

**BOSNA I HERCEGOVINA  
FEDERACIJA BiH  
UNSKO-SANSKI KANTON  
JU GIMNAZIJA „BIHAĆ“**

**MATURSKI RAD IZ PREDMETA WEB TEHNOLOGIJE  
INTERAKTIVNI MATERIJAL ZA UČENJE C++ PROGRAMSKOG JEZIKA**

**Mentor: prof. Muharem Mušić**

**Učenik: Mirza Mahmutović**

**Bihać, maj 2022.**

## SADRŽAJ

<b>1. UVOD .....</b>	<b>1</b>
<b>2. MATERIJAL ZA UČENJE C++ PROGRAMSKOG JEZIKA.....</b>	<b>2</b>
2.1 UVODNA OBLAST.....	2
2.2 RAZGRANATA STRUKTURA.....	3
2.3 CIKLIČNA STRUKTURA .....	5
2.4 FUNKCIJE.....	6
2.5 NIZOVI.....	7
<b>3. IZRADA WEB APLIKACIJE.....</b>	<b>9</b>
3.1 BAZA PODATAKA.....	9
3.2 WEB STRANICA.....	11
<b>4. ZAKLJUČAK .....</b>	<b>58</b>
<b>LITERATURA.....</b>	<b>59</b>

## 1. UVOD

Živimo u svijetu tehnologije, okruženi smo računarima i pametnim uređajima. Sve te uređaje koje koristimo na svakodnevnoj bazi, morao je jednim dijelom neko da isprogramira. Koristimo razne mobilne aplikacije, web aplikacije, računarske programe, video igrice koji su zasnovani na programiranju i izrađeni u jednom od programskih jezika današnjice.

Upravo zbog toga je programiranje jedna od najtraženijih vještina našeg vremena. I svako ko je programirao, programira ili planira programirati sigurno je čuo za C++ programski jezik i njegov utjecaj u svijetu tehnologije i računara.

C++ je programski jezik opće namjene i srednje razine s podrškom za objektno orijentisano programiranje. Pripada porodici C programskih jezika, a neosporna je njegova dominacija u svijetu video igrica. C++ je programski jezik koji se koncentrira na detalje i kao takav je izuzetno učinkovit i prilagodljiv programski jezik. Specifičan je po svojoj brzini izvođenja i minimalnoj količini memorije koju programi napisani u njemu zauzimaju. Samim tim, omogućava računarima da optimalno koriste svoju procesorsku snagu i memorijske kapacitete.

Uzimajući u obzir sve njegove karakteristike i prednosti u odnosu na druge programske jezike možemo shvatiti zašto se kao programski jezik toliko razvio i postao jedan od najpoznatijih. Iz svih tih razloga se i obrađuje u mnogim, kako srednjim, pa tako i u osnovnim školama.

Projekat kreiran u ovom radu nastoji olakšati učenje C++ programskog jezika učenicima, ali ujedno i olakšati način predavanja profesorima. Učenici imaju mogućnost pristupa, putem korisničkog naloga, nastavnom materijalu, objavama profesora, te događajima vezanih za njihov razred. Ujedno profesorima je olakšana kontrola nastavnog materijala, komunikacija sa učenicima koja se može vršiti putem same web aplikacije.

## 2. MATERIJAL ZA UČENJE C++ PROGRAMSKOG JEZIKA

Materijal za učenje C++ programskog jezika se sastoji od pet oblasti, u kojem svaka oblast sadrži određen broj lekcija. Svaka lekcija ima svoj video, u kojem se objašnjava kroz pisanje koda, zatim tekstualno objašnjenje, savjet i određen broj primjera. Oblasti sadržane u materijalu su:

1. *Uvod*
2. *Razgranata struktura*
3. *Ciklična struktura*
4. *Funkcije*
5. *Nizovi*

### 2.1 Uvodna oblast

Uvodna oblast sadrži četiri zasebne lekcije, a to su:

1. *Instalacija i Postavljanje DEV-C++*
2. *Struktura C++*
3. *Varijable i Konstante*
4. *Osnovni Operatori*

Kao i što sami naslovi lekcija govore, uvodna oblast se prije svega bavi instalacijom i pokretanjem razvojnom okruženja DEV-C++ koje je korišteno u cjelokupnom materijalu.

Nakon toga dolazi lekcija *Struktura C++*, u čijem se videu piše prvi program i objašnjavaju osnove C++ programskog jezika. Govori se o upotrebi datoteka, gdje se u svakom programu uključuje iostream datoteka linijom koda `#include<iostream>`, zatim biblioteka, gdje koristimo standardnu biblioteku linijom koda `using namespace std;`, main funkcije, odnosno glavne funkcije programa, zatim važnosti tačke-zarez na kraju svake izjave kao i o jednolinijskim - `//` i višelinijskim - `/**/` komentarima koji se mogu koristiti prilikom programiranja. Pored toga lekcija sadrži dva primjera sa objašnjenjem u kojima se detaljnije objašnjavaju već spomenute stvari.

Sljedeća lekcija unutar ove oblasti jeste *Varijable i Konstante* gdje se upoznajemo sa pojmom varijable(promjenjive). U videu su opisani različiti tipovi varijabli koje možemo susresti u C++ programskom jeziku, a to uključuje `int`(cjelobrojni tip podatka), `double`(tip podatka sa decimalom), `float`(tip podatka sa decimalom), `char`(jedan karakter)... Prvi sporedni primjer sadrži tabelu tih istih varijabli u kojoj piše njihova veličina i opseg vrijednosti. Drugi primjer je praktična primjena varijabli u programu u kojem se računa razlika dvije varijable i sprema u treću. U trećem primjeru je prikazana inicijalizacija varijabli, gdje se pri samoj deklaraciji odmah dodijeli i vrijednost varijabli. Četvrti primjer prikazuje poseban tip podatka – string. Stringovi su tekstualni tipovi podatka koji služe za spremanje riječi, više riječi, rečenice ili više rečenica. Na kraju, u zadnjem primjeru objašnjen je pojam konstante, kad se koriste i kako se inače nazivaju.

Lekcija *Osnovni Operatori* nas upoznaje sa aritmetičkim operatorima `+`, `-`, `*`, `/`, te sa načinom na koji funkcionišu. Ovdje se ističe operator dijeljenja, čiji rezultat zavisi od tipa podataka koji učestvuju u dijeljenju. Također nije moguće dijeliti sa 0, pa pokušaj toga će rezultirati sa greškom u programu. U primjerima su objašnjeni operator `sizeof()` – vraća veličinu promjenjive, redoslijed operatora `*`, `/` i `%`(modulo) imaju prednost u odnosu na `+` i `-`, operatori za unos `<<` i `>>`, koji u kombinaciji sa ključnim riječima `cin` i `cout` omogućavaju unos podataka sa tastature i ispis podataka u konzolu. Razlika je kod podataka tipa string, gdje se za unos koristi funkcija `getline()`.

## 2.2 Razgranata Struktura

Kako u ostalim programskim jezicima, pa tako i u C++ postoje različite programske strukture. Programi koji su pisani u prvoj oblasti su programi linijske strukture, gdje se pri svakom pokretanju uvijek izvrši ista radnja neovisno o unosu korisnika. Korištenjem *if else* iskaza i *switch* iskaza omogućujemo programu različit tijek, ovisno o rezultatu postavljenog uslova. Ovo je vrlo važna struktura bez koje bi mogućnost rješavanja zadataka računarom bila vrlo ograničena. Sadržane lekcije unutar oblasti Razgranata Struktura su:

1. *If Else*
2. *Logički Izrazi*
3. *Switch Iskaz*
4. *Matematičke Funkcije*

## 5. Funkcije Za Rad Sa Stringovima

U prvoj lekciji se upoznajemo sa `if else` izjavom koja bira između dvije ili više alternativnih akcija, zavisno od logičkog izraza unutar zagrade. U videu je objašnjena upotreba ove izjave kroz praktični primjer, a zatim imamo dva sporedna primjera u lekciji. Prvi sporedni primjer govori o `if-else-if` izjavi gdje imamo više uslova za provjeru, pa se na osnovu unešene vanjske temperature ispisuje određena poruka. Drugi primjer računa kvadratnu jednačinu gdje se provjerava rezultat determinante, pa na osnovu toga i uređuje ispis i računa rezultat. U ovom primjeru u kodu fali jedna ključna riječ, pa se od korisnika traži da unese potrebnu riječ da bi kod bio ispravan.

Druga lekcija nas upoznaje sa logičkim izrazima koje možemo pronaći pri korištenju `if else` izjave. Objašnjeni su relacijski operatori(`==`, `!=`, `<`, `>`, `<=`, `>=`) te način na koji funkcionišu. Zatim, objašnjeni su i logički operatori(`&&`, `||` ili `and`, `or`). Za kraj, objašnjen je primjer u kojem se kombinuju i logički i relacijski operatori.

U sljedećoj lekciji se upoznajemo sa `switch` iskazom. `Switch` iskaz funkcionise na način da određena varijabla prolazi kroz `switch` i provjerava se da li se podudara sa jednom od mogućnosti sa popisa. Ukoliko se podudara, izvršava se kod koji dolazi nakon tog slučaja sve do ključne riječi *break*. Ukoliko se vrijednost ne podudara ni sa jednom od mogućnosti iz `switch` iskaza, izvršava se kod koji se nalazi iza ključne riječi *default*, ukoliko je postavljena. Prikazana su dva sporedna primjera, gdje se u prvom provjerava unešena operacija, te na osnovu toga vrši računanje, a u drugom se provjerava da li je unešeno slovo samoglasnik ili suglasnik.

Lekcija *Matematičke Funkcije* nije nužno vezana za ragranatu strukturu, ali kako se pišu kompleksniji programi dolazi i potreba da se koriste određene funkcije. Objašnjene su funkcije za stepenovanje i korjenovanje(`pow()` i `sqrt()`), zatim logaritamske funkcije(`log()` i `log10()`), te ostale funkcije kao što su `ceil()` – funkcija za pronalazak najmanjeg cijelog broja većeg od zadanog, `floor()` – funkcija za pronalazak najvećeg cijelog broja manjeg od zadanog, `fabs()` – funkcija za računanje apsolutne vrijednosti i `fmod()` – funkcija za računanje ostatka pri dijeljenju dva broja tipa `float`.

Zadnja lekcija u ovoj oblasti, *Funkcije Za Rad Sa Stringovima* je u ovoj oblasti uključena iz istog razloga iz kojeg je uključena lekcija prije nje. Ovdje je detaljnije objašnjena već prije

spomenuta funkcija za unos stringa – *getline()*, zatim objašnjene su funkcije za uređivanje stringa, kao što su *replace()* – funkcija koja određeni dio stringa zamjenjuje sa drugim, *erase()* – funkcija koja briše određeni dio stringa, *clear()* – funkcija koja briše sav string, *toupper()* – funkcija koja određeni karakter unutar stringa pretvara u veliko slovo, *tolower()* – funkcija koja određeni karakter unutar stringa pretvara u malo slovo. Za kraj objašnjene su funkcije za provjeravanje veličine stringa, gdje imamo funkcije *size()* i *length()* koje vraćaju dužinu stringa i funkciju *empty()* koja provjerava da li je string prazan ili ne.

## 2.3 Ciklična Struktura

Pored linijske i razgranate postoji još i ciklična struktura. Ona je vrlo korisna ukoliko je u programu potrebno određenu radnju ponoviti više puta. Pa da se ne bi isti dio koda pisao više puta, koristimo petlje. Lekcije sadržane u ovoj oblasti su:

1. *For Petlja*
2. *While Petlja*
3. *Do While Petlja*

For petlja je najčešće korištena petlja u ovom programskom jeziku pa je iz tog razloga ona i prva petlja koja se uči u ovom materijalu. Ona se obično koristi kad kroz neku cjelobrojnu varijablu treba proći više puta jednakim koracima. Počinje ključnom riječi *for*, koju unutar zagrada slijede tri uslova koji kompajleru govore šta da radi sa kontrolnom varijablom. U videu je objašnjena *for* petlja kroz praktičan primjer, a pored toga u lekciji su sadržana tri sporedna primjera. Prvi primjer služi za sumu prvih deset prirodnih brojeva. Drugi primjer koristi *if* uslov unutar same *for* petlje, a u trećem primjeru nedostaje jedan dio koda, pa se od korisnika traži unos potrebnog dijela da kod bude ispravan.

