Web-aplikacija za evidenciju učenika u nastavi tjelesnog odgoja

Dulibić, Marta

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 491

WEB-APLIKACIJA ZA EVIDENCIJU UČENIKA U NASTAVI TJELESNOG ODGOJA

Marta Dulibić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 491

WEB-APLIKACIJA ZA EVIDENCIJU UČENIKA U NASTAVI TJELESNOG ODGOJA

Marta Dulibić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

DIPLOMSKI ZADATAK br. 491

Pristupnica:	**Marta Dulibić (0036525495)**
Studij:	Računarstvo
Profil:	Programsko inženjerstvo i informacijski sustavi
Mentor:	prof. dr. sc. Boris Milašinović

Zadatak: **Web-aplikacija za evidenciju učenika u nastavi tjelesnog odgoja**

Opis zadatka:

Potrebno je izraditi web-aplikaciju i odgovarajuće web-servise koji će omogućiti nastavnicima tjelesnog odgoja evidenciju podataka o učenima. Podaci o učenicima mogu se obaviti direktnim unosom kroz aplikaciju ili uvozom iz Excela. U zajedničkim podacima za sve korisnike trebaju se nalaziti uobičajene discipline iz nastave tjelesnog odgoja (npr. bacanje kugle, 60m, 100m, ...) koje pojedini nastavnik može uključiti u evidenciju, ali i definirati vlastite. Aplikacija treba omogućiti unos/uvoz rasporeda koji treba prikazati u obliku kalendara, a podaci iz kalendara se koriste za automatski odabir podataka o predmetu i disciplini prilikom unosa podataka. Osim uobičajene deskriptivne statistike vizualizirane kroz grafove i histograme, potrebno je implementirati i klasteriranje učenika temeljem različitih varijabli (visina, masa, rezultati po disciplinama...). Podaci o učenicima moraju se moći izvesti u Excel i u PDF. Svaki učenik može vidjeti svoje podatke. Prijavu nastavnika i učenika izvesti korištenjem vanjske usluge i protokola OAuth2/OpenID Connect.

Rok za predaju rada: 28. lipnja 2024.

Sadržaj

Uvod

U današnjem digitalnom dobu, sve više procesa i sustava u obrazovanju digitalizirano je radi jednostavnijeg pohranjivanja i arhiviranja podataka. E-dnevnik, aplikacija namijenjena vođenju razredne knjige u elektroničkom obliku, ističe se kao jedna od tih platformi koju danas većina nastavnika u Hrvatskoj koristi svakodnevno u nastavi. Osim redovnih društvenih i prirodoslovnih predmeta, nastavnici Tjelesne i zdravstvene kulture također mogu upisivati svoje ocjene i bilješke, no često se susrećem s potrebom korištenja zasebnih priručnika u papirnatom obliku kako bi detaljnije bilježili neke discipline koje nisu dostupne u E-dnevniku. Takav pristup ponekad izaziva poteškoće jer nastavnici tjelesnog nemaju uvijek isti nastavni plan te im nedostaju ili ne koriste određene kategorije iz tih priručnika. Zabrinjavajući je broj mladih koji većinu vremena provode pred laptopom ili mobitelom, što dovodi do minimalne fizičke aktivnosti, što se jasno očituje i kroz rezultate postignute na nastavi tjelesnog. Često se odlučuju za brze i nezdrave varijante obroka/međuobroka koje nisu dovoljno hranjive te ne osiguravaju dugotrajnu sitost. Općenito, mladi pokazuju nedostatak interesa i brige za vlastito zdravlje. S obzirom na sve navedeno, sve je važnije voditi zapis ili statistiku podataka o razvoju učenika, uključujući njihova morfološka obilježja te funkcionalne i motoričke sposobnosti. Ti podaci mogu biti od ključne važnosti prilikom istraživanja čiji zaključci bi se mogli iskoristiti za promicanje veće brige oko zdravlja i fizičke spreme mladih ljudi. Kao krajnji cilj, želja je poboljšati inicijative usmjerene na poticanje djece na zdraviji način života.

Kroz istraživanje koje će biti prezentirano u nastavku rada, korišteni su stvarni podaci učenika prvog razreda srednje škole. Provedena je analiza i klasteriranje podataka dobivenih iz inicijalnih testiranja na početku školske godine, fokusirajući se na njihove funkcionalne i morfološke sposobnosti. Dobiveni rezultati su identificirali parametre koji su međusobno korelirani te omogućili podjelu učenika prema kojoj nastavnici mogu prilagoditi nastavu kako bi izvukli maksimum iz svakog pojedinca ili identificirali te riješili određene probleme kod pojedinih učenika. Stoga, kao moguće rješenje, u sklopu ovog diplomskog rada stvorena je platforma slična e-Dnevniku, nazvana e-TZK Dnevnik, koja je namijenjena isključivo nastavnicima tjelesne kulture i njihovim učenicima. Glavna funkcionalnost ove aplikacije je pružanje mjesta gdje će nastavnici tjelesne kulture moći na

jednom mjestu prikupljati sve podatke o svakom učeniku po razredima te imati pregled nad poviješću njihovih rezultata. Time će im biti olakšano praćenje napretka učenika te će imati bolji uvid u njegovu tjelesnu sliku i zdravstveno stanje. U slučaju potrebe, ovi podaci će im omogućiti da poduzmu odgovarajuće korake kako bi poboljšali tjelesnu kondiciju i zdravlje učenika.

1. Analiza i klasteriranje podataka o motoričkim i funkcionalnim sposobnostima učenika

1.1. Pregled analize

Cilj provedene analize je bilo istražiti povezanost između morfoloških obilježja, motoričkih i funkcionalnih sposobnosti učenika srednje škole. Za testnu skupinu odabrane su učenici/e 1.razreda srednje škole (15 godina).

Kao morfološka obilježja uzete su visina, težina, opseg podlaktice, opseg struka te indeks tjelesne mase (engl. *ITM*). Za motoričke i funkcionalne sposobnosti prikupljeni su podaci kroz inicijalne testove koje se provode u školama jednom na početku i kraju godine:

1. **Taping** (broj ponavljanja)
2. **Skok u dalj** (cm)
3. **Podizanje trupa** (cm)
4. **Pretklon** (cm)
5. **Izdržaj u visu** (u sekundama)

Kao ciljevi analize postavljeno je sljedeće:

1. Utvrditi korelacije između pojedinih parametara kako bismo identificirali relevantne faktore.
2. Primijeniti analizu svojstvenih komponenti (PCA - Principal Component Analysis) radi smanjenja dimenzionalnosti podataka.
3. Provesti klastersku analizu kako bismo identificirali slične grupe učenika prema njihovim sposobnostima.

Koraci analize su redom: učitavanje podataka učenika, korelacija između parametara te primjena PCA i algoritma k-srednjih vrijednosti.

1.2. Učitavanje podataka

Korišteni podaci su pripremljeni u obliku *.csv* datoteke s sljedećim stupcima redom:

1. **Razred**
2. **Dob** (godina)
3. **Spol** (m/ž)
4. **Visina** (cm)
5. **Težina** (kg)
6. **OS** (cm) - opseg struka
7. **OP** (cm) - opseg podlaktice
8. **ITM** - indeks tjelesne mase
9. **Taping** (broj ponavljanja)
10. **Skok u dalj** (cm)
11. **Podizanje trupa** (broj ponavljanja)
12. **Pretklon** (cm)
13. **UZV** (s) - izdržaj u visini

Za učitavanja podataka korištena je Python biblioteka *Pandas* koja omogućava čitanje podataka iz .csv datoteka:

```
# Import potrebnih biblioteka
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
%matplotlib inline

# Učitavanje podataka
file_path = '/Users/mdulibic/Downloads/podaci.csv'
data = pd.read_csv(file_path)
```

Kôd 1.2.1 – Program za učitavanje podataka iz .csv datoteke

Podaci koji su nedostajali određenim učenicima su označeni s "-" i uklonjeni prilikom učitavanja podataka. Stupci *Razred* i *Dob(Godina)* su izbrisani jer su ispitanici učenici prvog razreda srednje škole.

Konačno, prvi redovi podataka su ispisani radi preglednosti u analizi:

```
# Filtriranje nečistih podataka
data.replace('-', np.nan, inplace=True)
data.replace('- ', np.nan, inplace=True)
data['Spol (M/Ž)'] = data['Spol (M/Ž)'].map({'M': 0, 'Ž': 1})
del data["Razred"]
del data['Dob (godina)']
data = data.astype(float)

# Pregled prvih redaka podataka
data.head()
```

Kôd 1.2.2 – Program za filtriranje nečistih podatka i pregled prvih redaka podataka

	Spol (M/Ž)	Visina (cm)	Težina (kg)	OS (cm)	OP (cm)	ITM	Taping	Skok u dalj	Podizanje trupa	Pretklon	Izdržaj UVZ
0	1.0	164.0	56.0	70.0	22.0	20.8	36.0	176.0	49.0	84.0	46.0
1	1.0	173.0	56.0	66.0	20.0	18.7	36.0	188.0	32.0	100.0	10.0
2	1.0	166.0	55.0	70.0	22.0	19.2	38.0	186.0	43.0	82.0	30.0
3	1.0	160.0	60.0	75.0	24.0	22.9	37.0	NaN	43.0	82.0	30.0
4	1.0	181.0	63.0	69.0	24.0	19.4	38.0	175.0	44.0	91.0	21.0

Slika 1.2.1 – Ispis prvih redaka podataka

1.3. Korelacija između parametara

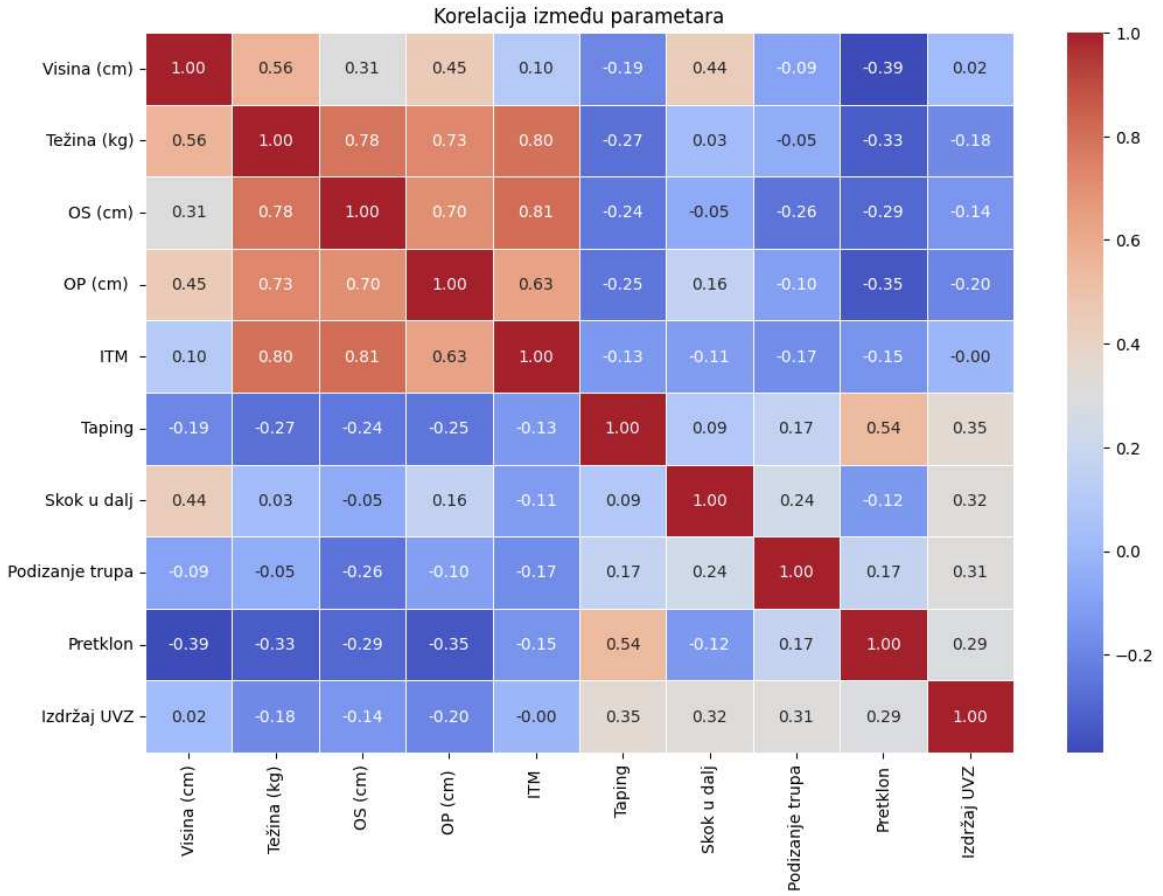
Kao sljedeći korak, provesti ćemo analizu korelacija između parametara. Za izračunavanje korelacijske matrice koristi se metoda *".corr()"* koja se nalazi u biblioteci *Pandas*. Ta metoda je dio Pandas *DataFrame* objekta i koristi se za izračunavanje *Pearsonove korelacije* između svih numeričkih varijabli u DataFrame-u. *Pearsonov koeficijent korelacije* koristi se za procjenu linearnih veza između varijabli s kontinuiranim normalno distribuiranim podacima. Vrijednost koeficijenta varira između *+1* (savršena pozitivna korelacija) i *-1* (savršena negativna korelacija), gdje predznak označava *smjer veze* (pozitivna ili negativna) [4].

Za vizualizaciju korelacija korištena je toplinska karta (engl. *heat map*) gdje svaka ćelija prikazuje vrijednost Pearsonovog koeficijenta korelacije između dviju varijabli. Funkcija *"heatmap"* se nalazi u Python biblioteci *Seaborn*.

```
# Analiza korelacija
correlation_matrix = data.drop('Spol (M/Ž)', axis=1).corr()
```

```
# Vizualizacija korelacija pomoću heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=.5)
plt.title('Korelacija između parametara')
plt.show()
```

Kôd 1.4.1 – Program za analizu korelacija među parametrima



Slika 1.4.1 – Prikaz toplinske karte korelacija među parametrima

U obzir ćemo uzeti samo srednje i snažne korelacije:

- **$0.3 \leq |r| < 0.5$**: *Srednja korelacija*
- **$|r| \geq 0.5$**: *Snažna korelacija*

Pozitivne korelacije:

- Visina i težina
- Opseg podlaktice i visina
- Težina i opseg struka, opseg podlaktice i ITM (*potvrda EDA analize*)
- Visina i skok u dalj (*potvrda EDA analize*)
- Opseg podlaktice i pretklon
- Taping i pretklon (snažna korelacija)

Negativne korelacije:

- Visina i pretklon *(potvrda EDA analize*)
- Težina i pretklon
- Opseg podlaktice i pretklon

Ove korelacije mogu biti korisne nastavnicima u prilagodbi nastave i evaluaciji učenika.

Na primjer, pozitivne korelacije mogu pomoći u stvaranju strategija prilagođene nastave kako bi se učenicima pomoglo u postizanju maksimalnih rezultata. S druge strane, negativna korelacija također može pružiti smjernice za identifikaciju područja koja zahtijevaju posebnu pažnju i podršku.

1.4. Primjena PCA te K-means clustering algoritma

U ovom dijelu bavit ćemo se provođenjem PCA analize te K-means clustering algortima.

1.4.1. PCA (Principal Component Analysis)

Principal Component Analysis (PCA) je tehnika smanjenja dimenzionalnosti koja se koristi za transformaciju originalnih značajki u novi skup značajki, tzv. glavne komponente. Glavna svrha PCA-a je smanjiti broj dimenzija podataka, zadržavajući pritom što veći postotak varijacije [5]. U našoj analizi, PCA će nam pomoći smanjiti dimenzionalnost morfoloških i motoričkih podataka, čime ćemo zadržati ključne informacije i olakšati interpretaciju rezultata.

1.4.2. K-Means clustering algoritam

K-Means clustering je algoritam za grupiranje podataka koji dijeli skup podataka u klastere na temelju sličnosti između podataka. Ovaj algoritam zahtijeva unaprijed definiran broj klastera (*k*), a zatim iterativno pridružuje podatke klasterima kako bi minimizirao varijancu

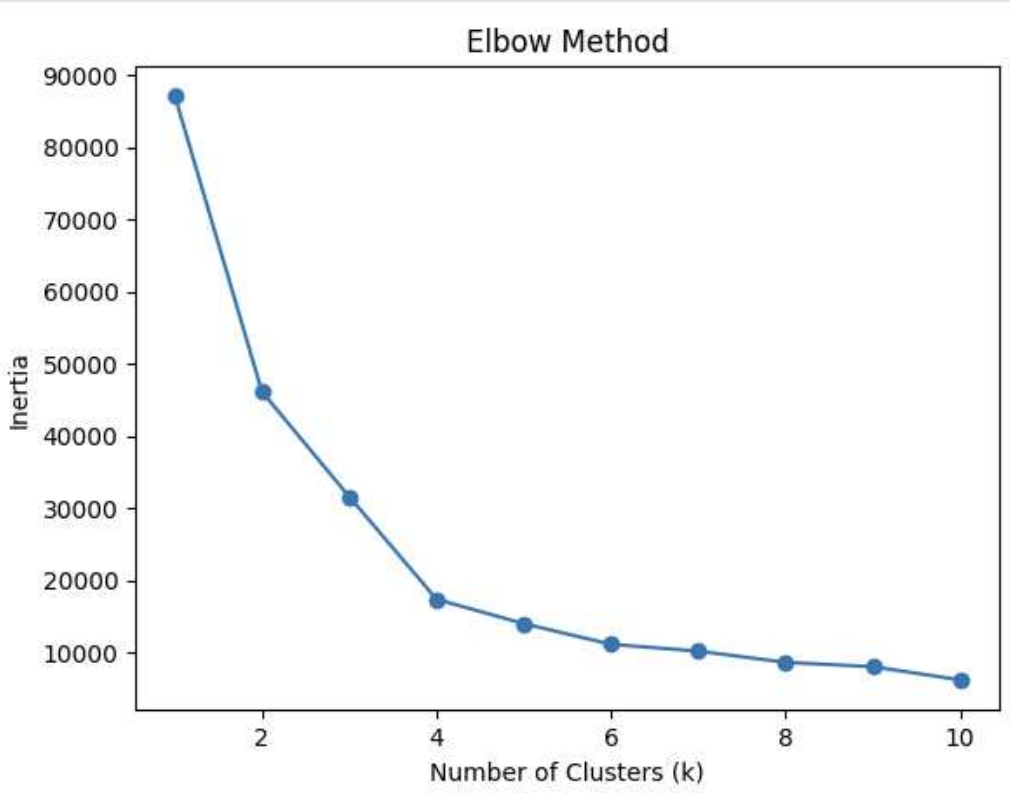
unutar svakog klastera [6]. U kontekstu naše analize, želimo koristiti K-Means clustering kako bismo grupirali učenike prema njihovim morfološkim i motoričkim sposobnostima. Ova grupiranja omogućit će nam identifikaciju sličnosti i razlika među različitim skupinama učenika, što može biti od pomoći u prilagodbi programa nastave tjelesnog.

1.4.3. Odabir broja klastera kroz Elbow metodu

Pri odabiru broja klastera za K-Means clustering, jedna od često korištenih metoda je *Elbow metoda*. Ova metoda pomaže identificirati optimalan broj klastera tako da se promatra inercija u ovisnosti o broju klastera. *Inercija* predstavlja mjeru udaljenosti između podataka unutar istog klastera. Što je inercija manja, to su podaci unutar klastera sličniji [7]. Elbow metoda traži točku na grafu gdje dodavanje dodatnih klastera prestaje značajno smanjivati inerciju, čime se stvara *"lakat" (elbow)* na grafu.

```
def elbow_method(inp_data, pca_n=2):
    data = inp_data.copy()

    # Primjena PCA analize na ulazne podatke
    pca = PCA(n_components=pca_n)
    data_pca = pca.fit_transform(data)

    inertias = []

    # Računanje broj klastera uz pomoć Elbow Metode
    for k in range(1, 11):
        kmeans = KMeans(n_clusters=k, random_state=42)
        kmeans.fit(data_pca)
        inertias.append(kmeans.inertia_)

    plt.plot(range(1, 11), inertias, marker='o')
    plt.xlabel('Number of Clusters (k)')
    plt.ylabel('Inertia')
    plt.title('Elbow Method')
    plt.show()
```

Kôd 1.5.1 – Program za pronalazak broj klastera pomoću Elbow metode



Slika 1.5.1 – Graf s prikazom inercije kod Elbow Metode

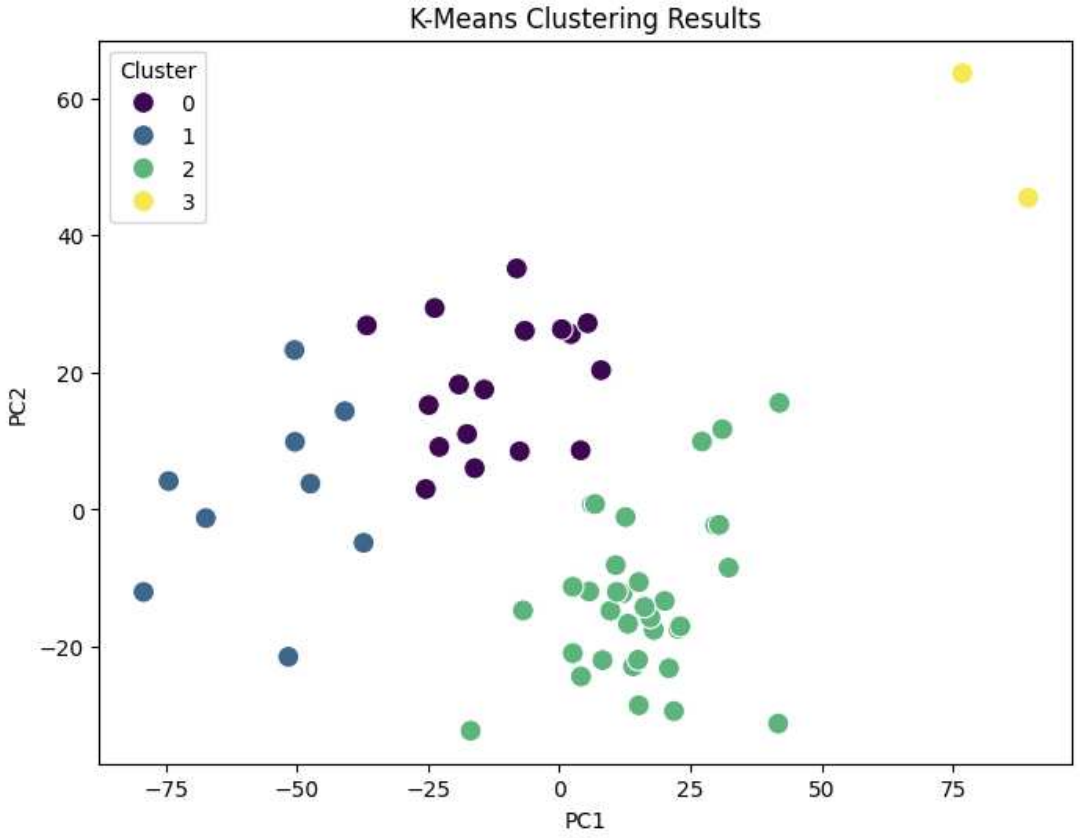
Provođenjem Elbow metode utvrdili smo da je idealan *broj klastera 4*.

1.4.4. Primjena K-Means clustering algoritma

Nakon utvrđivanja broja klastera, pozivamo funkciju *kmeans* s četiri klastera. Funkcija *"fit_transform"* nad objektom *PCA* za smanjenje dimenzionalnosti podataka se nalazi u Python biblioteci *sklearn.decomposition*, dok klasteriranje provodimo pozivom funkcije "*fit_predict"* nad objektom *Kmeans* koja se nalazi u Python biblioteci *sklearn.cluster.*

```
def kmeans(inp_data, res_k=4, pca_n=2):
    data = inp_data.copy()

    pca = PCA(n_components=pca_n)
    data_pca = pd.DataFrame(pca.fit_transform(data),        columns=[f'PC{i}' for i in
range(1, pca_n+1)])

    kmeans = KMeans(n_clusters=res_k, random_state=42)
    data_pca['Clusters'] = kmeans.fit_predict(data_pca)
    return data_pca
```

Kôd 1.5.2 – Program za klasteriranje učenika metodom K-means clustering



Slika 1.5.2 – Rezultati klasteriranje učenika metodom K-Means clustering

Primjenom PCA i K-Means algoritma, jasno su identificirana 4 klastera:

1. Klaster 0 (0 ženskih ispitanika, 17 muških ispitanika)
2. Klaster 1 (0 ženskih ispitanika, 9 muških ispitanika)
3. Klaster 2 (33 ženskih ispitanika, 1 muških ispitanika)
4. Klaster 3 (1 ženskih ispitanika, 1 muških ispitanika)

Primjećujemo da se dva ispitanika izdvajaju od ostalih i grupirani su u zasebni klaster. Također, primjećujemo da je jedan muški ispitanik grupiran sa ženskim ispitanicima. Za ispitanika "broj 51" u klasteru 3 pretpostavljamo da je rezultat podizanja trupa krivo zapisan ili je ispitanik iznimno ispod granice (vrijednost 9 je jako niska u usporedbi s ostalim), dok je ispitanik 18 ženska osoba s većom tjelesnom težinom (ITM) u usporedbi s ostalim ženskim ispitanicima, što može biti razlog njezina grupiranja u zasebni klaster.

```
whole_data.loc[out_data[out_data['Clusters']==3].index]
```

Kôd 1.5.3 – Program za izvlačenje podataka o učenicima iz klastera 3

	Spol (M/Ž)	Visina (cm)	Težina (kg)	OS (cm)	OP (cm)	ITM	Taping	Skok u dalj	Podizanje trupa	Pretklon
18	1.0	171.0	71.0	90.0	28.0	24.3	30.0	100.0	20.0	50.0
51	0.0	176.0	86.0	100.0	37.5	28.1	26.0	110.0	9.0	40.0

Slika 1.5.2 – Podaci o učenicima iz klastera 3

Iz podataka za muškog ispitanika koji je grupiran sa ženskim ispitanicima u klasteru 2, možemo pretpostaviti da je to zbog njegove smanjene tjelesne težine (49 kg) u usporedbi s ostalim muškim ispitanicima.

```
tmp = whole_data.loc[out_data[out_data['Clusters']==2].index]
tmp[tmp['Spol (M/Ž)']==0]
```

Kôd 1.5.3 – Program za izvlačenje podataka o muškim ispitanicima iz klastera 2

	Spol (M/Ž)	Visina (cm)	Težina (kg)	OS (cm)	OP (cm)	ITM	Taping	Skok u dalj	Podizanje trupa	Pretklon
55	0.0	175.5	49.0	55.5	23.0	16.6	25.0	188.0	33.0	62.0

Slika 1.5.2 – Podaci o učeniku iz klastera 2

1.5. Zaključak analize

Uvidom u korelaciju između morfoloških, funkcionalnih i motoričkih sposobnosti, nastavnici mogu identificirati veze između različitih aspekata učenikovog razvoja. To im omogućuje da ciljano rade na poboljšanju određenih vještina ili sposobnosti kako bi učenici postigli bolje rezultate u drugim područjima. Također, otkrivanje parametara koji su međusobno povezani može poslužiti kao upozorenje za nastavnike da razgovaraju s roditeljima o potencijalnim problemima koji se mogu javiti u budućnosti.

Identifikacijom klastera učenika olakšava im se ocjenjivanje i odabir načina rada, te organizacija timskih aktivnosti ili natjecanja. Također, pomaže im u prepoznavanju učenika koji bi mogli trebati dodatnu podršku ili prilagođeni pristup.

Kao zaključak, provedba ovakve analize može pomoći nastavnicima tjelesnog i zdravstvenog odgoja prilikom kreiranja nastavnog sadržaja kako bi ga maksimalno prilagodili svojim učenicima i njihovim ciljevima.

2. Specifikacija programske potpore

Web aplikacija *e-TZK dnevnik* ima tri tipa korisnika: administrator, nastavnik i učenik. U sljedećim potpoglavljima bit će opisani korisnički zahtjevi koje aplikacija treba ispuniti za svaku od korisničkih uloga.

2.1. Korisnički zahtjevi

Za korištenje aplikacije, korisnik se mora prijaviti u sustav kako bi mu se dodijelila ispravna uloga (administrator, nastavnik ili učenik). Korisnici se ne mogu samostalno registrirati u sustav već moraju biti dodani u sustav od strane administratora.

2.1.1. Administrator

Administrator može registrirati nove korisnike u sustav manualnim unosom potrebnih podataka (spol, ime, prezime, korisničko ime, email, rola, škola, razred/i te lozinka) ili uvozom njihovih podataka kroz *CSV* datoteku. Također, administrator može dodavati i nove škole u sustav uvozom podataka kroz *CSV* datoteku. Sučelje aplikacije omogućava administratoru pregled svih učenika i nastavnika po školama uz mogućnost filtriranja, dodavanje novih aktivnosti te dodavanje novih podaktivnosti za postojeće aktivnosti u sustavu. Administrator ima opciju brisanja i ažuriranja podataka svakog korisnika. Nadalje, može izmijeniti osnovne podatke učenika i njihova morfološka obilježja (visina, težina). Administrator također može dodati u sustav nove nastavne aktivnosti te dodati nove podaktivnosti za postojeće aktivnosti u sustavu.

2.1.2. Nastavnik

Nastavnik ima uvid u raspored nastave u obliku kalendara. Kroz kalendar može pregledati sve nadolazeće nastavne sate te njihove detalje, uključujući naziv i opis nastavnog sata, odabranu aktivnost (podaktivnost), razred i vrijeme održavanja. Također, nastavnik ima mogućnost ažuriranja podataka o nastavnom satu te njegovog brisanja iz sustava. Novi satovi se mogu dodati u kalendar putem zasebne forme, odabirom odgovarajućeg vremenskog okvira unutar samo kalendara ili uvozom cijelog rasporeda nastave putem CSV datoteke.

Prilikom izrade novog nastavnog sata, nastavnik može pridijeliti određenu oznaku (labelu) koja pomaže u grupiranju sličnih nastavnih satova te postoji mogućnost dodavanja vlastitih oznaka u sustav, koje su dostupne samo njima.

Nastavniku je uz planer prikazan prečac do nadolazećeg nastavnog sata. Odabirom prečaca, nastavniku se otvara opcija upisivanja rezultata za učenike razreda za aktivnost koja je upisana za taj nastavni sat. Nastavnik ima opciju preskočiti učenike koji u tom trenutku nisu prisutni ili su onemogućeni izvršavati zadatke tijekom sata. Aplikacija omogućuje nastavnicima pregled svih njihovih razreda. Za svaki razred, aplikacija pruža pregled popisa učenika s mogućnošću filtriranja. Za svakog učenika nastavnik ima opciju ažuriranja podataka, pregled detalja o učeniku, što uključuje statistiku učenika po postignutim rezultatima u svakoj školskoj godini.

Nadalje, za svaki razred nastavnik može pregledati njihove rezultate s mogućnošću filtriranja te ima mogućnost izvoza tih rezultata u *Excel* format. Osim pregleda postojećih rezultata, nastavnik ima mogućnost unosa novih rezultata po razredu odabirom željene aktivnosti i podaktivnosti te unosom ostvarenog rezultata i mjerne jedinice tog rezultata. Uz popis učenika i njihove rezultate, nastavnik ima mogućnost uvida u deskriptivnu statistiku razreda vizualiziranu kroz grafove i histograme. Za svaki razred omogućen mu je pregled histograma prosječnih rezultata razreda po odabranoj aktivnosti/podaktivnosti te popis rezultata učenika poredanih po uspješnosti u toj kategoriji. Također, ima mogućnost provođenja klasterske analize nad razredom te izvoza njenog izvješća u *PDF* formatu. Nastavnik može pregledati svoje generalne informacije (ime, prezime, email, škola, razredi kojima predaje). Naposljetku, ima mogućnost dodavanja novih aktivnosti i podaktivnosti za postojeće aktivnosti kao i administrator.

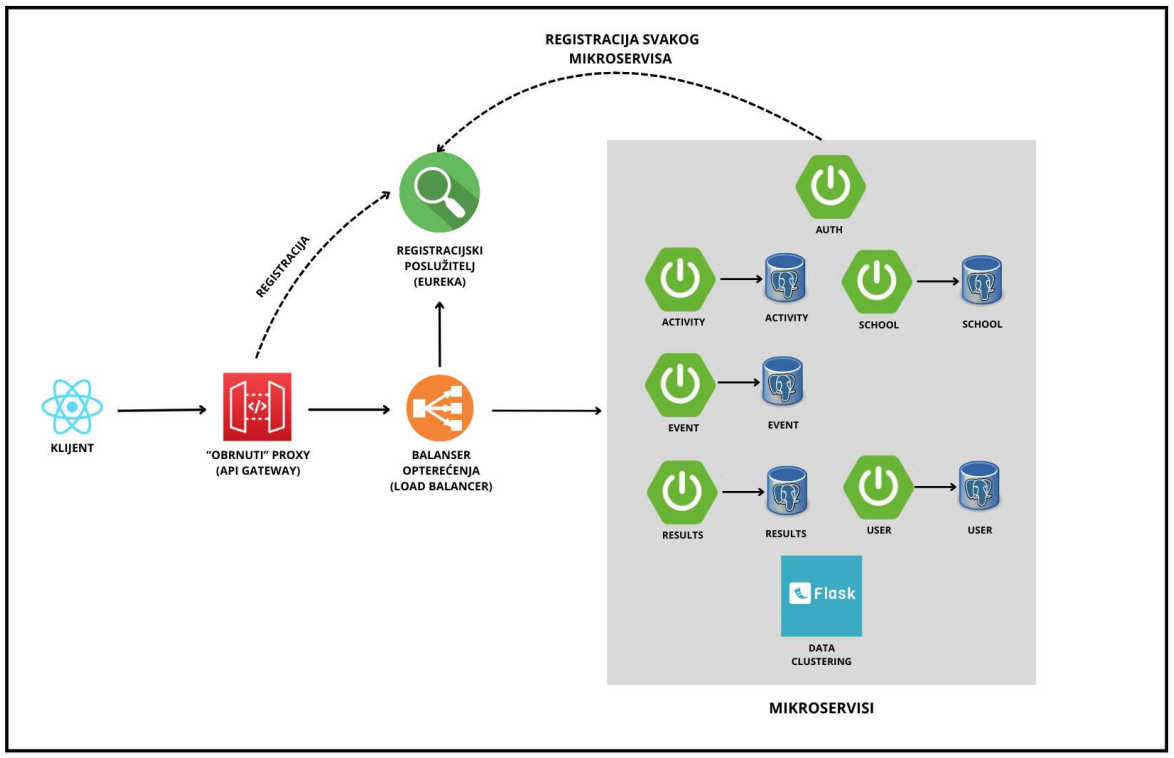
2.1.3. Učenik

Učenik ima mogućnost pregled rasporeda nastave u obliku kalendara. Kroz kalendar može pregledati sve svoje nadolazeće nastavne sate i njihove detalje, uključujući naziv i opis nastavnog sata, aktivnost/podaktivnost te vrijeme održavanja.Također, učenik ima mogućnost pregleda svog profila i njegovih detalja, kao i nastavnik.

3. Arhitektura sustava

Glavne komponente arhitekture sustava su:

1. Klijent (**React**)
2. Poslužitelj (**Mikroservisna arhitektura** sa **obrnutim proxyem** i **registracijskim poslužiteljem** - ***Spring Boot/Flask*** mikroservisi, ***Spring Cloud Eureka*** za registracijski poslužitelj, ***Spring Cloud Routing Gateway*** za "obrnuti" proxy)
3. Baza podataka (**PostgreSQL**)



Slika 3.1. - Arhitektura sustava

3.1. Klijent

React

Za izradu klijentske aplikacije odabrana je biblioteka *React*. *React* je besplatna i otvorena Javascript biblioteka za izradu korisničkih sučelja temeljena na komponentama. Ključna prednost Reacta nad nekim drugim alternativama je u tome što ponovno renderira samo

one dijelove stranice koji su se promijenili, izbjegavajući nepotrebno ponovno renderiranje nepromijenjenih DOM elemenata. [5]

Aplikacija je pisana u *Typescript*-u, programskom jeziku koji proširuje *Javascript* dodavanjem statičkog tipiziranja koji pomaže pri otkrivanju i ispravljanju grešaka prilikom razvoja aplikacije.

