
Renata Zuvić	Izostanak				

Studentski rad:
Studenti

Opis aktivnosti:
Razgovor s korisnicima
Pisanje zadaće

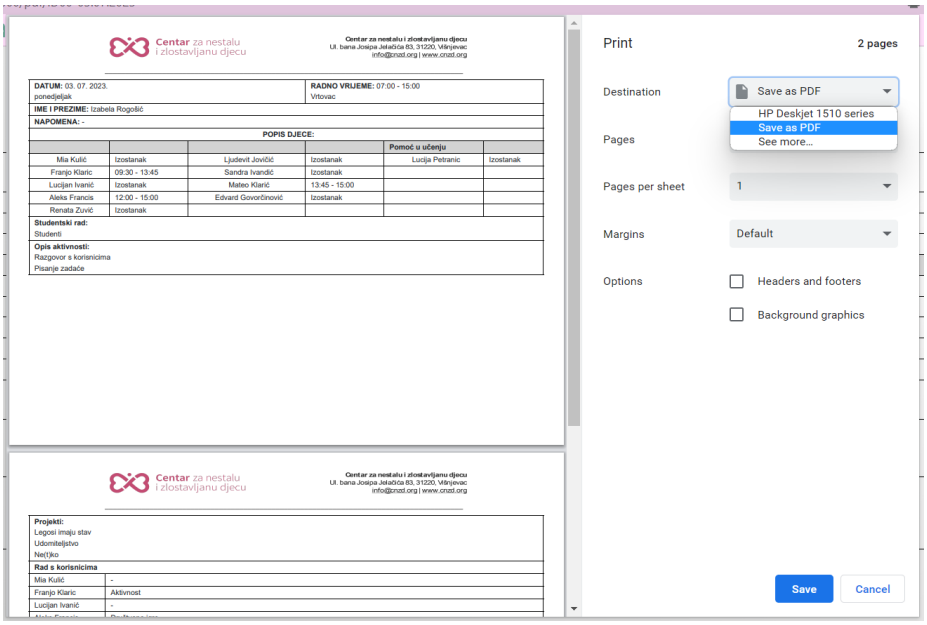
Projekti:
Legosi imaju stav
Udomiteljstvo
Ne(t)ko

Rad s korisnicima	
Mia Kulić	-
Franjo Klaric	Aktivnost
Lucijan Ivanić	-
Aleks Francis	Društvene igre
Renata Zuvić	-
Ljudevit Jovičić	-
Sandra Ivandić	-
Mateo Klarić	Društvene igre
Lucija Petranic	-
Edvard Govorčinović	-

Potpis odgajate

Slika 17. Stranica s izgledom PDF-a

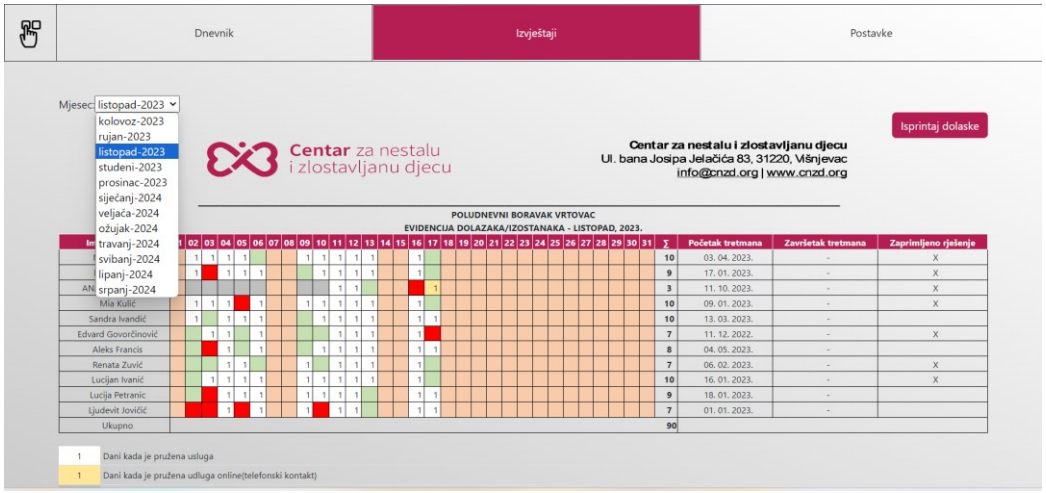
Pritiskom na gumb Isprintaj u sredini gore, uključuje se dijalog u kojem se zatim dnevnik rada može spremiti u PDF obliku i isprintati. Nakon zatvaranja ovog dijaloga se dnevnik označava kao gotov. Primjer dijaloga nalazi se na slici 18.