*While* petlja provjerava da li je neki uslov ispunjen, pa ako jeste ide se u iteraciju, odnosno, ponavljanje tijela petlje. Lekcija sadrži tri primjera. Prvi primjer računa sumu prvih deset prirodnih brojeva. Drugi primjer koristi *if* uslov unutar petlje, a u trećem primjeru fali ključna riječ, pa se od korisnika traži unos.

Treća, i zadnja, lekcija jeste do while petlja koja se od while petlje razlikuje po tome što prvo ulazi u tijelo petlje, pa se tek onda provjerava uslov same petlje. I ova lekcija sadrži tri primjera. Prvi primjer računa sumu prvih deset prirodnih brojeva. Drugi primjer koristi if uslov unutar same petlje, a u trećem uslovu fali dio koda, pa se od korisnika traži da unese to što fali.

U sve tri lekcije se koriste slični primjeri upravo iz razloga da bi se istakla sličnost ove tri petlje koje nam C++ pruža. Veliki broj zadataka se može uraditi u sve tri petlje, pa je izbor petlje koja će se koristiti ličan.

## 2.4 Funkcije

U ovoj oblasti u materijalu se napokon upoznajemo sa funkcijama, koje predstavljaju temelj strukturnog programiranja. Cilj ove oblasti jeste da se shvati koliko funkcije olakšavaju pisanje, korekciju i održavanje programa. Lekcije sadržane u ovoj oblasti su:

1. *Uvod u Funkcije*
2. *Tipovi Funkcija*
3. *Argumenti i Parametri*
4. *Lokalne i Vanjske Varijable*

U lekciji Uvod u Funkcije su objašnjeni main funkcija, a zatim i deklaracija, definicija i poziv spostvene funkcije. Prvi sporedni primjer nam govori o deklaraciji(prototipu) funkcije. Tu je potrebno navesti povratni tip funkcije, ime funkcije, tip parametara i nazive parametara. Drugi nam govori o definiciji funkcije, gdje pored zaglavlja funkcije imamo i tijelo funkcije. Tijelo funkcije govori šta funkcija radi te može imati *return* izraz zavisno od tipa funkcije. I zadnji primjer nam govori o pozivu funkcije, gdje je potrebno navesti ime funkcije, te unijeti parametre ukoliko ih funkcija zahtjeva.

Funkcija može vratiti jednu vrijednost, i to tipa kojeg je i sama funkcija, ukoliko je to funkcija sa vraćenom vrijednosti. Takve funkcije u svom tijelu sadrže ključnu riječ *return* za vraćanje vrijednosti. Funkcija koja ne vraća nikakvu vrijednost je *void* funkcija. Ova lekcija sadrži



dva sporedna primjera u kojima su dodatno objašnjene *void* funkcije i funkcije sa vraćenom vrijednosti.

Funkcija može da ima svoje parametre. Ono što razlikujemo kod funkcija su argumenti i parametri. Parametri predstavljaju formalne parametre funkcije i oni se uvijek nalaze u zaglavlju ili prototipu funkcije. Argumenti predstavljaju aktuelne(stvarne) parametre i oni se pojavljuju prilikom poziva funkcije. Unutar lekcije su sadržana još dva primjera da bi se bolje shvatila razlika između ta dva pojma.

Zadnja lekcija ove oblasti se dotiče lokalnih i vanjskih varijabli. Lokalne varijable su varijable koje su definisane unutar funkcije, pa su zbog toga i vidljive samo unutar te funkcije. Vanjske varijable, za razliku od lokalnih, su varijable koje su definisane izvan svih funkcija, pa su i vidljive unutar svih funkcija. Oba tipa varijabli su automatski promjenjive varijable. To znači da se lokalne varijable pri završetku funkcije brišu, a vanjske varijable se brišu pri završetku programa. Međutim dodavanjem ključne riječi *static* ispred deklaracije lokalne varijable, ona poprima karakteristiku vanjske varijable pa traje sve do završetka programa kad se briše.

## 2.5 Nizovi

Zadnja oblast u materijalu za učenje C++ govori o nizovima. Nizovi nam omogućavaju da predstavimo cijelu skupinu vrijednost, za razliku od varijabli gdje možemo spremiti samo pojedinačnu vrijednost. Lekcije sadržane unutar ove oblasti su:

1. *Jednodimenzionalni Nizovi*
2. *Dvodimenzionalni Nizovi*
3. *Sortiranje Nizova*

Iz priloženih lekcija možemo vidjeti da u C++ programskom jeziku razlikujemo dva tipa nizova, a to su jednodimenzionalni i dvodimenzionalni.

U prvoj lekciji se upoznajemo sa prvom vrstom nizova. Upoznajemo se sa elementima niza i indeksima tih elemenata. Elementi niza moraju biti istog tipa podatka kao cjelokupan niz, a njima

pristupamo pomoću indeksa. Indeksi kod nizova kreću od 0, pa tako ako imamo niz od pet elemenata, njihovi indeksi će biti 0,1,2,3,4. Uz dva sporedna primjera bolje je prikazano korištenje nizova sa for petljom, te inicijalizacija niza.

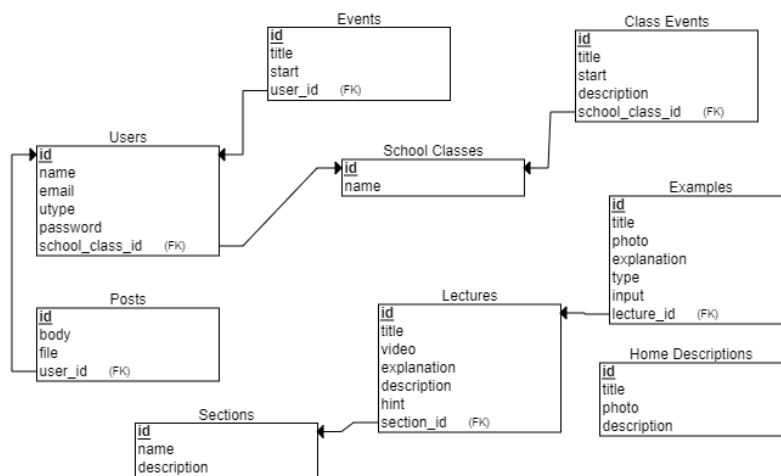
Dvodimenzionalni nizovi(matrice) imaju dva indeksa. Odnosno, svaki element dvodimenzionalnog niza ima svoj red i svoju kolonu. Da bismo koristili for petlju uz matrice, moramo koristiti ugniježdenu for petlju, pri čemu vanjska for petlja prolazi kroz prvi indeks matrice(red), a unutrašnja for petlja prolazi kroz drugi indeks(kolonu).

Sa ovom lekcijom, i ovom oblasti završen je materijal za učenje C++ programskog jezika. Naravno, pored ovoga ima još mnogo toga da se uči u C++, međutim ovaj materijal se drži onih oblasti koje se obrađuju u školama.

### 3. IZRADA WEB APLIKACIJE

Web aplikacija je rađena većinski u Laravel framework-u za programski jezik php. Korišten je i Livewire framework, koji služi kao dodatak Laravel-u i olakšava kreiranje dinamičnih web aplikacija. U izradi web aplikacije korištene su komponente koje Livewire pruža. Kreiranjem Livewire komponente, kreiraju se dva fajla(Class file i Blade file) koji su međusobno povezani. Class file sadrži klasu unutar koje se definišu funkcije, komunicira sa bazom podataka i sa drugim fajlom. Blade file sadrži HTML kod u kombinaciji sa php-om, CSS i eventualno JavaScript i to je dio koji vidimo na ekranu. Pored toga, naknadno je korišten Laravel Jetstream koji omogućava lakšu implementaciju autentifikacije u vidu login stranice, stranice za registraciju, te dodatnih mogućnosti kao što su slanje mail-ova za promjenu passworda. Web aplikacija je također povezana s bazom podataka, koja je unaprijed kreirana isto kroz Laravel. Naravno, korišteni su još HTML, CSS i JavaScript.

#### 3.1 Baza podataka



Slika 1. : Relacijski model baze podataka

Prvo je kreiran relacijski model baze podataka(Slika 1.), nakon čega je kroz Laravel kreirana sama baza podataka.

Php kod za kreiranje *users* tabele:

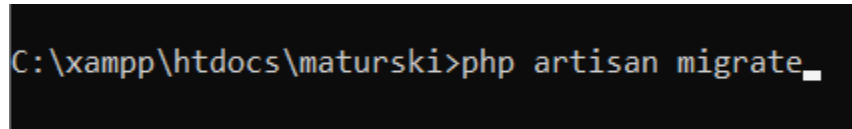
```
Schema::create('users', function (Blueprint $table) {
    $table->id();
    $table->string('name');
    $table->string('email')->unique();
    $table->timestamp('email_verified_at')->nullable();
    $table->string('password');
    $table->rememberToken();
    $table->foreignId('current_team_id')->nullable();
    $table->string('profile_photo_path', 2048)->nullable();
    $table->timestamps();
    $table->string('utype')->default('UCE')->comment('PROF za profesora i
UCE za učenika');
});
}
```

Php kod za dodavanje 'section\_id' atributa u 'lectures' tabelu:

```
Schema::table('lectures', function (Blueprint $table) {
    $table->bigInteger('section_id')->unsigned();
    $table->foreign('section_id')->references('id')->on('sections')-
    >onDelete('cascade');
});
```

Php kod za mijenjanje 'file' atributa u 'posts' tabeli:

```
Schema::table('posts', function (Blueprint $table) {
    $table->string('file')->nullable()->change();
});
```



```
C:\xampp\htdocs\maturski>php artisan migrate_
```

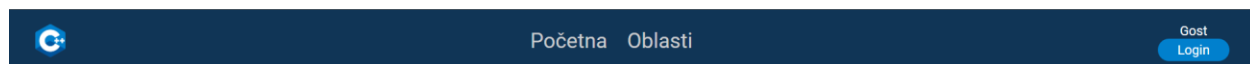
Slika 2. : Komanda za prijenos baze podataka

Nakon kreiranja svih tabela i svih izmjena koje su napravljene potrebno je pokrenuti komandu `php artisan migrate` (slika 2.) unutar command prompt-a da bi se iste tabele kreirale na mysql serveru. Kao rezultat toga dobijemo bazu podataka na *phpmyadmin* serveru (Slika 3.).

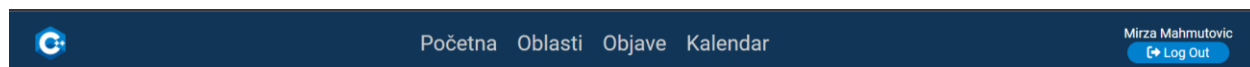
Tabelle	Aktion	Datensätze	Typ	Kollation	Größe	Überhang
<input type="checkbox"/> class_events	★	9	InnoDB	utf8mb4_unicode_ci	32,0 KiB	-
<input type="checkbox"/> events	★	3	InnoDB	utf8mb4_unicode_ci	48,0 KiB	-
<input type="checkbox"/> examples	★	50	InnoDB	utf8mb4_unicode_ci	32,0 KiB	-
<input type="checkbox"/> failed_jobs	★	0	InnoDB	utf8mb4_unicode_ci	32,0 KiB	-
<input type="checkbox"/> home_descriptions	★	3	InnoDB	utf8mb4_unicode_ci	16,0 KiB	-
<input type="checkbox"/> lectures	★	19	InnoDB	utf8mb4_unicode_ci	32,0 KiB	-
<input type="checkbox"/> migrations	★	34	InnoDB	utf8mb4_unicode_ci	16,0 KiB	-
<input type="checkbox"/> password_resets	★	1	InnoDB	utf8mb4_unicode_ci	32,0 KiB	-
<input type="checkbox"/> personal_access_tokens	★	0	InnoDB	utf8mb4_unicode_ci	48,0 KiB	-
<input type="checkbox"/> posts	★	11	InnoDB	utf8mb4_unicode_ci	32,0 KiB	-
<input type="checkbox"/> school_classes	★	11	InnoDB	utf8mb4_unicode_ci	32,0 KiB	-
<input type="checkbox"/> sections	★	5	InnoDB	utf8mb4_unicode_ci	16,0 KiB	-
<input type="checkbox"/> section_successes	★	0	InnoDB	utf8mb4_unicode_ci	48,0 KiB	-
<input type="checkbox"/> sessions	★	1	InnoDB	utf8mb4_unicode_ci	48,0 KiB	-
<input type="checkbox"/> users	★	13	InnoDB	utf8mb4_unicode_ci	64,0 KiB	-
15 Tabellen	Gesamt	160	InnoDB	utf8mb4_general_ci	528,0 KiB	0 B