Vite

Kao razvojni poslužitelj odabran je *Vite*, alat za konfiguraciju web aplikacija koji olakšava razvoj jer rješava potrebne postavke prije kretanja u razvoj aplikacije kao što su pokretanje razvojnog okruženje, prevođenje koda u sintaksu koju stariji preglednici mogu razumjeti, postavljanje poslužitelja s hot reloading-om koji automatski osvježava stranicu pri promjenama koda, itd.

Vite, za razliku od svog prethodnika *Create React App* (CRA), ne gradi ponovno cijelu aplikaciju prilikom svake promjene, što ga čini dosta bržim. To je postignuto podjelom aplikacije u dvije kategorije: *ovisnosti* i *izvorni kod*. Ovisnosti su stvari koje se jako rijetko mijenjaju tijekom razvoja, dok se izvorni kod poziva samo kad je potreban. [6]

Tailwind

Za dizajn i izgradnju web aplikacije, korišten je okvir *Tailwind*. Tailwind je CSS okvir, koji za razliku od *bootstrap*-a ili čistog *"vanilla"* CSS-a, sadrži već predefinirane klase koje se koriste direktno u HTML-u. Prije korištenja u svom projektu, potrebno ga je instalirati kao npm paket i kreirati *tailwind.config.js* datoteku.

Datoteka *tailwind.config.js* sadrži putanje do datoteka gdje će on biti omogućen. Dodatno, mogu se konfigurirati boje, margine, teme, itd. [7]

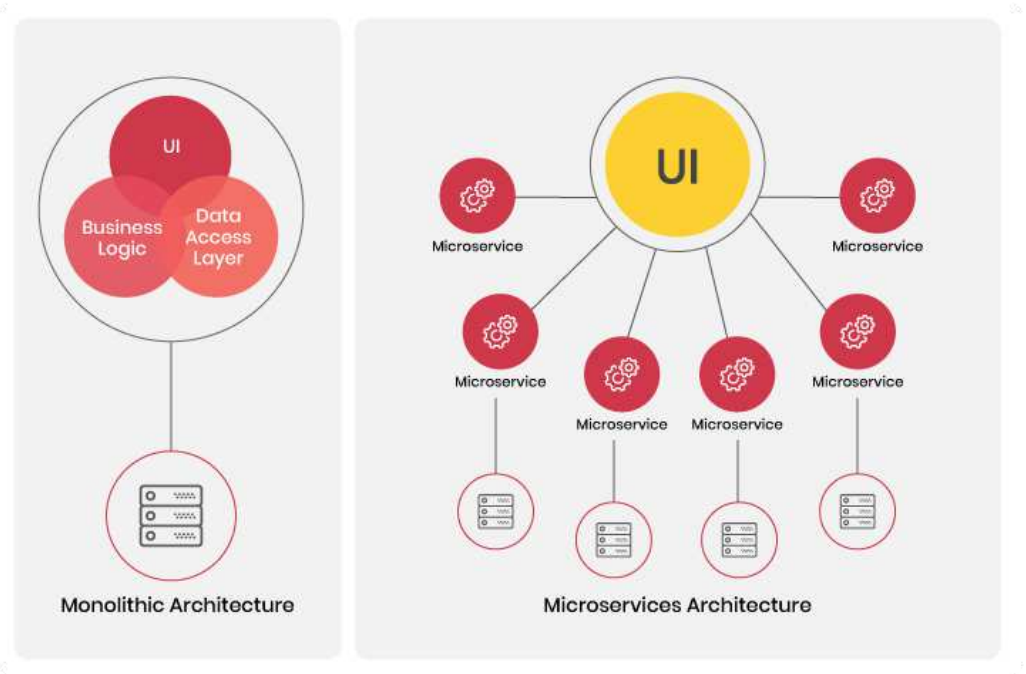
Shadcn/UI

Za određene UI komponente u projektu korištene su komponente iz biblioteke "Shadcn/UI". "Shadcn/UI" nije tipična UI biblioteka kao što su Material-UI ili Radix jer datoteke njenih komponenti nisu uvezene kao vanjska ovisnost (dependency), već se nalaze unutar same strukture projekta. Komponente su izgrađene na temelju Radix biblioteke, pružajući osnovne elemente poput tablica, formi, kartica i drugih vizualnih elemenata.

3.2. Poslužitelj

Mikroservisna arhitektura

Mikroservisna arhitektura je pristup razvoju softvera u kojem se aplikacija razbija na manje, neovisne servise. Za razliku od monolitne arhitekture, svaki mikroservis enkapsulira specifičnu poslovnu logiku ili funkcionalnost aplikacije. Prednosti mikroservisne arhitekture su međusobna slaba povezanost (*"loosely coupling"*), neovisno isporučivanje, skalabilnost i izolacija podataka [8].



Slika 3.2. - Pregled monolitne arhitekture naspram mikroservisne

Za izradu mikroservisa moguće je koristit bilo koji programski jezik sve dok drugi mikroservisi mogu komunicirati s njim.

Za potrebe ovog projekta, za izradu mikroservisa korišteni su programski okviri Spring Boot i Flask. Za mikroservise koji upravljaju školama, aktivnostima, rezultatima, statistikom te korisnicima, odabran je Spring Boot zbog jednostavnosti razvijanja CRUD operacija te dobro dokumentiranih i ispitanih vanjskih ovisnosti za generiranje PDF izvješća te izvoza podataka u *CSV/Excel* format. Za autentifikacijski servis također je odabran Spring Boot zbog korištenja vanjske ovisnosti Spring Security, koja omogućuje

integraciju generiranja i provjere sesija korisnika s JWT tokenom, povezivanje na OAuth2 autorizacijski server za prijavu s Google računom te autorizaciju prilikom pristupanja ruta sa strane klijenta. Također, *"obrnuti"* proxy i registracijski poslužitelj implementirani su kao Spring Boot aplikacije, što čini logičnim da se i ostali mikroservisi implementiraju u toj tehnologiji.

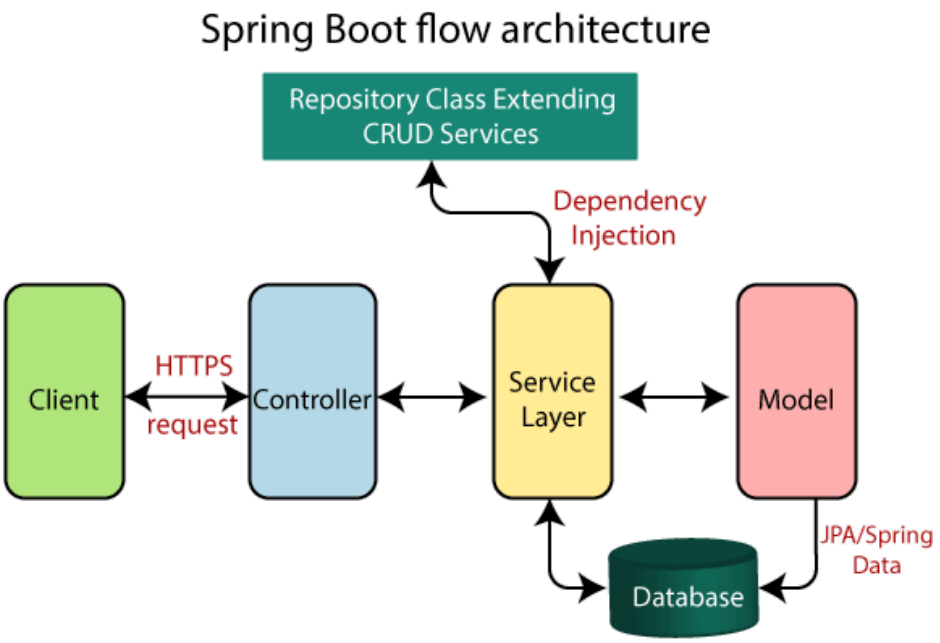
Za mikroservis za klasteriranje podataka odabran je Flask zbog prednosti koje pruža Python. Java biblioteke ne nude podršku kao Python za analizu i vizualizaciju podataka. Python ima bogat ekosustav biblioteka poput *pandas*, *numpy, matplotlib*, *seaborn* i *scikit-learn*, koje su ključne za kreiranje i prikazivanje grafova te analizu podataka unutar mikroservisa. Dodatno, Flask omogućuje jednostavnu registraciju na *Spring Cloud Eureka* poslužitelj, čime se olakšava integracija u postojeći sustav mikroservisa.

Spring Boot

Spring Boot je razvojni okvir otvorenog koda temeljen na Javi koji služi za razvijanje samostalnih i produkcijski spremnih aplikacija.

Jedna od glavnih prednosti korištenja Spring Boot-a je što zahtijeva minimalno konfiguracije prije pokretanja (nema potrebe za složenom XML konfiguracijom) što ga čini jednostavnim za korištenje, povećava produktivnost i ubrzava sam razvojni proces Spring aplikacija [11].

Za potrebe ovog projekta, za arhitekturu Spring Boot mikroservisa odabrana je *MVC* (engl. Model-View-Controller) arhitektura.



Slika 3.3. - Spring Boot MVC arhitektura [15]

Flask

Flask je razvojni okvir napisan u Pythonu koji omogućuje jednostavan razvoj web aplikacija. Baziran je na WSGI (eng. Web Server Gateway Interface) okviru, koji pruža standardiziranu specifikaciju za komunikaciju web poslužitelja i Python aplikacija.

Klasificira se kao *"mikrookvir"* jer ne uključuje sloj apstrakcije baze podataka - ORM (eng. Object Relational Manager) niti druge komponente koje su već sadržane u nekim drugim razvojnim okvirima.

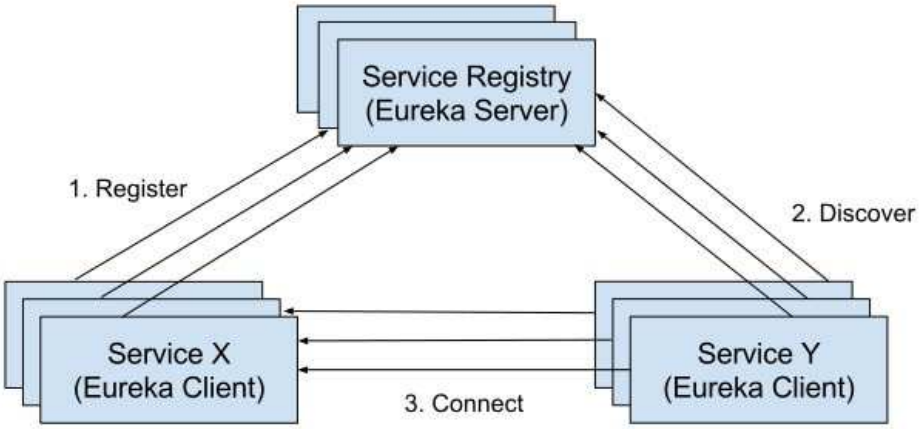
Međutim, Flask podržava proširenja koja mogu dodati funkcionalnosti aplikacije kao da su implementirane u samom Flasku, kao što su objektno-relacijsko mapiranje, autorizacija, i mnoge druge funkcionalnosti. [13]

Registracijski poslužitelj

Kako bi omogućili sustavu da automatski otkrije nove mikroservise i kako bi sustav bio transparentniji, skalabilniji i tolerantniji na greške, sustav koristi registracijski poslužitelj koji omogućuje mikroservisima da se međusobno otkriju.

Glavna ideja registracijskog poslužitelja je da djeluje kao središnji registar za sve usluge te nudi pregled njihovog trenutnog statusa. Klijenti mogu postavljati upite registracijskom poslužitelju kako bi odredili lokaciju određene usluge [9], dok ga mikroservisi koriste za

međusobno pronalaženje i komunikaciju bez statički definirane IP adrese i port-a. Jedino što je "fiksna točka" je što se svaka usluga mora registrirati u registar [10].



Slika 3.4. - Registracija i međusobno otkrivanje usluga putem registracijskog poslužitelja [16]

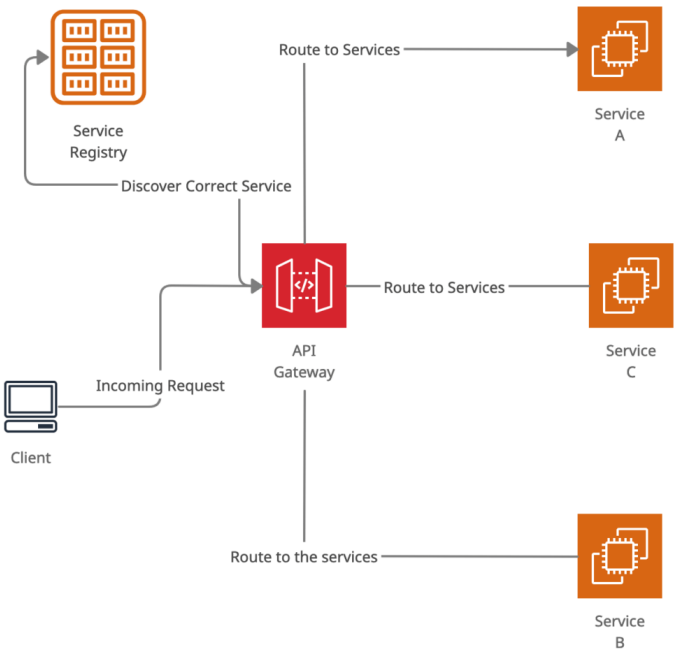
Za potrebe ovog projekta, kao registracijski poslužitelj korišten je Spring Cloud Netflix - Eureka Server kojeg je razvio Netflix te se koristi za otkrivanje usluga mikroservisa u distribuiranom okruženju.

Api Gateway - "obrnuti" proxy

Za upravljanje HTTP prometom između mikroservisa i klijenta, korišten je "obrnuti" proxy (eng. "reverse" proxy). Još ga nazivamo i Api Gateway jer djeluje kao ulazna točka za sve dolazne API zahtjeve sa klijenta prema poslužitelju. Sadrži popis dopuštenih IP adresa, politike ponovnih pokušaja te pravila za ograničavanje i regulaciju zahtjeva.

Također, API Gateway se može koristiti i kao autentifikacijski/autorizacijski sloj prije propuštanja zahtjeva prema poslužitelju.

Veza između registracijskog poslužitelja i API Gateway-a je u tome što API Gateway koristi informacije iz registra za usmjeravanje zahtjeva do odgovarajućeg mikroservisa. API Gateway postavlja upit registru kako bi odredio lokaciju ciljnog mikroservisa, a zatim prosljeđuje dolazni zahtjev toj usluzi.



Slika 3.5. - Pronalazak ispravnog servisa u registru/Usmjeravanje zahtjeva kroz API Gateway do željenog servisa [17]

Na taj način, API Gateway djeluje kao most između klijentske aplikacije i mikrousluga, pružajući jedinstvenu ulaznu točku za dolazne zahtjeve i apstrahirajući temeljnu složenost mikrousluga [8].

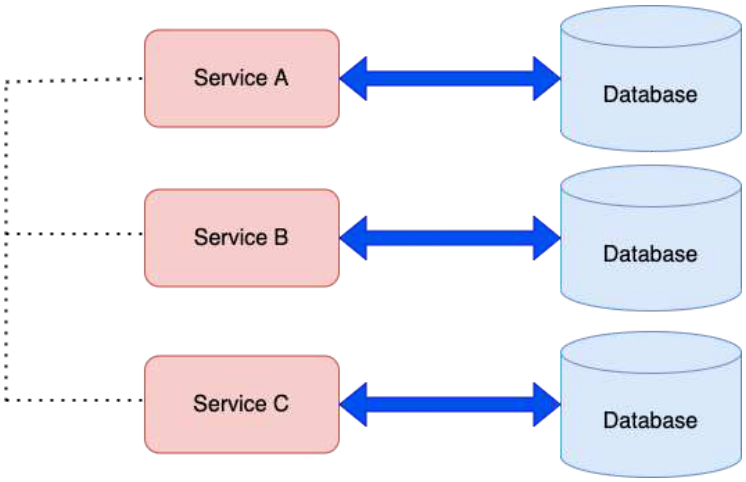
3.3. Baza podataka

Kao baza podataka, odabrana je objektno-relacijska baza *PostgreSQL*.

"Database per service" obrazac

S obzirom da poslužiteljska strana aplikacije koristi mikroservisnu arhitekturu, korišten je dizajn obrazac *"Database per service"* za upravljanje bazom podataka.

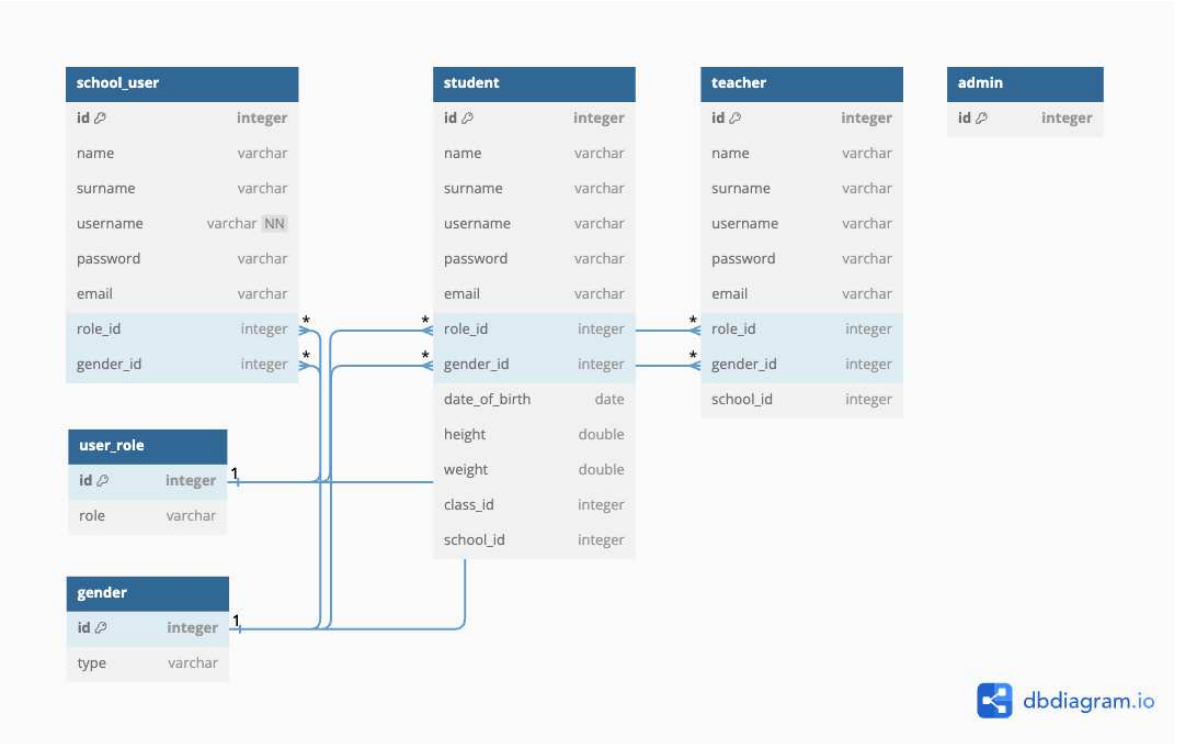
"Database per service" obrazac definira da svaka mikrousluga ima odvojenu bazu podataka, a njezini podaci su dostupni drugim mikroservisima samo putem njezinog API-ja. Transakcije mikrousluge uključuju samo njezinu bazu podataka, što znači da je svaki mikroservis potpuno neovisan u pogledu pohrane i upravljanja podacima što omogućuje bolju izolaciju podataka, poboljšava skalabilnost te pomaže pri postizanju međusobne slabe povezanosti (*"loosely coupling"*). [14]



Slika 3.6. - "Database per service" obrazac [18]

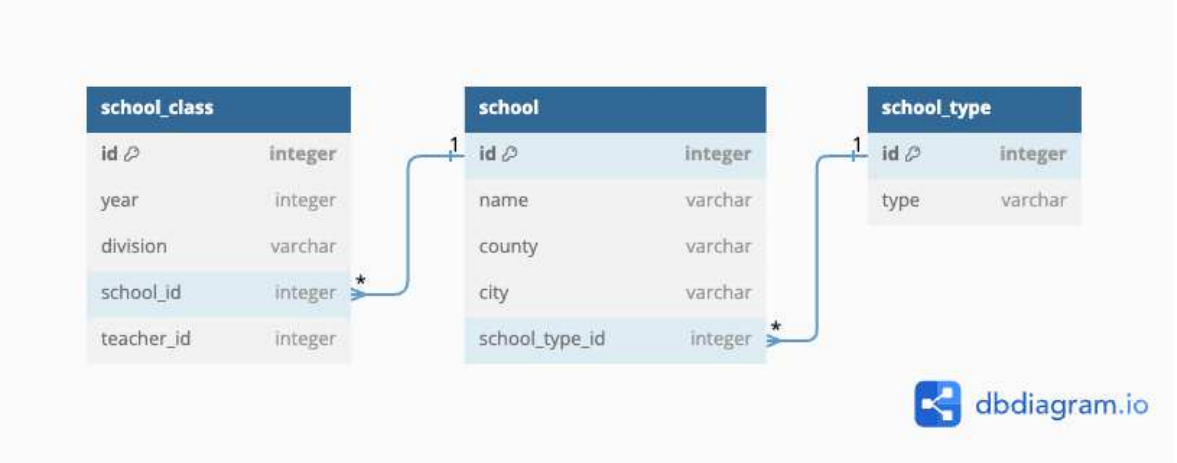
Konceptualni model baze podataka

Postoje tri tipa korisnika u sustavu: učenik, nastavnik i admin. Osnovne informacije koje svaki korisnik sadrži su ime, prezime, korisničko ime, lozinka, e-mail, rola i spol. Svaki učenik dodatno sadrži informacije o datumu rođenja, visini, težini, razredu kojem pripada i školi, dok svaki nastavnik sadrži informaciju o školi kojoj pripada.



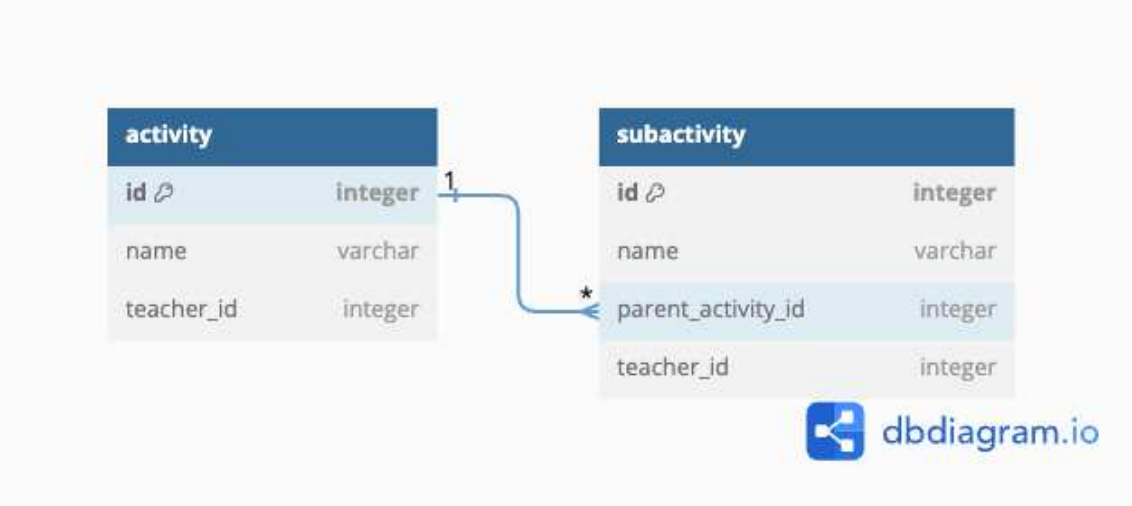
Slika 3.7. - Model baze podataka - servis za upravljanje korisnicima

U sustavu je moguće evidentirati više škola za koje znamo ime škole, županiju, grad i stupanj škole (osnovna, srednja). Svaka škola ima više razreda, a svaki razred ima svog pripadnog nastavnika.



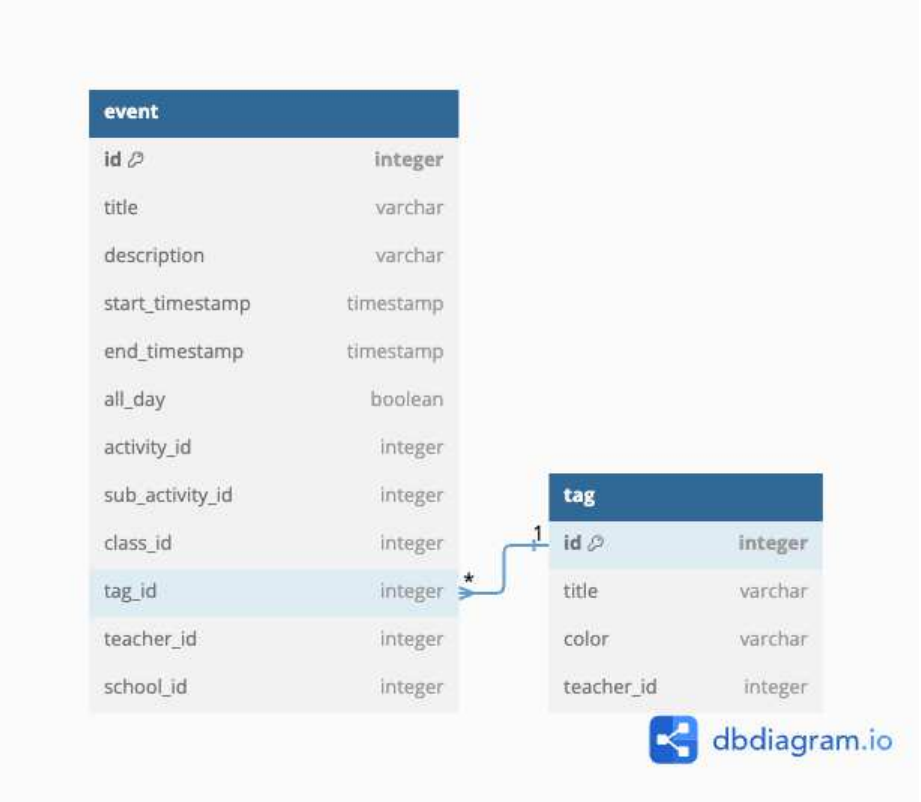
Slika 3.8. - Model baze podataka - servis za upravljanje školama

Sustav sadrži aktivnosti i pripadne podaktivnosti koje nastavnik može uključiti u svoj raspored nastave.



Slika 3.9. - Model baze podataka - servis za upravljanje aktivnostima

Svaki nastavnik ima svoj raspored nastave u koji može dodavati razredne sate. Svaki razredni sat se sastoji od naslova, opisa, vremenskog trajanja, aktivnosti i podaktivnosti koje će se održavati taj sat, te razreda koji će prisustvovati tom satu. Razredni sat može dodatno sadržavati oznaku (labelu) koja ga identificira u određenu kategoriju.



Slika 3.10. - Model baze podataka - servis za upravljanje raspored nastave

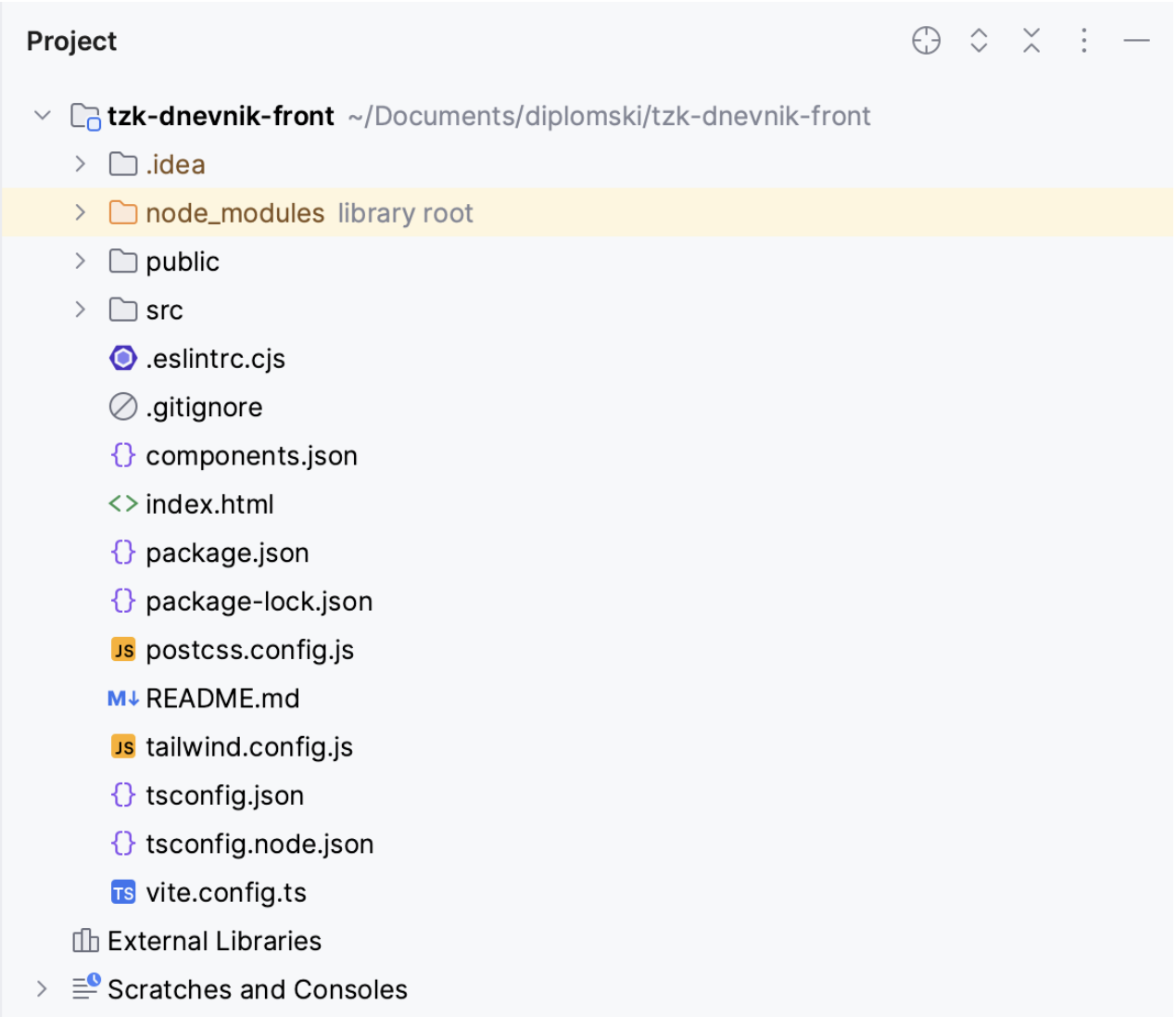
Za svakog učenika, nastavnik može upisati ostvareni rezultat za neku aktivnost/podaktivnost.

Slika 3.11. - Model baze podataka - servis za upravljanje rezultatima

4. Implementacijski detalji aplikacije

4.1. Klijentska strana aplikacije

Projektna struktura klijentske aplikacije je sljedeća:

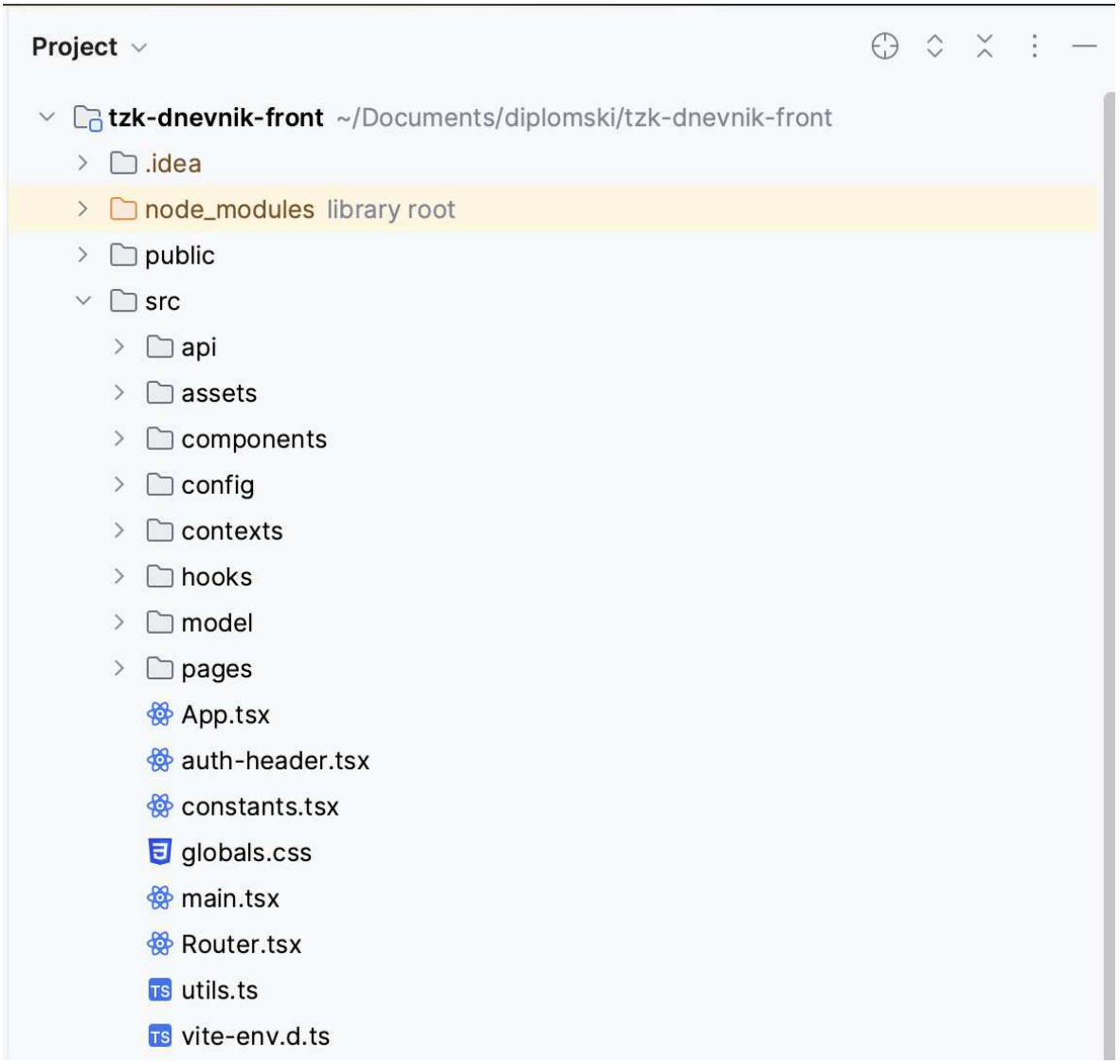


Slika 4.1. - Projektna struktura klijenta

Struktura direktorija je sljedeća:

- *node_modules*: Sadrži pakete vanjskih biblioteka koje se koriste u projektu
- *src*: Sadrži izvorne datoteke koda
    - *api*: Direktorij s API pozivima prema poslužitelju
    - *components*: Direktorij s React komponentama
    - *pages*: Direktorij s stranicama
    - *Router.tsx*: Datoteka s konfiguracijom mogućih ruta u aplikaciji

- *tsconfig.json, tsconfig.node.json*: Konfiguracijske datoteke za korištenje Typescript-a
- *tailwind.config.js*: Konfiguracijska datoteka za korištenje Tailwind CSS okvira



Slika 4.2. - Projektna struktura klijenta - pregled direktorija sa izvornim datotekama

4.1.1. Postavke usmjeravanja

Ulazna točka u aplikaciju je datoteka *App.tsx* koja u sebi ima omotač za promjenu teme aplikacije te pristupnu točku za navigaciju kroz aplikaciju. Za routing koristi se biblioteka *"react-router-dom"*.