Slika 18. Prozor dijaloga za ispis

3.4.3. Mjesečni izvještaj dolazaka/izostanaka

Na stranici Izvještaji nalazi se evidencija dolazaka i izostanaka. Ona je vidljiva na slici 19. Gore lijevo se nalazi element u kojem odabiremo za koji mjesec i godinu trebamo izvještaj. Ovdje se nalaze svi mjeseci koji su između navedenih početka i završetka školske godine (namješteno u postavkama). Zatim je vidljiva slika zaglavlja i naziv evidencije. Mjesto i vrijeme se mijenjaju ovisno o poludnevnom boravku i odabranom mjesecu.



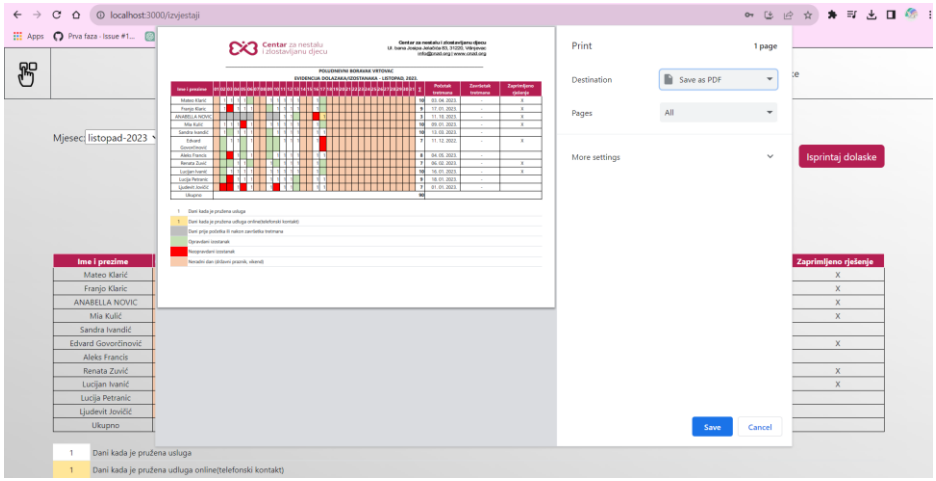
Slika 19. Izgled stranice za izvješća

Zatim se nalazi tablica. U tablici su nabrojani korisnici koji su upisani u poludnevni boravak. Za svakog korisnika navedeno je po danu je li mu pružena usluga, je li mu usluga pružena online i tako dalje prema legendi na slici 20. Bitno je napomenuti da bi djetetu bila pružena usluga online u dnevniku rada pod termin treba pisati online.

1	Dani kada je pružena usluga
1	Dani kada je pružena udluga online(telefonski kontakt)
	Dani prije početka ili nakon završetka tretmana
	Opravdani izostanak
	Neopravdani izostanak
	Neradni dan (državni praznik, vikend)

Slika 20. Legenda za tablicu evidencija dolazaka/izostanaka

Pritiskom na gumb Isprintaj dolaske u gornjem desnom uglu otvara nam se prozor gdje je vidljivo kako izgleda dokument te ga je moguće ispisati ili spremiti u PDF obliku. Primjer ovog dijaloga vidljiv je na slici 21.



Slika 21. Prikaz prozora za pregled, printanje i spremanje tablice dolazaka/izostanaka

3.5. Svakodnevno korištenje programa

Kako je i prije rečeno, dnevnik je rađen s motivacijom digitalizacije i ubrzavanja rada poludnevnih boravaka koji djeluju u sklopu Centra za nestalu i zlostavljanu djecu.

Nakon registracije potrebno je otići na postavke i unijeti sve vrijednosti.