Slika 3. : Baza podataka na phpmyadmin serveru

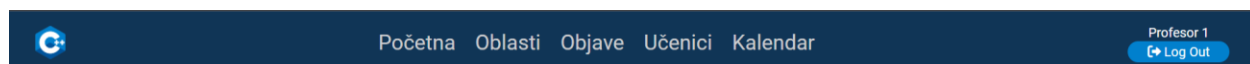
### 3.2 Web stranica



Slika 4. : Navigacijska traka za gosta



Slika 5. : Navigacijska traka za učenika



Slika 6. : Navigacijska traka za profesora

Blade file kod za navigacijske trake(Slika 1. Slika 2. i Slika 3.):

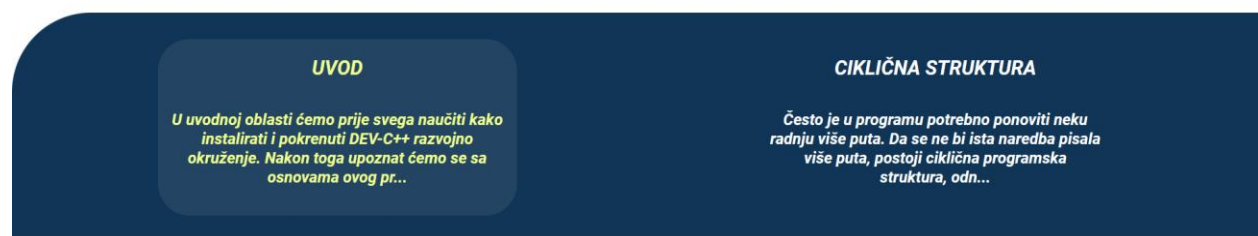
```
<header class="main-header">
    <div>
        <a href="/" class="main-header_brand">
            
        </a>
    </div>
    <div>
        <nav class="main-nav">
            <ul class="main-nav_items">
                <li class="main-nav_item"><a href="/">Početna</a></li>
                <li class="main-nav_item"><a href="{{ route('sekcije')
}}">Oblasti</a></li>
                @if (Route::has('login'))
                    @auth
                        <li class="main-nav_item"><a href="{{ route('objave')
}}">Objave</a></li>
                        @if (Auth::user()->utype === 'PROF')
                            <li class="main-nav_item"><a href="{{
route('ucenici') }}">Učenici</a></li>
                            <li class="main-nav_item"><a href="{{
route('kalendar') }}">Kalendar</a></li>
                        @elseif(Auth::user()->utype === 'UCE')
                            <li class="main-nav_item"><a href="{{
route('kalendar.razred', ['razred_name' => Auth::user()->razred->name])
}}">Kalendar</a></li>
                        @endif
                    @endauth
                @endif
            </ul>
        </nav>
    </div>
    <div>
        <div>
            @if (Route::has('login'))
                @auth
                    <span class="main-header_profile">{{ Auth::user()->name
}}</span>
                    <form method="POST" class="main-header_logout" action="{{
route('logout') }}">
                        @csrf
                        <a class="main-header_logout_button" href="{{
route('logout') }}" onclick="event.preventDefault();
```

```

                                this.closest('form').submit(); "
role="button">
                                <i class="fas fa-sign-out-alt"></i>
                                {{ __('Log Out') }}
                                </a>
                                </form>
                                @else
                                <span class="main-header_profile">Gost</span>
                                <div class="main-header_logout">
                                <a href="{{ route('login') }}" class="main-
header_login_button">Login</a>
                                </div>
                                @endauth
                                @endif
                                </div>
                                </div>
                                </header>

```

Kod iznad provjerava da li je korisnik prijavljen(i da li je prijavljen kao učenik ili profesor), te na osnovu toga dodaje ili oduzima mogućnosti sa navigacijske trake. Kada prvi put uđemo na stranicu vidimo navigacijsku traku(Slika 1.), gdje imamo tri linka. Ukoliko smo se prijavili kao učenik vidimo navigacijsku traku(Slika 2.) gdje dobijamo više opcija. I na kraju, ukoliko se prijavimo kao profesor dobijamo sva ovlaštenja i imamo najveći broj mogućnosti za izbor na navigacijskoj traci(Slika 3.). Važno je i spomenuti da je navigacijska traka sadržana na svakoj stranici.



Slika 7. : Početna stranica – slider

Class file kod za slider(Slika 7.):

```

$homeSliders = Section::inRandomOrder()->limit(2)->get();
$hSDescription = [];

```

```

foreach($homeSliders as $key=>$homeSlider)
{
    $hSDescription[$key] = substr($homeSlider->description, 0,
150).'...';
}

```

Blade file kod za slider(Slika 7.):

```

<section id="home-start" class="slide-in">
    @foreach ($homeSliders as $key=>$hS)
        <div class="home-start_quote">
            <a href="{ route('lekcije', ['sekcija_id' => $hS->id]) }}">
            <h1>{{ $hS->name }}</h1>
            <div class="home-start_quote-text">
                <p>{{ $hSDescription[$key] }}</p>
            </div>
            </a>
        </div>
    @endforeach
</section>

```

U kodu iznad se prije svega u class file-u uzimaju dvije nasumično odabrane oblasti iz baze podataka, nakon čega se uzima samo prvih 150 karaktera njihovog opisa radi ljepšeg izgleda na stranici. Zatim se unutar blade file-a te iste sekcije sačuvane u nizu *\$homeSliders* ispisuju na stranicu.

## C++



### Šta je C++ programski jezik? [🔗](#)

U svojoj srži C++ programski jezik niz je uputa koje programer upisuje na određeni način, držeći se pritom pravila i poštivajući opću logiku. Postoji niz jezika kojima se programeri mogu koristiti, no C++ se pokazao izrazito korisnim za razvoj videoigara.

### Koliko je teško naučiti programirati u C++? [🔗](#)

C++ spada u kategoriju tzv. "mid-level" jezika - jezika na

```

1 #include <iostream>
2 using namespace std;
3
4 int getSum(int *arr, int n){
5     int sum = 0;
6     for (int i = 0; i < n; i++)
7         sum += arr[i];
8     return sum;
9 }
10
11 // bad array, how, length
12

```

Slika 8. Početna stranica – o C++ programskom jeziku



### Koliko je teško naučiti programirati u C++?



C++ spada u kategoriju tzv. "mid-level" jezika - jezika na granici između mašinskog koda i ljudske logike slaganja riječi i rečenica. On je na pola puta između nula i jedinica koje razumije računar i našeg jezika. Učenje C++ programskog jezika nije teško za osobe koje su svjesne u šta se upuštaju.

```
1 int getsum(int *arr, int n){
2     int sum = 0;
3     for (int i = 0; i < n; i++)
4     {
5         sum += arr[i];
6     }
7     return sum;
8 }
9
10 // ARRAY OPERATORS
11 // bad_array.new_length
12 // is_array
13 int main ()
14 {
15     // is_array.known_bounds
16     // is_array.unknown_bounds
17     int arr[10];
18     // restrict_arr
19     int n = 10;
20     // add_value_reference_t
21     for (int i = 0; i < n; i++)
22     {
23         arr[i] = i * 10;
24     }
25     // add_value_reference_helper
26     // int sum = getsum(arr, n);
27     // int sum = getsum(arr, n);
28     cout << "Value of array elements : " << sum << endl;
29     return 0;
30 }
```



### Brzina, kontrola i moć



C++ programski jezik dosta se koncentrira na detalje, što isprva zna zbunjivati početnike i dovoditi do grešaka. Ovakav pristup detaljima omogućava jeziku da bude učinkovit i prilagodljiv. Problem učenja ovog programskog jezika nije u tome da je težak ili zahtjevan, već je iznimno širok i moćan.

Slika 9. Početna stranica – o C++ programskom jeziku

Class file kod za opis C++ programskog jezika(Slika 8. i Slika 9.):

```
$homes = HomeDescription::all();
```

Blade file kod za opis C++ programskog jezika(Slika 8. i Slika 9.):

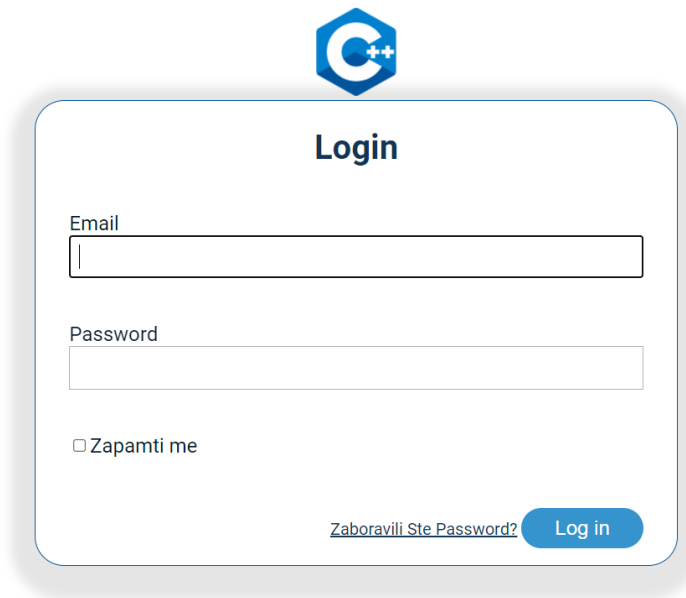
```
<section id="administrators">
    <h1 class="administrators-title">C++</h1>
    @foreach ($homes as $key=>$home)
        @if ($key % 2 != 0)
            <div class="story" id="admin-2">
                <div class="story_info">
                    <h1 class="story_name">{{ $home->title }}
                    @if (Route::has('login'))
                        @auth
                            @if (Auth::user()->utype === 'PROF')
                                <a href="{ route('home.uredi',
['home_id' => $home->id]) }}" class="edit"><i title="Uredi" class="fa-solid fa-
pen-to-square"></i></a>
                            @endif
                        @endauth
                    @endif
                </h1>
                <p class="story_text">{{ $home->description }}</p>
            </div>
            <div class="story_image-box">
                photo) }}"
alt="" class="story_image">
            </div>
        @endif
    @endforeach
</section>
```

```

        </div>
    </div>
    @else
        <div class="story" id="admin-1">
            <div class="story_image-box">
                photo) }}"
alt="" class="story_image">
            </div>
            <div class="story_info">
                <h1 class="story_name">{{ $home->title }}
                @if (Route::has('login'))
                    @a
                        @if (Auth::user()->utype === 'PROF')
                            <a href="{{ route('home.uredi',
['home_id' => $home->id]) }}" class="edit"><i title="Uredi" class="fa-solid fa-
pen-to-square"></i></a>
                        @endif
                    @endauth
                @endif
                </h1>
                <p class="story_text">{{ $home->description }}</p>
            </div>
        </div>
    @endif
@endforeach
</section>

```

U class file kodu vidimo da se iz baze podataka uzimaju svi podaci iz tabele za opis C++. Nakon čega se prenose u blade file, gdje se i ispisuju. S tim da se provjerava da li se ispisuje parna ili neparna iz razloga što se prvi opis postavlja na desnu stranu, a njegova slika na lijevu, a u drugom se slučaju slika stavlja na desnu stranu, a opis na lijevu.



*Slika 10. Login stranica*

Blade file kod za Login stranicu(Slika 10.):

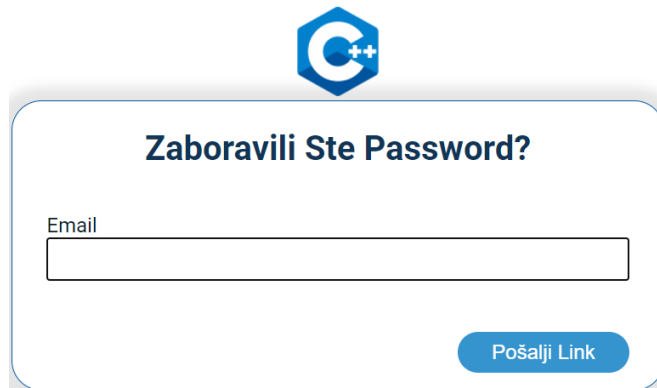
```
<x-guest-layout>
  <main>
    <div class="login-image-box">
      
    </div>
    <div class="login-box">
      <h1>Login</h1>
      <x-jet-validation-errors class="mb-4" />
      <form class="login-form" method="POST" action="{{ route('login') }}">
        @csrf
        <div class="login-form_group">
          <label for="username">Email</label>
          <input type="email" id="email" placeholder="Unesite email"
name="email" value="{{ old('email') }}" required autofocus>
        </div>
        <div class="login-form_group">
          <label for="password">Password</label>
          <input type="password" placeholder="Unesite password"
name="password" required autocomplete="current-password">
        </div>
        <div class="login-form_group">
          <label for="password">
```

```

        <input type="checkbox" name="remember" id="rememberme"
value="forever"><span>Zapamti me</span>
    </label>
</div>
<div class="login-submit">
    <a href="{ route('password.request') }">Zaboravili Ste
Password?</a>
    <button type="submit">Log in</button>
</div>
</form>
</div>
</main>
</x-guest-layout>

```

Stranica za login je napravljena uz pomoć Jetstream dodatka. Sadrži formu koja provjerava da li su unešeni podaci ispravni, te ako nisu ispisuje adekvatnu poruku. Također sadrži i link za slučaj ukoliko je korisnik zaboravio password. Klikom na taj link odlazimo na stranicu gdje unosimo email za promjenu passworda(Slika 11.).