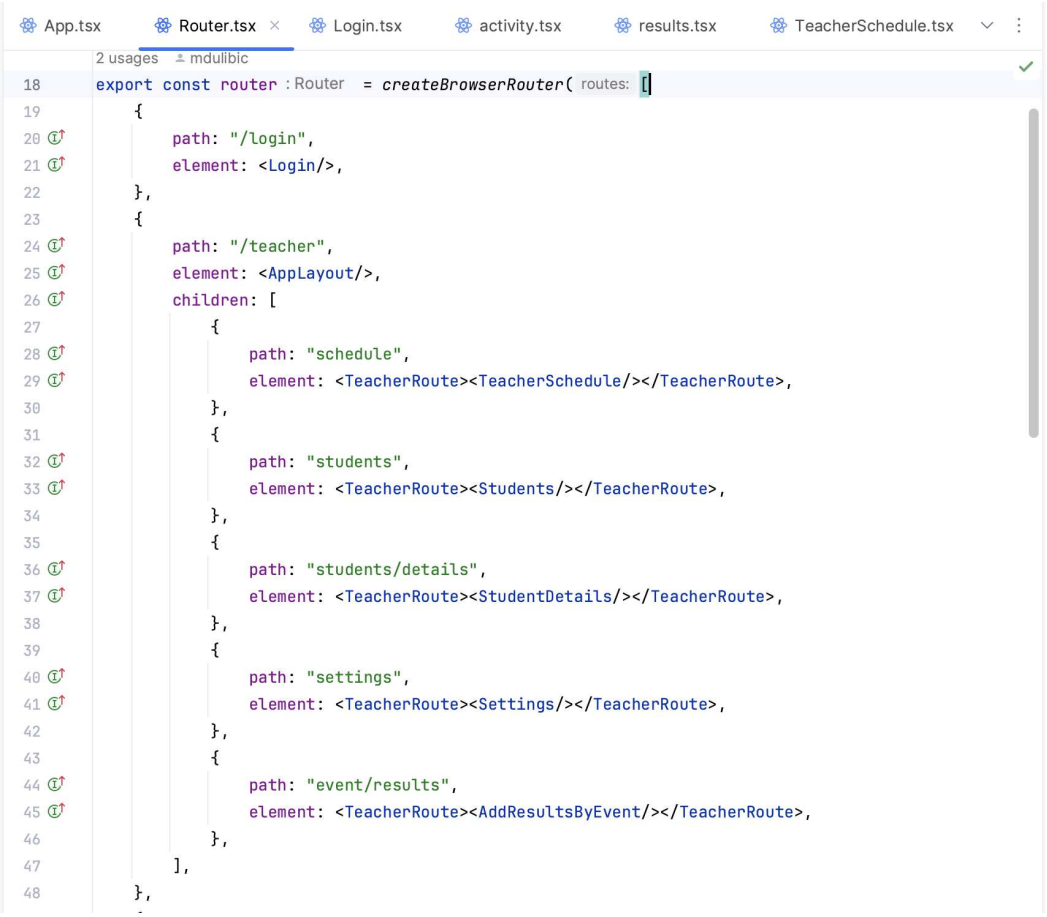
App.tsx | Router.tsx | Login.tsx | activity.tsx | results.tsx | TeacherSchedule.tsx

```
1  import { RouterProvider } from "react-router-dom";
2  import { ThemeProvider } from "./contexts/ThemeContext";
3  import { router } from "./Router";
4
   2 usages  mdulibic
5  export default function App() : JSX.Element {
6      return (
7          <ThemeProvider>
8              <RouterProvider router={router} />
9          </ThemeProvider>
10     )
11 }
```

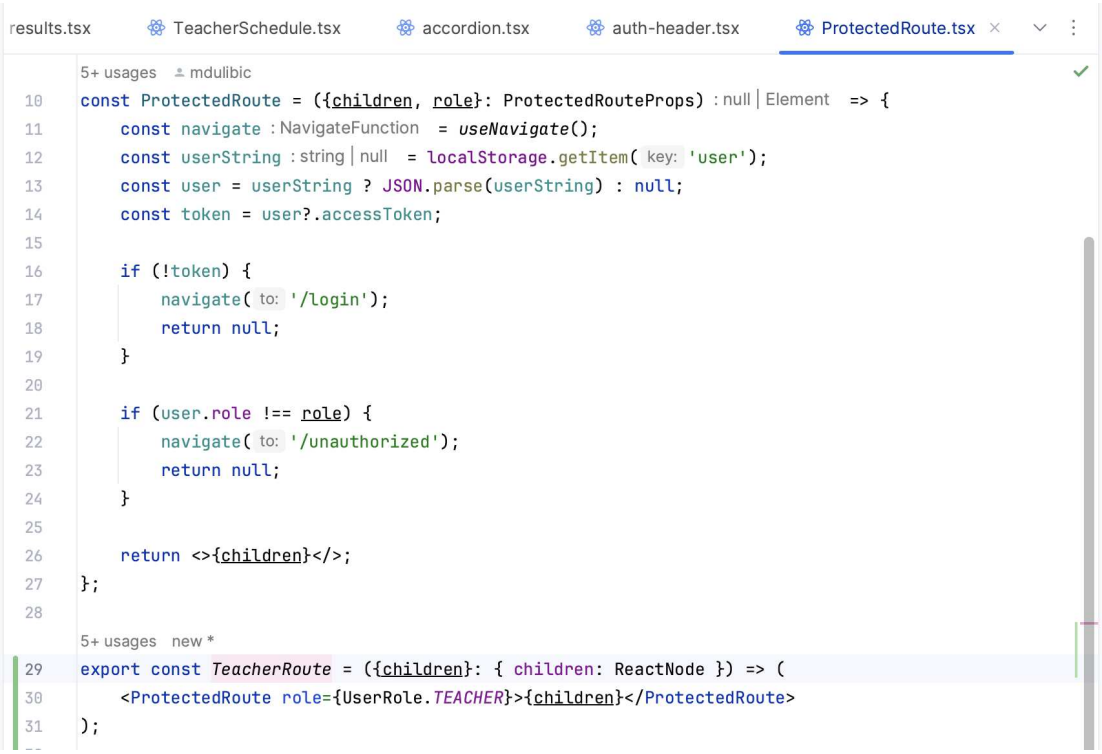
Slika 4.3. - Ulazna točka u klijentsku aplikaciju

Datoteka *Router.tsx* sadrži sve moguće rute koje su dostupne u aplikaciji. S obzirom na to da aplikacija ima tri različite vrste korisnika (administrator, nastavnik, učenik), rute su konfigurirane tako da svaki korisnik može pristupiti samo onim rutama za koje je autoriziran prilikom prijave u sustav.

Prije nego što korisnik pristupi određenoj ruti, u lokalnom spremištu aplikacije provjerava se postoji li pohranjen korisnik i koja je njegova uloga. Ako u trenutnoj sesiji nema spremljenog korisnika, korisnik se preusmjerava na stranicu za prijavu. Ako korisnik nema prava za pristup određenoj stranici, preusmjerava ga se na stranicu s porukom o pogrešci.



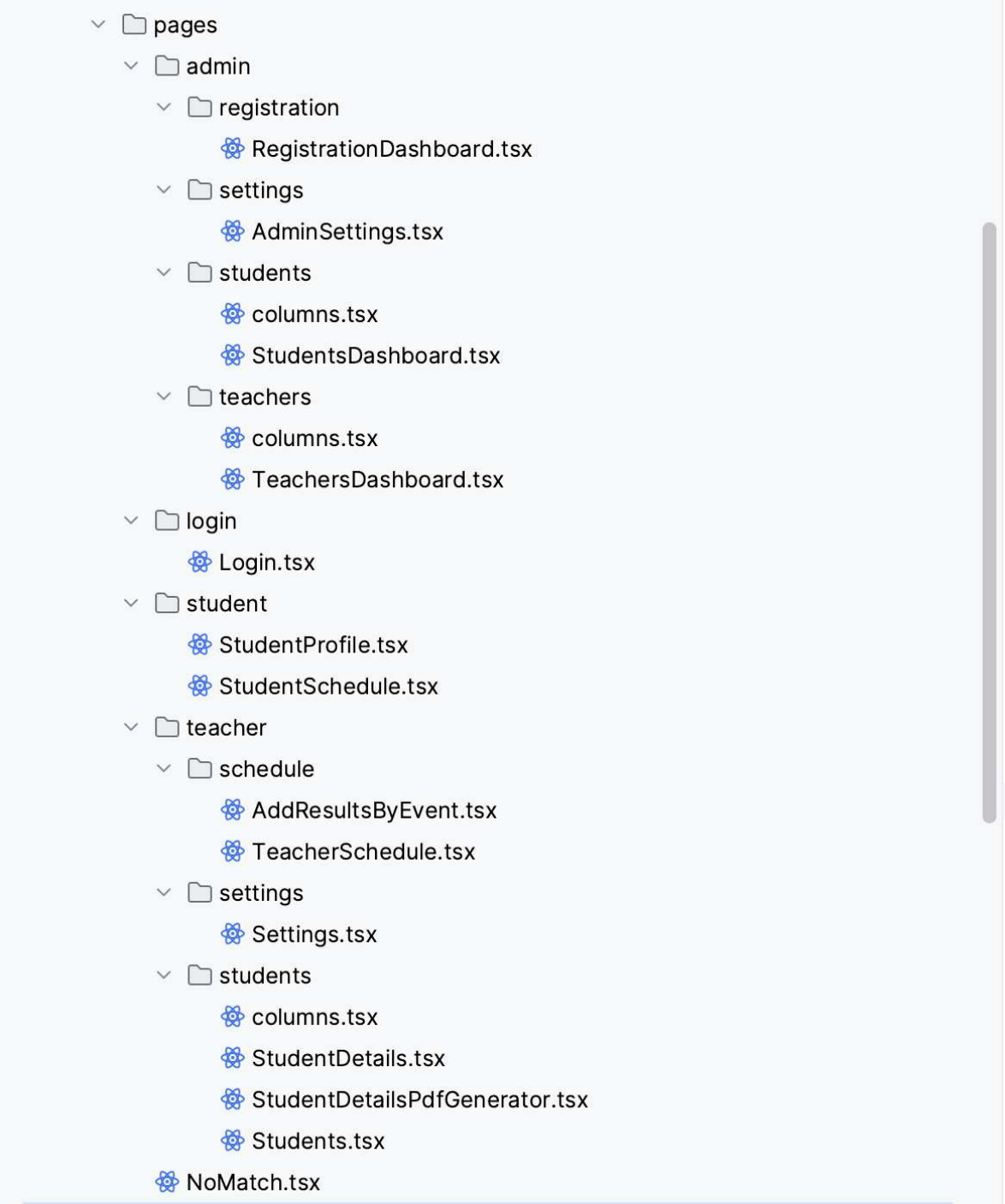
Slika 4.4. - Primjer konfiguracije za korisnika sa rolom *"Nastavnik"*



Slika 4.5. - Primjer autorizacije ruta za korisnika sa rolom *"Nastavnik"*

4.1.2. Pregled stranica i komponenti

Stranice u aplikaciji su organizirane u direktorijima prema ulogama korisnika i funkcionalnostima koje pružaju. Svaka uloga korisnika (administrator, nastavnik, učenik) ima vlastiti direktorij, unutar kojeg su grupirane stranice ili komponente relevantne za tu ulogu.

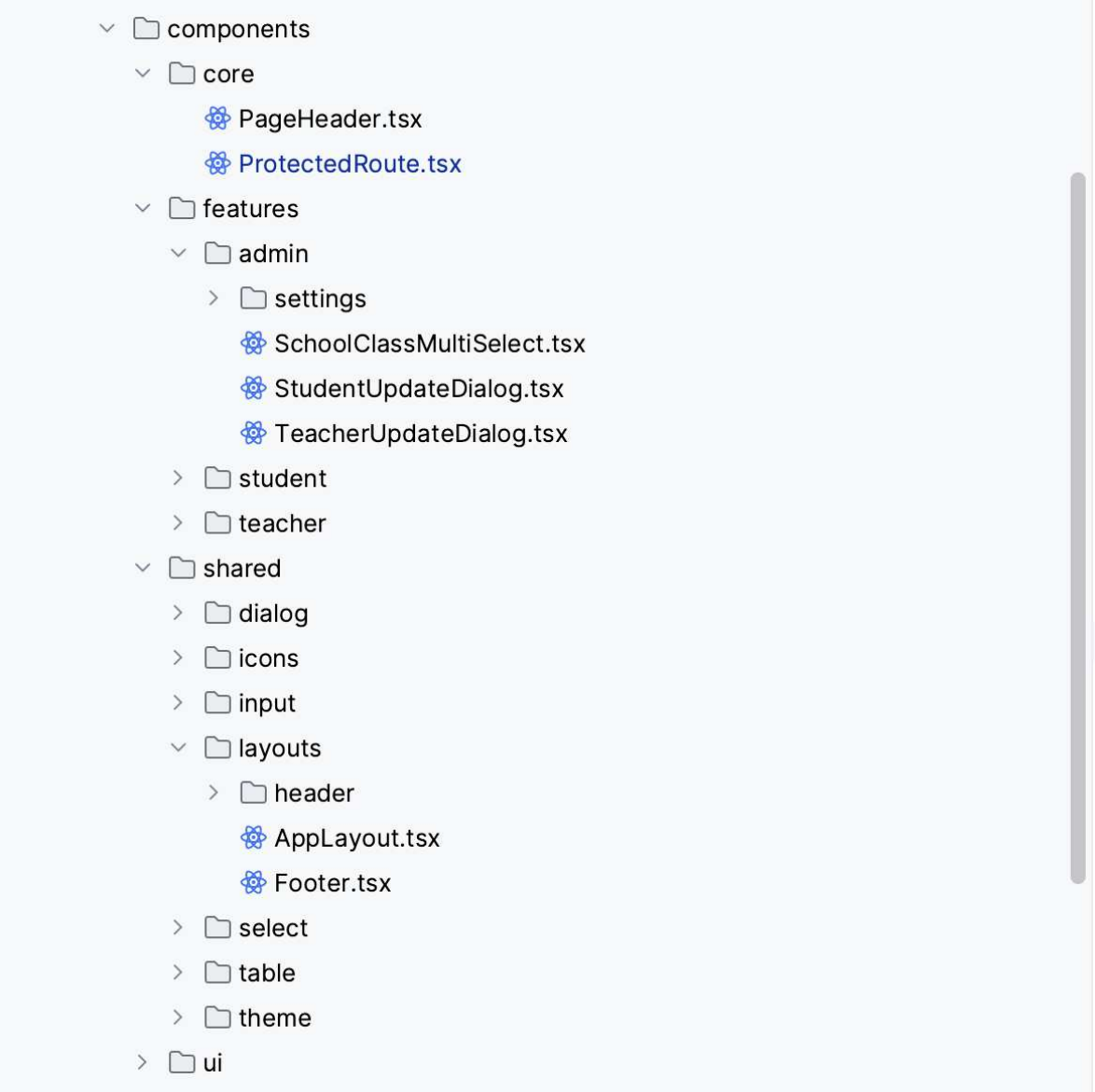


Slika 4.6. - Struktura *pages* direktorija

Na primjer, direktorij *"'admin"* sadrži pod-direktorije poput *"registration'"* za registraciju korisnika, *"students"/"teachers"* za upravljanje učenicima/nastavnicima, i 'settings' za pregled postavki aplikacije. Slično tome, direktoriji *"login"*, *"teacher"* i *"student"* sadrže funkcionalnosti specifične za prijavu u sustav, nastavnike i učenike.

React komponente u aplikaciji su organizirane u četiri glavna direktorija:

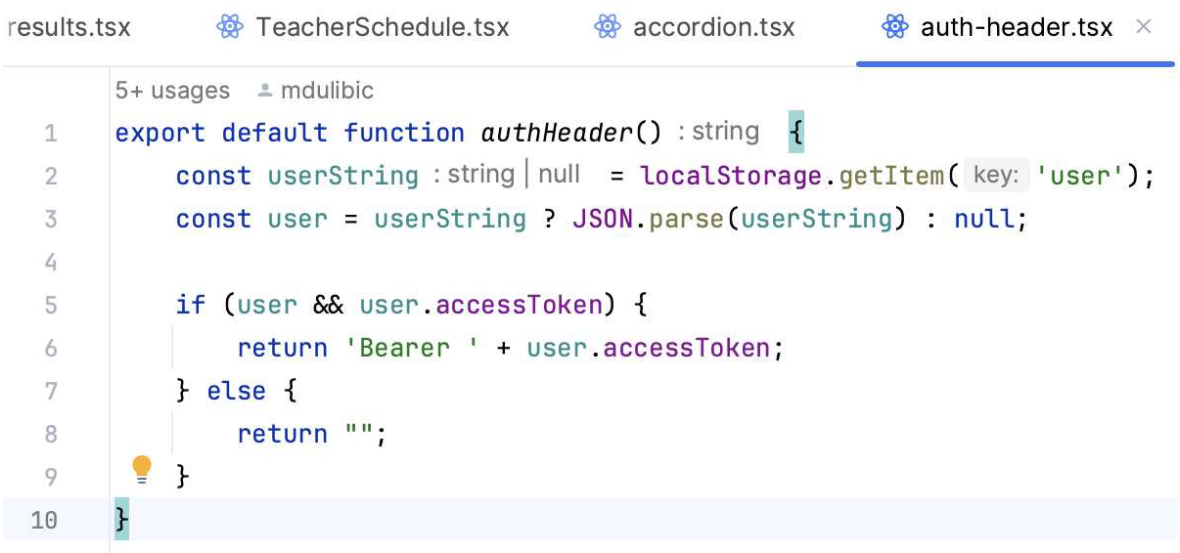
- *core*: Sadrži primitivne komponente (*"low level"*) komponente kao što su glavna navigacijska traka i wrapper datoteka za autorizaciju ruti
- *features*: Korijenski direktorij sa komponente vezane za neke specifične funkcionalnosti aplikacije (često vezani uz pojedinu stranicu)
    - npr. *StudentUpdateDialog* - dijalog komponenta koja se koristi prilikom ažuriranja podataka o učeniku
- *shared*: Komponente koje se dijele kroz cijeli projekt (npr. dijalozi, ikone, polja za unos, tablice, itd.)
- *ui*: Sadrži "*Shadcn/UI"* komponente koje se koriste u projektu



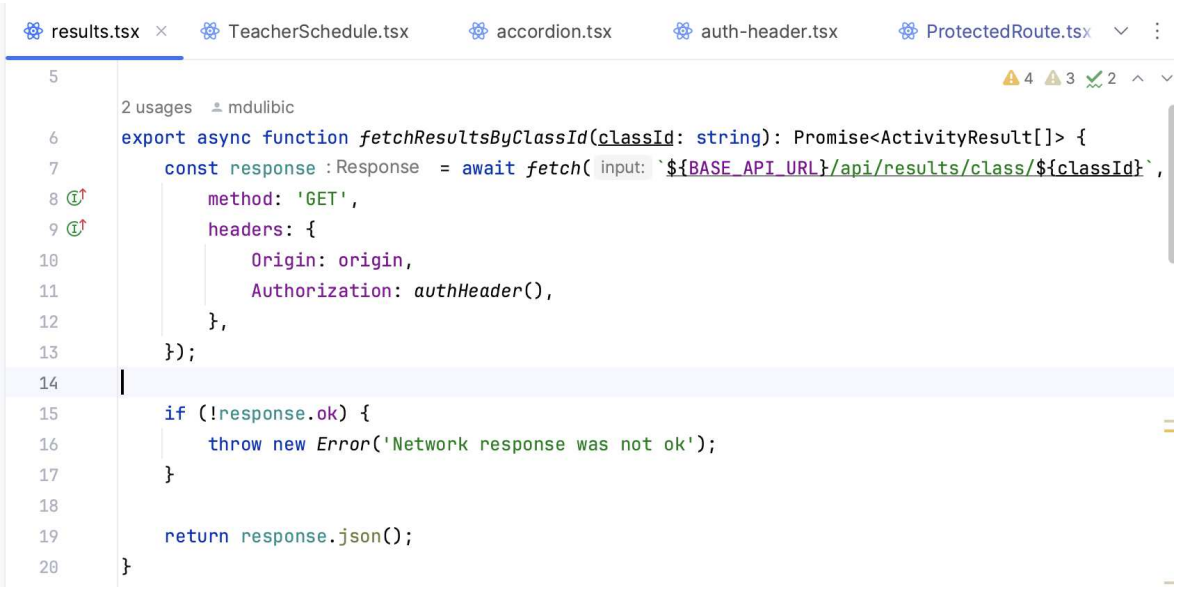
Slika 4.7. - Struktura *components* direktorija

4.1.3. Komunikacija s poslužiteljem

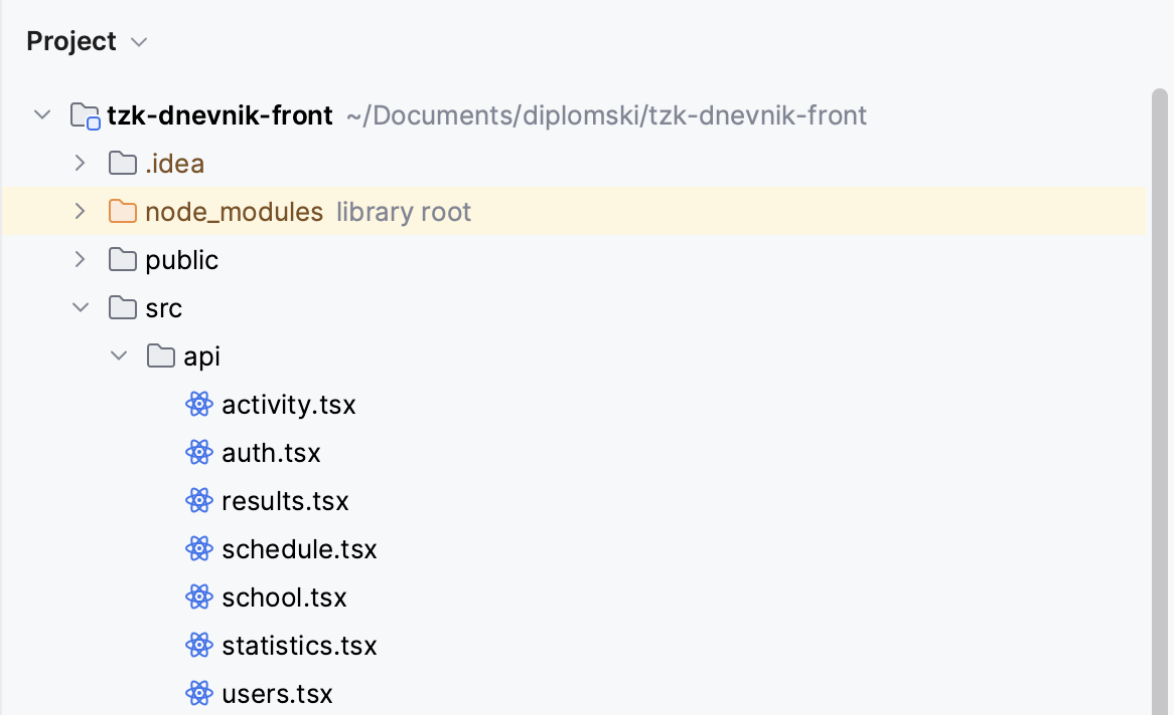
Za pozive prema API-u koristi se JavaScript *Fetch API*. Nakon uspješne prijave u sustav, za daljnju autorizaciju prema poslužitelju koristi se pristupni token (eng. *access token*). Taj token se sprema u lokalno spremište aplikacije i šalje se u obliku *Bearer* tokena prilikom svakog API poziva.



Slika 4.8. - Dohvaćanje pristupnog tokena



Slika 4.9. - Primjer API poziva - dohvaćanje rezultata za razred



Slika 4.10. - Pregled datoteka s API pozivima

4.2. Poslužiteljska strana aplikacije

Poslužiteljska strana aplikacija se sastoji od registracijskog poslužitelja, API gateway-a, šest Spring Boot mikroservisa i jednog Flask mikroservisa.

4.2.1. Registracijski poslužitelj

Za izgradnju registracijskog poslužitelja korišten je *Spring Cloud Netflix Eureka Server*.

Za korištenje Eureka servera u Spring Boot projektu potrebno je dodati vanjsku ovisnost *spring-cloud-starter-netflix-eureka-server* u *build.gradle* datoteku.

Kako bi se aplikacija identificirala kao registracijski poslužitelj, potrebno ju je označiti s `@EnableEurekaServer` anotacijom te postaviti port na kojem će se pokretati (port za Eureka server je 8761).

```
mdulibic
@SpringBootApplication
@EnableEurekaServer
public class EurekaServerApplication {

    mdulibic
    public static void main(String[] args) { SpringApplication.run(EurekaServerApplication.class, args); }

}
```

Slika 4.11. - Spring Boot aplikacija sa omogućenim Eureka serverom

EurekaServerApplication.java | build.gradle (EurekaServer) | application.properties

```
1  spring.application.name=EurekaServer
2  server.port=8761
3  eureka.client.register-with-eureka=false
4  eureka.client.fetch-registry=false
5
```

Slika 4.12. - Postavke za Eureka server u application.properties

Nakon pokretanja aplikacije, odlaskom na adresu `http://localhost:8761` možemo vidjeti popis svih registriranih usluga.

4.2.2. API gateway

Za izgradnju API gatewaya korišten je *Spring Cloud Router Gateway*. Da bi se API gateway omogućio u Spring Boot projektu, potrebno je dodati sljedeće vanjske ovisnosti u *build.gradle* datoteku:

1. *spring-cloud-starter-gateway-mvc* za funkcionalnosti API gatewaya.
2. *spring-cloud-starter-netflix-eureka-client* za registraciju servisa na registracijski poslužitelj.

API gateway sadrži konfiguraciju ruta kako bi znao preusmjeriti API pozive od klijenta na odgovarajući mikroservis.

```
@Bean
public RouteLocator routes(RouteLocatorBuilder builder) {
    return builder.routes()
            .route( id: "data-clustering-service", r -> r.path("/data-clustering/**") BooleanSpec
                    .filters(f -> f.filter(filter)) UriSpec
                    .uri("lb://data-clustering-service"))
            .route( id: "auth-service", r -> r.path("api/auth/**") BooleanSpec
                    .filters(f -> f.filter(filter)) UriSpec
                    .uri("lb://auth-service"))
            .route( id: "activity-service", r -> r.path("api/activities/**") BooleanSpec
                    .filters(f -> f.filter(filter)) UriSpec
                    .uri("lb://activity-service"))
            .route( id: "event-service", r -> r.path("api/events/**") BooleanSpec
                    .filters(f -> f.filter(filter)) UriSpec
                    .uri("lb://event-service"))
            .route( id: "result-service", r -> r.path("api/results/**") BooleanSpec
                    .filters(f -> f.filter(filter)) UriSpec
                    .uri("lb://result-service"))
            .route( id: "school-service", r -> r.path("api/school/**") BooleanSpec
                    .filters(f -> f.filter(filter)) UriSpec
                    .uri("lb://school-service"))
            .route( id: "user-service", r -> r.path("api/users/**") BooleanSpec
                    .filters(f -> f.filter(filter)) UriSpec
                    .uri("lb://user-service"))
            .route( id: "statistics-service", r -> r.path("api/statistics/**") BooleanSpec
                    .filters(f -> f.filter(filter)) UriSpec
                    .uri("lb://statistics-service"))
            .build();
}
```

Slika 4.13. - Konfiguracija ruta

Također sadrži datoteku *AuthenticationFilter* koja provjerava postoji li valjani JWT token u headeru poziva. Ako postoji, prosljeđuje poziv potrebnom mikroservisu, a ako ne postoji, vraća HTTP Error CODE 403 FORBIDDEN.

```java
@Component
public class AuthenticationFilter implements GatewayFilter {

    2 usages
    private final RouterValidator routerValidator;
    2 usages
    private final JwtUtil jwtUtil;

    new *
    @Autowired
    public AuthenticationFilter(RouterValidator routerValidator, JwtUtil jwtUtil) {
        this.routerValidator = routerValidator;
        this.jwtUtil = jwtUtil;
    }

    new *
    @Override
    public Mono<Void> filter(ServerWebExchange exchange, GatewayFilterChain chain) {
        ServerHttpRequest request = exchange.getRequest();

        if (routerValidator.isSecured.test(request)) {
            if (this.isAuthMissing(request)) {
                return this.onError(exchange, HttpStatus.UNAUTHORIZED);
            }

            final String token = this.getAuthHeader(request);

            if (!jwtUtil.validateJwtToken(token)) {
                return this.onError(exchange, HttpStatus.FORBIDDEN);
            }
        }
        return chain.filter(exchange);
```

Slika 4.13. - Autentifikacijski filter prije propuštanja poziva

Komponenta *RouterValidator* određuje koje putanje se moraju provjeriti prije preusmjeravanja na mikroservis. U našem slučaju, sve rute osim ruta prema auth mikroservisu su zaštićene.

```java
@Component
public class RouterValidator {

    1 usage
    public static final List<String> openApiEndpoints = List.of(
            e1: "/api/auth/**"
    );

    1 usage
    public Predicate<ServerHttpRequest> isSecured =
            request -> openApiEndpoints
                    .stream()
                    .noneMatch(uri -> request.getURI().getPath().contains(uri));

}
```

Slika 4.14. - Validacija rute prije preusmjeravanja na ispravan mikroservis

4.2.3. Mikroservisi

Svaki mikroservis se mora registrirati na Eureka server kako bi bio prepoznatljiv i dostupan za druge servise. Da bi se mikroservis registrirao na Eureka server, potrebno ga je anotirati s `@EnableEurekaClient`.

Da bi mikroservisi mogli međusobno komunicirati i slati podatke, također je potrebno konfigurirati *RestTemplate* s anotacijom `@LoadBalanced` te dodati anotaciju `@EnableFeignClient`. *RestTemplate* će omogućiti mikroservisu da koristi balansiranje opterećenja (eng. *"load balancing"*) kada šalje zahtjeve prema drugim servisima registriranim na Eureka serveru.

```
@EnableDiscoveryClient
@EnableFeignClients
@SpringBootApplication
public class AuthServiceApplication {

    new *
    public static void main(String[] args) { SpringApplication.run(AuthServiceApplication.class, args); }

    new *
    @LoadBalanced
    @Bean
    RestTemplate restTemplate() { return new RestTemplate(); }
}
```

Slika 4.15. - Primjer konfiguracije mikroservisa

4.2.3.1 Autentifikacijski servis

Mikroservis za autentifikaciju izlaže sučelja za registraciju i prijavu u sustav. Korisnik se može prijaviti u sustav putem korisničkog imena i lozinke ili Google računa.

```
@PostMapping("/register")
public ResponseEntity<Void> registerUser(@RequestBody RegisterUserDto user) throws IOException {
    if (userClient.registerUser(user)) {
        return ResponseEntity.ok().build();
    } else {
        throw new IllegalArgumentException("Username already exists.");
    }
}

new *
@PostMapping("/login")
public ResponseEntity<Object> loginUser(@RequestBody LoginUserDto userDto) {
    UserDto user = userClient.checkUserCredentials(userDto);
    if (user != null) {
        return authenticateUser(user);
    } else {
        throw new IllegalArgumentException("Invalid username/password.");
    }
}
```

Slika 4.16. - API sučelja

SecurityConfig datoteka služi za konfiguriranje sigurnosnih pravila vezanih uz autorizaciju korisnika u Spring aplikaciji. Ona definira postavke kao što su enkripcija lozinke, provjera ispravnosti JWT tokena prilikom API poziva te omogućuje prijavu korisnika putem OAuth2 protokola, posebno integriranog s Google računom.

```
@Bean
SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
    http
            .csrf(AbstractHttpConfigurer::disable)
            .authorizeHttpRequests(auth -> {
                auth.requestMatchers("/api/auth/**").permitAll();
                auth.requestMatchers("/oauth2/authorization/google/**").permitAll();
                auth.anyRequest().authenticated();
            })
            .oauth2Login(oath2 -> {
                oath2
                        .userInfoEndpoint(userInfoEndpointConfig ->
                                userInfoEndpointConfig.userService(oauthUserService))
                        .loginPage(frontendUrl + "/login").permitAll()
                        .defaultSuccessUrl(frontendUrl + "/")
                        .successHandler(oAuth2LoginSuccessHandler);
            })
            .exceptionHandling(e -> e
                    .authenticationEntryPoint(new HttpStatusEntryPoint(HttpStatus.UNAUTHORIZED))
            )
            .logout(l -> l
                    .logoutUrl("/logout")
                    .logoutSuccessUrl(frontendUrl + "/login").permitAll()
                    .invalidateHttpSession(true)
                    .deleteCookies( ...cookieNamesToClear: "JSESSIONID")
                    .clearAuthentication(true)
                    .logoutRequestMatcher(new AntPathRequestMatcher("/logout"))
            );

    http.addFilterBefore(authenticationJwtTokenFilter(), UsernamePasswordAuthenticationFilter.class);

    return http.build();
```

Slika 4.17. - Konfiguracija sigurnosnog filtera unutar *SecurityConfig* klase

Nakon uspješne prijave, korisnicima se dodjeljuje JWT token koji se koristi za autorizirane pristupe API-ju.

```
private ResponseEntity<Object> authenticateUser(UserDto userDto) {
    Authentication authentication = authenticationManager.authenticate(
            new UsernamePasswordAuthenticationToken(userDto.getEmail(), userDto.getPassword()));

    SecurityContextHolder.getContext().setAuthentication(authentication);
    String jwt = jwtUtils.generateJwtToken(authentication);

    CustomUserDetails userDetails = (CustomUserDetails) authentication.getPrincipal();

    return ResponseEntity.ok(new JwtResponse(jwt,
            userDto.getId(),
            userDetails.getUsername(),
            userDto.getName(),
            userDto.getSurname(),
            userDto.getEmail(),
            userDto.getRole().name()));
}
```

Slika 4.18. - Metoda za autentifikacijski korisnika u sustav

4.2.3.2 Servis za upravljanje korisnicima

Mikroservis za korisnike izlaže sučelja za upravljanje nastavnicima i učenicima kao što su dodavanje/brisanje korisnika iz sustava, dohvaćanje korisnika za određenu školu ili školski razred, ažuriranje podataka o korisnicima.

```
@PostMapping("/student/add")                                                    ⚠4 ⚠3
public ResponseEntity<Student> addStudent(@RequestBody Student student) {
    Student savedStudent = studentService.addStudent(student);
    return ResponseEntity.status(HttpStatus.CREATED).body(savedStudent);
}

new *
@DeleteMapping("/student/delete/{studentId}")
public ResponseEntity<Void> deleteStudent(@PathVariable Long studentId) {
    studentService.deleteStudent(studentId);
    return ResponseEntity.noContent().build();
}

new *
@GetMapping("/student/all")
public ResponseEntity<List<Student>> getAllStudents() {
    List<Student> students = studentService.getAll();
    return ResponseEntity.ok(students);
}

new *
@GetMapping("/student/school/{schoolId}")
public ResponseEntity<List<Student>> getStudentsBySchool(@PathVariable Long schoolId) {
    List<Student> students = studentService.getStudentsBySchool(schoolId);
    return ResponseEntity.ok(students);
}

new *
@GetMapping("/student/{studentId}")
public ResponseEntity<Student> getStudentById(@PathVariable Long studentId) {
    Student student = studentService.getStudentById(studentId).orElse( other: null);
```

Slika 4.19. - Prikaz API sučelja za upravljanje učenicima

```java
@PostMapping("/teacher/add")
public ResponseEntity<Teacher> addTeacher(@RequestBody Teacher teacher) {
    Teacher savedTeacher = teacherService.addTeacher(teacher);
    return ResponseEntity.status(HttpStatus.CREATED).body(savedTeacher);
}

new *
@DeleteMapping("/teacher/delete/{teacherId}")
public ResponseEntity<Void> deleteTeacher(@PathVariable Long teacherId) {
    teacherService.deleteTeacher(teacherId);
    return ResponseEntity.noContent().build();
}

new *
@GetMapping("/teacher/all")
public ResponseEntity<List<Teacher>> getAllTeachers() {
    List<Teacher> teachers = teacherService.getAllTeachers();
    return ResponseEntity.ok(teachers);
}

new *
@GetMapping("/teacher/school/{schoolId}")
public ResponseEntity<List<Teacher>> getTeachersBySchool(@PathVariable Long schoolId) {
    List<Teacher> teachers = teacherService.getTeachersBySchool(schoolId);
    return ResponseEntity.ok(teachers);
}

new *
@GetMapping("/teacher/{teacherId}")
public ResponseEntity<Teacher> getTeacherById(@PathVariable Long teacherId) {
    Teacher teacher = teacherService.getTeacherById(teacherId).orElse( other: null);
```

Slika 4.20. - Prikaz API sučelja za upravljanje nastavnicima

4.2.3.3 Servis za upravljanje školama

Mikroservis za škole izlaže sučelja za upravljanje školama kao što je dodavanje novih škola, dohvaćanje/dodavanje školskih razreda, uvoz škola kroz CSV datoteku, itd.

```
@RestController
@RequestMapping("/api/school")
public class SchoolController {

    @Autowired
    private SchoolService schoolService;

    mdulibic
    @GetMapping("/all")
    public List<School> getAll() { return schoolService.getSchools(); }

    mdulibic
    @GetMapping("/classes")
    public List<SchoolClass> getAllClasses() { return schoolService.getClasses(); }

    mdulibic
    @GetMapping("/classes/{schoolId}")
    public List<SchoolClass> getClassesForSchool(@PathVariable Long schoolId) {
        return schoolService.getClassesBySchool(schoolId);
    }

    mdulibic
    @PostMapping("/class")
    public SchoolClass createClass(@RequestParam int year, @RequestParam char division) {
        return schoolService.createClass(year, division);
    }

    mdulibic
    @PostMapping("/enroll/{schoolId}")
    public ResponseEntity<String> enrollStudentsFromCsv(@RequestParam("role") UserRole role, @RequestParam("file") M
        try {
            if (role == UserRole.ROLE_STUDENT) {
```