Program je zamišljen da se koristi pomoću dva računala, ali ga je naravno moguće prilagoditi i jednom. Zamišljeno je da se odgajatelj/odgajateljica prijavi na svoje računalo, kojemu samo on/ona ima pristup i otvori stranicu Administracija te zatim stranicu Dnevnik rada i unese današnji datum. Nakon toga se prijavi na računalo kojem i djeca imaju pristup (može biti i tablet ili nešto slično) i pokrene stranicu Upisi, te unese današnji datum. Kako koje dijete dolazi ono se označi. Kada odgajatelj/odgajateljica odluči može pritisnuti gumb Pošalji na stranici Upisi. Nakon toga pritisne gumb Osvježi na svojem računalu i djeci koja su se do tada upisala se dodaje termin i aktivnost. Ako tjedan nije dobar potrebno je postaviti i tjedan. Kako djeca rade nešto, moguće je odmah to upisati u tablicu aktivnosti. Preporuča

se spremiti dnevnik čim se u njemu nešto promjeni (pogotovo kod termina i aktivnosti korisnika), a obavezno ga je potrebno spremiti prije nego li se ponovno pritisne tipka Pošalji na stranici Upisi. U slučaju da se pritisne tipka Pošalji i Osvježi, prije nego li se spreme promjene, promjene će bit kao da se nisu ni dogodile, odnosno ponovno će se upisati zadane vrijednosti za aktivnosti korisnika i termine korisnika.

Ideja programa je da stranica Upisi zamjeni potpisnu listu, a stranica Dnevnik zamjeni bilježnicu u koju su do sada odgajatelji/odgajateljice pisali sve svoje bilješke vezane za dnevnik rada.

Na kraju radnog dana, kada se sve upiše i spremi, klikne se na link pogledaj PDF i ako sve izgleda u redu, pritisne se na Isprintaj. Nakon toga je moguće odmah isprintati dnevnik rada ili ga spremiti u PDF obliku i isprintati ga kasnije.

Kada je potrebno napraviti mjesečni izvještaj, odlazi se na stranicu Izvještaji, odabire se željeni mjesec i izvještaj će biti spreman. Potrebno ga je samo isprintati ili spremiti u PDF obliku.

4. Zaključak

Pri stvaranju aplikacije Dnevnik rada korištena su mnoga znanja stečena na različitim kolegijima, a posebno na programiranje u .NET okolini, web programiranje na strani poslužitelja i projektiranje informacijskih sustava. Na zadnje navedenom je i započeto stvaranje aplikacije. Uz to bilo je potrebno dodatno istraživanje i učenje novih stvari kako bi se projekt realizirao. Poseban naglasak je bio stavljen na ASP.NET Core i React.

Osim novih tehničkih znanja pri izradi projekta došlo je do razvoja mekih vještina (engl. *soft skills*) jer je aplikacija nastala u suradnji s Centrom za nestalu i zlostavljanu djecu te je rađena s pravim klijentom u mislima te su klijenti iznosili svoja mišljenja i potrebe za vrijeme stvaranja aplikacije. Neke od mekih vještina koje su bile potrebne za ovaj projekt su pregovaranje, pisana komunikacija, primanje povratni informacija, kreativnost i prilagodljivost. Također budući da je aplikaciju moguće koristiti u stvarnom svijetu već neko vrijeme, bilo je potrebno odvojiti produkciju i razvoj aplikacije te popravljati greške na koje su nailazili stvarni korisnici aplikacije.

5. Popis literature

Knjige:

[1] Carić T. i Buntić M.: Uvod u relacijske baze podataka, Sveučilište u Zagrebu, Zagreb, 2015.

[2] Bipin C. Desai: An Introduction to Database System, Montreal, Canada, 1990.

[3] Svetlin Nakov & Co.: Fundamentals of Computer Programming with C#, Sofia, 2013.

[4] Thomas A. Powell: The Complete Reference HTML & CSS, New York, 2010.

[5] Dr. Axel Rauscmayer: JavaScript For Impatient Programmers, 2022.

Članci u časopisima:

[6] E. F. Codd (srpanj 1970.): A Relational Model of Data for Large Shared Data Banks.
Časopis Communications of the ACM, Vol. 13., br. 6., str. 377-387

Internetski izvori:

[7] TechTarget, Microsoft SQL Server, Pristupljeno: 12. kolovoz 2023., https://www.techtarget.com/searchdatamanagement/definition/SQL-Server

[8] informIT, Anders Hejlsberg, Pristupljeno: 14. studenog 2023., https://www.informit.com/authors/bio/eef7a7d9-cd27-450e-90ae-aa39474a1b6e

[9] C# The modern, innovative, open-source programming language for building all your apps, Pristupljeno: 14. studenog 2023., https://dotnet.microsoft.com/en-us/languages/csharp

[10] What is .NET Framework, Pristupljeno: 25.studenog 2023., https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet-framework

[11] ASP.NET MVC Overview, Pristupljeno: 13. ožujka 2024., https://learn.microsoft.com/en-us/aspnet/mvc/overview/older-versions-1/overview/asp-net-mvc-overview