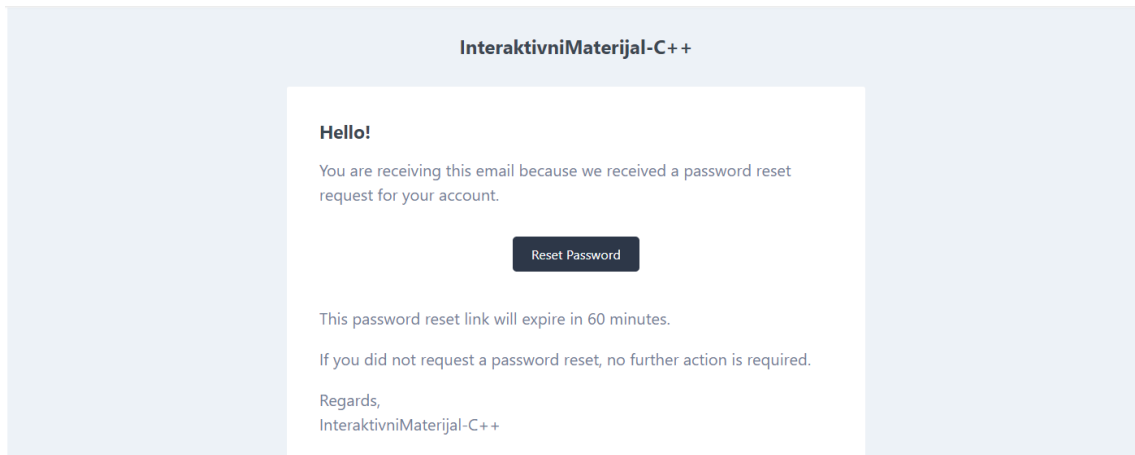
**Zaboravili Ste Password?**

Email

Pošalji Link

Slika 11. Zaboravljen password

Radi se o formi koja provjerava da li postoji korisnički račun sa unesim email-om, te ako postoji šalje mail sa linkom stranice za promjenu passworda(Slika 12.).

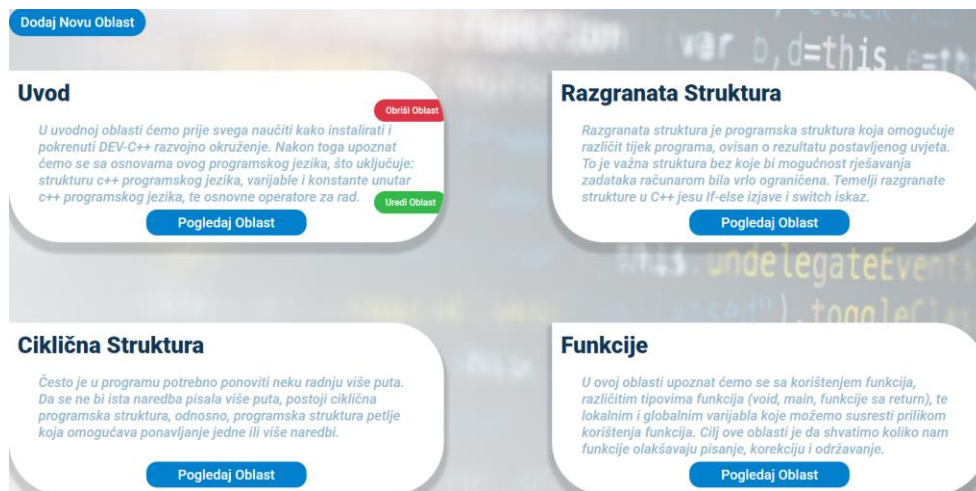


*Slika 12. : Mail sa linkom za promjenu passworda*

Klikom na dugme u mail-u(Slika 12.), otvorit će nam se stranica za promjenu passworda(Slika 13.).

The image displays a web form titled "Promijenite Password" (Change Password). At the top of the form is a logo consisting of a blue hexagon with a white 'C' and two white plus signs. The form contains three input fields: "Email" with the value "m.mirza2304@gmail.com", "Password", and "Potvrdite Password". A blue button labeled "Promijenite Password" is located at the bottom right of the form.

*Slika 13. Stranica za promjenu passworda*



Slika 14. – Oblasti za učenje C++

Class file kod za stranicu sa oblastima(Slika 14.):

```
class SekcijeComponent extends Component
{
    public function render()
    {
        $sekcije = Section::all();
        return view('livewire.sekcije-component', ['sekcije' => $sekcije])-
>layout('layouts.base');
    }

    public function deleteSection($id)
    {
        $sek = Section::find($id);
        $sek->delete();
    }
}
```

Blade file kod za stranicu sa oblastima(Slika 14.):

```
@push('styles')
    <link rel="stylesheet" href="{{ asset('assets/css/lekcije.css') }}">
@endpush
<main>
    @if (Route::has('login'))
        @auth
            @if (Auth::user()->utype === 'PROF')
```

```

        <div class="add-new">
            <a href="{{ route('oblast.dodaj') }}" class="add-new-
button">Dodaj Novu Oblast</a>
        </div>
    @endif
    @endauth
@endif
<div id="lekcije">
    @foreach ($sekcije as $sekcija)
        <div class="lekcija lekcija-l">
            <div class="lekcija-heading">
                <h1 class="lekcija-title">{{ $sekcija->name }}</h1>
                @if (Route::has('login'))
                    @auth
                        @if (Auth::user()->utype === 'PROF')
                            <a href="#" onclick="confirm('Da li ste sigurni
da želite obrisati ovu oblast?') || event.stopImmediatePropagation()"
wire:click.prevent="deleteSection({{ $sekcija->id }})" class="lekcija-remove-
button">Obriši Oblast</a>
                            <a href="{{ route('sekcija.uredi', ['sekcija_id'
=> $sekcija->id]) }}" class="lekcija-edit-button">Uredi Oblast</a>
                        @endif
                    @endauth
                @endif
            </div>
            <p class="lekcija-info">{{ $sekcija->description }}</p>
            <div class="lekcija-zapocni"><a href="{{ route('lekcije',
['sekcija_id' => $sekcija->id]) }}">Pogledaj Oblast</a></div>
        </div>
    @endforeach
</div>
    {{-- <div id="paginacija">
        {{ $sekcije->links() }}
    </div> --}}
</main>

```

U class file kodu možemo vidjeti cijelu klasu vezanu za stranicu sa oblastima(Slika 14.). Sadrži funkciju *render* pomoću koje se varijable prenose u blade file i funkciju *deleteSection* u kojoj se briše sekcija iz baze podataka ukoliko se klikne na dugme 'Obriši oblast'.

U kodu blade file-a vidimo ispis svih sekcija korištenjem *foreach* petlje gdje se ispisuju naslov, opis, dugme da se otvori oblast. Ukoliko je korisnik prijavljen i ukoliko je taj korisnik

profesor, on će hoveranjem preko jedne od oblasti dobiti mogućnost da obriše ili uredi oblast. Klikom na link 'Uredi oblast' otvara se nova stranica gdje se uređuje izabrana oblast(Slika 15.). Također klikom na link 'Dodaj Novu oblast' otvara se stranica sa formom za dodavanje nove oblasti(Slika 16.).

Slika 15. : Stranica za uređivanje oblasti

Class file kod za stranicu za uređivanje oblasti(Slika 15.):

```
class UrediSekcijuComponent extends Component
{
    public $section_id;
    public $name;
    public $description;

    public function mount($sekcija_id)
    {
        $this->section_id = $sekcija_id;
        $sekcija = Section::find($sekcija_id);
        if($sekcija === null)
            abort(404);
        $this->name = $sekcija->name;
        $this->description = $sekcija->description;
    }

    public function updated($fields)
    {
        $this->validateOnly($fields,[
```



```

        'name' => [
            'required',
            Rule::unique('sections')->ignore($this->section_id),
            'max:100'
        ],
        'description' => 'required|max:300',
    ]);
}

public function updateSection()
{
    $this->validate([
        'name' => [
            'required',
            Rule::unique('sections')->ignore($this->section_id),
            'max:100'
        ],
        'description' => 'required|max:300',
    ]);

    $sekcija = Section::find($this->section_id);
    $sekcija->name = $this->name;
    $sekcija->description = $this->description;
    $sekcija->save();
    session()->flash('oblast_poruka', 'Oblast je uspješno uređena');
}

public function render()
{
    return view('livewire.admin.uredi-sekciju-component')-
>layout('layouts.base');
}
}

```

Blade file kod za stranicu za uređivanje oblasti(Slika 15.):

```

<main>
    <div class="all">
        <a href="{{ route('sekcije') }}" class="all-button">Sve Oblasti</a>
    </div>
    <div class="register-box">
        <h1 class="register-heading">Uredite Oblast</h1>
        @if (Session::has('oblast_poruka'))

```

```

        <div class="alert alert-success" role="alert">{{
Session::get('oblast_poruka') }}</div>
    @endif
    <form class="register-form" method="POST"
wire:submit.prevent="updateSection()">
        @csrf
        <div class="register-form_group">
            <label for="name">Naziv Oblasti</label>
            <input type="text" placeholder="Naziv Oblasti" id="name"
name="name" wire:model='name'>
            @error('name')
                <p class="text-danger">{{ $message }}</p>
            @enderror
        </div>
        <div class="register-form_group">
            <label for="description">Opis Oblasti</label>
            <textarea class="register-description" name="description"
placeholder="Opis Oblasti" id="description" cols="30" rows="10"
wire:model='description'></textarea>
            @error('description')
                <p class="text-danger">{{ $message }}</p>
            @enderror
        </div>
        <div class="register-submit">
            <button type="submit">Uredi</button>
        </div>
    </form>
</div>
</main>

```

U Class file-u možemo vidjeti mount funkciju koja automatski učitava izabranu oblast. Zatim funkciju updated koja u stvarnom vremenu provjerava da li je korisnik unio ispravne podatke, te ako nije izvještava ga o tome. Funkcija updateSection se pokreće nakon što korisnik klikne na dugme uredi i ona u konačnosti još jednom provjeri ispravnost korisničkog unosa, te ako je sve ispravno uređuje oblast i sprema je u bazu podataka.

U Blade file-u možemo vidjeti formu koja je povezana sa varijablama iz class file-a i u koju korisnik unosi podatke. Klikom na link 'Sve oblasti' korisnik se vraća nazad na prijašnju stranicu.

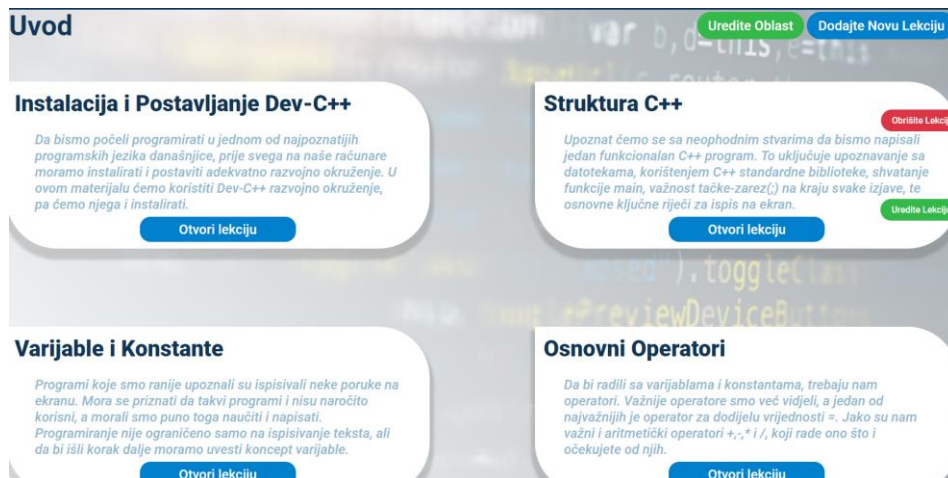
Slika 16. : Stranica za dodavanje nove oblasti

Class file kod za stranicu za dodavanje nove oblasti(Slika 16.):

```
public function addSection()  
{  
    $this->validate([  
        'name' => 'required|unique:sections,name|max:100',  
        'description' => 'required|max:300|min:150'  
    ]);  
    $section = new Section();  
    $section->name = $this->name;  
    $section->description = $this->description;  
    $section->save();  
    session()->flash('oblast_poruka', 'Uspješno ste dodali novu oblast');  
}
```

U class file kodu vidimo funkciju addSection koja se pokreće klikom na dugme 'Dodaj' i ona opet provjerava korisnikov unos, te ako je unos ispravan sprema oblast u bazu podataka.