Slika 4.21. - Prikaz API sučelja u servisu za škole

4.2.3.4 Servis za upravljanje aktivnostima

Mikroservis za aktivnosti izlaže sučelja za upravljanje aktivnostima i podaktivnostima kao što je dodavanje novih aktivnosti/podaktivnosti, dohvaćanje liste aktivnosti/podaktivnosti, itd.

```
mdulibic
@RestController
@RequestMapping("/api/activities")
public class ActivityController {

    @Autowired
    private ActivityService activityService;

    mdulibic
    @GetMapping("/all")
    public ResponseEntity<List<Activity>> getAllActivities() {
        List<Activity> activities = activityService.getAll();
        return ResponseEntity.ok(activities);
    }

    mdulibic
    @PostMapping("/add")
    public ResponseEntity<Activity> addActivity(@RequestBody String name) {
        Activity activity = activityService.addActivity(name);
        return ResponseEntity.status(HttpStatus.CREATED).body(activity);
    }

    mdulibic
    @PostMapping("/add/subactivity/{activityId}")
    public ResponseEntity<SubActivity> addSubActivity(@PathVariable Long activityId, @RequestBody String name) {
        SubActivity subActivity = activityService.addSubActivity(activityId, name);
        return ResponseEntity.status(HttpStatus.CREATED).body(subActivity);
    }
}
```

Slika 4.22. - Prikaz API sučelja u servisu za aktivnosti

4.2.3.5 Servis za upravljanje rezultatima

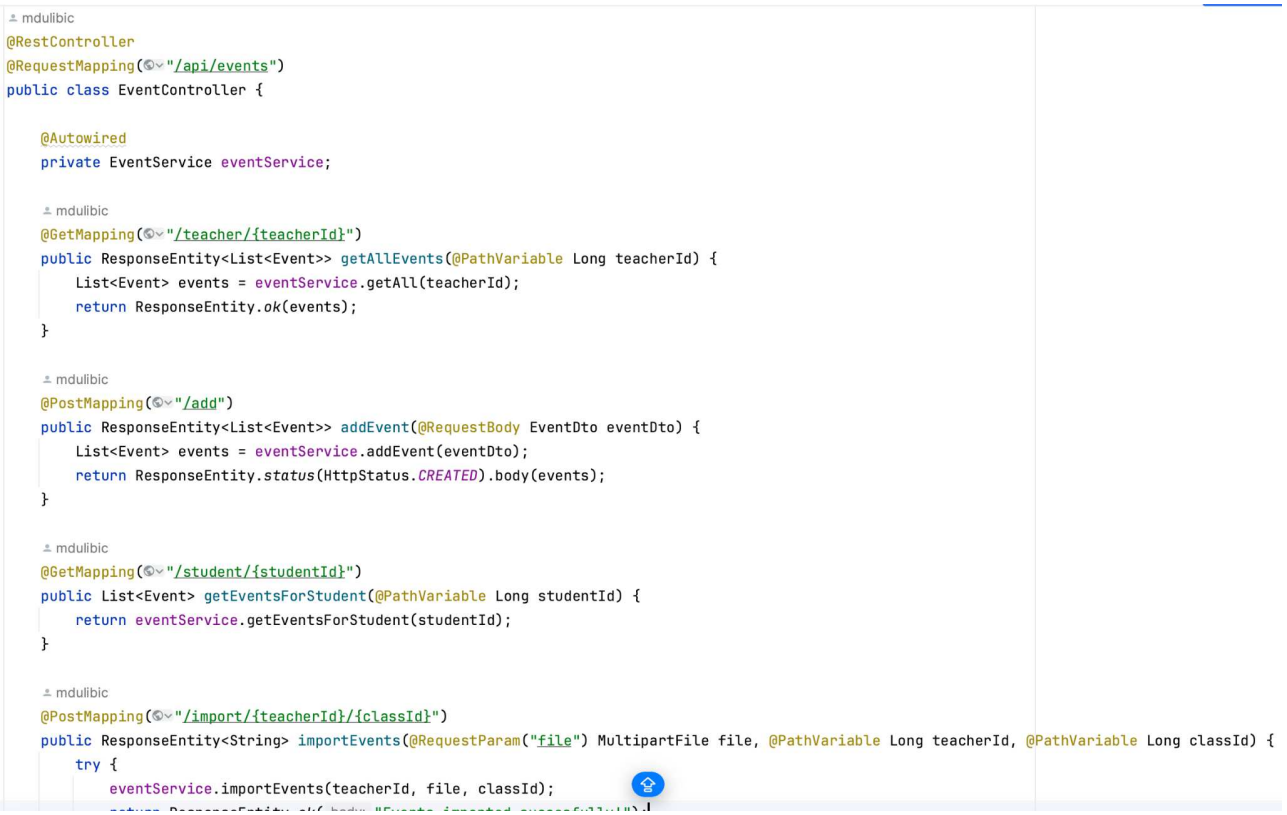
Mikroservis za rezultate izlaže sučelja za upravljanje rezultatima učenika kao što je dodavanje novih rezultata za učenika, dohvaćanje rezultata razreda ili pojedinog učenika, generiranje statistike iz povijesti rezultata, itd.

```
mdulibic
@RestController
@RequestMapping("/api/results")
public class ResultController {

    @Autowired
    ResultService resultService;

    mdulibic
    @GetMapping("/all")
    public ResponseEntity<List<Result>> getResults() {
        List<Result> results = resultService.getAllResults();
        return ResponseEntity.ok(results);
    }

    mdulibic
    @GetMapping("/class/{classId}")
    public ResponseEntity<List<Result>> getResultsByClassId(@PathVariable Long classId) {
        List<Result> results = resultService.getResultsByClassId(classId);
        return ResponseEntity.ok(results);
    }

    mdulibic
    @GetMapping("/student/{studentId}")
    public ResponseEntity<List<Result>> getResultsByStudentId(@PathVariable Long studentId) {
        List<Result> results = resultService.getResultsByStudentId(studentId);
        return ResponseEntity.ok(results);
    }
```

Slika 4.23. - Prikaz API sučelja u servisu za rezultate

4.2.3.6 Servis za upravljanje rasporedom nastave

Mikroservis za raspored nastave izlaže sučelja za upravljanje rasporedom nastave kao što je dodavanje novih rezultata za učenika, dohvaćanje rezultata razreda ili pojedinog učenika, generiranje statistike iz povijesti rezultata, itd .

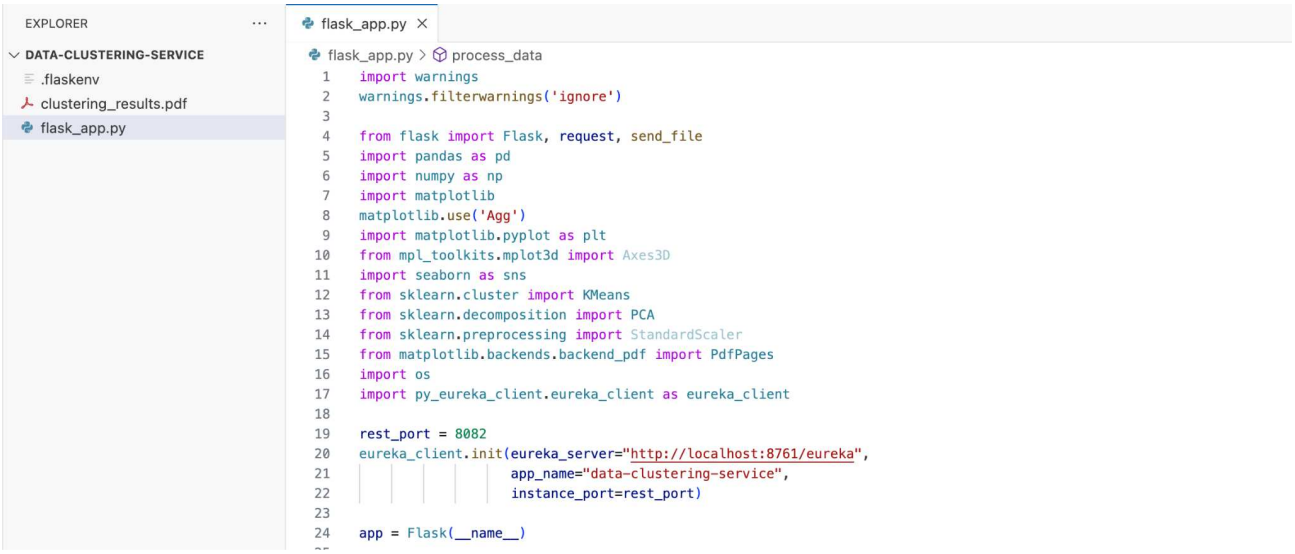


```
mdulibic
@RestController
@RequestMapping("/api/events")
public class EventController {

    @Autowired
    private EventService eventService;

    mdulibic
    @GetMapping("/teacher/{teacherId}")
    public ResponseEntity<List<Event>> getAllEvents(@PathVariable Long teacherId) {
        List<Event> events = eventService.getAll(teacherId);
        return ResponseEntity.ok(events);
    }

    mdulibic
    @PostMapping("/add")
    public ResponseEntity<List<Event>> addEvent(@RequestBody EventDto eventDto) {
        List<Event> events = eventService.addEvent(eventDto);
        return ResponseEntity.status(HttpStatus.CREATED).body(events);
    }

    mdulibic
    @GetMapping("/student/{studentId}")
    public List<Event> getEventsForStudent(@PathVariable Long studentId) {
        return eventService.getEventsForStudent(studentId);
    }

    mdulibic
    @PostMapping("/import/{teacherId}/{classId}")
    public ResponseEntity<String> importEvents(@RequestParam("file") MultipartFile file, @PathVariable Long teacherId, @PathVariable Long classId) {
        try {
            eventService.importEvents(teacherId, file, classId);
```

Slika 4.24. - Prikaz API sučelja u servisu za raspored nastave

4.2.3.7 Servis za analizu i klasteriranje podataka

Mikroservis za analizu i klasteriranje podataka izlaže sučelje za generiranje PDF izvještaja putem klasterske analize podataka o učeniku. Ovaj mikroservis je jedini implementiran u Flasku zbog prednosti koje Python nudi za analizu i vizualizaciju podataka, poput podrške za biblioteke poput pandas, numpy, matplotlib, seaborn i scikit-learn, koje su ključne za provedbu klasterske analize. Cijeli izvorni kod mikroservisa nalazi se unutar datoteke flask_app.py. Za registraciju na Eureka registracijski poslužitelj, potrebno je pozvati funkciju za inicijalizaciju Eureka klijenta.



Slika 4.25. - Registracija Flask servisa na Eureka server

Mikroservis izlaže jednu API rutu koja prima *CSV* datoteku s podacima učenika. Na temelju primljenih podataka, mikroservis izvodi analizu kako bi identificirao korelacije između različitih parametara te primjenjuje klastersku analizu za grupiranje učenika prema njihovim karakteristikama.

flask_app.py

```python
@app.route('/data-clustering/generate-pdf', methods=['POST'])
def generate_pdf():
    if 'file' not in request.files:
        return "No file part", 400

    file = request.files['file']
    if file.filename == '':
        return "No selected file", 400

    data = process_data(file)

    pdf_file = '/Users/mdulibic/Documents/diplomski/flask-back/clustering_results.pdf'

    if not os.path.exists(pdf_file):
       os.makedirs(pdf_file)

    with PdfPages(pdf_file) as pdf:
        add_intro_text(pdf)
        add_data_table(data, pdf)
        del data["Učenik"]
        create_heatmap(data, gender=1, pdf=pdf)
        create_heatmap(data, gender=0, pdf=pdf)

        whole_data = data.drop('Izdržaj UVZ', axis=1).dropna(ignore_index=True)
        data_pca, kmeans = kmeans_clustering(whole_data, res_k=4, pca_n=2)

        plot_clusters(data_pca, kmeans, pdf)

        add_cluster_info_to_pdf(data_pca, whole_data, kmeans, pdf)

    return send_file(pdf_file, as_attachment=True)
```

Slika 4.26. - API ruta za generiranje izvješća o klasterskoj analizi

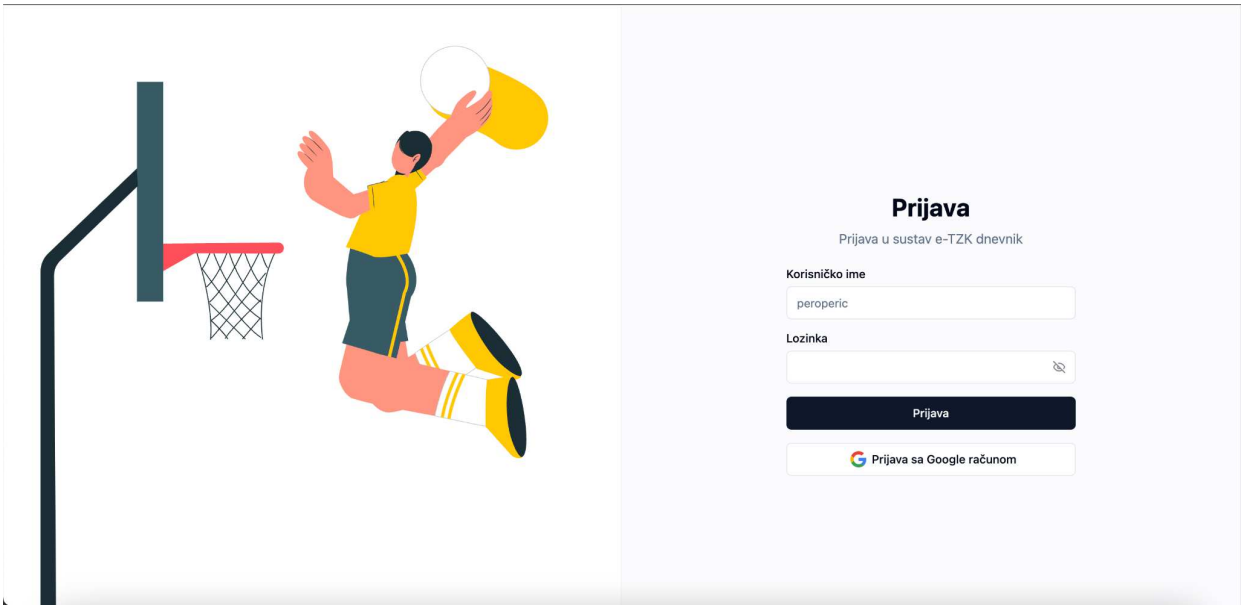
```python
def kmeans_clustering(data, res_k=4, pca_n=2):
    pca = PCA(n_components=pca_n)
    data_pca = pd.DataFrame(pca.fit_transform(data), columns=[f'PC{i}' for i in range(1, pca_n+1)])

    inertias = []
    for k in range(1, 11):
        kmeans = KMeans(n_clusters=k, random_state=42)
        kmeans.fit(data_pca)
        inertias.append(kmeans.inertia_)

    plt.figure()
    plt.plot(range(1, 11), inertias, marker='o')
    plt.xlabel('Number of Clusters (k)')
    plt.ylabel('Inertia')
    plt.title('Elbow Method')
    return data_pca, kmeans
```

Slika 4.27. - Funkcija za provođenje klasterske analize nad skupom podataka

5. Korisničke upute

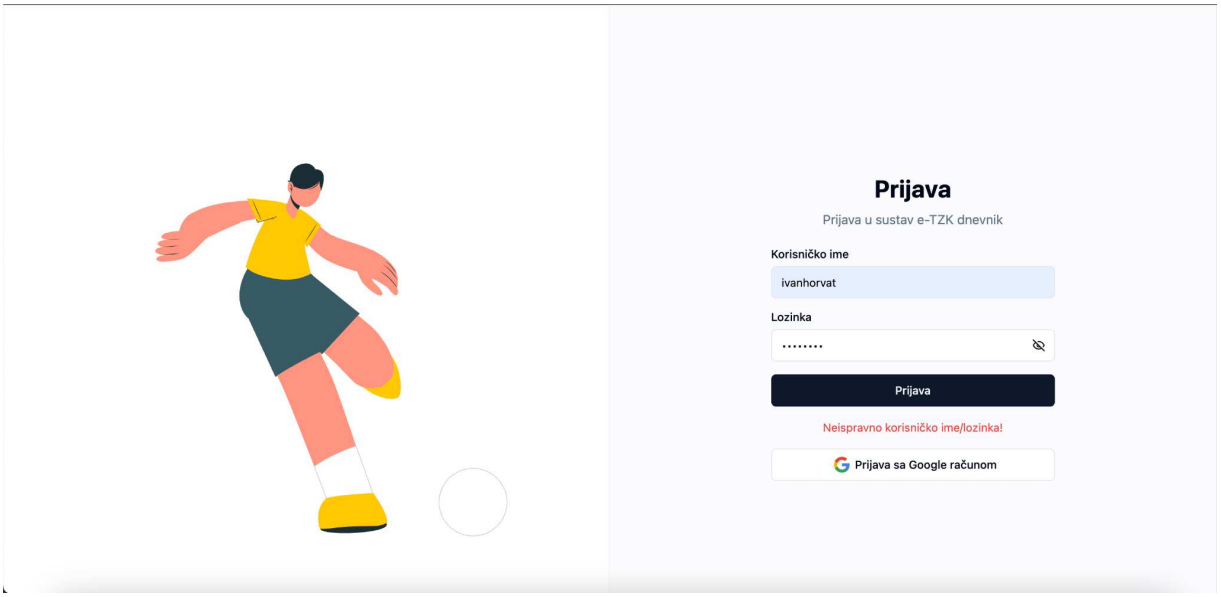
5.1. Prijava u sustav

Početna stranica omogućuje korisniku prijavu u sustav korisničkim imenom i lozinkom ili putem Google računa.



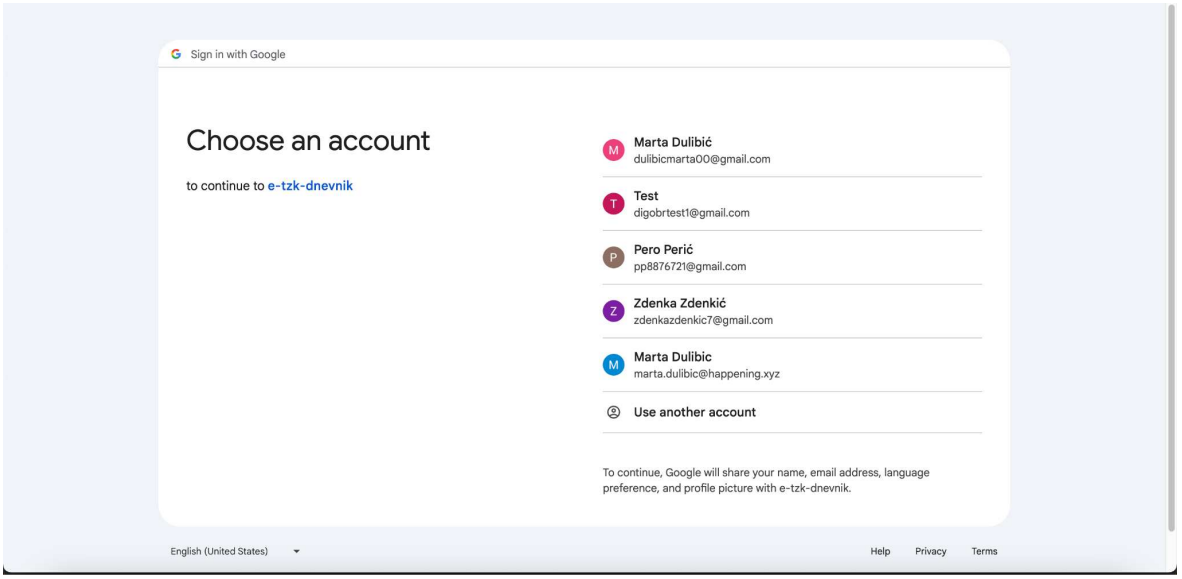
Slika 5.1.1 – Početna stranica sa formom za prijavu

Prilikom pogrešnog unosa korisničkog imena i/ili lozinke, sustav ispisuje poruku *"Neispravno korisničko ime/lozinka"*.



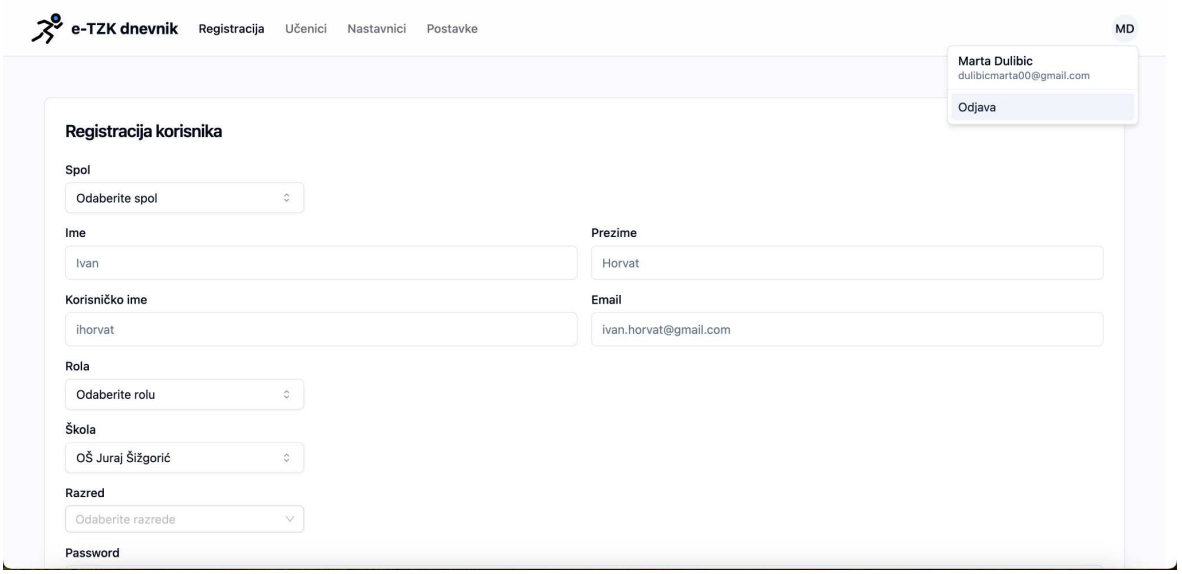
Slika 5.1.2 – Početna stranica - Neispravno korisničko ime/lozinka

Klikom na gumb *"Prijava sa Google računom"*, korisnik je preusmjeren na Googleovu autorizacijsku stranicu gdje mora odabrati račun s kojim se želi prijaviti u sustav.



Slika 5.1.3 – Stranica sa odabirom Google korisničkog računa

Nakon uspješne prijave, korisniku se otvara početna stranica ovisno o njegovoj ulozi (administrator, nastavnik ili učenik). U desnom gornjem kutu navigacijske trake, neovisno o ulozi, nalazi se ikona s korisnikovim inicijalima. Klikom na tu ikonu otvara se modal s opcijom za odjavu iz sustava.

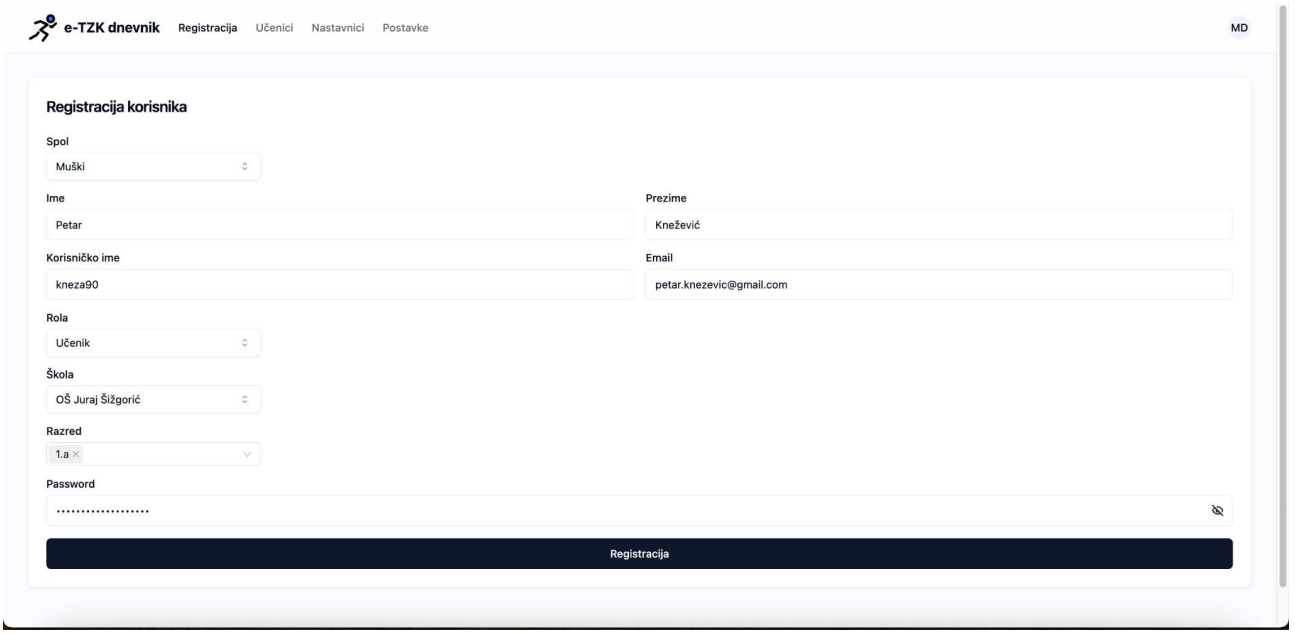


Slika 5.1.4 – Odjava iz sustava

5.2. Administratorsko sučelje

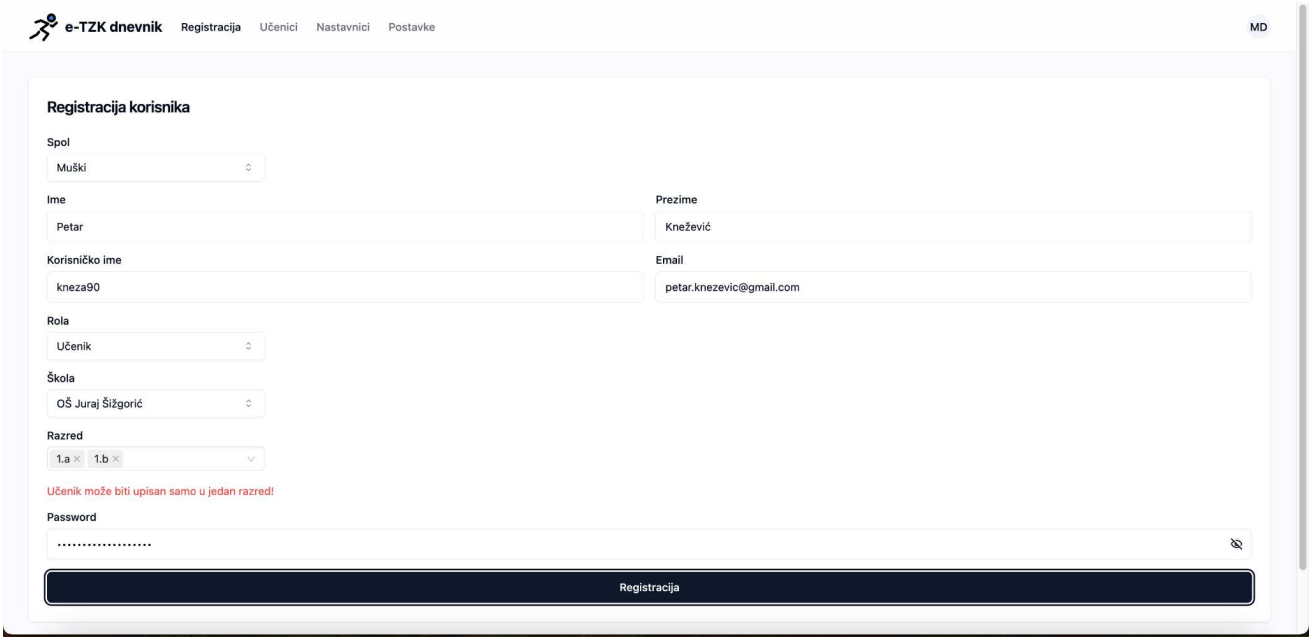
5.2.1. Registracija novih korisnika

Nakon uspješne prijave i autorizacije kao administrator, otvara se stranica sa formom za registraciju novih korisnika. Za dodavanje novih korisnika u sustav, potrebno je unijeti sljedeće podatke: ime, prezime, korisničko ime, email, lozinku, rola (nastavnik ili učenik), školu i razred.



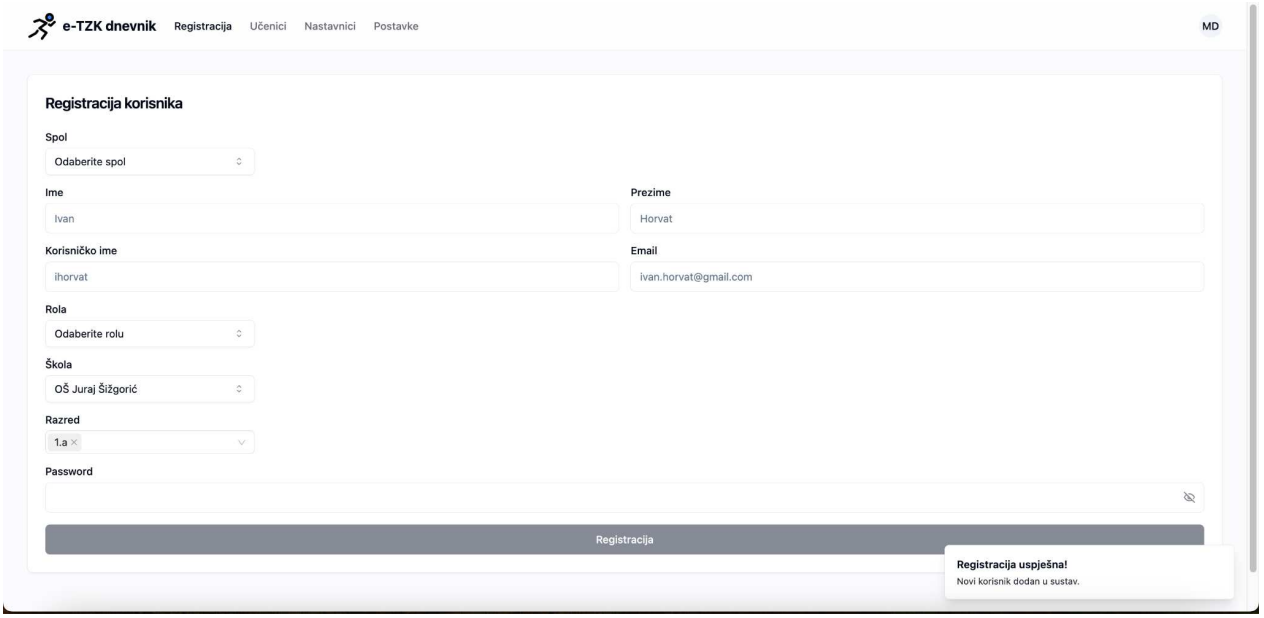
Slika 5.2.1 – Registracijska forma - ispunjeni podaci

Odabirom više od jednog razreda prilikom registracije za rolu "Učenik", sustav ce prikazati poruku upozorenja *"Učenik može biti upisan samo u jedan razred"* u odnosu na nastavnika kojem možemo odabrati vise razreda.



Slika 5.2.2 – Registracijska forma - poruka o pogrešci prilikom odabira za učenika

Nakon uspješne registracije, polja u formi za dodavanje novih korisnika se vraćaju u inicijalno stanje (spremna za unos novih podataka). Administrator je obaviješten o uspješnoj registraciji korisnika porukom koja se prikazuje u donjem desnom kutu ekrana.

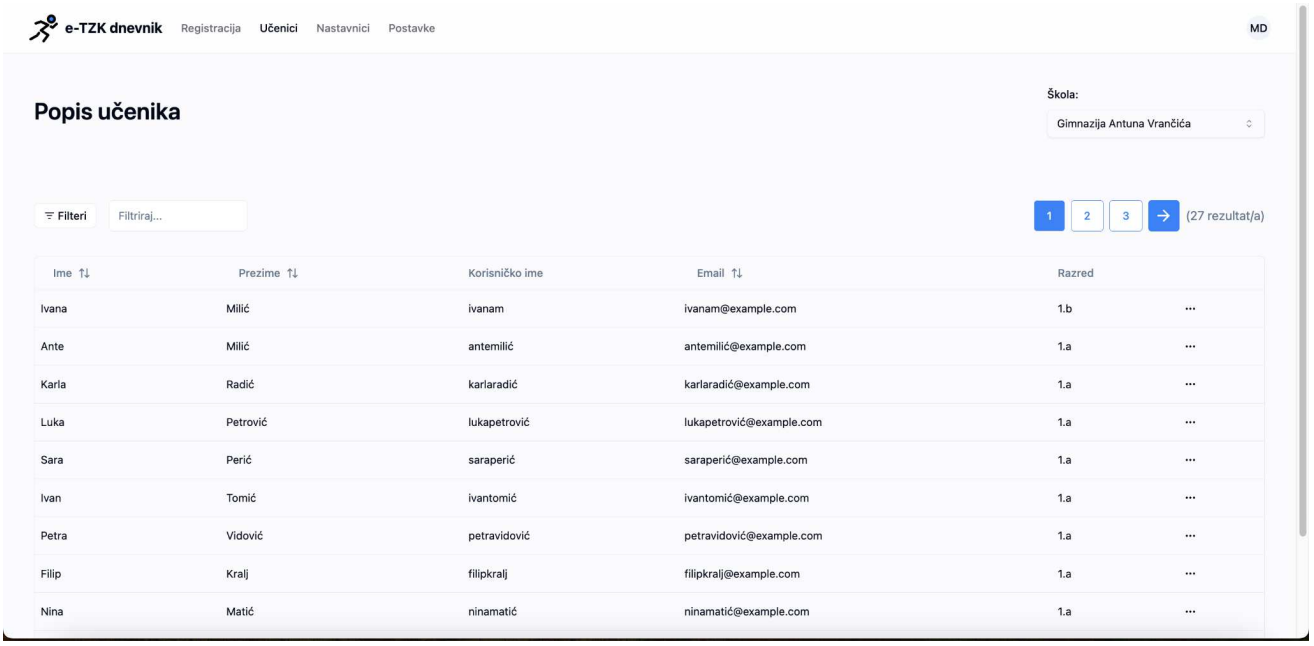


Slika 5.2.3 – Registracijska forma - uspješna registracija

5.2.2. Pregled i upravljanje korisnicima

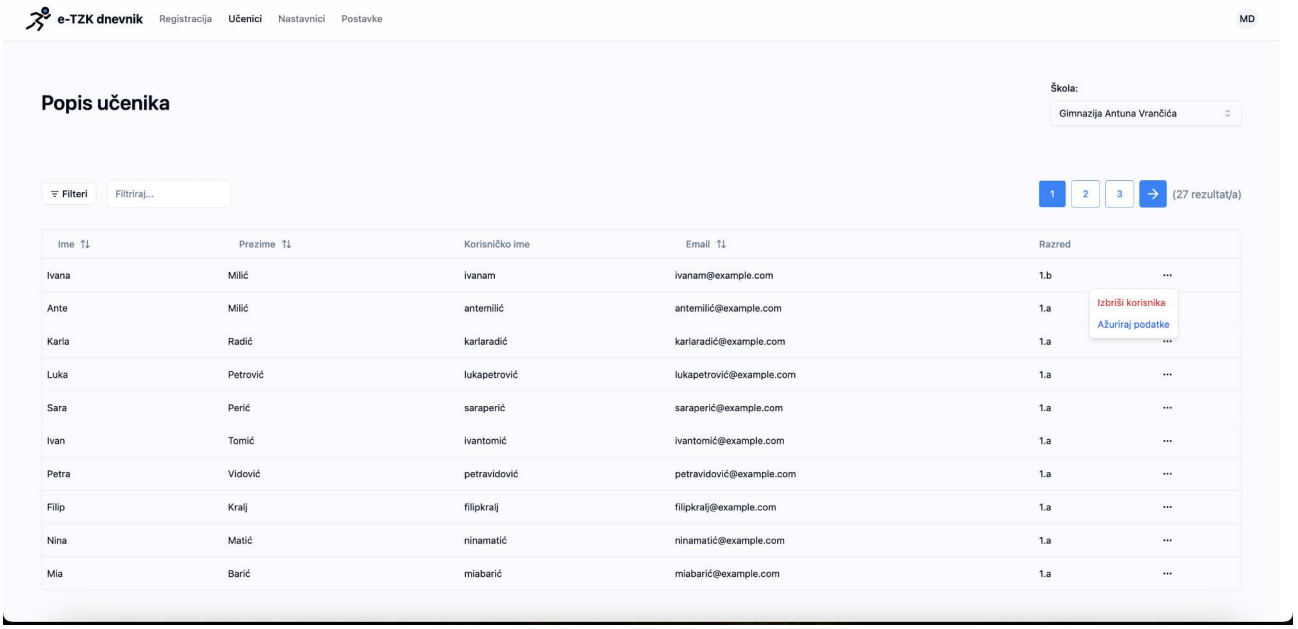
Odabirom opcije *"Učenici"* na navigacijskoj traci, otvara se stranica s popisom učenika. Na vrhu lijevog kutka ekrana nalazi se izbornik za odabir škole. Nakon odabira škole iz

izbornika, prikazat će se popis učenika s paginacijom, prikazujući po 10 učenika po stranici.