[12] ASP.NET MVC Architecture, Pristupljeno:13. ožujka 2024., https://www.tutorialsteacher.com/mvc/mvc-architecture

[13] Entity Framework, Pristupljeno: 24. travnja 2024., https://learn.microsoft.com/en-us/aspnet/entity-framework

[14] What is Entity Framework? Pristupljeno: 24. travnja 2024., https://www.entityframeworktutorial.net/entityframework6/what-is-entityframework.aspx

[15] Uvod u HTML, Pristupljeno: 25. travnja 2024., https://tesla.carnet.hr/mod/book/view.php?id=5430&chapterid=885

[16] A Brief History of HTML, Pristupljeno: 25. travnja 2024., https://www.washington.edu/accesscomputing/webd2/student/unit1/module3/html_history.html#:~:text=The%20first%20version%20of%20HTML,HTML%20as%20an%20XML%20language./

[17] Javascript - osnove, Pristupljeno: 21. svibnja 2024., https://programiranje.com.hr/javascript/osnove

[18] Javascript, Pristupljeno: 21. svibnja 2024., https://developer.mozilla.org/en-US/docs/Web/JavaScript

[19] CSS software, Pristupljeno: 22. svibnja 2024., https://www.w3.org/Style/CSS/software

[20] Build fast, responsive sites with Bootstrap, Pristupljeno: 22. svibnja 2024. https://getbootstrap.com/

6. Popis slika

Veleučilište u Virovitici

OBRAZAC 5

IZJAVA O AUTORSTVU

Ja, HELENA JERBIĆ

izjavljujem da sam autor/ica završnog/diplomskog rada pod nazivom

IZRADA APLIKACIJE ZA IZRADU DNEVNIKA RADA KORISTEĆI PROGRAMSKE TEHNOLOGIJE ASP.NET CORE I REACT

Svojim vlastoručnim potpisom jamčim sljedeće:

- da je predani završni/diplomski rad isključivo rezultat mog vlastitog rada koji se temelji na mojim istraživanjima i oslanja se na objavljenu literaturu, a što pokazuju korištene bilješke i bibliografija,
- da su radovi i mišljenja drugih autora/ica, koje sam u svom radu koristio/la, jasno navedeni i označeni u tekstu te u popisu literature,
- da sam u radu poštivao/la pravila znanstvenog i akademskog rada.

Potpis studenta/ice

Jerbić H

Veleučilište u Virovitici

OBRAZAC 6

ODOBRENJE ZA OBJAVLJIVANJE ZAVRŠNOG/DIPLOMSKOG RADA U DIGITALNOM REPOZITORIJU

Ja, HELENA JERBIĆ

dajem odobrenje za objavljivanje mog autorskog završnog/diplomskog rada u nacionalnom repozitoriju odnosno repozitoriju Veleučilišta u Virovitici u roku od 30 dana od dana obrane.

Potvrđujem da je za pohranu dostavljena završna verzija obranjenog završnog/diplomskog rada.

Ovom izjavom, kao autor navedenog rada dajem odobrenje i da se moj rad, bez naknade, trajno javno objavi i besplatno učini dostupnim na sljedeći način (zaokružiti):

(a) Rad u otvorenom pristupu
b) Rad dostupan nakon: _____ (upisati datum)
c) Pristup svim korisnicima iz sustava znanosti i visokog obrazovanja RH
d) Pristup korisnicima matične ustanove
e) Rad nije dostupan (*u slučaju potrebe dodatnog ograničavanja pristupa Vašem završnom/diplomskom radu, podnosi se pisani obrazloženi zahtjev*).

U slučaju dostupnosti rada prethodno označeno od a) do d), ovom izjavom dajem pravo iskorištavanja mog ocjenskog rada kao autorskog djela pod uvjetima Creative Commons licencije (zaokružiti):

1) CC BY (Imenovanje)
2) CC BY-SA (Imenovanje – Dijeli pod istim uvjetima)
3) CC BY-ND (Imenovanje – Bez prerada)
4) CC BY-NC (Imenovanje – Nekomercijalno)
5) CC BY-NC-SA (Imenovanje – Nekomercijalno – Dijeli pod istim uvjetima)
(6)) CC BY-NC-ND (Imenovanje – Nekomercijalno – Bez prerada)

Ovime potvrđujem da mi je prilikom potpisivanja ove izjave pravni tekst licencija bio dostupan te da sam upoznat s uvjetima pod kojim dajem pravo iskorištavanja navedenog djela.

Potpis studenta/ice

jerbić H