U blade file-u opet je sadržana forma koja je povezana sa varijablama iz class file-a. Pored forme imamo i link 'Sve oblasti' što nas vraća na prijašnju stranicu.



Slika 17. : Stranica sa lekcijama unutar izabrane oblasti

Unutar class file-a imamo tri funkcije koje funkcionišu na isti način kao funkcije prikazane u kodu za stranicu sa oblastima(Slika 14.). Prvo imamo funkciju mount koja na osnovu izabrane oblasti iz baze podataka uzima lekcije. Funkcija render povezuje class file i blade file, dok funkcija deleteLecture briše lekciju zajedno sa svim njenim primjerima ukoliko korisnik klikne na dugme 'Obriši lekciju'.

Kod u blade file-u funkcioniše na isti princip kao i kod stranice sa oblastima(Slika 14.). Putem foreach petlje se ispisuju lekcije, a profesor ima dodatne mogućnosti. Dodatne mogućnosti su brisanje i uređivanje lekcija, zatim uređivanje oblasti i dodavanje nove lekcije. Klikom na link 'Uredite oblast' otvorit će se stranica za uređivanje oblasti(Slika 15.). Klikom na link 'Dodajte Novu Lekciju' otvara se stranica sa formom za dodavanje nove lekcije(Slika 18. i Slika 19.). Imamo još i link za uređivanje lekcije čijim klikom nam se otvara stranica za uređivanje odabrane lekcija(Slika 20. i Slika 21.).

Sve Lekcije

## Dodajte Novu Lekciju

Naslov

Opis Lekcije

Youtube Video Link

Objašnjenje Lekcije

Savjet

Slika 18. : Stranica za dodavanje nove lekcije

Savjet

Dodatni Primjeri

**Primjer( Slika + Objašnjenje )**

Naslov Primjera

Slika Primjera

No file chosen

Objašnjenje Primjera

Slika 19. : Stranica za dodavanje nove lekcije

Class file kod za stranicu za dodavanje nove lekcije(Slika 18. i Slika 19.):

```
class DodajLekcijuComponent extends Component
```

```

{
    use WithFileUploads;

    public $section_id;

    public $title;
    public $video;
    public $explanation;
    public $hint;
    public $description;

    public $example_type;
    public $inputs = [];
    public $examples = [];
    public $example_values = [];

    public function mount($sekcija_id)
    {
        $this->section_id = $sekcija_id;
        $sekcija = Section::find($sekcija_id);
        if($sekcija === null)
            abort(404);
    }

    public function addExample()
    {
        if($this->example_type == 2 || $this->example_type == 1)
        {
            array_push($this->inputs, $this->example_type);
            array_push($this->examples, $this->example_type);
        }
    }

    public function removeExample($key)
    {
        unset($this->inputs[$key]);
        unset($this->examples[$key]);
    }

    public function updated($fields)
    {
        $this->validateOnly($fields,[
            'title' => 'required|max:100',
            'video' => 'required|max:50',
            'explanation' => 'required|max:500',

```

```

        'description' => 'required|max:300|min:150'
    ]);
    foreach($this->inputs as $ikey=>$input)
    {
        $this->validateOnly($fields,[
            'example_values.'.$ikey.'.title' => 'required|max:100',
            'example_values.'.$ikey.'.photo' => 'required|mimes:jpeg,png',
            'example_values.'.$ikey.'.explanation' => 'required|max:500',
        ]);
        if($input == 1)
        {
            $this->validateOnly($fields,[
                'example_values.'.$ikey.'.input' => 'required|max:30'
            ]);
        }
    }
}

public function addLecture()
{
    $this->validate([
        'title' => 'required|max:100',
        'video' => 'required|max:50',
        'explanation' => 'required|max:500',
        'description' => 'required|max:300|min:150'
    ]);
    foreach($this->inputs as $ikey=>$input)
    {
        $this->validate([
            'example_values.'.$ikey.'.title' => 'required|max:100',
            'example_values.'.$ikey.'.photo' => 'required|mimes:jpeg,png',
            'example_values.'.$ikey.'.explanation' => 'required|max:500',
        ]);
        if($input == 1)
        {
            $this->validate([
                'example_values.'.$ikey.'.input' => 'required|max:30'
            ]);
        }
    }
    $lekciya = new Lecture();
    $lekciya->title = $this->title;
    $lekciya->explanation = $this->explanation;
    $lekciya->video = preg_replace(

```

```

        "\s*[a-zA-Z\\/\?:\.\.]*youtu(be.com/watch?v=|.be/)([a-zA-Z0-9\-\_]+)([a-zA-Z0-9\\/\*\\-\_\\?\\&\\;%\\=\.]*)/i",
        "//www.youtube.com/embed/$2",
        $this->video
    );
    $lekciya->section_id = $this->section_id;
    $lekciya->description = $this->description;
    if($this->hint)
    $lekciya->hint = $this->hint;
    $lekciya->save();
    foreach($this->example_values as $key=>$example_values)
    {
        foreach($this->inputs as $ikey=>$input)
        {
            if($key == $ikey)
            {
                $example = new Example();
                $example->title = $example_values['title'];
                $example->explanation = $example_values['explanation'];
                $example->lecture_id = $lekciya->id;
                $photoName = Carbon::now()-
>timestamp.$key.'.'. $example_values['photo']->extension();
                $example_values['photo']->storeAs('public/examples',
$photoName);

                $example->photo = $photoName;
                if($input == 1)
                {
                    $example->input = $example_values['input'];
                    $example->type = 1;
                }
                else if($input == 2)
                {
                    $example->type = 2;
                }
                $example->save();
            }
        }
    }
    session()->flash('lekciya_poruka', 'Lekcija je uspješno dodana');
}

public function messages()
{
    return [

```



```

        'example_values.*.title.required' => 'The example title field is
required',
        'example_values.*.title.max' => 'An example title must not be greater
than 100 characters',
        'example_values.*.explanation.required' => 'The example explanation
field is required',
        'example_values.*.explanation.max' => 'An example explanation must
not be greater than 500 characters',
        'example_values.*.photo.required' => 'The example photo field is
required',
        'example_values.*.photo.mimes' => 'The example photo must be a file
of type: jpeg, png',
        'example_values.*.input.required' => 'The example input field is
required',
        'example_values.*.input.max' => 'An example input must not be greater
than 30 characters'
    ];
}

public function render()
{
    return view('livewire.admin.dodaj-lekciju-component')-
>layout('layouts.base');
}
}

```

Blade file kod za stranicu za dodavanje nove lekcije(Slika 18. i Slika 19.):

```

<main>
    <div class="all">
        <a href="{{ route('lekcije',['sekcija_id' => $section_id]) }}"
class="all-button">Sve Lekcije</a>
    </div>
    <div class="register-box">
        <h1 class="register-heading">Dodajte Novu Lekciju</h1>
        @if (Session::has('lekcija_poruka'))
            <div class="alert alert-success" role="alert">{{
Session::get('lekcija_poruka') }}</div>
        @endif
        <form class="register-form" method="POST" enctype="multipart/form-data"
wire:submit.prevent="addLecture()">
            @csrf
            <div class="register-form_group">
                <label for="title">Naslov</label>

```

```

        <input type="text" placeholder="Naslov" id="title" name="title"
wire:model='title'>
        @error('title')
        <p class="text-danger">{{ $message }}</p>
        @enderror
    </div>
    <div class="register-form_group">
        <label for="description">Opis Lekcije</label>
        <textarea class="register-description" name="description"
placeholder="Opis" id="description" cols="30" rows="10"
wire:model='description'></textarea>
        @error('description')
        <p class="text-danger">{{ $message }}</p>
        @enderror
    </div>
    <div class="register-form_group">
        <label for="video">Youtube Video Link</label>
        <input type="text" placeholder="Youtube Video Link" id="video"
name="video" wire:model='video'>
        @error('video')
        <p class="text-danger">{{ $message }}</p>
        @enderror
    </div>
    <div class="register-form_group">
        <label for="explanation">Objašnjenje Lekcije</label>
        <textarea class="register-description" name="explanation"
placeholder="Objašnjenje" id="explanation" cols="30" rows="10"
wire:model='explanation'></textarea>
        @error('explanation')
        <p class="text-danger">{{ $message }}</p>
        @enderror
    </div>
    <div class="register-form_group">
        <label for="hint">Savjet</label>
        <textarea class="register-description" name="hint"
placeholder="Savjet" id="hint" cols="30" rows="10" wire:model='hint'></textarea>
    </div>
    <div class="register-form_group">
        <label for="school_class_id">Dodatni Primjeri</label>
        <div class="register-flex">
            <select name="" id="" wire:model="example_type">
                <option value="">Izaberite Tip Primjera</option>
                <option value="1">Slika + objašnjenje + unos</option>
                <option value="2">Slika + objašnjenje</option>
            </select>

```

```

        <button wire:click.prevent="addExample()" class="gen-
pass">Dodaj Primjer</button>
    </div>
</div>
@foreach ($inputs as $key=>$value)
    @if ($value == 2)
        <div class="register-flex-example">
            <h2 class="register-example-title">Primjer( Slika +
Objasnjenje )</h2>
            <button type="button" class="remove-example"
wire:click.prevent="removeExample({{ $key }})">Obriši</button>
        </div>
    @elseif($value == 1)
        <div class="register-flex-example">
            <h2 class="register-example-title">Primjer( Slika +
Objasnjenje + Unos )</h2>
            <button type="button" class="remove-example"
wire:click.prevent="removeExample({{ $key }})">Obriši</button>
        </div>
    @endif
    <div class="register-form_group">
        <label for="title">Naslov Primjera</label>
        <input type="text" placeholder="Naslov Primjera"
id="title" name="title" wire:model="example_values.{{ $key }}.{{ 'title' }}">
    </div>
    <div class="register-form_group">
        <label for="photo">Slika Primjera</label>
        <input type="file" name="photo" id="photo"
wire:model="example_values.{{ $key }}.{{ 'photo' }}">
    </div>
    @isset($example_values[$key]['photo'])
        @if ($example_values[$key]['photo']->extension() == 'jpg'
|| $example_values[$key]['photo']->extension() == 'png')
            
        @endif
    @endisset
    <div class="register-form_group">
        <label for="explanation">Objasnjenje Primjera</label>
        <textarea class="register-description" name="explanation"
placeholder="Objasnjenje Primjera" id="explanation" cols="30"
rows="10" wire:model="example_values.{{ $key }}.{{ 'explanation' }}"></textarea>
    </div>
    @if ($value == 1)
        <div class="register-form_group">

```

```

        <label for="input">Tačan Unos</label>
        <input type="text" placeholder="Tačan Unos Primjera"
id="input" name="input" wire:model="example_values.{{ $key }}.{{ 'input' }}">
    </div>
    @endif
@endforeach
<div class="register-form_group">
    @error("example_values*.title")
        <p class="text-danger">{{ $message }}</p>
    @enderror
    @error("example_values*.photo")
        <p class="text-danger">{{ $message }}</p>
    @enderror
    @error("example_values*.explanation")
        <p class="text-danger">{{ $message }}</p>
    @enderror
    @error("example_values*.input")
        <p class="text-danger">{{ $message }}</p>
    @enderror
</div>
<div class="register-submit">
    <button type="submit">Dodaj</button>
</div>
</form>
</div>
</main>

```

Unutar class file koda vidimo prvo funkciju mount koja provjerava da li postoji sekcija za koju je korisnik odabrao da dodaje lekciju. Nakon toga vidimo funkciju addExample koja se pokreće klikom na dugme 'Dodaj Primjer', i funkciju removeExample, koja se pokreće klikom na dugme 'Obriši'. Funkcija updated u realnom vremenu provjerava korisnikov unos, a funkcija addLecture provjerava korisnikov unos nakon što je kliknuo dugme 'Dodaj', te ukoliko je ispravan unos sprema u bazu podataka lekciju i njene primjere. Funkcija messages uređuje poruke koje se ispisuju na ekran ukoliko je korisnik unos pogrešan.