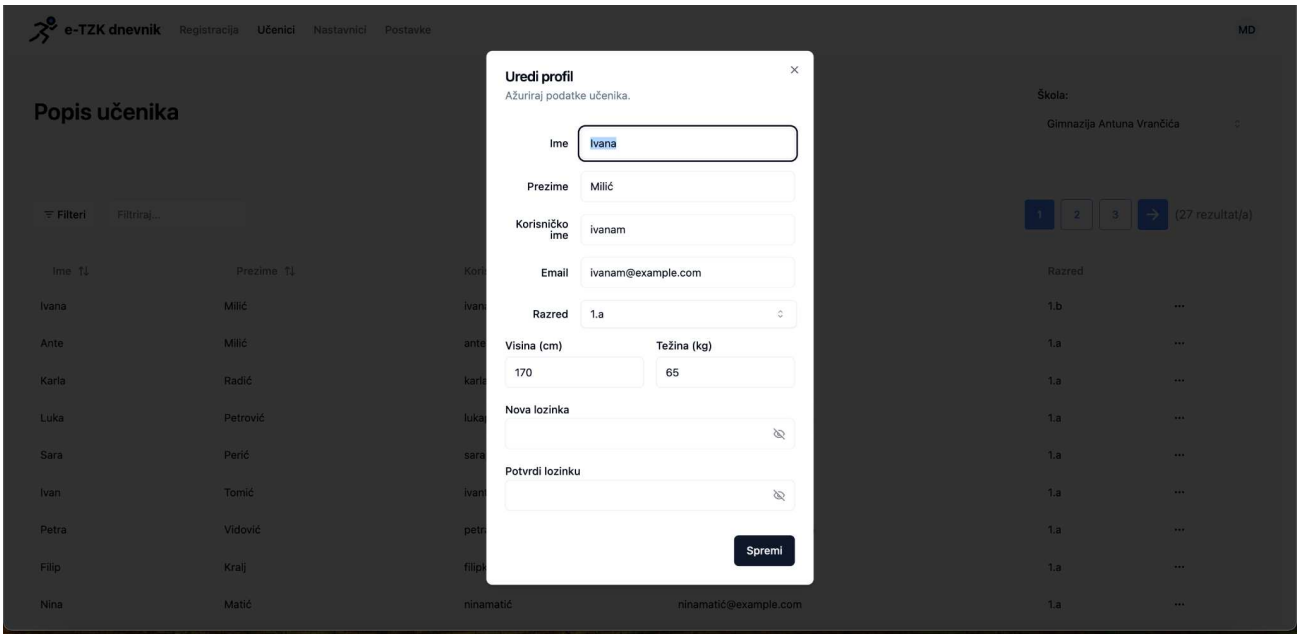


Slika 5.2.4 – Popis učenika - Paginacija

Na stranici postoji mogućnost filtriranja učenika prema predodređenim filterima kao što su ime, prezime i email. Svaki redak u tablici predstavlja jednog učenika. Klikom na *"tri točkice"* pored svakog učenika, administratoru se otvaraju opcije za brisanje korisnika iz sustava i ažuriranje njegovih podataka.

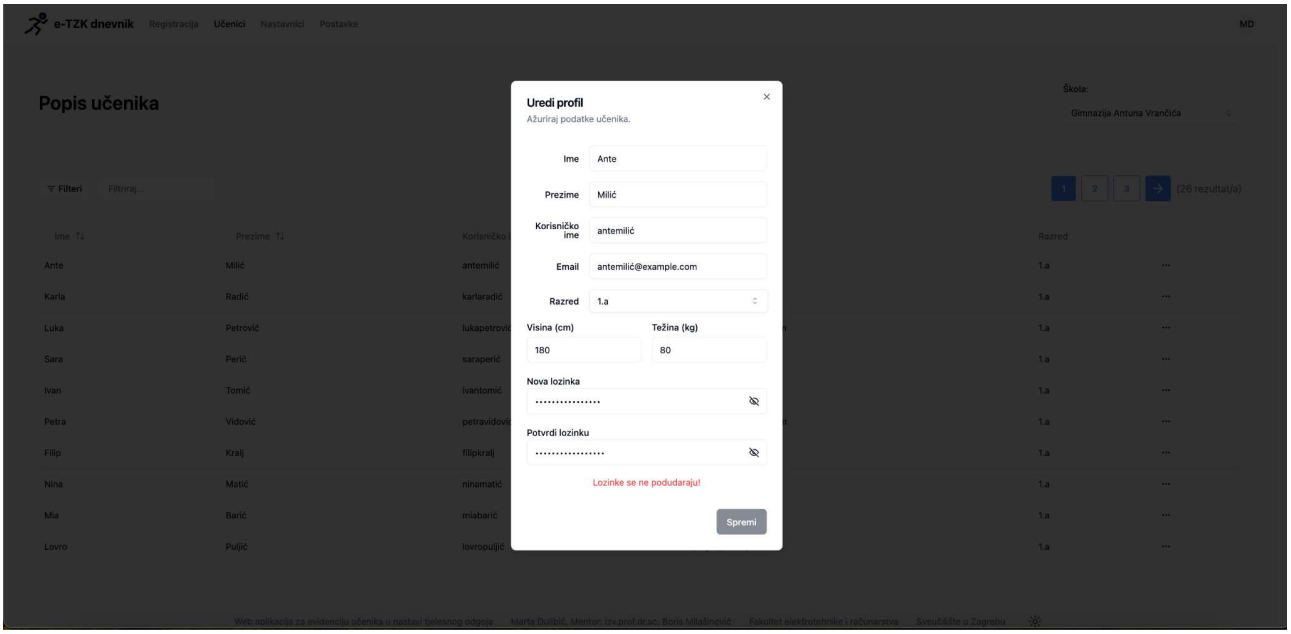


Slika 5.2.5 – Popis učenika - Izbornik za brisanje i ažuriranje



Slika 5.2.6 – Popis učenika - Dijalog za uređivanje podataka o učeniku

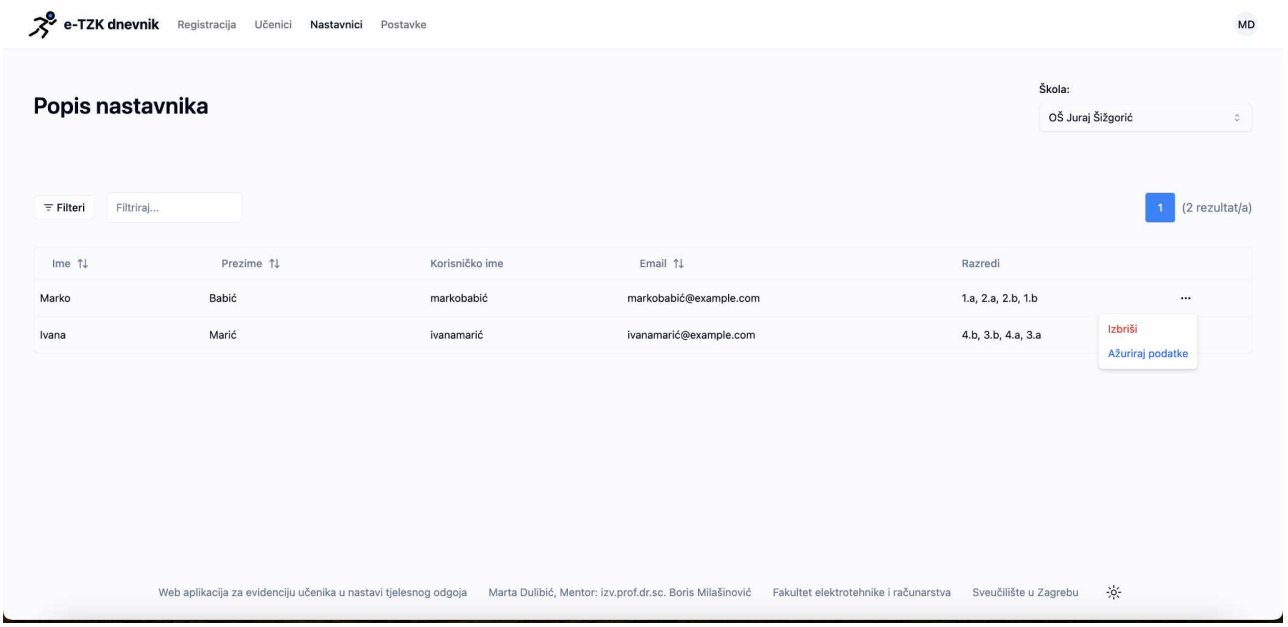
U slučaju nepodudaranja unosa nove lozinke, korisniku se ispisuje poruka *"Lozinke se ne podudaraju"*.



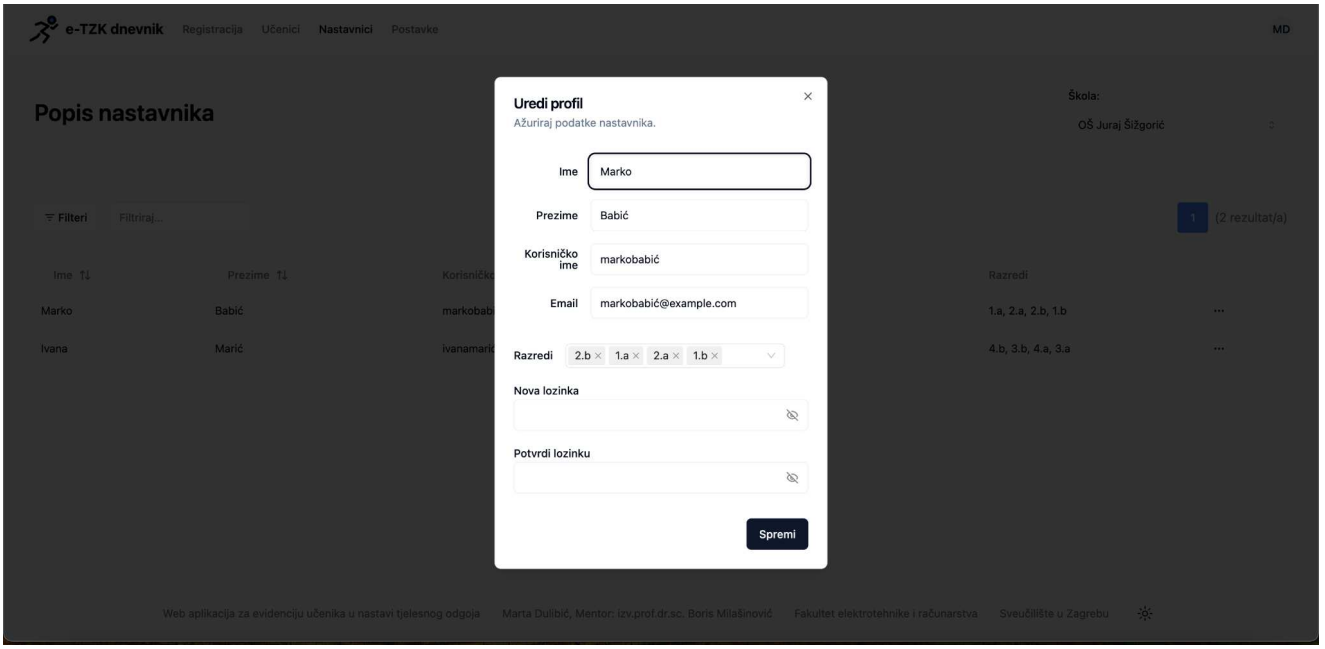
Slika 5.2.7 – Popis učenika - Poruka o ne podudaranje lozinku

Odabirom opcije *"Nastavnici"* na navigacijskoj traci, otvara se stranica s popisom nastavnika. Analogno kao i kod opcije *"Učenici"*, u desnom gornjem kutu ekrana nalazi se izbornik za odabir škole. Nakon odabira škole iz izbornika, prikazat će se popis nastavnika s paginacijom, prikazujući 10 nastavnika po stranici. Na stranici postoji mogućnost filtriranja nastavnika prema predodređenim filterima kao što su ime, prezime i email. Svaki

redak u tablici predstavlja jednog nastavnika. Klikom na *"..."* pored svakog učenika, administratoru se otvaraju opcije za brisanje korisnika iz sustava i ažuriranje njegovih podataka.



Slika 5.2.8 – Popis nastavnika - Izbornik za brisanje i ažuriranje

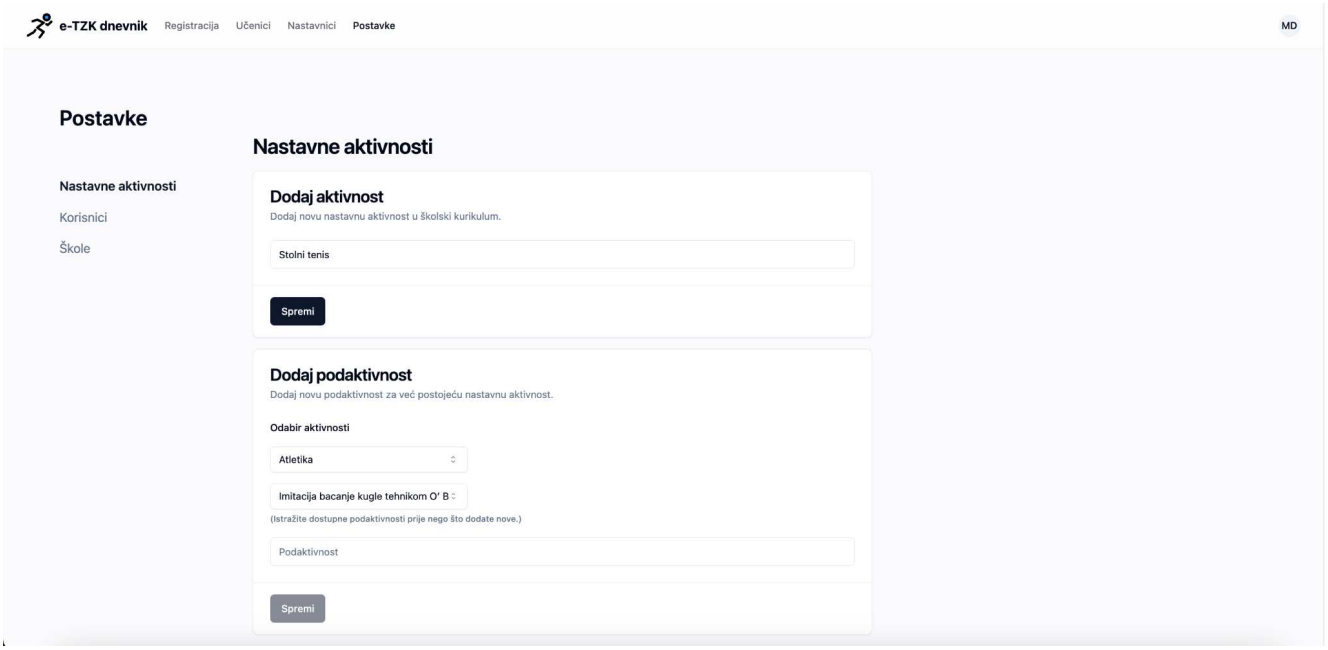


Slika 5.2.9 – Popis nastavnika - Dijalog za uređivanje podataka o nastavniku

Slično kao i kod učenika, pri unosu novih lozinki koje se ne podudaraju, korisnik će dobiti poruku upozorenja.

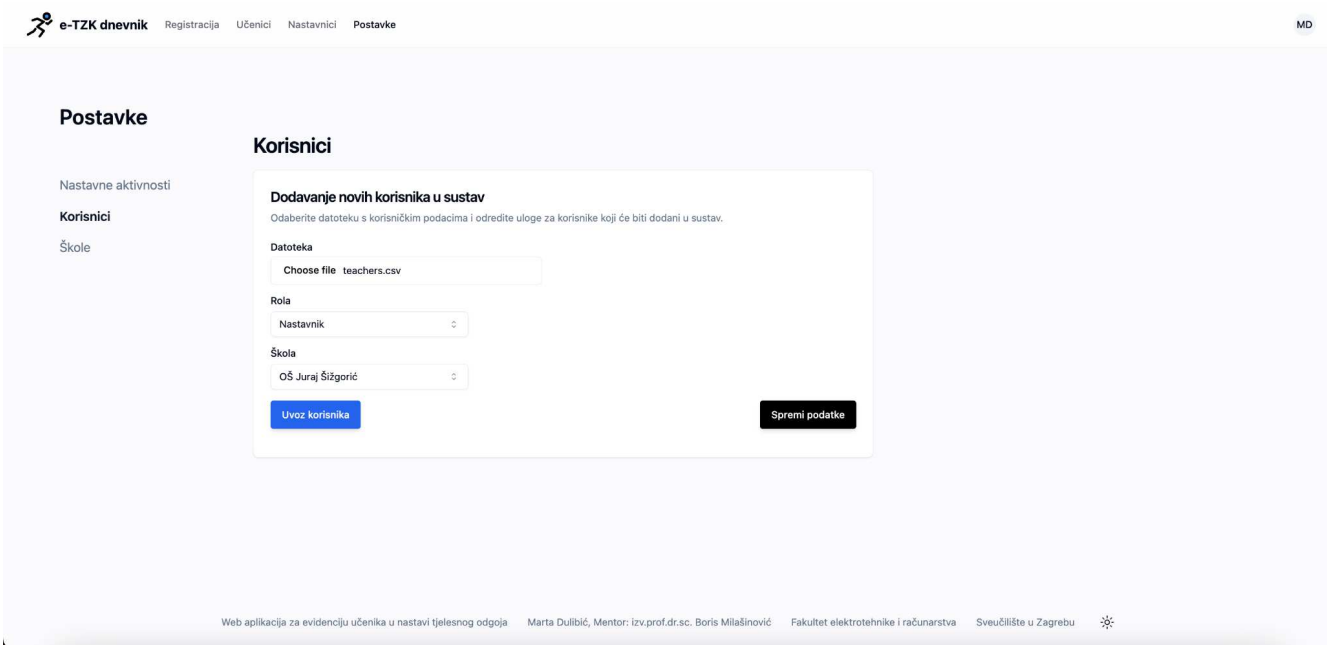
5.2.3. Upravljanje postavkama

Odabirom opcije *"Postavke"* na navigacijskoj traci, otvara se stranica s formama za dodavanje nove aktivnosti ili nove podaktivnosti za postojeće aktivnosti.



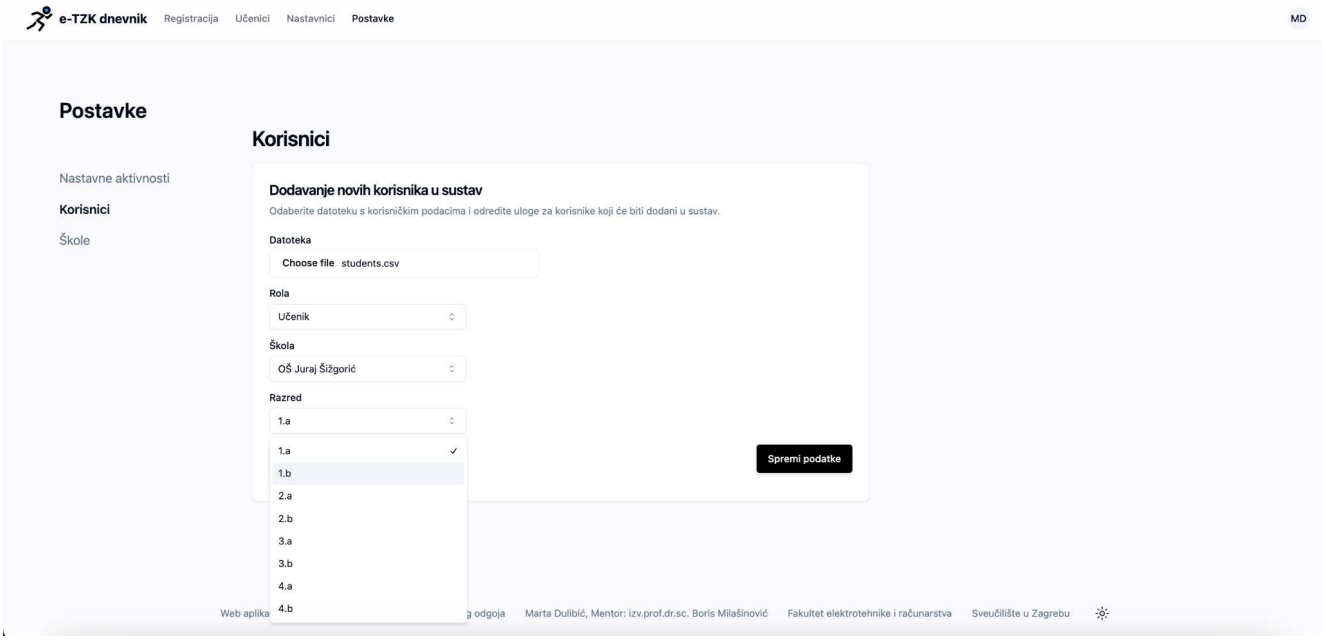
Slika 5.2.10 – Postavke - Forme za dodavanje aktivnosti/podaktivnosti

Za daljnju navigaciju na stranici postoji bočna navigacijska traka s opcijama *"Nastavne aktivnosti"* (što je inicijalno otvoren sadržaj), *"Korisnici"* i *"Škole"*. Odabirom opcije *"Korisnici"*, otvara se forma za dodavanje novih korisnika u sustav putem CSV datoteke. Nakon odabira datoteke sa željenim podacima, odabire se rola za koju se žele dodati korisnici te škola.



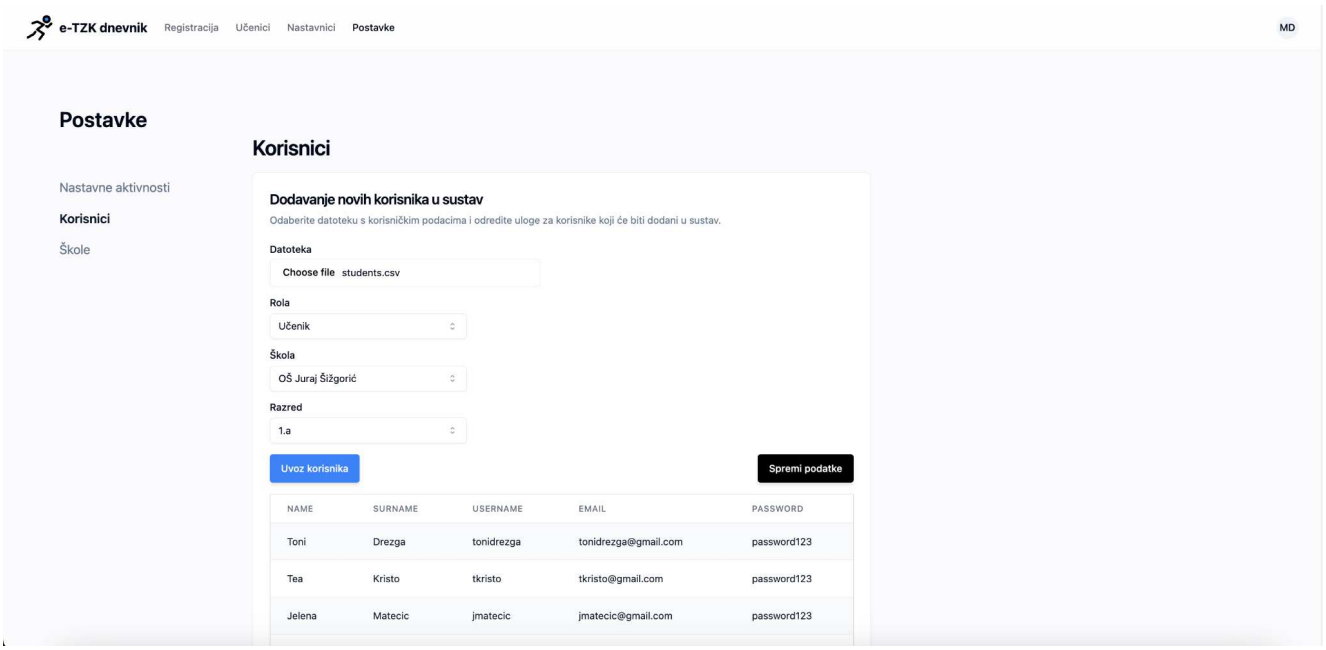
Slika 5.2.11 – Postavke - Forma za dodavanje korisnika (nastavnika)

Za dodavanje učenika, dodatno se može odabrati u koji razred ih se želi upisati.



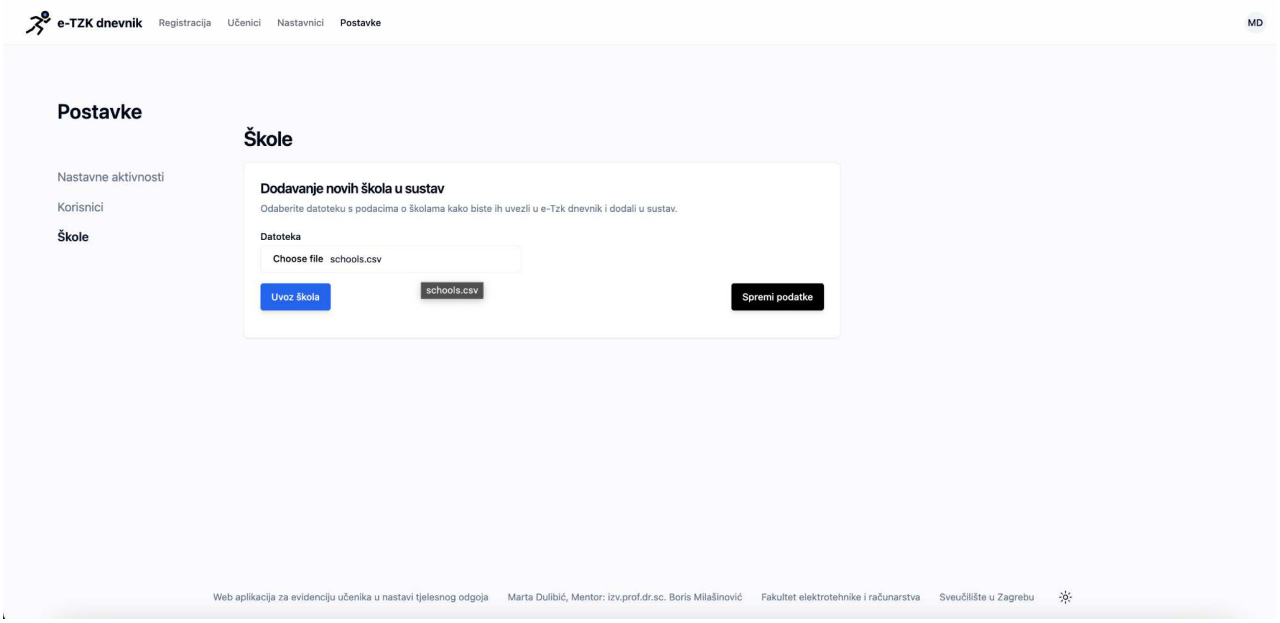
Slika 5.2.12 – Postavke - Forma za dodavanje korisnika (učenika)

Klikom na gumb *"Uvoz podataka"*, na stranici se prikazuje tablica s korisnicima čiji se podaci mogu pregledati prije samog dodavanja u sustav klikom na gumb *"Spremi podatke"*.



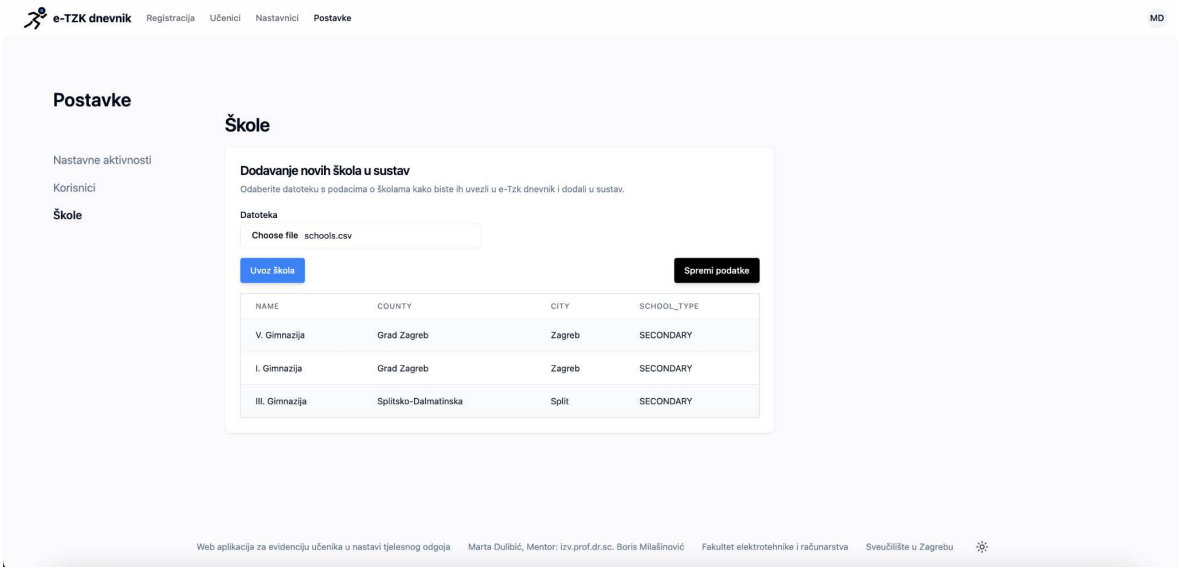
Slika 5.2.13 – Postavke - Forma za dodavanje korisnika - pregled podataka

Odabirom opcije "*Škole*", otvara se forma dodavanje novih škola u sustav.



Slika 5.2.14 – Postavke - Forma za dodavanje škola

Nakon odabira datoteke s željenim podacima, analogno kao i kod uvoza novih korisnika, klik na gumb *"Uvoz podataka"*, na stranici će prikazati tablicu s školama čiji se podaci mogu pregledati prije samog dodavanja u sustav klikom na gumb *"Spremi podatke"*.

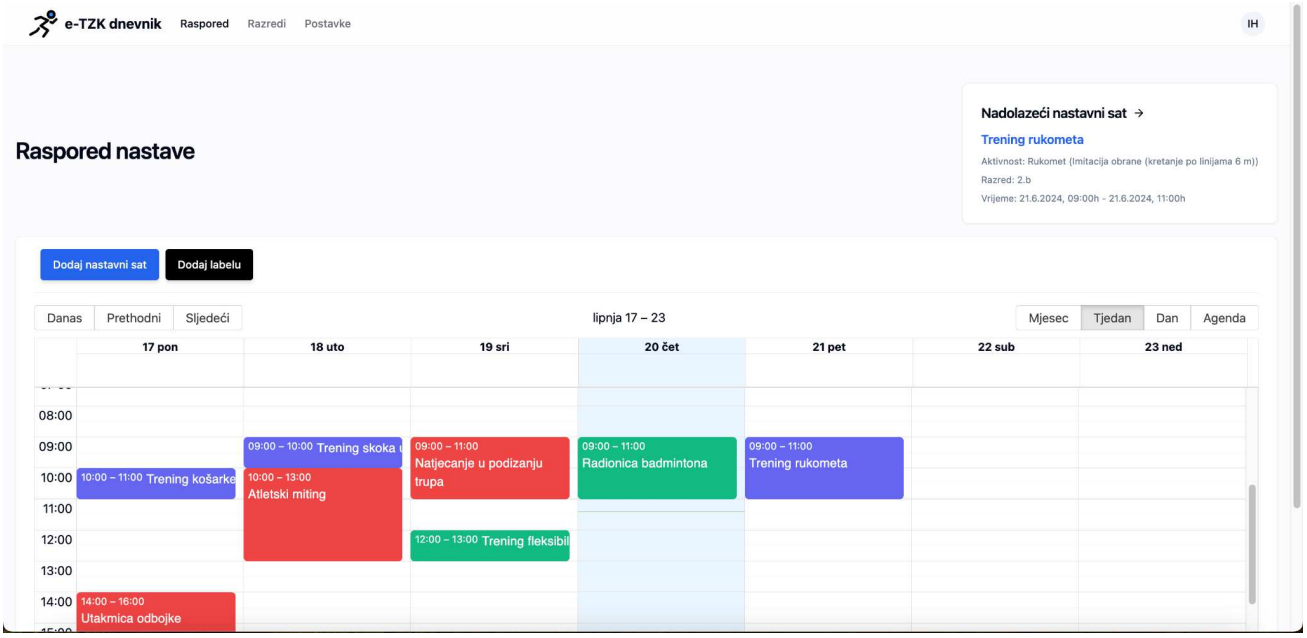


Slika 5.2.15 – Postavke - Forma za dodavanje škola - pregled podataka

5.3. Sučelje za nastavnike

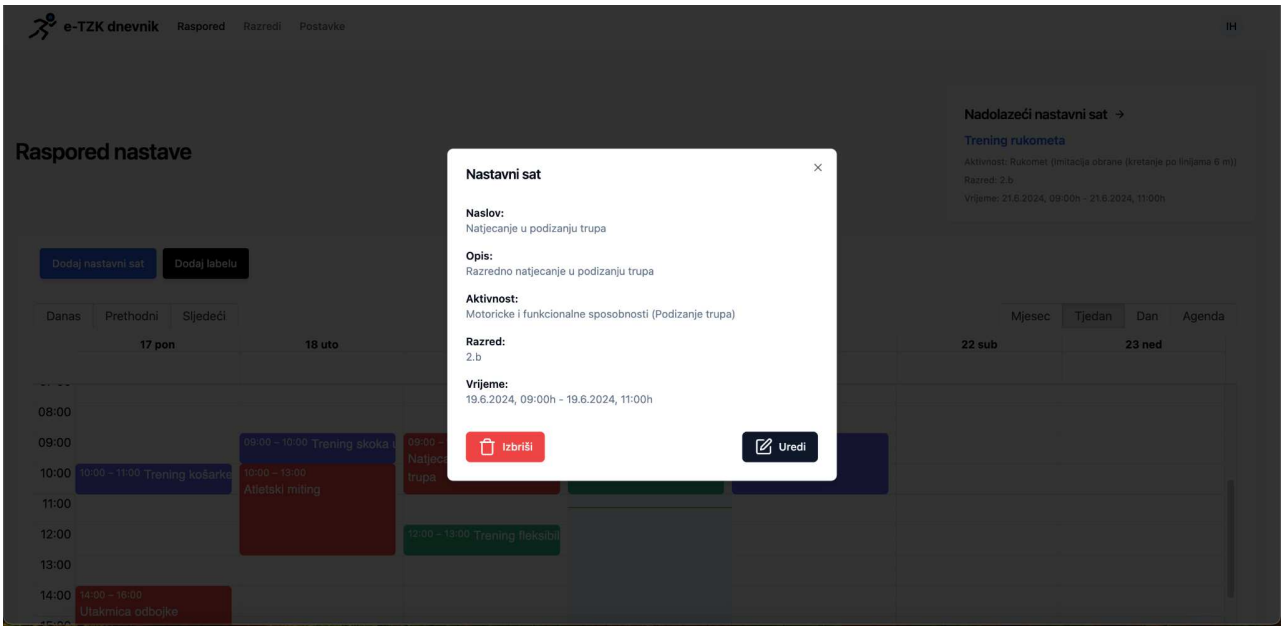
5.3.1. Pregled i upravljanje rasporedom

Nakon uspješne prijave i autorizacije kao nastavnik, otvara se stranica sa rasporedom nastave u obliku kalendara. Kroz kalendar se mogu pregledati svi nadolazeće nastavni sati u danu, tjednu ili mjesecu.