Unutar class file-a vidimo opet formu za unos podataka, u kojoj je svaki input povezan sa varijablom iz class file-a. Vidimo i dugme 'Sve lekcije' koje nas vraća na prijašnju stranicu.

## Uredi Lekciju

Naslov

Instalacija i Postavljanje Dev-C++

Opis Lekcije

Da bismo počeli programirati u jednom od najpoznatijih programskih jezika današnjice, prije svega na naše računare moramo instalirati i postaviti adekvatno razvojno okruženje. U ovom materijalu ćemo koristiti Dev-C++ razvojno okruženje, pa ćemo njega i instalirati.

Youtube Video Link

//www.youtube.com/embed/oHRtnIEblbo

Objašnjenje Lekcije

Da bismo počeli programirati u jednom od najpoznatijih programskih jezika današnjice, prije svega na naše računare moramo instalirati i postaviti adekvatno razvojno okruženje. U ovom materijalu ćemo koristiti Dev-C++ razvojno okruženje, pa ćemo njega i instalirati.

Savjet

Savjet

Slika 20. Stranica za uređivanje lekcije

Savjet

Savjet

Dodatni Primjeri

Izaberite Tip Primjera

Dodaj Primjer

Uredi

Slika 21. : Stranica za uređivanje lekcije

```
public function updateLecture()
{
    $this->validate([
        'title' => 'required|max:100',
        'video' => 'required|max:50',
        'explanation' => 'required|max:500',
    ]);
}
```

```

        'description' => 'required|max:300|min:150'
    ]);
    foreach($this->inputs as $ikey=>$input)
    {
        $this->validate([
            'example_values.'.$ikey.'.title' => 'required|max:100',
            'example_values.'.$ikey.'.explanation' => 'required|max:500',
        ]);
        if($input == 1)
        {
            $this->validate([
                'example_values.'.$ikey.'.input' => 'required|max:30'
            ]);
        }
        if(!$this->example_values[$ikey]['photo'])
        {
            $this->validate([
                'example_values.'.$ikey.'.newphoto' =>
                'required|mimes:jpeg,png'
            ]);
        }
    }
    $this->lekcija->title = $this->title;
    $this->lekcija->explanation = $this->explanation;
    $this->lekcija->video = preg_replace(
        "/\s*[a-zA-Z\\\/:\\.]*youtu(be.com\\/watch\\?v=|\\.be\\/)([a-zA-Z0-9\\-
_]+)([a-zA-Z0-9\\/*\\-\\_\\?\\&\\;%\\=\\.]*)/i",
        "//www.youtube.com/embed/$2",
        $this->video
    );
    $this->lekcija->description = $this->description;
    if($this->hint)
    $this->lekcija->hint = $this->hint;
    $this->lekcija->save();
    foreach($this->example_values as $key=>$example_values)
    {
        foreach($this->inputs as $ikey=>$input)
        {
            if($key == $ikey)
            {
                if(isset($example_values['id']))
                {
                    foreach($this->lekcija->primjeri as $primjer)
                    {
                        if($primjer->id == $example_values['id'])

```

```

        $example = $primjer;
    }
    $example->title = $example_values['title'];
    $example->explanation = $example_values['explanation'];
    $example->lecture_id = $this->lekcija->id;
    if($example_values['newphoto'])
    {
        if($example_values['photo'])
        {
            if(file_exists(public_path('storage').'/'.$example
_values['photo'])))
            {
                unlink(public_path('storage').'/'.$example_val
ues['photo']));
            }
        }
        $photoName = Carbon::now()-
>timestamp.$key.'.'.$example_values['newphoto']->extension();
        $example_values['newphoto']->
>storeAs('public/examples', $photoName);
        $example->photo = $photoName;
    }
}

else
{
    $example = new Example();
    $example->title = $example_values['title'];
    $example->explanation = $example_values['explanation'];
    $example->lecture_id = $this->lekcija->id;
    $photoName = Carbon::now()-
>timestamp.$key.'.'.$example_values['newphoto']->extension();
    $example_values['newphoto']->storeAs('public/examples',
$photoName);

    $example->photo = $photoName;
}
if($input == 1)
{
    $example->input = $example_values['input'];
    $example->type = 1;
}

else if($input == 2)
{
    $example->type = 2;
}

```

```

    }

    $example->save();
}
}
}
session()->flash('lekcija_poruka', 'Lekcija je uspješno uređena');
}

```

Unutar class file-a vidimo funkciju updateLecture koja na kraju provjerava unos korisnika, te ukoliko je ispravan sprema njegove izmjene.

Unutar blade file-a je ista forma kao za dodavanje nove lekcije.



Slika 22. : Stranica odabrane lekcije

Class file kod za stranicu odabrane lekcije(Slika 22.):

```

class LekcijaComponent extends Component
{
    public $lekcija;
    public $lek_id;
    public $ytId;
    public $vidExists;
    public $input = [];
    public $correct = [];
    public $check = [];

    public $next;
}

```



```

use WithPagination;

public function mount($lekcija_id)
{
    if(!Auth::check())
    {
        return redirect()->route('login');
    }
    $this->lek_id = $lekcija_id;
    $this->lekcija = Lecture::find($this->lek_id);
    if($this->lekcija === null)
    abort(404);
    $yt = explode('/', $this->lekcija->video);
    $this->ytId = end($yt);
    $this->vidExists = $this->yt_exists($this->ytId);
    foreach($this->lekcija->primjeri as $key=>$primjer)
    {
        if($primjer->type == '1')
        {
            $this->correct[$key] = $primjer->input;
            $this->check[$key] = 3;
        }
    }
    $this->next = Lecture::where('id', '>', $lekcija_id)->where('section_id',
'=', $this->lekcija->section_id)->min('id');
}

public function yt_exists($videoID) {
    $theURL = "http://www.youtube.com/oembed?url=$videoID&format=json";
    $headers = get_headers($theURL);
    if(substr($headers[0], 9, 3) !== "400" && substr($headers[0], 9, 3) !==
"401" && substr($headers[0], 9, 3) !== "404")
    return 1;
    else
    return 0;
}

public function checkInput($key)
{
    if(strtolower($this->input[$key]) == strtolower($this->correct[$key]))
    $this->check[$key] = 1 ;
    else
    $this->check[$key] = 0;
}

```

```

public function render()
{
    return view('livewire.lekcija-component')->layout('layouts.base');
}
}

```

Blade file kod za stranicu odabrane lekcije(Slika 22.):

```

<main>
<div class="backdrop"></div>
<div class="lekcija-box">
    <h1 class="lekcija-heading">{{ $lekcija->title }}</h1>
    <div class="primjer">
        <h2 class="primjer-heading">Video + Objašnjenje</h2>
        <div class="primjer-body">
            <div class="primjer-body_img-container">
                @if ($vidExists == 1)
                    <iframe width="560" height="315" src="{{ $lekcija->video
}} " title="YouTube video player" frameborder="0" allow="accelerometer; autoplay;
clipboard-write; encrypted-media; gyroscope; picture-in-picture"
allowfullscreen></iframe>
                @else
                    <h2>Nije moguće pronaći video.</h2>
                @endif
            </div>
            <div class="primjer-body_text">
                <h2 class="objasnjenje-heading">Objašnjenje</h2>
                <p class="objasnjenje-text">{{ $lekcija->explanation }}</p>
                @if ($lekcija->hint)
                    <button class="primjer-body_rjesenje"
onclick="primjer1()">Pogledaj Savjet</button>
                    <div class="rjesenje_hidden">
                        <p>{{ $lekcija->hint }}</p>
                    </div>
                @endif
            </div>
        </div>
    </div>
</div>
@php
    $k=-1;
@endphp
@foreach ($lekcija->primjeri as $key=>$primjer)

```

```

@if ($primjer->type == '2')
    @php
        $k++;
    @endphp
    <div class="primjer">
        <h2 class="primjer-heading">{{ $primjer->title }}</h2>
        <div class="primjer-body">
            <div class="primjer-body_img-container" id="img-
container_3">
                photo) }}" alt="">
            </div>
            <div class="primjer-3-btn-div">
                <button class="primjer3-btn" onclick="primjer3({{ $k
}})">Prikaži Objašnjenje</button>
            </div>
            <div class="primjer-3-objasnjenje">{{ $primjer-
>explanation }}</div>
        </div>
    </div>
@else
    <div class="primjer">
        <h2 class="primjer-heading">{{ $primjer->title }}</h2>
        <div class="primjer-body" id="primjer2-body">
            <div class="primjer-body_img-container" id="img-
container_2">
                photo) }}" alt="">
            </div>
            <div class="primjer-body_text">
                <h2 class="primjer2-title">{{ $primjer->explanation
}}</h2>
                <p class="primjer2-subtitle">U polje ispod napiši šta
fali kodu na slici</p>
                <form action="" wire:submit.prevent="checkInput({{
$key }}">
                    @csrf
                    @if ($check[$key] == 1)
                        <input type="text" class="primjer2-input
correct" wire:model="input.{{ $key }}">
                    @elseif($check[$key] == 0)
                        <input type="text" class="primjer2-input
incorrect" wire:model="input.{{ $key }}">
                    @else

```

```

                                <input type="text" class="primjer2-input"
wire:model="input.{{ $key }}">
                                @endif
                                <button type="submit" class="primjer2-
btn">Potvrđi</button>
                                </form>
                                </div>
                                </div>
                                </div>
                                @endif
                                @endforeach
                                <div class="bubble-body">
                                    <div class="bubble" title="Početak lekcije"
onclick="window.scrollTo({ top: 0, behavior: 'smooth' });"><i class="fas fa-
arrow-up"></i></div>
                                    @if ($next)
                                        <a href="{{ route('lekcija', ['lekcija_id' => $next ]) }}"><div
class="bubble" title="Sljedeća Lekcija"><i class="fa-solid fa-arrow-
right"></i></div></a>
                                    @else
                                        <a href="{{ route('lekcije', ['sekcija_id' => $lekcija-
>section_id ]) }}"><div class="bubble" title="Oblast"><i class="fa-solid fa-
arrow-right"></i></div></a>
                                    @endif
                                </div>
                                </div>
</main>

@push('scripts')
<script>
    backdrop = document.querySelector('.backdrop');
    rjesenje = document.querySelector('.rjesenje_hidden');
    btn1 = document.querySelector('.primjer-body_rjesenje');
    pr3o = document.querySelectorAll('.primjer-3-objasnjenje');
    var i;

    function primjer1()
    {
        backdrop.style.display = "block";
        rjesenje.style.display = "block";
        rjesenje.classList.remove("close");
        rjesenje.classList.add("open");

        btn1.classList.remove("open");
        btn1.classList.add("close");
    }

```

```

    }

    backdrop.addEventListener('click', function(){
        backdrop.style.display = "none";
        if(btn1 !== null)
        {
            if(btn1.classList.contains("close"))
            {
                btn1.classList.remove("close");
                btn1.classList.add("open");
                rjesenje.classList.remove("open");
                rjesenje.classList.add("close");
                rjesenje.style.display = "none";
            }
        }
        if(pr3o[i].style.display == "block")
        {
            pr3o[i].style.display = "none";
            pr3o[i].style.zIndex = "1";
        }

    });

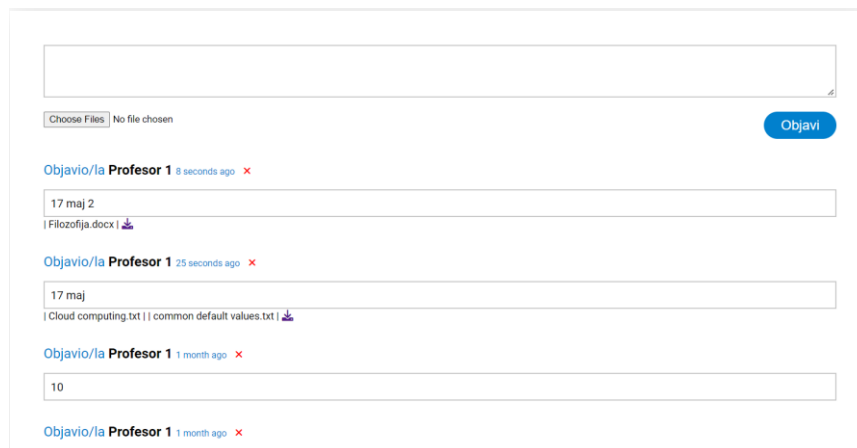
    function primjer3(key)
    {
        i=key;
        backdrop.style.display = "block";
        pr3o[key].style.display = "block";
        pr3o[key].style.zIndex = "200";
    }

</script>
@endpush

```

U class file-u vidimo funkciju mount u kojoj se provjerava da li je korisnik prijavljen, te ako nije vraća ga na login stranicu, jer nema pristup lekcijama. Zatim se iz baze podataka uzima odabrana lekcija. Imamo i funkciju yt\_exists koja provjerava da li se može pronaći youtube video sa linkom vezanim uz lekciju, te ako ne može vraća vrijednost 0. Također imamo i funkciju checkInput koja se pokreće klikom na dugme 'Potvrdi' kod primjera sa unosom, ukoliko lekcija ima takav primjer.