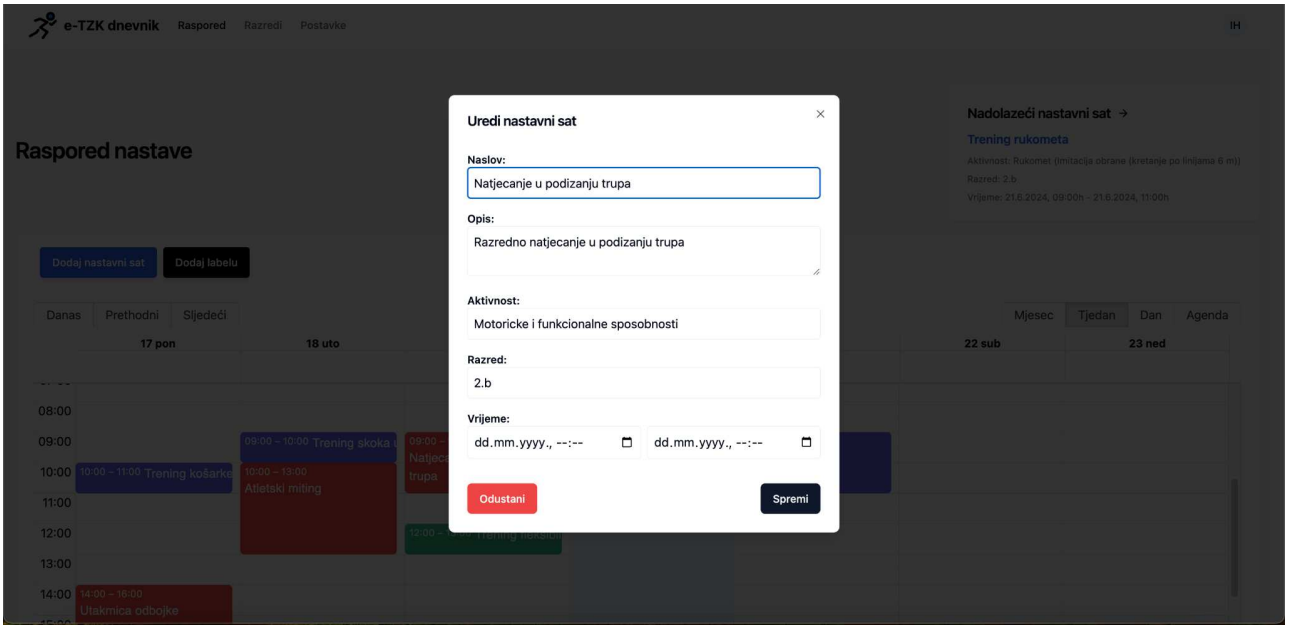
Slika 5.3.1 – Raspored - pregled kalendara nastave

Klikom na nastavni sat u kalendaru, otvara se dijalog s detaljima koji uključuje naziv i opis nastavnog sata, odabranu aktivnost/podaktivnost, razred i vrijeme održavanja. Dijalog također sadrži gumbe za brisanje sata (*"Izbriši"*) iz rasporeda te za uređivanje podataka ("Uredi").



Slika 5.3.2 – Raspored - pregled detalja o nastavnom satu

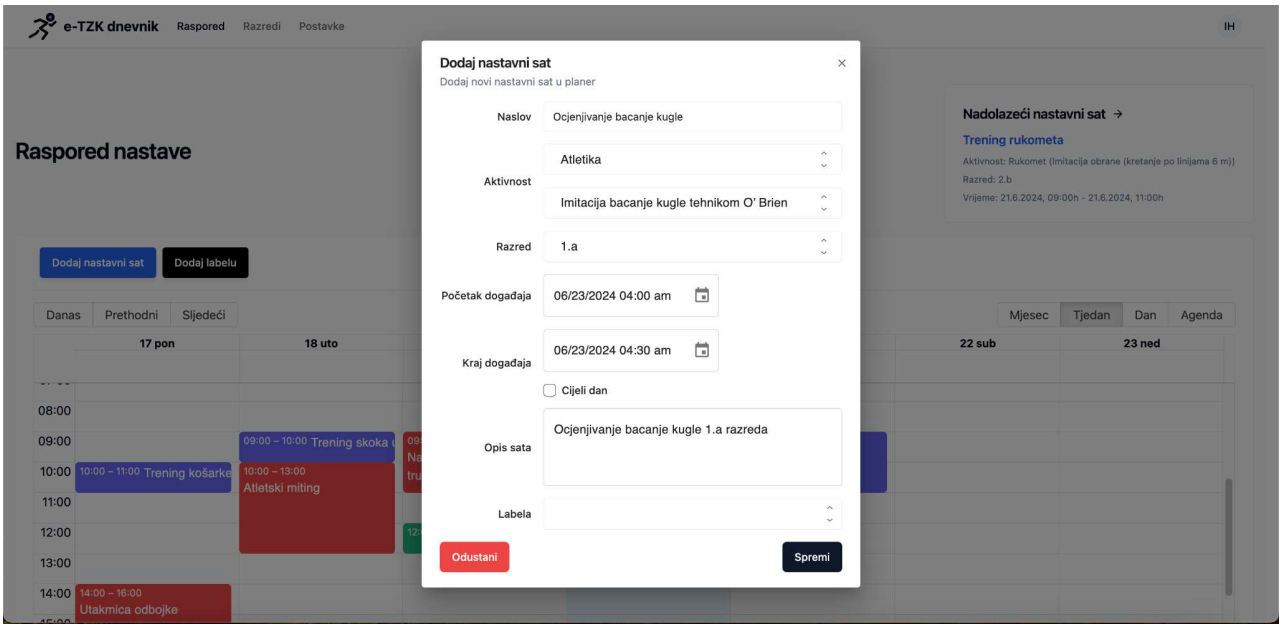
Klikom na gumb *"Uredi"* u dijalogu, omogućava se uređivanje sadržaja. Nakon unosa potrebnih promjena, novi podaci se spremaju klikom na gumb *"Spremi"*. Ako nastavnik odustane od uređivanja, može izaći iz dijaloga klikom na gumb *"Odustani"*.



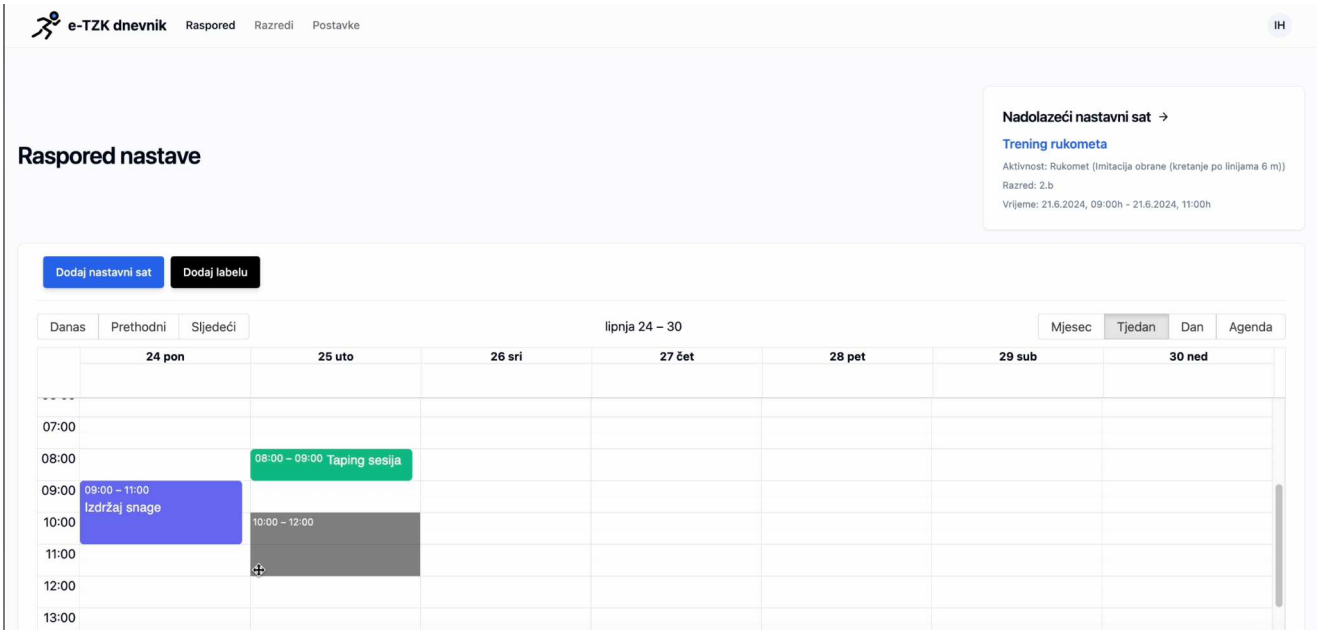
Slika 5.3.3 – Raspored - uređivanje detalja o nastavnom satu

Novi nastavni satovi mogu se dodati u kalendar putem zasebne forme klikom na gumb *"Dodaj nastavni sat"* (Slika 5.23) ili odabirom odgovarajućeg vremenskog okvira unutar samog kalendara (Slika 5.24, Slika 5.25).

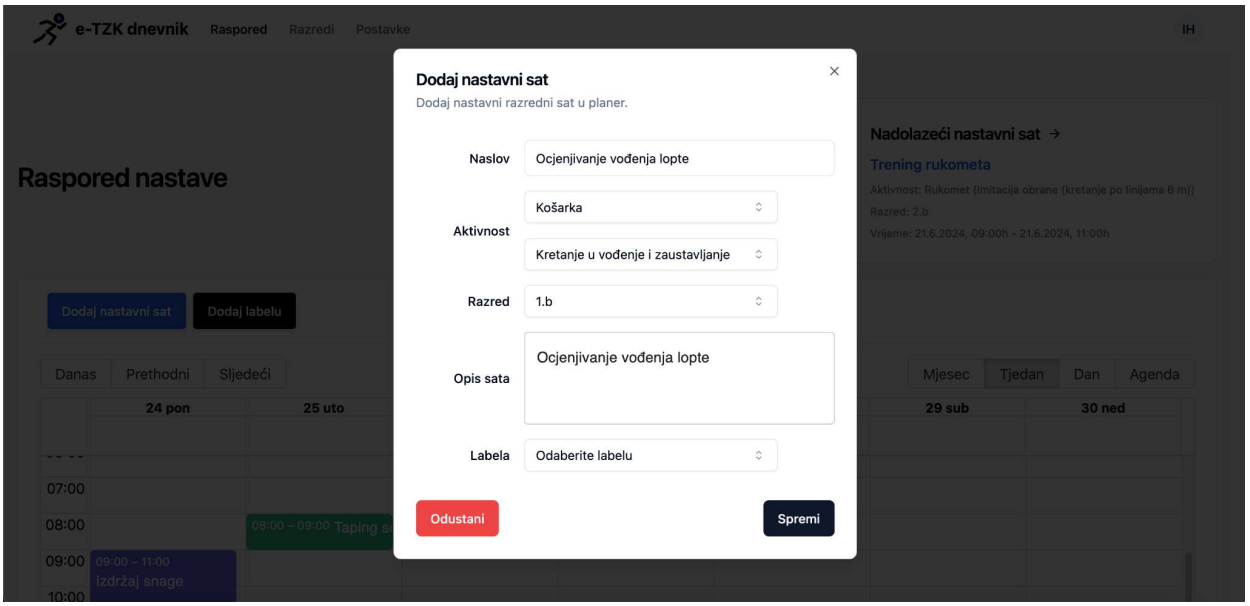
Za dodavanje nastavnog sata potrebno je unijeti naslov, odabrati aktivnost/podaktivnost koja će se odvijati, razred, te postaviti početak i kraj sata. Dodatno se može unijeti opis sata i odabrati jednu od dostupnih oznaka (labela).



Slika 5.3.4 – Raspored - dodavanje novog nastavnog sata klikom na gumb *"Dodaj nastavni sat"*

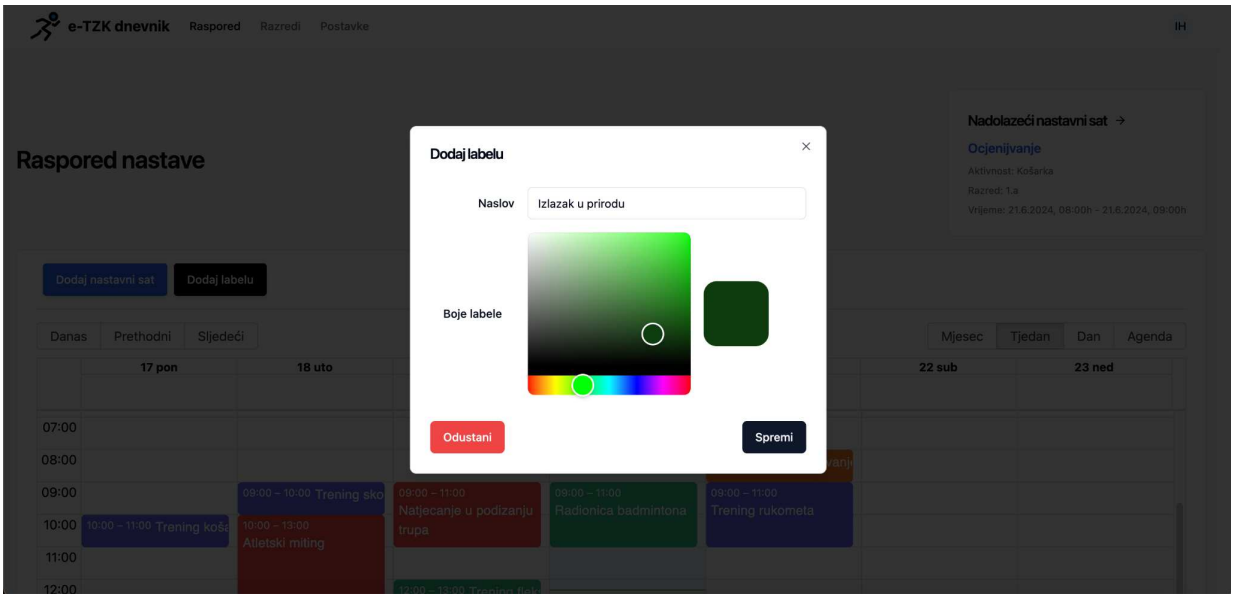


Slika 5.3.5 – Raspored - odabir vremenskog okvira unutar kalendara za dodavanje novog sata



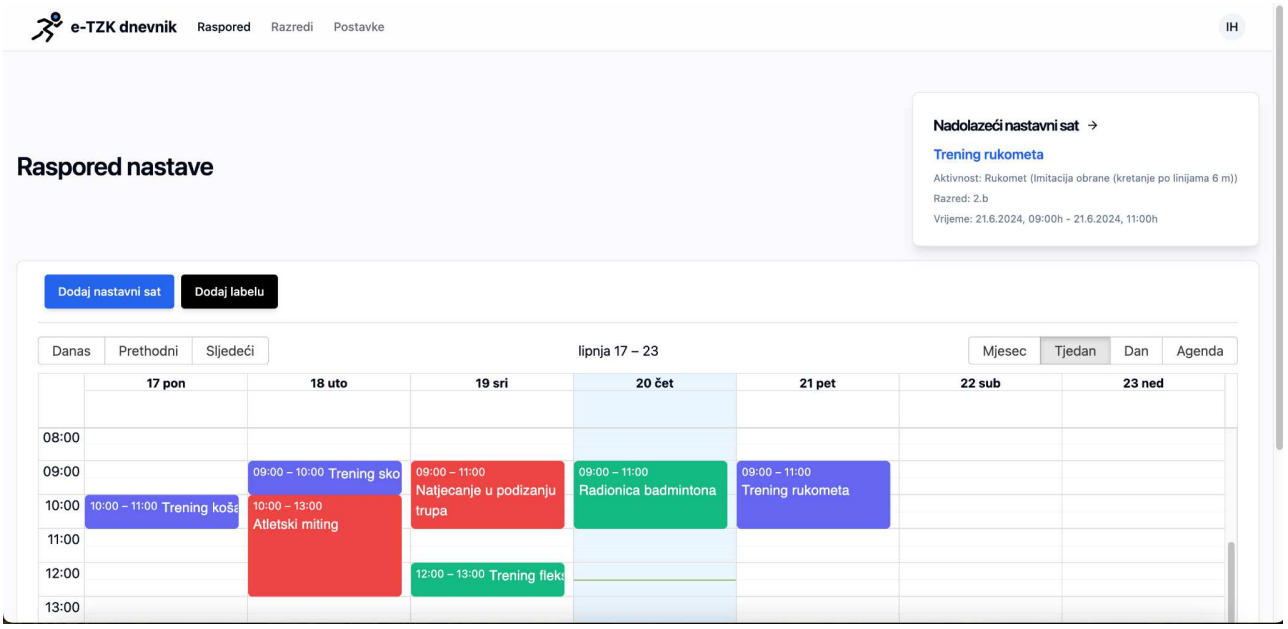
Slika 5.3.6 – Raspored - dodavanje novog nastavnog sata

Klikom na gumb *"Dodaj labelu"*, otvara se dijalog s mogućnošću dodavanja vlastitih oznaka u sustav. Nastavnik može koristiti ove oznake za posebno "označavanje" nastavnih sati. Prilikom dodavanja nove oznake, nastavnik treba unijeti njen naziv i odabrati boju kojom će biti obojena kartica s nastavnim satom kojem je ta oznaka dodijeljena (Slika 5.24).



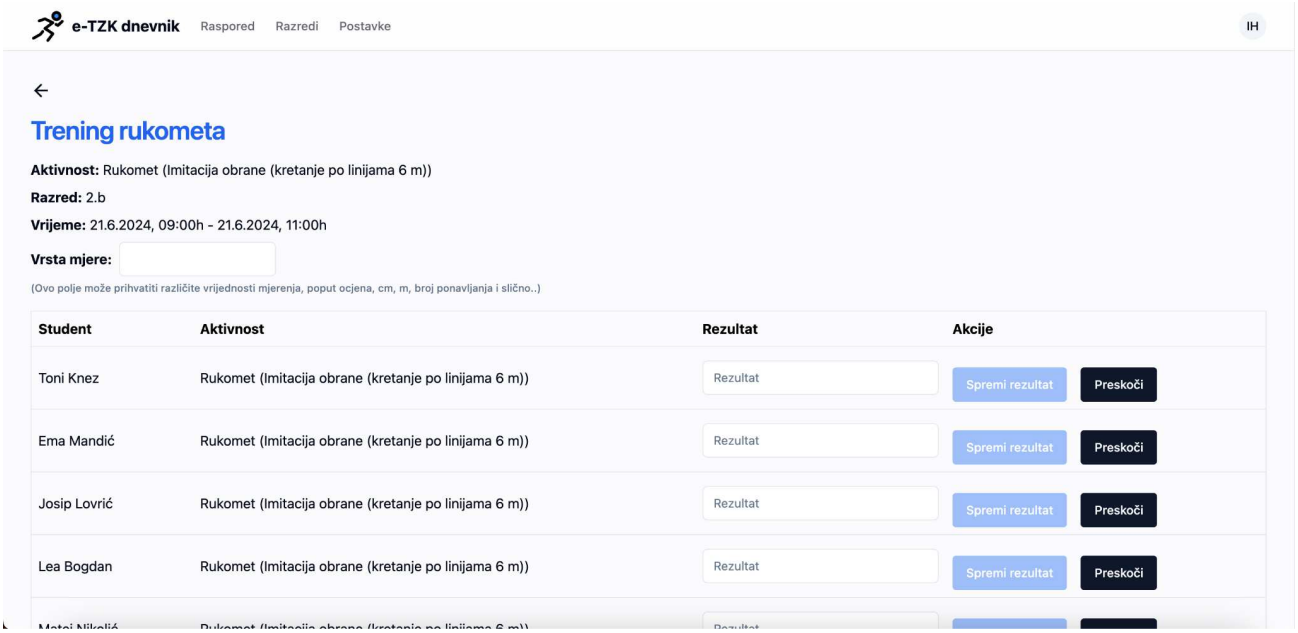
Slika 5.3.7 – Raspored - dodavanje nove labele

U gornjem desnom kutu iznad kalendara nalazi se prečac do nadolazećeg nastavnog sata.



Slika 5.3.8– Raspored - prečac do nadolazećeg nastavnog sata

Klikom na prečac, nastavniku se otvara nova stranica za upisivanje rezultata učenika za aktivnost koja je planirana za taj nastavni sat. Nastavnik može pregledati popis učenika u razredu i unijeti njihove rezultate. Također, ima mogućnost preskočiti učenike koji u tom trenutku nisu prisutni ili su onemogućeni za izvršavanje aktivnosti.



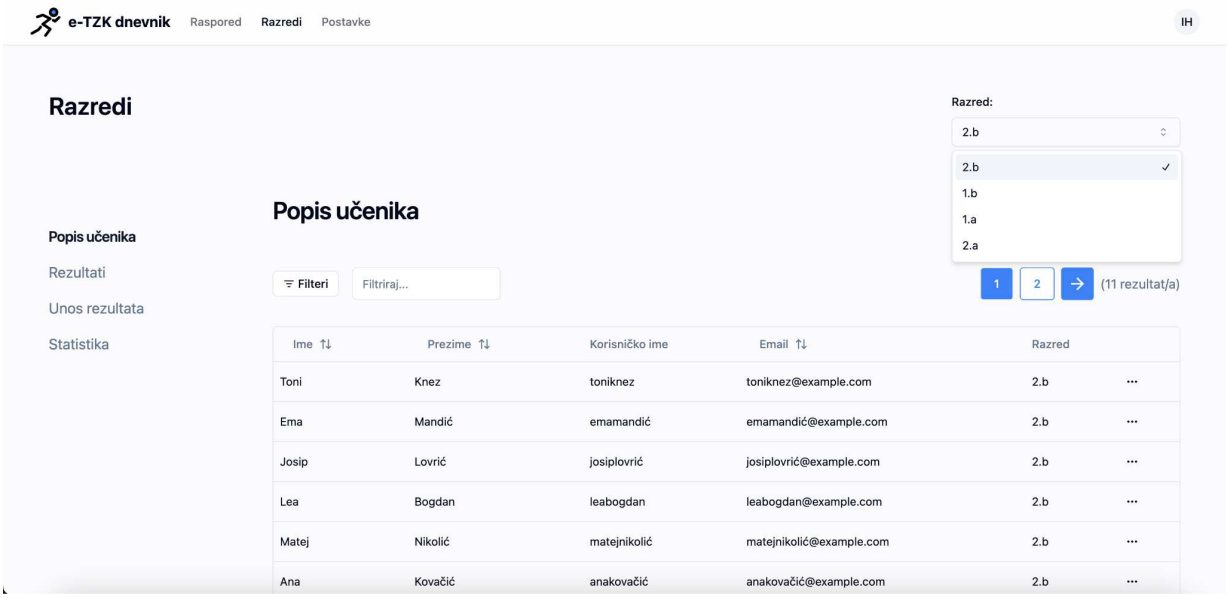
Slika 5.3.9 – Dodavanje rezultata za planiranu aktivnost

5.3.2. Pregled i upravljanje razredima

Odabirom opcije *"Razredi"* na navigacijskoj traci, otvara se stranica s popisom učenika. Za daljnju navigaciju na stranici postoji bočna navigacijska traka s opcijama *"Popis učenika"* (što je inicijalno otvoren sadržaj), *"Rezultati"*, *"Unos rezultata"* i *"Statistika"*. U gornjem desnom kutu ekrana nalazi se izbornik za odabir razreda.

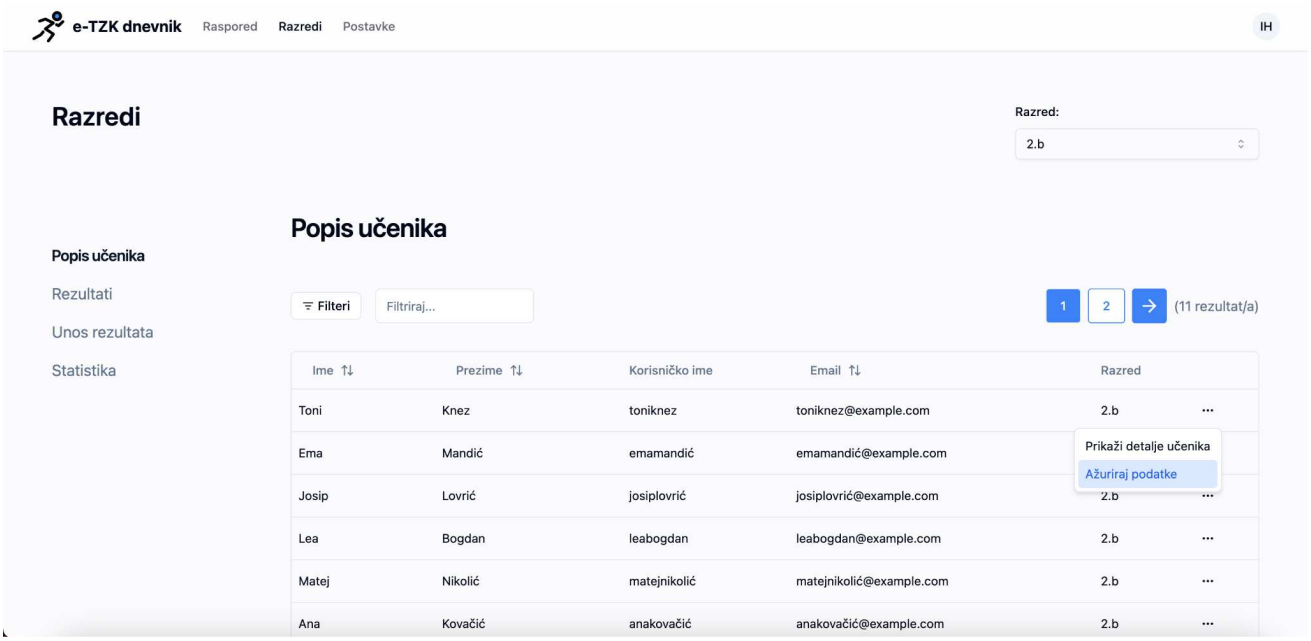
5.3.2.1 Popis učenika

Odabirom opcije *"Popis učenika"*, otvara se stranica s popisom učenika odabranog razreda s paginacijom, prikazujući po 10 učenika po stranici.



Slika 5.3.10 – Popis učenika - odabir razreda

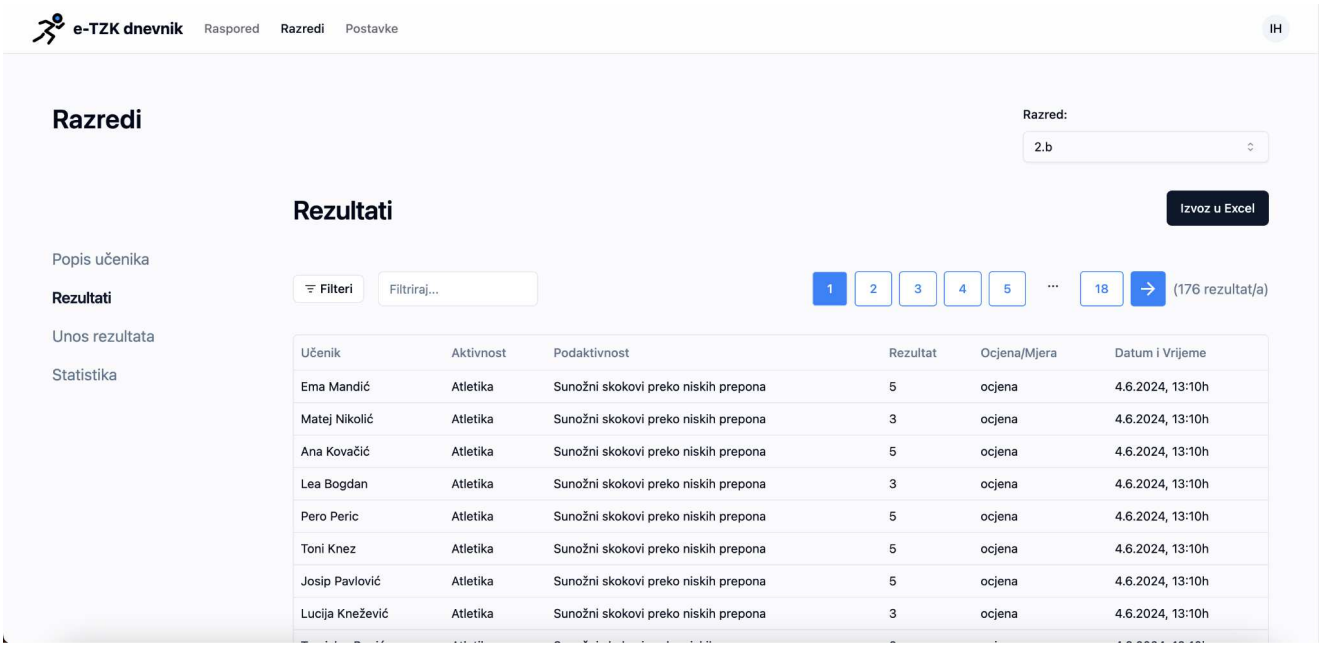
Na stranici postoji mogućnost filtriranja učenika prema predodređenim filterima kao što su ime, prezime i email. Svaki redak u tablici predstavlja jednog učenika. Klikom na *"tri točkice"* pored svakog učenika, otvaraju se opcije za pregled detalja i ažuriranje podataka o učeniku (vidi Slika 5.10). Odabirom opcije *"Prikaži detalje o učeniku"* otvara se nova stranica s profilom učenika, što će biti detaljno objašnjeno u poglavlju 5.3.3.



Slika 5.3.11 – Popis učenika - izbornik sa akcijama

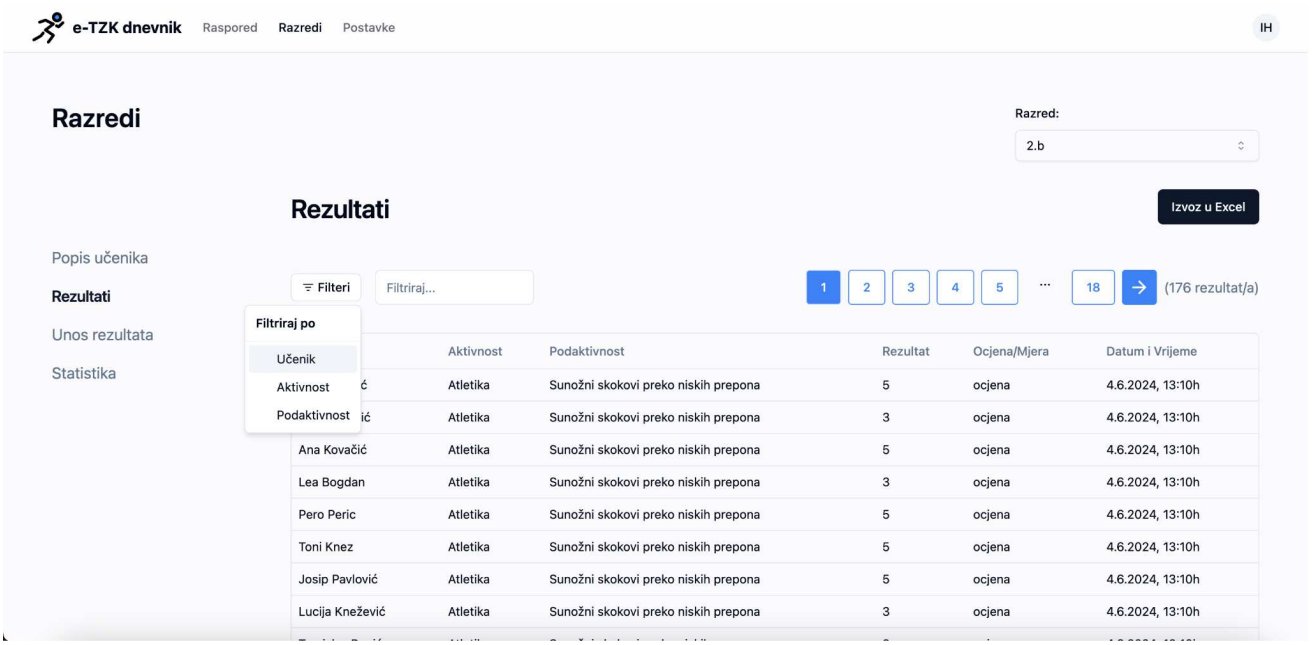
5.3.2.2 Rezultati

Odabirom opcije *"Rezultati"*, otvara se stranica s popisom rezultata učenika odabranog razreda s paginacijom, prikazujući po 10 rezultata po stranici.

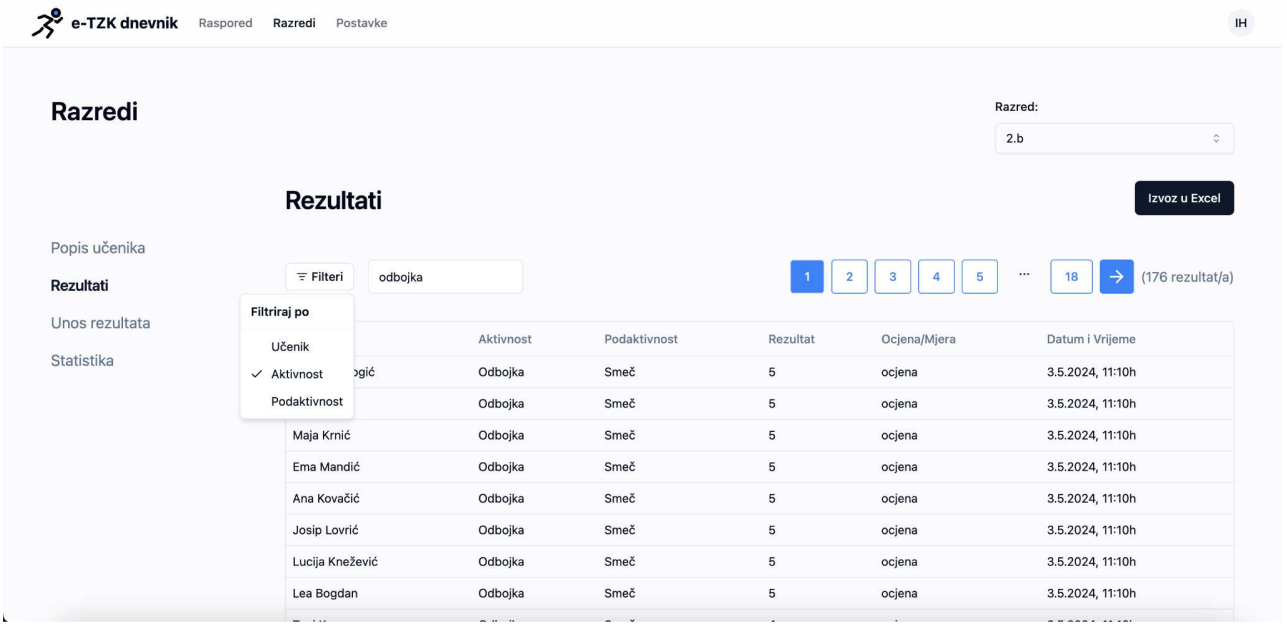


Slika 5.3.12 – Rezultati učenika - paginacija

Na stranici postoji mogućnost filtriranja rezultatima prema predodređenim filterima kao što su učenik, aktivnost i podaktivnost.