U blade file-u se ispisuju svi podaci vezani za lekciju. Provjerava se koju je vrijednost vratila funkcija `yt_exists`, te na osnovu toga se ispisuje poruka. Lekcija također sadrži dva buttona, jedan vraća korisnik na vrh stranice, a drugi vodi korisnika na sljedeću lekciju iste oblasti. Ukoliko nema takve lekcije, onda ga vraća nazad u oblast, gdje može da vidi sve lekcije. Pored toga sadrži i JavaScript kod, pomoću kojeg se povećava interakcija korisnika sa samim lekcijama. Služi za otvaranje i zatvaranje objašnjenja kod primjera lekcije.



Slika 23. : Stranica za objave

Class file kod za stranicu za objave(Slika 23.):

```
class ObjaveComponent extends Component
{
  use WithFileUploads;
  use WithPagination;

  public $body;
  public $files;
  public $fileCounter=0;

  public function mount()
  {
    if(!Auth::check())
    {
      return redirect()->route('login');
    }
  }

  public function render()
```

```

{
    $objave = Post::orderBy('created_at', 'DESC')->paginate(10);
    return view('livewire.objave-component', ['objave' => $objave])-
>layout('layouts.base');
}

public function post()
{
    $this->validate([
        'body' => 'required'
    ]);
    $objava = new Post();
    $objava->body = $this->body;
    $objava->user_id = Auth::user()->id;
    if($this->files)
    {
        $filename = '';
        foreach($this->files as $key=>$file)
        {
            $fname = Carbon::now()->timestamp.$key.'/'.'$file-
>getClientOriginalName();
            $file->storeAs('files', $fname);
            if($key == 0)
                $filename = $fname;
            else
                $filename = $filename .','.'$fname;
        }
        $objava->file = $filename;
    }
    $objava->save();
    $this->reset(['body', 'files']);
    $this->fileCounter++;
    session()->flash('objava_poruka', 'Uspješno ste dodali novu objavu!');
}

public function deletePost($id)
{
    $objava = Post::find($id);
    if($objava->file)
    {
        $files = explode(',', $objava->file);
        foreach($files as $file)
        {
            if($file)
            {

```

```

        if(file_exists(storage_path('app/files'.'/'.$file)))
        {
            $fname = explode('/', $file);
            File::deleteDirectory(storage_path('app/files'.'/'.$fname
[0]));

            // unlink(storage_path('app/files'.'/'.$file));
        }
    }
}
$objjava->delete();
session()->flash('objjava_poruka', 'Objava je uspješno obrisana');
}

public function downloadFiles($id)
{
    $zipFile = 'objava.zip';
    $zip = new ZipArchive();
    $zip->open($zipFile, ZipArchive::CREATE | ZipArchive::OVERWRITE);
    $objjava = Post::find($id);
    $files = explode(',', $objjava->file);
    foreach($files as $file)
    {
        if(file_exists(storage_path('app/files/'.$file)))
        {
            $zipName = explode('/', $file);
            $zip->addFile(storage_path('app/files/'.$file), $zipName[1]);
        }
    }
    $zip->close();
    return response()->download($zipFile);
}
}

```

Blade file kod za stranicu za objave(Slika 23.):

```

<main>
    <div class="posts">
        @if (Route::has('login'))
            @auth
                @if (Auth::user()->utype === 'PROF')
                    @if (Session::has('objava_poruka'))
                        <div class="alert alert-success" role="alert">{{
Session::get('objava_poruka') }}</div>

```



```

        @endif
        <form action="" enctype="multipart/form-data" class="posts-
form" wire:submit.prevent="post()">
            @csrf
            <textarea name="body" id="" placeholder="Write a post!"
wire:model="body"></textarea>
            @error('body')
                <p class="text-danger">{{ $message }}</p>
            @enderror
            <input type="file" wire:model="files" id="{{ $fileCounter
}}"> multiple>

            <button type="submit">Objavi</button>
        </form>
    @endif
    @endauth
    @endif
    @if ($objave->count()>0)
        @foreach ($objave as $objava)
            <div class="post">
                <p class="post-head">Objavio/la <span class="post-name">{{
$objava->korisnik->name }} </span><span class="post-time"> {{ $objava->created_at-
>diffForHumans() }}</span>
                    @if (Gate::allows('delete-post', $objava))
                        <a href="#" onclick="confirm('Da li ste sigurni da
želite obrisati ovu objavu?') || event.stopImmediatePropagation()"
wire:click.prevent="deletePost('{{ $objava->id }})' title="Obriši Objavu"
class="post-delete"><i class="fa-solid fa-xmark"></i></a>
                    @endif
                </p>
                <div class="post-body">{{ $objava->body }}</div>
                @if ($objava->file)
                    @php
                        $files = explode(',', $objava->file);
                    @endphp
                    @foreach ($files as $file)
                        @php
                            $fname = explode('/', $file);
                        @endphp
                        <span class="post-files">| {{ $fname[1] }} |</span>
                    @endforeach
                    <a href="#" title="Preuzmi Datoteke" onclick="confirm('Da
li ste sigurni da želite preuzeti ove datoteke?') ||
event.stopImmediatePropagation()" wire:click.prevent="downloadFiles('{{ $objava-
>id }})'><i class="fa-solid fa-download"></i></a>
                @endif
            </div>
        @endforeach
    @endif

```

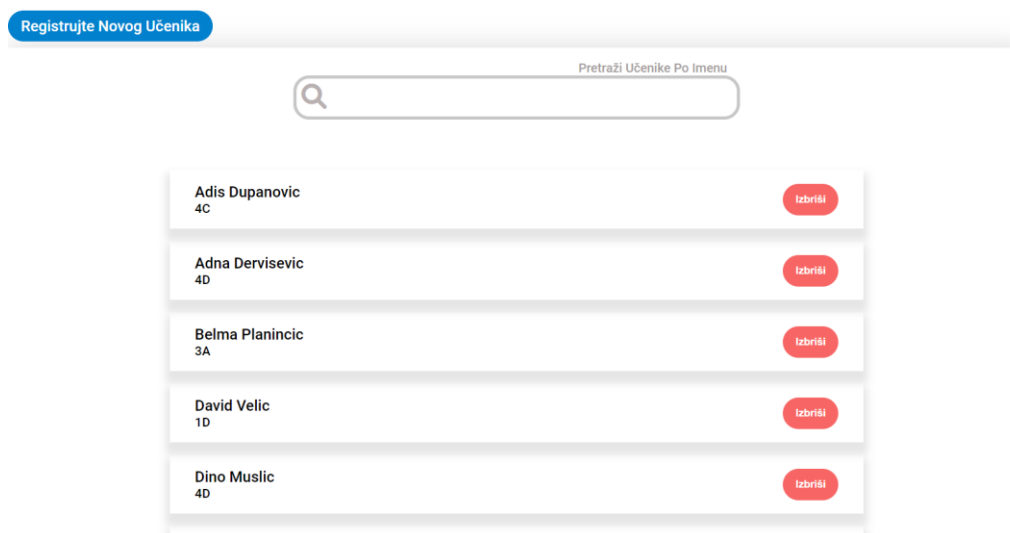
```

        </div>
    @endforeach
    <div style="margin-top: 1rem">
        {{ $objave->links() }}
    </div>
    @else
        <p>Trenutno Nema Objava</p>
    @endif
</div>
</main>

```

U class file-u imamo funkciju mount koja provjerava da li je korisnik prijavljen, te ukoliko nije vodi ga na login stranicu, jer nema pristup objavama. Zatim imamo funkciju updated, koja provjerava korisnikov unos u realnom vremenu, te funkciju post, koja se pokreće klikom na dugme 'Objavi' i koja sprema objavu u bazu podataka ukoliko je unos ispravan. Zatim imamo funkciju deletePost koja izabranu objavu briše iz baze podataka zajedno sa svim njenim datotekama, ukoliko ih ima. Na kraju imamo funkciju downloadFiles, koja sve datoteke izabrane objave sprema u zip te ih preuzima na korisnikov uređaj.

Unutar blade file-a imamo formu za unos objave, koja je vidljiva samo ako je korisnik profesor, iz razloga što samo profesor može objavljivati. Nakon toga se ispisuju sve objave, sa njihovim datotekama, ukoliko ih ima, a ako nema ispisuje se poruka 'Trenutno nema objava'. Također korisnik ima pravo da obriše svoju objavu, pa se provjerava da li objava pripada prijavljenu korisniku, te ako pripada ima mogućnost brisanja.



Slika 24. : Stranica sa popisom učenika

Class file kod za stranicu sa popisom učenika(Slika 24.):

```
class UceniciComponent extends Component
{
    use WithPagination;

    public $search;
    public function render()
    {
        $ucenici = User::orderBy('name', 'ASC')->where('utype', 'UCE')->where('name', 'LIKE', '%'.$this->search.'%')->paginate(10);
        return view('livewire.admin.ucenici-component', ['ucenici' => $ucenici])->layout('layouts.base');
    }

    public function deleteStudent($id)
    {
        $ucenik = User::find($id);
        $ucenik->delete();
        session()->flash('ucenici_poruka', 'Učenik je uspješno izbrisan iz baze podataka');
    }
}
```

Blade file kod za stranicu sa popisom učenika(Slika 24.):

```
<main>
    <div class="sign-in-student">
        <a href="{{ route('registracija') }}" class="sign-in-student-button">Registrujte Novog Učenika</a>
    </div>
    <div class="students">
        <form class="students-form">
            <label>Pretraži Učenike Po Imenu</label>
            <div class="input-icons">
                <i class="fa fa-search fa-2x icon"></i>
                <input type="text" name="search" class="input-field"
wire:model="search">
            </div>
        </form>
        @if ($ucenici->count() > 0)
            <div class="students-list">
```

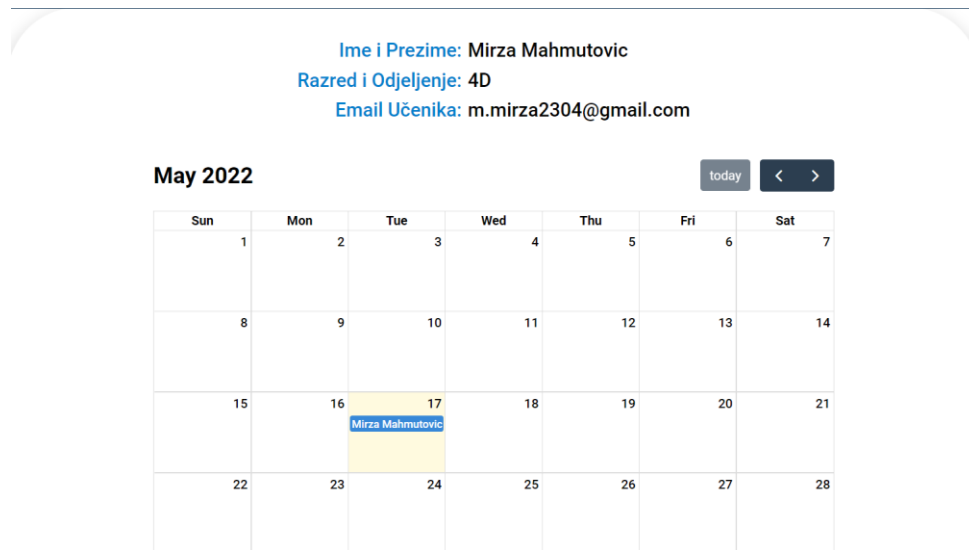
```

        @if (Session::has('ucenici_poruka'))
            <div class="alert alert-success" role="alert">{{
Session::get('ucenici_poruka') }}</div>
        @endif
        @foreach ($ucenici as $ucenik)
            <div class="student">
                <div class="info">
                    <a href="{{ route('ucenik',['ucenik_id' => $ucenik-
>id]) }}" class="info_a">
                        <div class="info_details">
                            <p class="name">{{ $ucenik->name }}</p>
                            <p class="class">{{ $ucenik->razred->name }}</p>
                        </div>
                    </a>
                </div>
                <div><button class="student-button" onclick="confirm('Da
li ste sigurni da želite obrisati ovog učenika?') ||
event.stopImmediatePropagation()" wire:click.prevent="deleteStudent({{ $ucenik-
>id }}">Izbriši</button></div>
            </div>
        @endforeach
        <div style="margin-top: 1rem">
            {{ $ucenici->links() }}
        </div>
    </div>
    @else
        <p>Nema učenika.</p>
    @endif
</div>
</main>

```

Ovo je stranica kojoj može pristupiti samo profesor. U class file-u vidimo funkciju deleteStudent koja briše odabranog učenika iz baze podataka. Pored toga, u render funkciju vidimo da se putem varijable \$search , u koju se sprema korisnikov unos, traže učenici iz baze podataka.

U blade file-u vidimo, prije svega link koji nas vodi na stranicu za registraciju novog učenika(Slika 26.). Zatim vidimo formu za pretragu učenika po bazi podataka, te spisak svih učenika, ukoliko ih ima. Klikom na učenika otvorit će nam se profil učenika(Slika 25.), a pored njegovog imena imamo i dugme za brisanje učenika.



Slika 25. : Stranica sa profilom učenika

Class file kod za stranicu sa profilom učenika(Slika 25.):

```
class UcenikComponent extends Component
{
    public $student_id;
    public $ucenik;
    public $events;

    public function mount($ucenik_id)
    {
        $this->student_id = $ucenik_id;
        $this->ucenik = User::find($this->student_id);
        if($this->ucenik === null || $this->ucenik->utype === 'PROF')
            abort(404);
    }

    public function getEvents()
    {
        $events = Event::select('id', 'title', 'start')->where('user_id', $this->ucenik_id)->get();
        return json_encode($events);
    }

    public function deleteStudent()
    {
        $this->ucenik->delete();
        session()->flash('ucenik_poruka', 'Učenik je uspješno obrisao');
    }
}
```

```

public function render()
{
    $this->events = json_encode($this->ucenik->events);
    return view('livewire.admin.ucenik-component')->layout('layouts.base');
}
}

```

Blade file kod za stranicu sa profilom učenika(Slika 25.):

```

<main>

    <div class="profil-box">
        @if (Session::has('ucenik_poruka'))
            <div class="alert alert-success" role="alert">{{
Session::get('ucenik_poruka') }}</div>
        @endif
        @if ($ucenik)
            <table align="center" cellpadding="6px">
                <tr>
                    <td class="label">Ime i Prezime:</td>
                    <td>{{ $ucenik->name }}</td>
                </tr>
                <tr>
                    <td class="label">Razred i Odjeljenje:</td>
                    <td>{{ $ucenik->razred->name }}</td>
                </tr>
                <tr>
                    <td class="label">Email Učenika:</td>
                    <td>{{ $ucenik->email }}</td>
                </tr>
            </table>
            <div wire:ignore>
                <div id="calendar">

                </div>
            </div>
        @endif
    </div>
    @if ($ucenik)
        <div class="remove-student">
            <a href="#" onclick="confirm('Da li ste sigurni da želite obrisati
ovog učenika?') || event.stopImmediatePropagation()"
wire:click.prevent="deleteStudent()" class="remove-student-button">Izbriši</a>
        </div>
    </if>

```

```

    @endif
</main>
@push('scripts')
<script src='{ asset('assets/fullcalendar/main.js') }'></script>
<script>

document.addEventListener('DOMContentLoaded', function() {
    var calendarEl = document.getElementById('calendar');
    var data = @this.events;
    var calendar = new FullCalendar.Calendar(calendarEl, {
        initialView: 'dayGridMonth',
        editable: false,
        eventInteractive: true,
        selectable: false,
        events: JSON.parse(data)
    });
    calendar.render();
});

</script>
@endpush

```

U class file kodu unutar mount funkcije, se provjerava da li postoji izabrani učenik u bazi podataka. Zatim u funkciji `getEvents` se uzimaju svi datumi kada je učenik ušao na stranicu. I imamo funkciju `deleteStudent` u kojoj se briše učenik.

U blade file kodu prije svega ispisani su osnovni podaci za učenika. Zatim je prikazan kalendar koji radi preko JavaScript, i na njemu su prikazani svi dani kad je učenik ulazio na stranicu.

Slika 26. : Stranica za registraciju novog učenika

Class file kod za stranicu za registraciju novog učenika(Slika 26.):

```
class DodajNovogUcenikaComponent extends Component
{
    public $name;
    public $email;
    public $school_class_id;
    public $password;

    public function registerStudent()
    {
        $this->validate([
            'name' => 'required|max:50',
            'email' => 'required|email|unique:users,email',
            'password' => 'required|min:8|max:20',
            'school_class_id' => 'required'
        ]);
        $ucenik = new User();
        $ucenik->name = $this->name;
        $ucenik->email = $this->email;
        $ucenik->utype = 'UCE';
        $ucenik->password = Hash::make($this->password);
        $ucenik->school_class_id = $this->school_class_id;
        $ucenik->save();
        session()->flash('registracija_poruka', 'Uspješno je registrovan novi
učenik');
    }

    public function generatePassword()
    {
        $this->password = Str::random(8);
    }
}
```

Blade file kod za stranicu za registraciju novog učenika(Slika 26.):

```
@push('scripts')
<script>
    pass = document.getElementById('password');
    passVisible = document.getElementById('pass-visible');
    passHidden = document.getElementById('pass-hidden');
    function switchPassword() {
        if(pass.getAttribute('type') === 'password')
        {
            pass.setAttribute('type', 'text')
        }
    }
}
```



```

        passVisible.style.display = 'block';
        passHidden.style.display = 'none';
    }

    else if(pass.getAttribute('type') === 'text')
    {
        pass.setAttribute('type', 'password')
        passVisible.style.display = 'none';
        passHidden.style.display = 'block';
    }
}
</script>
@endpush

```

Prije svega u class file kodu imamo funkciju update, koja u realnom vremenu provjeravamo unos korisnika. Zatim imamo funkciju RegisterStudent, koja sprema učenika u bazu podataka, ukoliko je unos validan. Zatim imamo funkciju generatePassword, koja se pokreće klikom na dugme 'Generiši Password' i koja je generiše password od 8 karaktera.

Unutar blade file-a imamo formu za unos podataka čiji su inputi povezani sa varijablama iz class file koda. Imamo i javascript kod kojim možemo mijenjati da li je password vidljiv ili ne. Ova stranica sadrži i link 'Pogledaj razrede' koji nas vodi na stranicu sa spiskom svih razreda(Slika 27.).

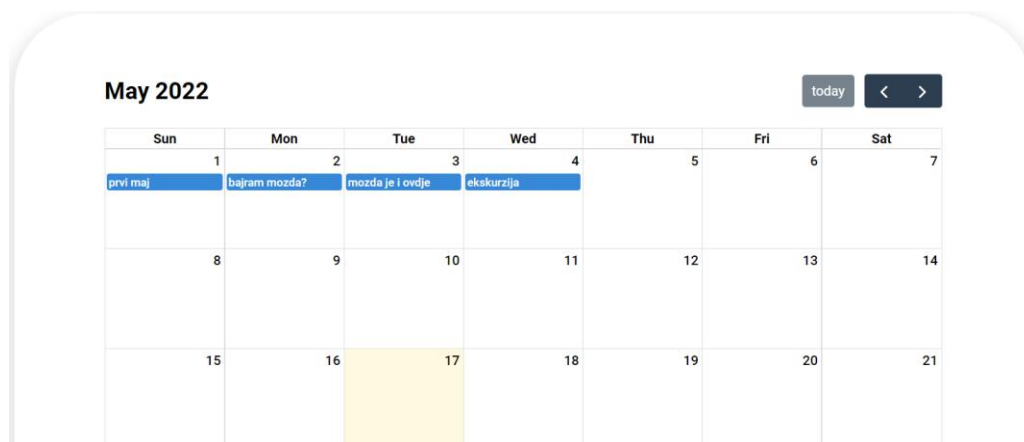


Slika 27. : Stranica sa spiskom razreda

Stranica sa spiskom razreda(Slika 27.) funkcioniše na isti način kao i stranica sa spiskom učenika(Slika 24.). Ima mogućnost pretrage razred, te njihovo brisanje.

*Slika 28. : Stranica za unos novog razreda*

Stranica za unos novog razreda ima formu u kojoj ima samo jedan input kroz koji se unosi razred i odjeljenje te se kroz funkciju provjerava da li je unos validan, te ako jeste sprema se u bazu podataka.



*Slika 29. : Stranica sa kalendarom*

Class file kod za stranicu sa kalendarom(Slika 29.):

```
public function addEvent()
{
    $this->validate([
        'title' => 'required|max:100',
        'class' => 'required',
        'date' => 'required'
    ]);
    $createDate = new DateTime($this->date);
    $strip = $createDate->format('Y-m-d');
```

```

        $class = SchoolClass::find($this->class);
        $event = new ClassEvent();
        $event->title = $this->title;
        $event->start = $strip;
        $event->description = $class->name . " - " . $this->title;
        $event->school_class_id = $this->class;
        $event->save();
        return redirect()->route('kalendar');
    }

    public function deleteEvent($id)
    {
        $event = ClassEvent::find($id);
        $event->delete();
        return redirect()->route('kalendar');
    }

    public function updateEvent($id, $date)
    {
        $event = ClassEvent::find($id);
        $event->start = $date;
        $event->save();
    }
}

```

Ukoliko je na stranicu prijavljen profesor, on može pristupiti kalendaru na kojem su sadržani svi događaji koji su u bazi podataka. Ima također opciju dodavanja, ažuriranja i brisanja događaja. S druge strane, ukoliko je na stranicu prijavljen učenik, on ima mogućnost da pogleda samo kalendar za svoj razred, odnosno, da pogleda događaje samo za svoj razred. Pored toga, učenik nema mogućnost dodavanja, brisanja ili ažuriranja događaja.

#### 4. ZAKLJUČAK

Kroz sami materijal za učenje C++ programskog jezika su obrađene osnovne stvari potrebne da bi se moglo efikasno programirati u C++. Kroz izradu ovog rada, kreirao sam projekat s namjenom da bude što interaktivniji i jednostavniji za shvatiti. Težio sam ka tome da ima što više praktičnih primjera jer se kroz praksu najbolje uči, pogotovo nešto kao što je programiranje. Svoje već unaprijed stečeno znanje o C++ programskom jeziku sam pokušao na najbolji mogući način pokazati kroz materijal koji sam sakupio. Te na kraju ovog rada mogu sa sigurnošću reći da je C++ programski jezik vrijedan učenja, jer onaj ko nauči efikasno programirati u C++, za njega se može reći da zna programirati. Odličan je programski jezik za rad i učenje same programske logike.

Pored samog interaktivnog materijala za učenje C++ programskog jezika, nastojao sam kroz ovaj rad napraviti projekat koji bi predstavljao platformu za lakše predstavljanje budućeg materijala. Web aplikacija je napravljena s ciljem da je koriste profesori i učenici. Smatram da profesoru omogućuje lakšu kontrolu materijala, lakšu komunikaciju sa učenicima, te lakše praćenje aktivnosti učenika. A za učenike smatram da je odlična stvar imati sav materijal na jednom mjestu zajedno sa sporednim stvarima kao što su objave profesora, kalendar razreda i mnoge druge opcije.

## LITERATURA

[1] Learn C++ Programming na adresi:

<https://www.programiz.com/cpp-programming>

[2] Nastavni materijal sa redovne nastave

[3] Šutalo, S. (2010). Osnove programiranja u jeziku C++ na adresi:

<https://sites.google.com/site/sandasutalo/>

[4] Zašto naučiti C++ programski jezik? Na adresi:

<https://machina.academy/machina-blog/zasto-nauciti-c-programski-jezik>

[5] W3Schools – C++ Tutorial na adresi:

<https://www.w3schools.com/cpp/default.asp>

**Datum predaje rada:** \_\_\_\_\_

**Komisija:**

**Predsjednik:** \_\_\_\_\_

**Ispitivač:** \_\_\_\_\_

**Član:** \_\_\_\_\_

**Mišljenje o radu:**

**Datum odbrane rada:** \_\_\_\_\_

**Ocjena o radu** : \_\_\_\_\_ (\_\_\_)