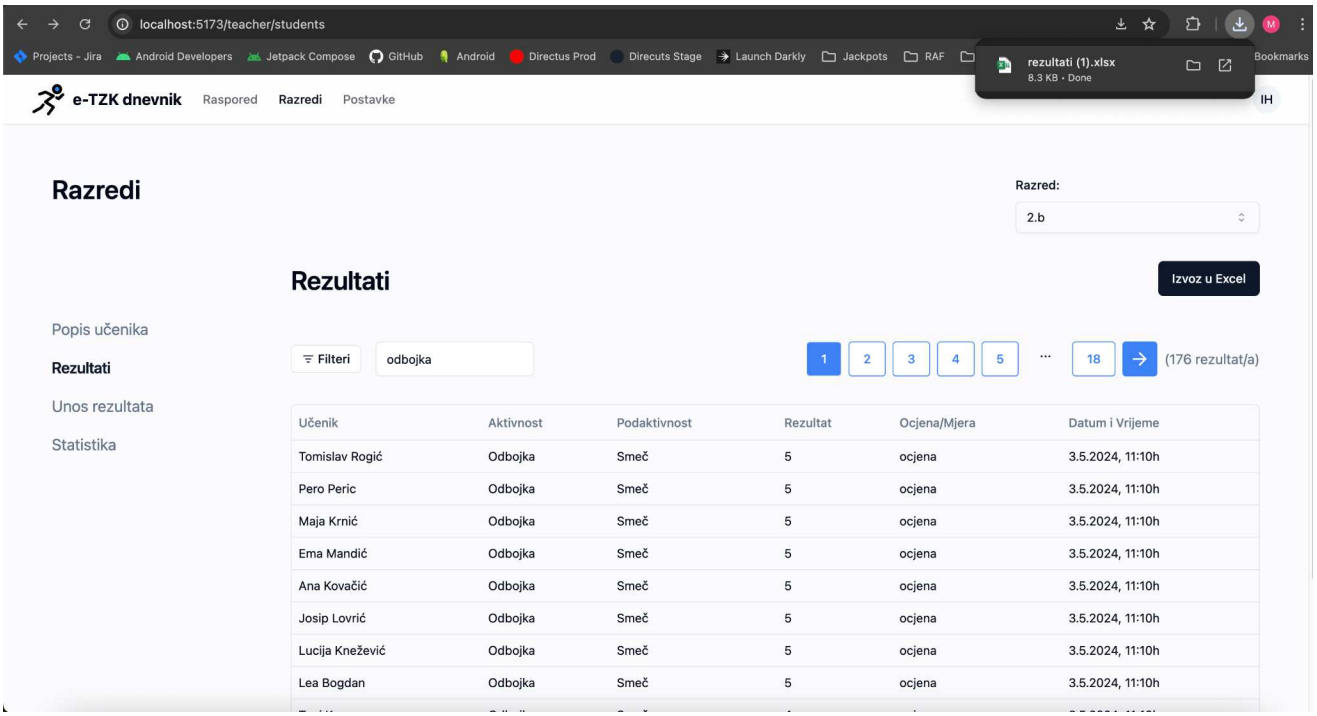


Slika 5.3.13 – Rezultati učenika - odabir filtera

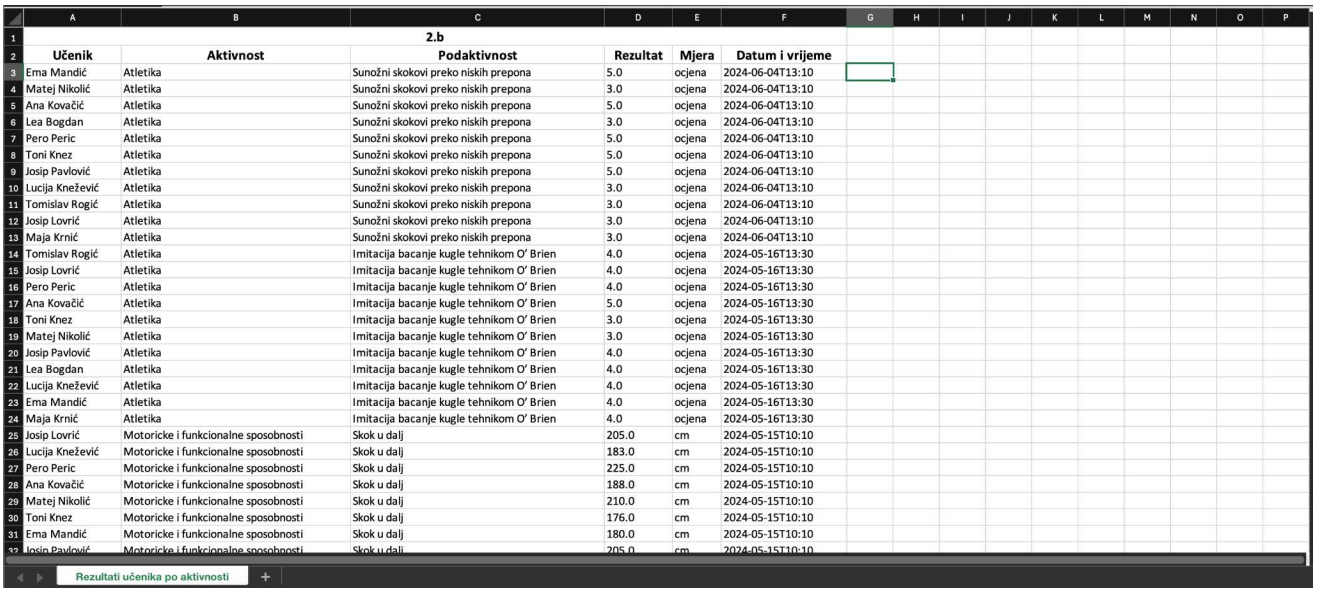


Slika 5.3.14 – Rezultati učenika - filtriranje po aktivnosti

Klikom na gumb *"Izvoz u Excel"* u gornjem desnom kutu, rezultati razreda se izvoze u Excel format i preuzimaju na lokalno računalo nastavnika. Nastavnik ih može pregledati ili ih koristiti za daljnju distribuciju i analizu.



Slika 5.3.15 – Rezultati učenika - preuzimanje exelice s rezultatima



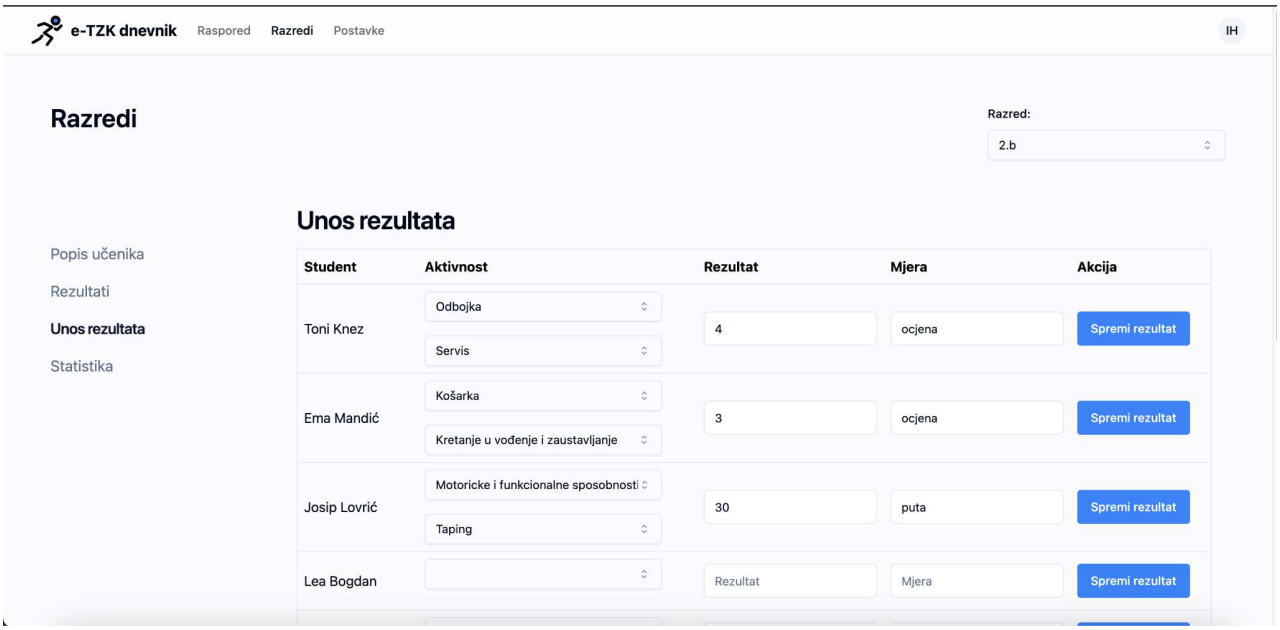
	A	B	C	D	E	F
1			2.b			
2	Učenik	Aktivnost	Podaktivnost	Rezultat	Mjera	Datum i vrijeme
3	Ema Mandić	Atletika	Sunožni skokovi preko niskih prepona	5.0	ocjena	2024-06-04T13:10
4	Matej Nikolić	Atletika	Sunožni skokovi preko niskih prepona	3.0	ocjena	2024-06-04T13:10
5	Ana Kovačić	Atletika	Sunožni skokovi preko niskih prepona	5.0	ocjena	2024-06-04T13:10
6	Lea Bogdan	Atletika	Sunožni skokovi preko niskih prepona	3.0	ocjena	2024-06-04T13:10
7	Pero Peric	Atletika	Sunožni skokovi preko niskih prepona	5.0	ocjena	2024-06-04T13:10
8	Toni Knez	Atletika	Sunožni skokovi preko niskih prepona	5.0	ocjena	2024-06-04T13:10
9	Josip Pavlović	Atletika	Sunožni skokovi preko niskih prepona	5.0	ocjena	2024-06-04T13:10
10	Lucija Knežević	Atletika	Sunožni skokovi preko niskih prepona	3.0	ocjena	2024-06-04T13:10
11	Tomislav Rogić	Atletika	Sunožni skokovi preko niskih prepona	3.0	ocjena	2024-06-04T13:10
12	Josip Lovrić	Atletika	Sunožni skokovi preko niskih prepona	3.0	ocjena	2024-06-04T13:10
13	Maja Krnić	Atletika	Sunožni skokovi preko niskih prepona	3.0	ocjena	2024-06-04T13:10
14	Tomislav Rogić	Atletika	Imitacija bacanje kugle tehnikom O' Brien	4.0	ocjena	2024-05-16T13:30
15	Josip Lovrić	Atletika	Imitacija bacanje kugle tehnikom O' Brien	4.0	ocjena	2024-05-16T13:30
16	Pero Peric	Atletika	Imitacija bacanje kugle tehnikom O' Brien	4.0	ocjena	2024-05-16T13:30
17	Ana Kovačić	Atletika	Imitacija bacanje kugle tehnikom O' Brien	5.0	ocjena	2024-05-16T13:30
18	Toni Knez	Atletika	Imitacija bacanje kugle tehnikom O' Brien	3.0	ocjena	2024-05-16T13:30
19	Matej Nikolić	Atletika	Imitacija bacanje kugle tehnikom O' Brien	3.0	ocjena	2024-05-16T13:30
20	Josip Pavlović	Atletika	Imitacija bacanje kugle tehnikom O' Brien	4.0	ocjena	2024-05-16T13:30
21	Lea Bogdan	Atletika	Imitacija bacanje kugle tehnikom O' Brien	4.0	ocjena	2024-05-16T13:30
22	Lucija Knežević	Atletika	Imitacija bacanje kugle tehnikom O' Brien	4.0	ocjena	2024-05-16T13:30
23	Ema Mandić	Atletika	Imitacija bacanje kugle tehnikom O' Brien	4.0	ocjena	2024-05-16T13:30
24	Maja Krnić	Atletika	Imitacija bacanje kugle tehnikom O' Brien	4.0	ocjena	2024-05-16T13:30
25	Josip Lovrić	Motoricke i funkcionalne sposobnosti	Skok u dalj	205.0	cm	2024-05-15T10:10
26	Lucija Knežević	Motoricke i funkcionalne sposobnosti	Skok u dalj	183.0	cm	2024-05-15T10:10
27	Pero Peric	Motoricke i funkcionalne sposobnosti	Skok u dalj	225.0	cm	2024-05-15T10:10
28	Ana Kovačić	Motoricke i funkcionalne sposobnosti	Skok u dalj	188.0	cm	2024-05-15T10:10
29	Matej Nikolić	Motoricke i funkcionalne sposobnosti	Skok u dalj	210.0	cm	2024-05-15T10:10
30	Toni Knez	Motoricke i funkcionalne sposobnosti	Skok u dalj	176.0	cm	2024-05-15T10:10
31	Ema Mandić	Motoricke i funkcionalne sposobnosti	Skok u dalj	180.0	cm	2024-05-15T10:10
32	Josip Pavlović	Motoricke i funkcionalne sposobnosti	Skok u dalj	205.0	cm	2024-05-15T10:10

Rezultati učenika po aktivnosti

Slika 5.3.16 – Rezultati učenika - pregled exelice s rezultatima

5.3.2.3 Unos rezultata

Odabirom opcije *"Unos rezultata"*, otvara se stranica s tablicom za unos novih rezultata po razredu. Nastavnik može odabrati željenu aktivnost i podaktivnost te unijeti ostvareni rezultat i mjerne jedinice.

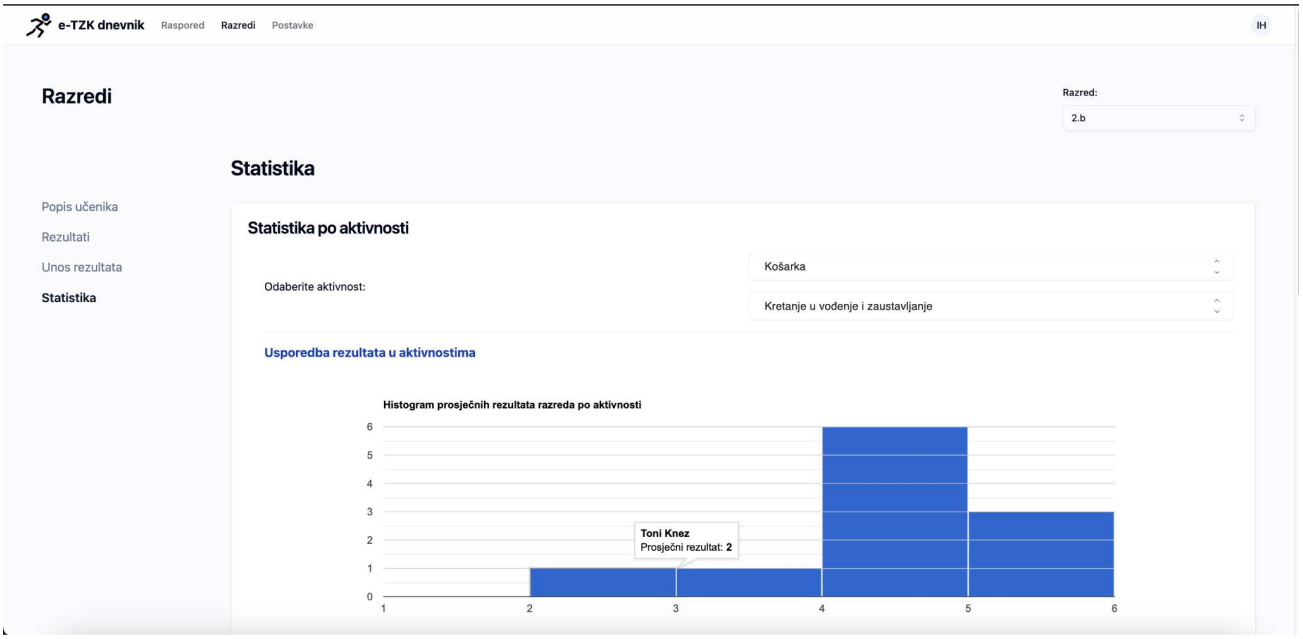


Slika 5.3.17 – Unos rezultata

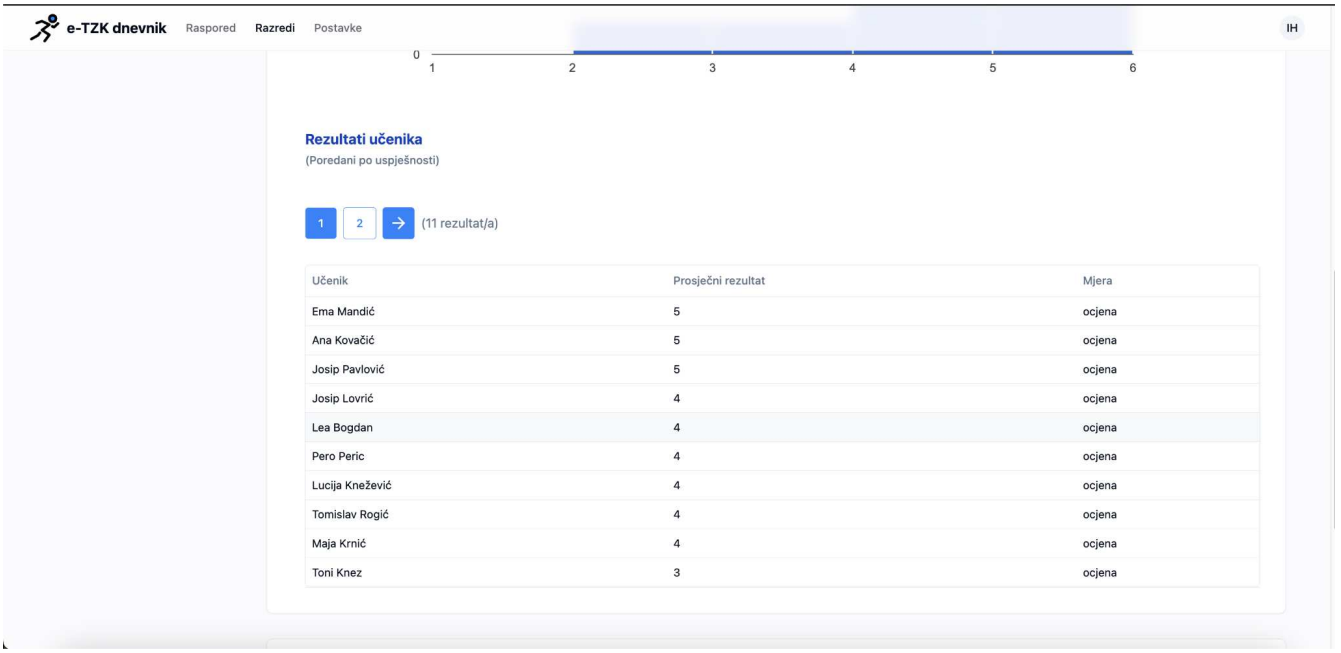
5.3.2.4 Statistika

Odabirom opcije *"Statistika"*, otvara se stranica s uvidom u deskriptivnu statistiku razreda, vizualiziranu kroz grafove i histograme.

Nakon odabira željene aktivnosti/podaktivnosti kroz izbornik, prikazuje se histogram prosječnih rezultata razreda po odabranoj aktivnosti/podaktivnosti (Slika 5.3.18), kao i popis rezultata učenika poredanih po uspješnosti u toj kategoriji (Slika 5.3.19).

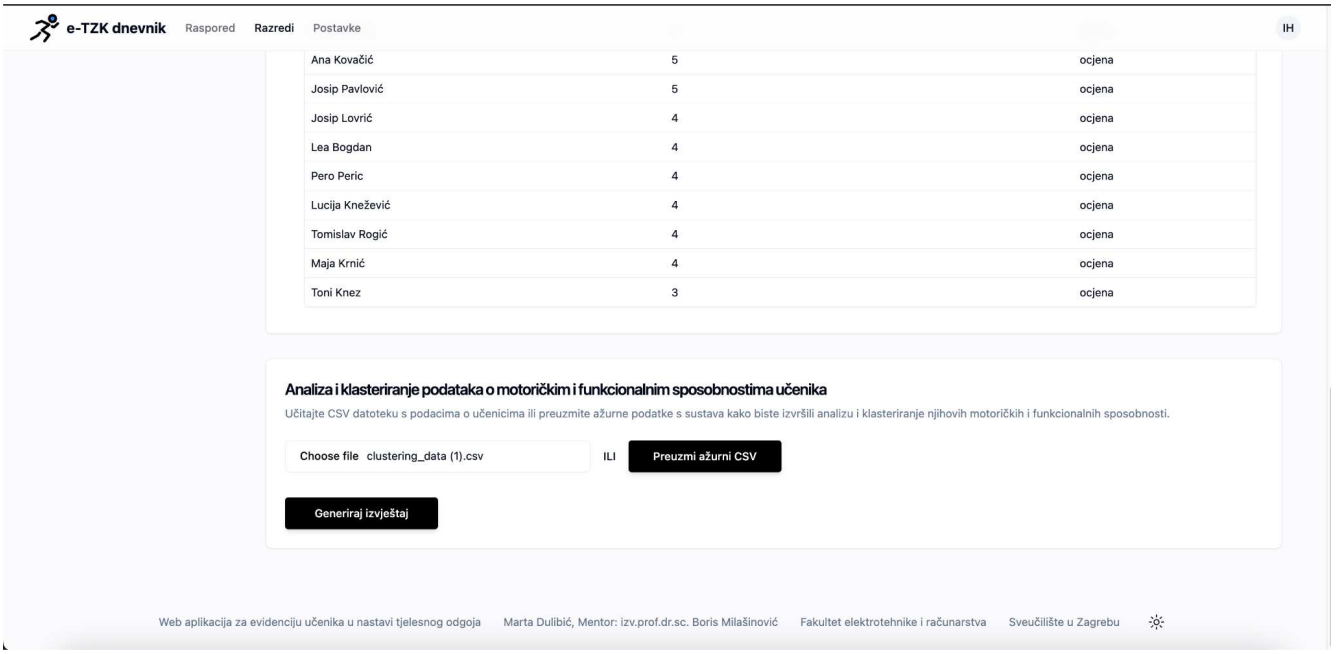


Slika 5.3.18 – Statistika - histogram prosječnih rezultata razreda

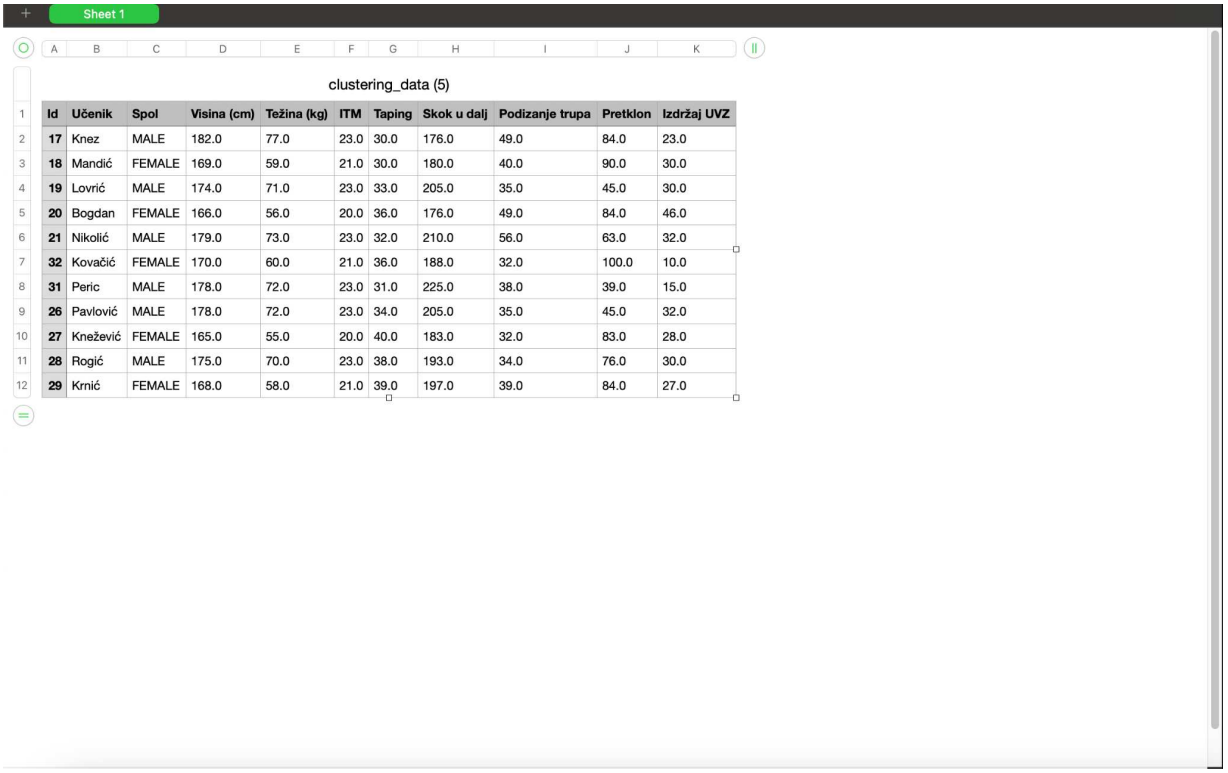


Slika 5.3.19 – Statistika - popis rezultata poredanih po uspješnosti

Uz deskriptivnu statistiku, nastavnik ima mogućnost provođenja klasterske analize nad razredom. Nastavnik može odabrati CSV datoteku s podacima o učenicima ili preuzeti nove podatke iz sustava klikom na gumb *"Preuzmi ažurni CSV"*.



Slika 5.3.20 – Statistika - klasterska analiza



Sheet 1

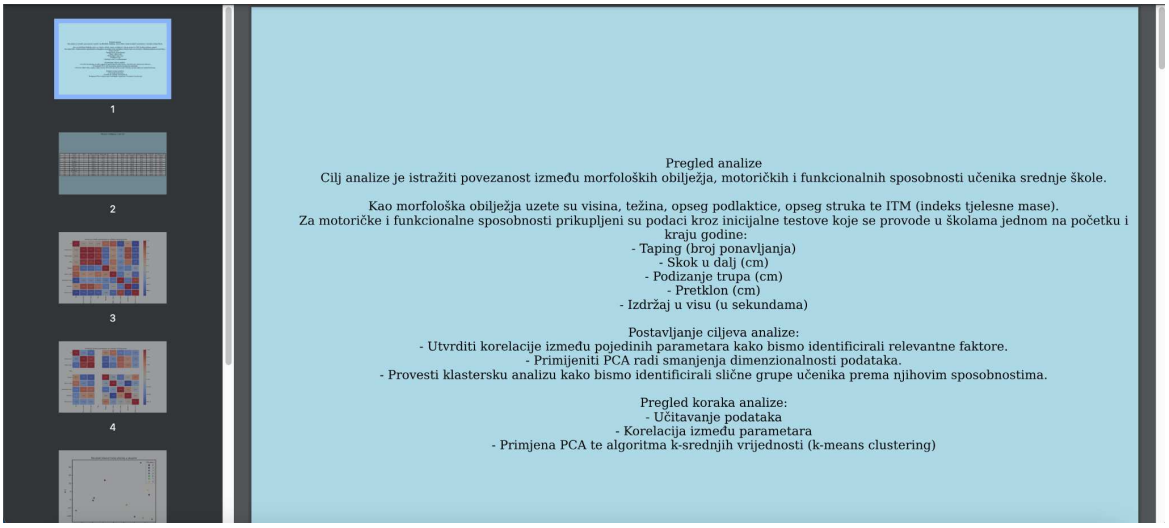
clustering_data (5)

Id	Učenik	Spol	Visina (cm)	Težina (kg)	ITM	Taping	Skok u dalj	Podizanje trupa	Pretklon	Izdržaj UVZ
17	Knez	MALE	182.0	77.0	23.0	30.0	176.0	49.0	84.0	23.0
18	Mandić	FEMALE	169.0	59.0	21.0	30.0	180.0	40.0	90.0	30.0
19	Lovrić	MALE	174.0	71.0	23.0	33.0	205.0	35.0	45.0	30.0
20	Bogdan	FEMALE	166.0	56.0	20.0	36.0	176.0	49.0	84.0	46.0
21	Nikolić	MALE	179.0	73.0	23.0	32.0	210.0	56.0	63.0	32.0
32	Kovačić	FEMALE	170.0	60.0	21.0	36.0	188.0	32.0	100.0	10.0
31	Peric	MALE	178.0	72.0	23.0	31.0	225.0	38.0	39.0	15.0
26	Pavlović	MALE	178.0	72.0	23.0	34.0	205.0	35.0	45.0	32.0
27	Knežević	FEMALE	165.0	55.0	20.0	40.0	183.0	32.0	83.0	28.0
28	Rogić	MALE	175.0	70.0	23.0	38.0	193.0	34.0	76.0	30.0
29	Krnić	FEMALE	168.0	58.0	21.0	39.0	197.0	39.0	84.0	27.0

Slika 5.3.20 – Statistika - preuzeta CSV datoteka s ažurnim podacima

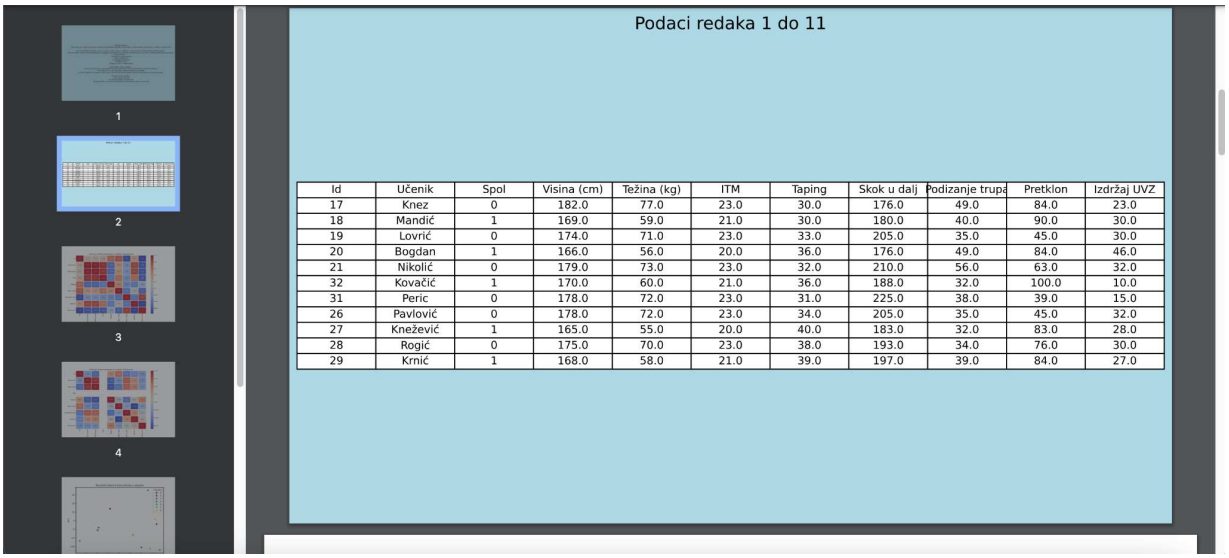
Nakon odabira željene datoteke, klikom na gumb *"Generiraj izvještaj"*, generirana klasterska analiza će se preuzeti u PDF formatu na njegovo lokalno računalo.

Izvještaj se sastoji od uvoda koji daje kratki pregled analize, njezine svrhe i koraka koje obuhvaća.



Slika 5.3.21 – Statistika - klasterska analiza - uvod

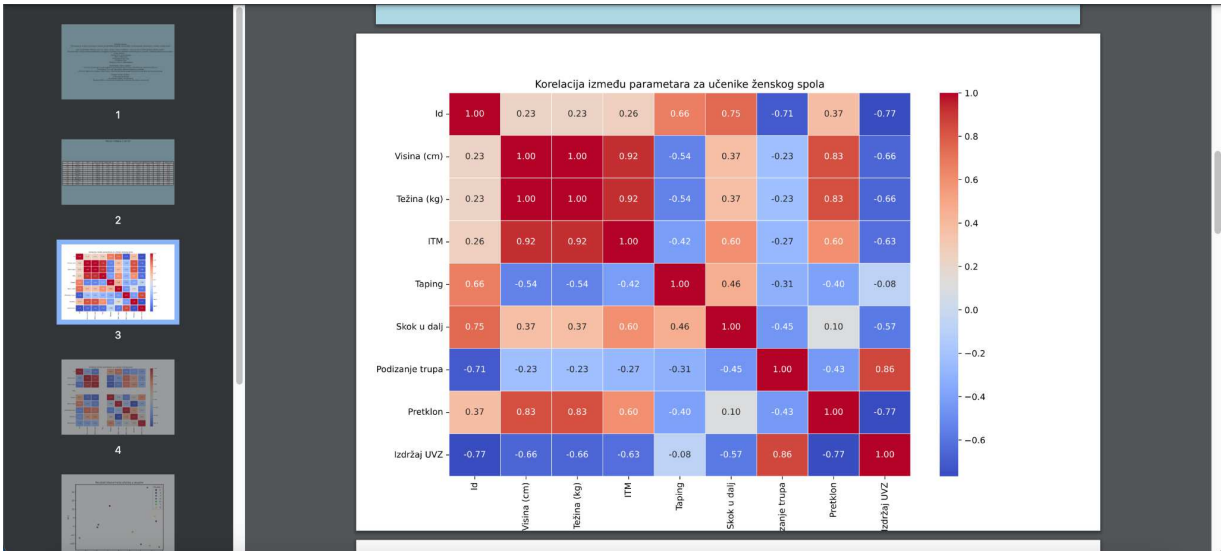
Nakon uvoda, prikazani su podaci korišteni za klastersku analizu u obliku tablice s podacima o učenicima.



Id	Učenik	Spol	Visina (cm)	Težina (kg)	ITM	Taping	Skok u dalj	Podizanje trupa	Pretklon	Izdržaj UVZ
17	Knez	0	182.0	77.0	23.0	30.0	176.0	49.0	84.0	23.0
18	Mandić	1	169.0	59.0	21.0	30.0	180.0	40.0	90.0	30.0
19	Lovrić	0	174.0	71.0	23.0	33.0	205.0	35.0	45.0	30.0
20	Bogdan	1	166.0	56.0	20.0	36.0	176.0	49.0	84.0	46.0
21	Nikolić	0	179.0	73.0	23.0	32.0	210.0	56.0	63.0	32.0
32	Kovačić	1	170.0	60.0	21.0	36.0	188.0	32.0	100.0	10.0
31	Peric	0	178.0	72.0	23.0	31.0	225.0	38.0	39.0	15.0
26	Pavlović	0	178.0	72.0	23.0	34.0	205.0	35.0	45.0	32.0
27	Knežević	1	165.0	55.0	20.0	40.0	183.0	32.0	83.0	28.0
28	Rogić	0	175.0	70.0	23.0	38.0	193.0	34.0	76.0	30.0
29	Krnić	1	168.0	58.0	21.0	39.0	197.0	39.0	84.0	27.0

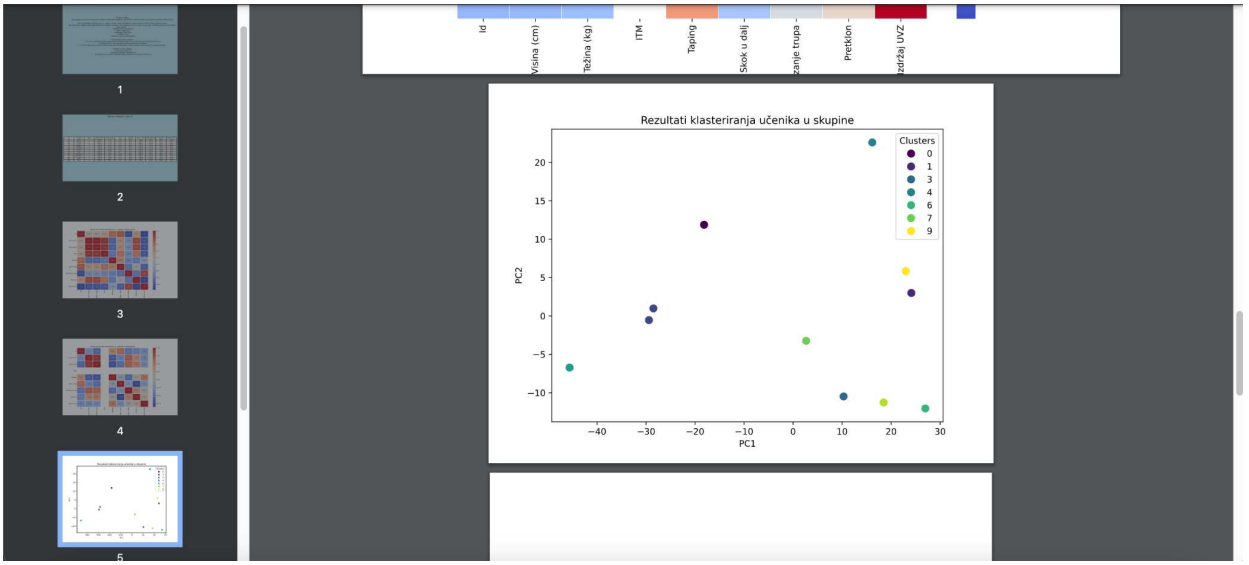
Slika 5.3.22 – Statistika - klasterska analiza - tablica s korištenim podacima

Prvi dio analize obuhvaća prikaz korelacije između parametara za učenike muškog i ženskog spola.



Slika 5.3.23 – Statistika - klasterska analiza - korelacija

Dalje, prikazan je graf koji ilustrira rezultate klasteriranja učenika u različite skupine.



Slika 5.3.24 – Statistika - klasterska analiza - klasteriranje

Kao posljednja stavka, prikazani su vizualni prikazi klastera i grupiranja učenika prema njihovom pripadanju određenim klasterima.

Slika 5.3.24 – Statistika - klasterska analiza - vizualizacija pripadnosti učenika klasteru

5.3.3. Pregled detalja o učeniku

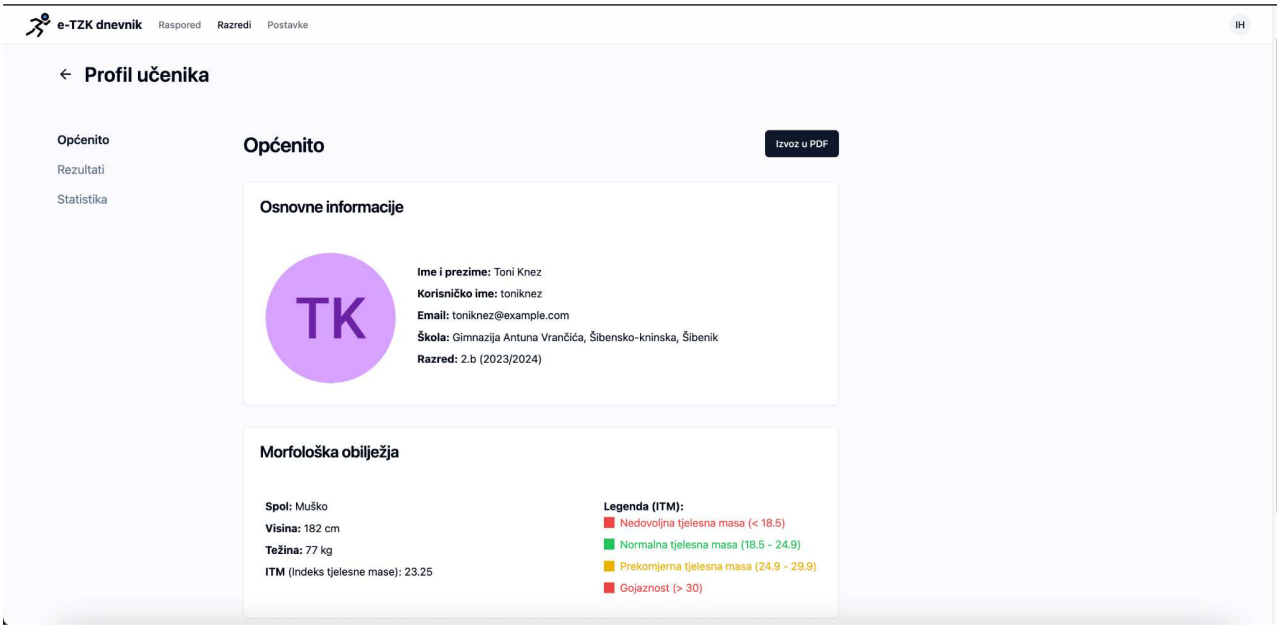
Kao što je spomenuto u poglavlju 5.3.2.1, na stranici sa popisom učenika, klikom na *"tri točkice"* pored svakog učenika, otvara se opcija za pregled detalja o učeniku (vidi Slika 5.3.11).

Klikom na opciju *"Prikaži detalje o učeniku"* otvara se nova stranica s profilom učenika.

Za daljnju navigaciju na stranici postoji bočna navigacijska traka s opcijama *"Općenito"* (što je inicijalno otvoren sadržaj), *"Rezultati"* i *"Statistika"*.

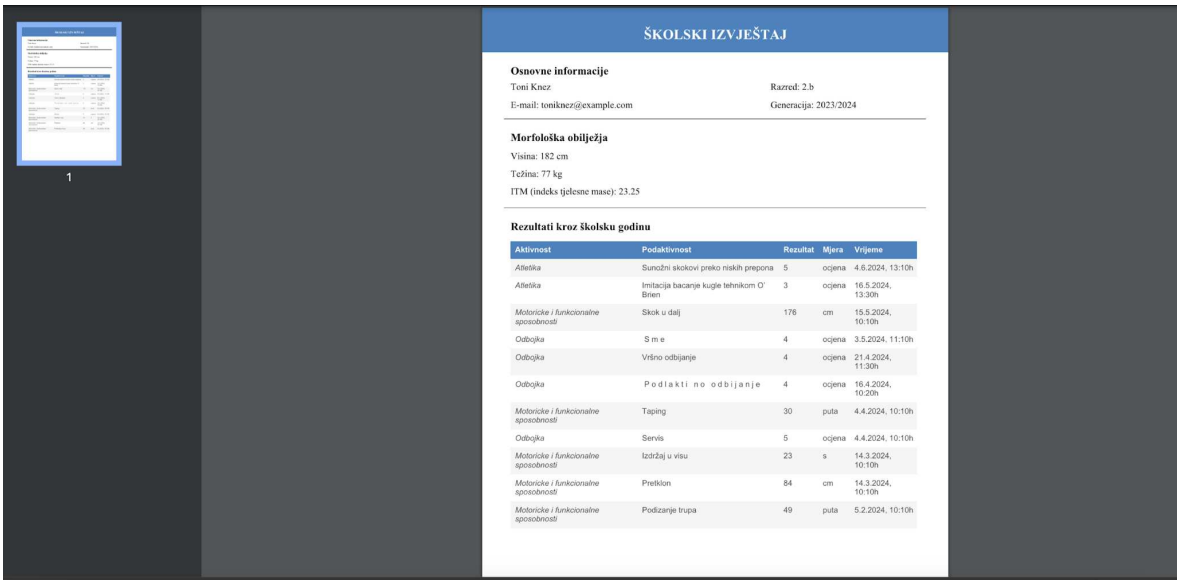
5.3.3.1 Općenito

Odabirom opcije *"Općenito"*, otvara se stranica s osnovnim informacijama o učeniku, uključujući ime, prezime, školu, razred i pripadnu generaciju. Također su prikazana morfološka obilježja učenika kao što su spol, težina, visina i indeks tjelesne mase (ITM). Za ITM je prikazana legenda koja definira raspon za normalno i problematično stanje tjelesne mase učenika.



Slika 5.3.25 – Profil učenika - općenito

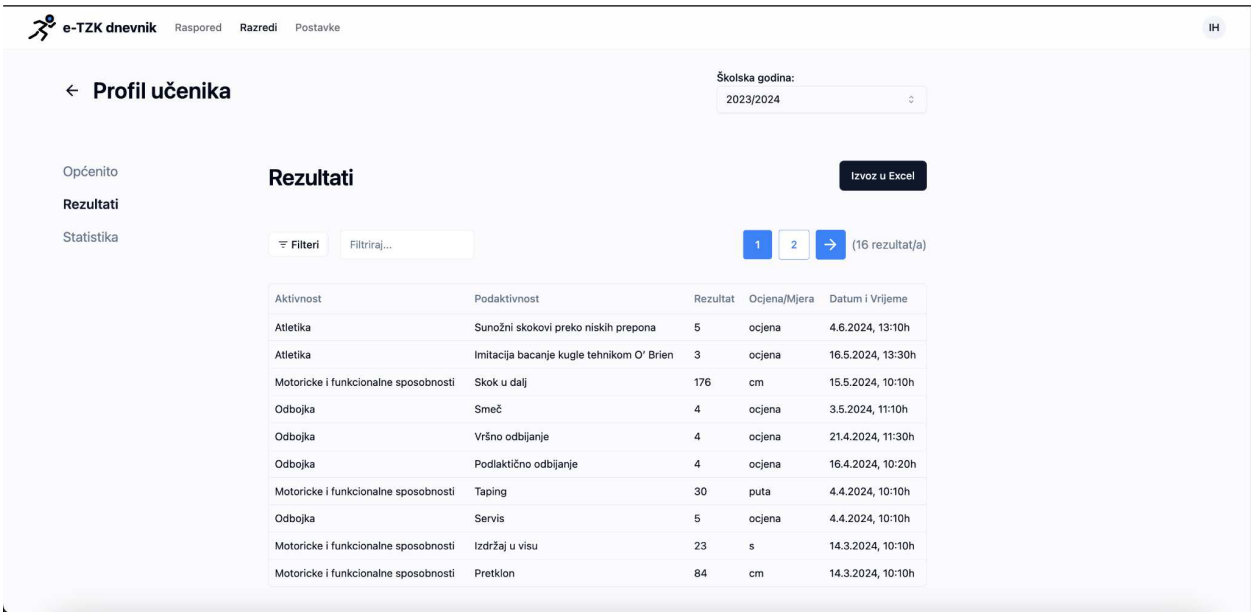
Klikom na gumb *"Izvoz u PDF"* u gornjem desnom kutu, na lokalno računalo korisnika će se preuzeti PDF izvješće s detaljima o učeniku.



Slika 5.3.26 – PDF izvještaj s detaljima učenika

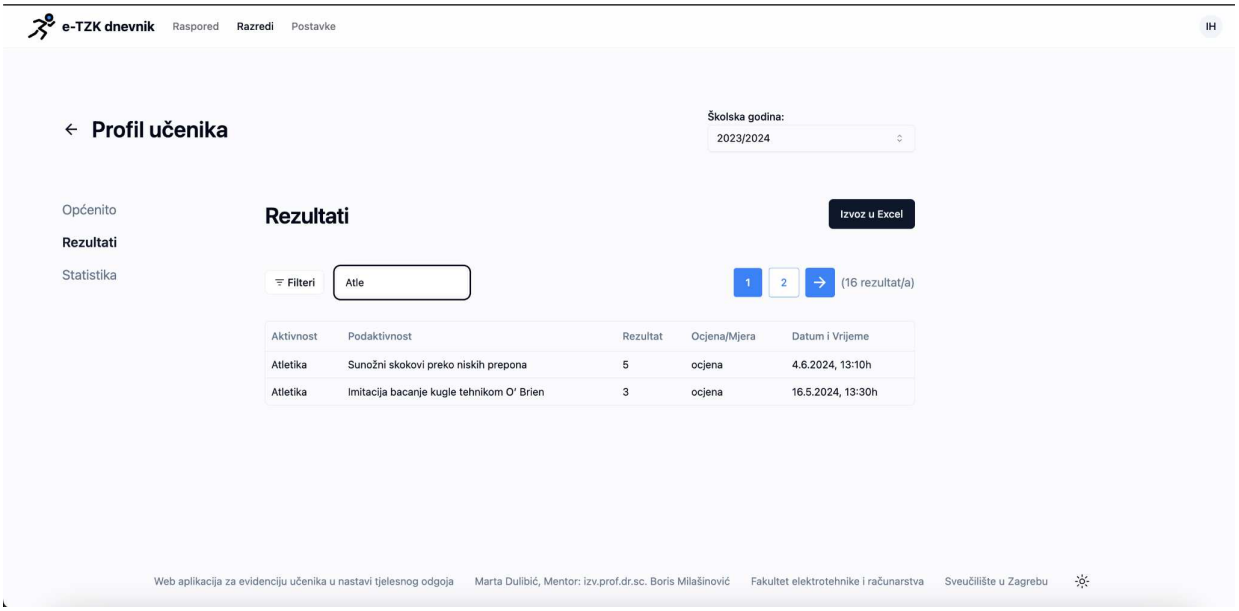
5.3.3.2 Rezultati

Odabirom opcije *"Rezultati"*, otvara se stranica s rezultatima učenika s paginacijom, prikazujući po 10 rezultata po stranici. U gornjem desnom kutu ekrana nalazi se izbornik za odabir školske godine za koju se žele pregledati rezultati.



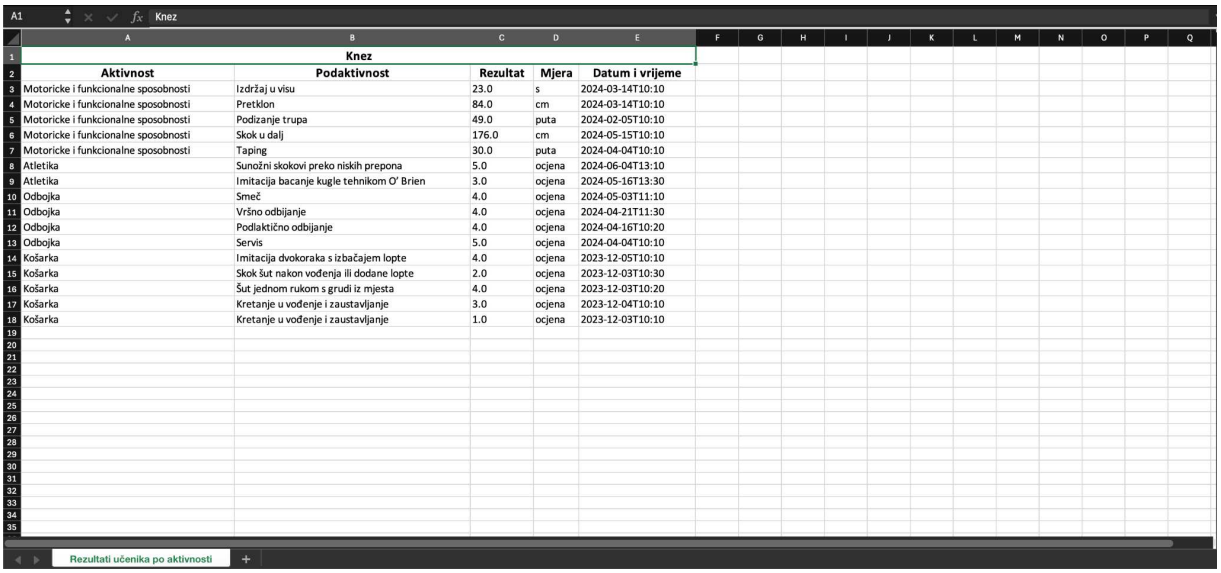
Slika 5.3.26 – Rezultati učenika po školskoj godini

Na stranici postoji mogućnost filtriranja rezultatima prema predodređenim filterima kao što su aktivnost i podaktivnost.



Slika 5.3.27 – Rezultati učenika - filtriranje

Klikom na gumb *"Izvoz u Excel"* u gornjem desnom kutu, rezultati učenika se izvoze u Excel format i preuzimaju na lokalno računalo nastavnika.



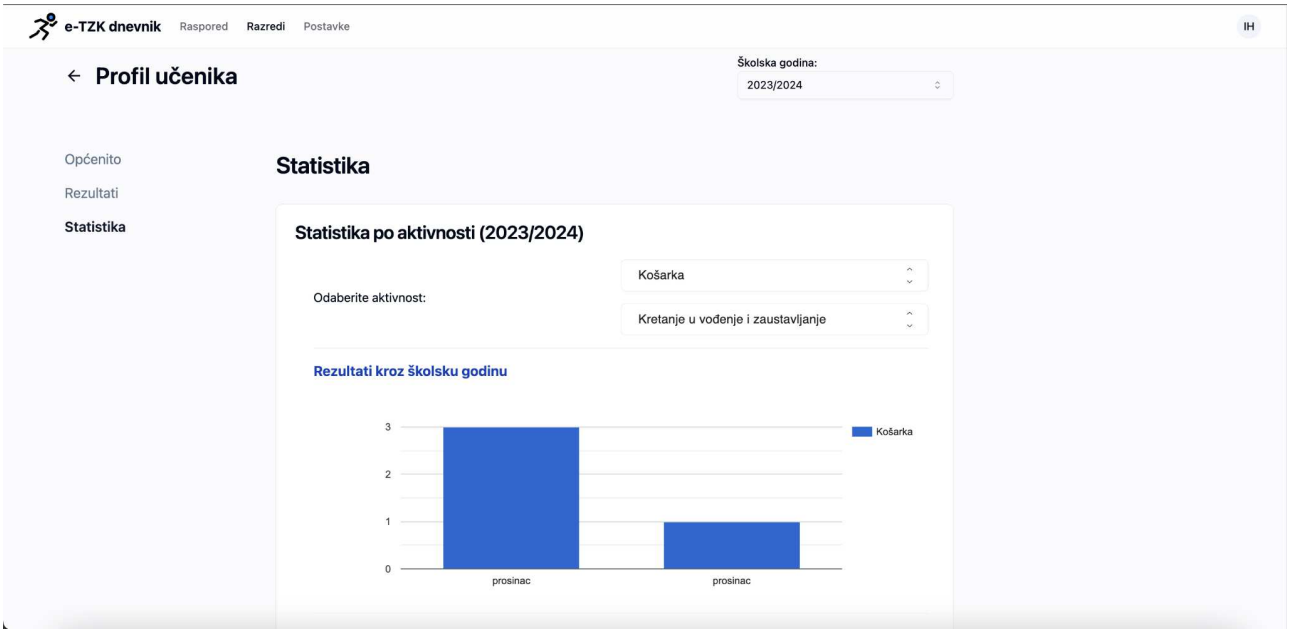
	A	B	C	D	E
1	Knez				
2	Aktivnost	Podaktivnost	Rezultat	Mjera	Datum i vrijeme
3	Motoricke i funkcionalne sposobnosti	Izdržaj u visu	23.0	s	2024-03-14T10:10
4	Motoricke i funkcionalne sposobnosti	Pretklon	84.0	cm	2024-03-14T10:10
5	Motoricke i funkcionalne sposobnosti	Podizanje trupa	49.0	puta	2024-02-05T10:10
6	Motoricke i funkcionalne sposobnosti	Skok u dalj	176.0	cm	2024-05-15T10:10
7	Motoricke i funkcionalne sposobnosti	Taping	30.0	puta	2024-04-04T10:10
8	Atletika	Sunožni skokovi preko niskih prepona	5.0	ocjena	2024-06-04T13:10
9	Atletika	Imitacija bacanje kugle tehnikom O' Brien	3.0	ocjena	2024-05-16T13:30
10	Odbojka	Smeč	4.0	ocjena	2024-05-03T11:10
11	Odbojka	Vršno odbijanje	4.0	ocjena	2024-04-21T11:30
12	Odbojka	Podlaktično odbijanje	4.0	ocjena	2024-04-16T10:20
13	Odbojka	Servis	5.0	ocjena	2024-04-04T10:10
14	Košarka	Imitacija dvokoraka s izbačajem lopte	4.0	ocjena	2023-12-05T10:10
15	Košarka	Skok šut nakon vođenja ili dodane lopte	2.0	ocjena	2023-12-03T10:30
16	Košarka	Šut jednom rukom s grudi iz mjesta	4.0	ocjena	2023-12-03T10:20
17	Košarka	Kretanje u vođenje i zaustavljanje	3.0	ocjena	2023-12-04T10:10
18	Košarka	Kretanje u vođenje i zaustavljanje	1.0	ocjena	2023-12-03T10:10

Slika 5.3.28 – Rezultati učenika - pregled exelice s rezultatima

5.3.3.3 Statistika

Odabirom opcije *"Statistika"*, otvara se stranica s uvidom u deskriptivnu statistiku učenika po školskoj godini, vizualiziranu kroz grafove i histograme.

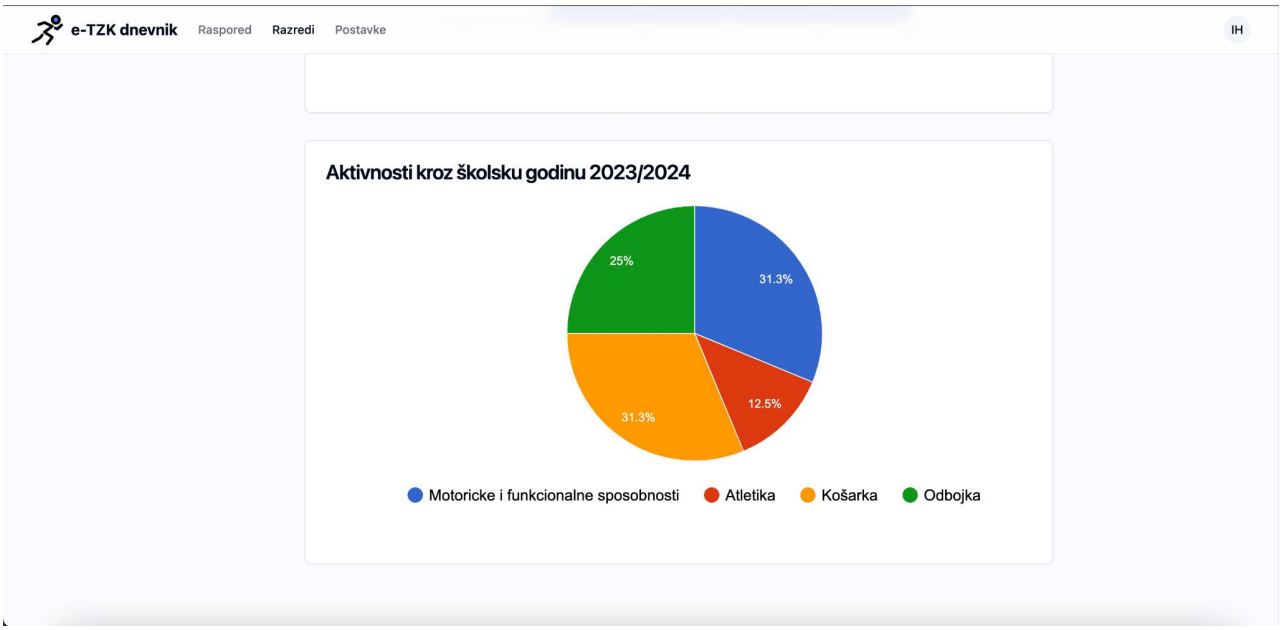
Nakon odabira željene aktivnosti/podaktivnosti kroz izbornik, prikazuje se graf promjene rezultata za odabranu kategoriju kroz školsku godinu (Slika 5.3.29), kao i histogram prosječnih rezultata razreda (Slika 5.3.30).



Slika 5.3.29 – Statistika - graf rezultata kroz školsku godinu

Slika 5.3.30 – Statistika - histogram prosječnih rezultata razreda

Također, na stranici je prikazan i kružni graf koji prikazuje postotke udjela različitih aktivnosti kroz školsku godinu.



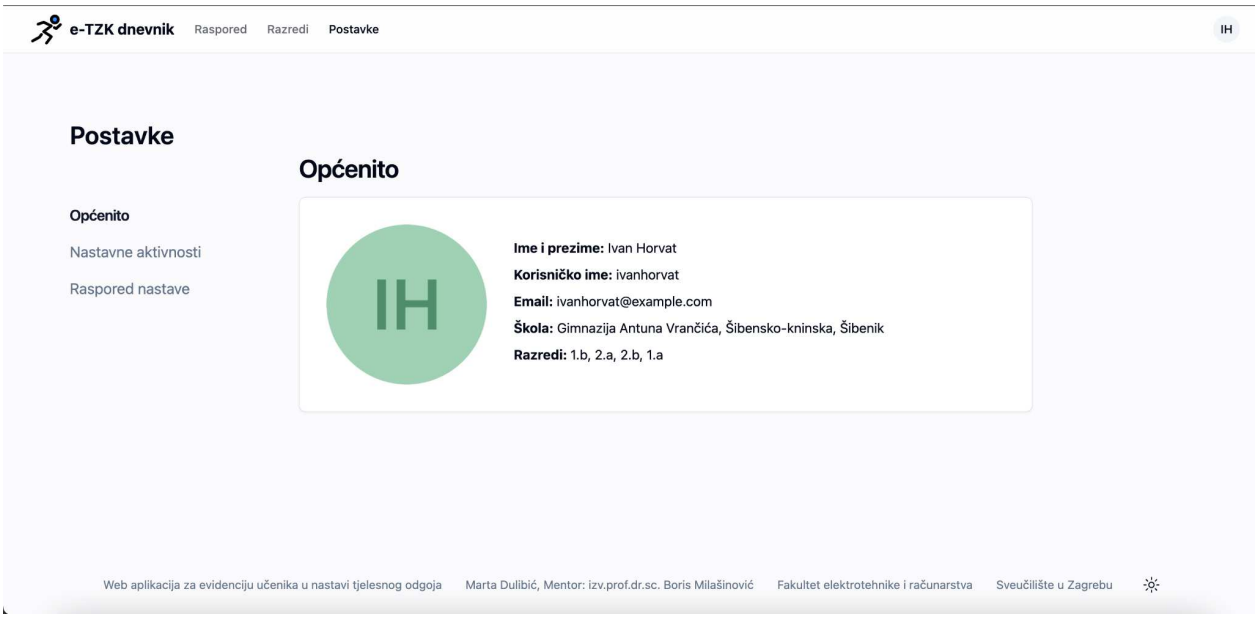
Slika 5.3.31 – Statistika - kružni graf sa udjelima aktivnosti

5.3.4. Pregled i upravljanje postavkama

Odabirom opcije *"Postavke"* na navigacijskoj traci, inicijalno se otvara stranica s općenitim informacijama o nastavniku. Za daljnju navigaciju na stranici postoji bočna navigacijska traka s opcijama *"Općenito"* (što je inicijalno otvoren sadržaj), *"Nastavne aktivnosti"i "Raspored nastave"*.

5.3.4.1 Općenito

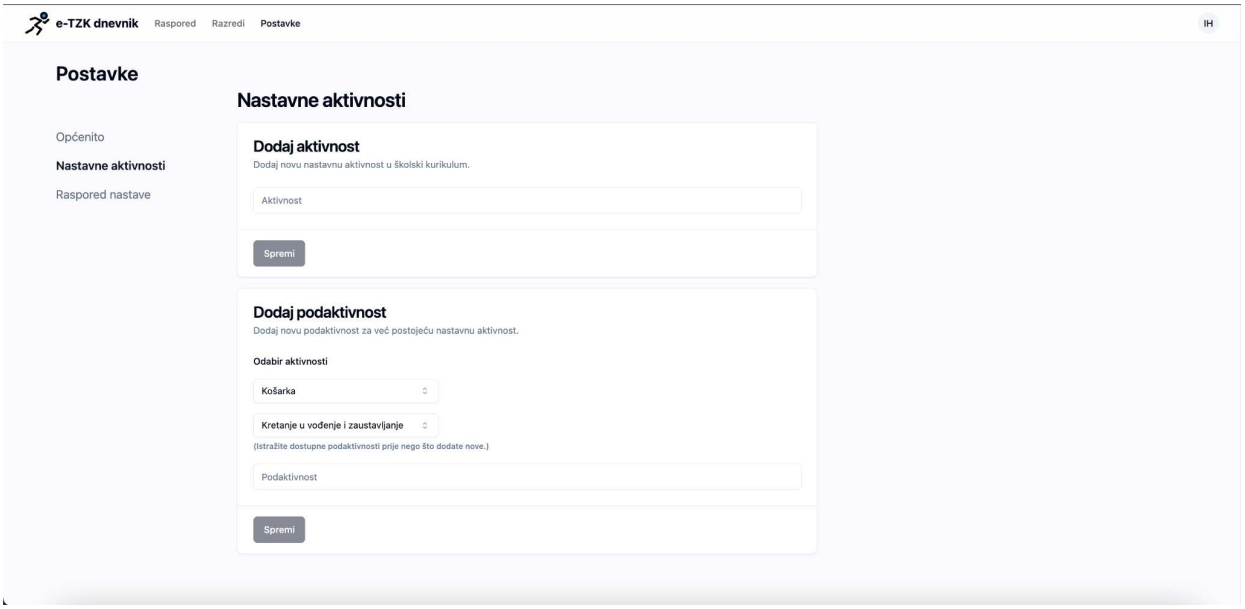
Odabirom opcije *"Općenito"*, otvara se stranica s osnovnim informacijama o nastavniku, uključujući ime, prezime, školu te razrede kojima predaje.



Slika 5.3.32 – Općenito - informacije o nastavniku

5.3.4.2 Nastavne aktivnosti

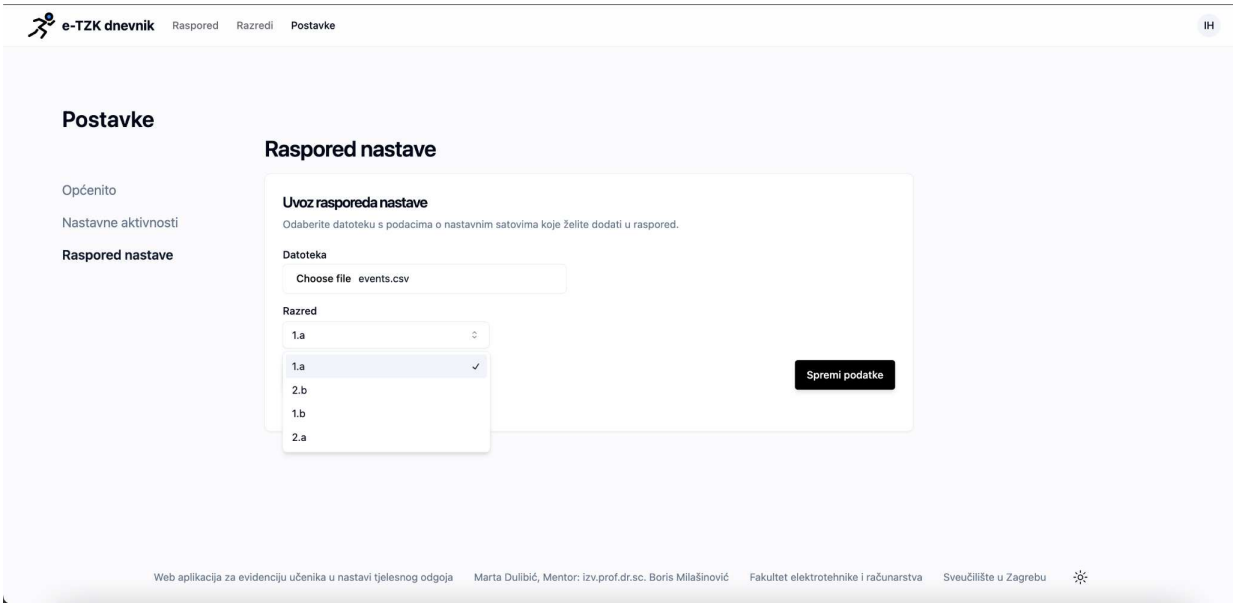
Odabirom opcije *"Nastavne aktivnosti"* na navigacijskoj traci, otvara se stranica s formama za dodavanje nove aktivnosti ili nove podaktivnosti za postojeće aktivnosti.



Slika 5.3.33 – Nastavne aktivnosti - dodavanje

5.3.4.3 Raspored nastave

Odabirom opcije *"Raspored nastave"*, otvara se forma za uvoz rasporeda nastave u sustav putem CSV datoteke. Nakon odabira datoteke s željenim podacima, odabire se razred za koji se žele dodati nastavni sati.

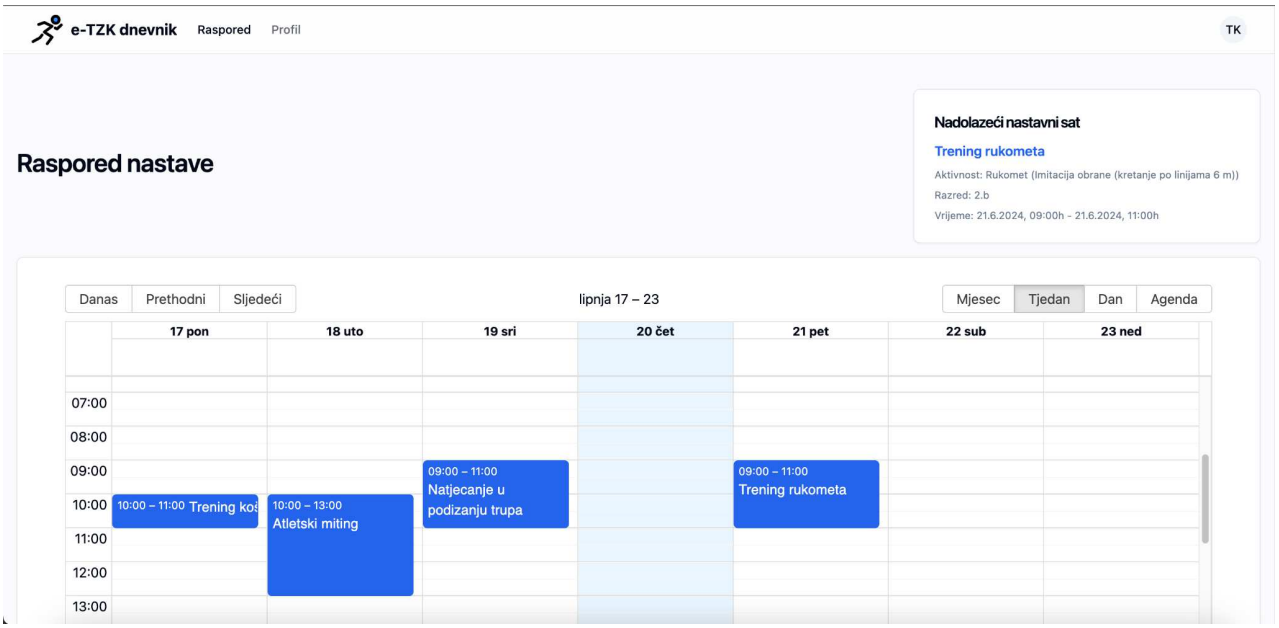


Slika 5.3.34 – Raspored nastave - odabir razreda

Nakon odabira datoteke s željenim podacima, klik na gumb *"Uvoz podataka"*, na stranici će prikazati tablicu s nastavnim rasporedom čiji se podaci mogu pregledati prije samog dodavanja u sustav klikom na gumb *"Spremi podatke"* (vidi kao primjer Slika 5.2.13) .

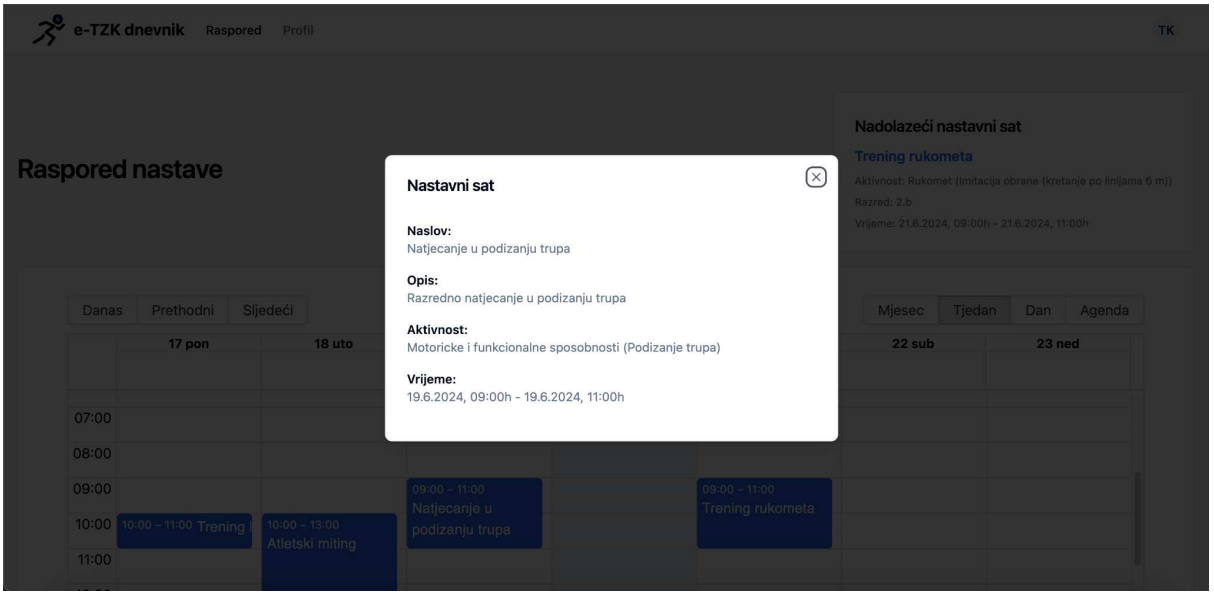
5.4. Sučelje za učenike

Nakon uspješne prijave i autorizacije kao učenik, otvara se stranica sa rasporedom nastave u obliku kalendara. Kroz kalendar se mogu pregledati svi nadolazeće nastavni sati u danu, tjednu ili mjesecu. U gornjem desnom kutu iznad kalendara nalazi se podsjetnik na nadolazeći nastavni sat.



Slika 5.3.34 – Raspored - pregled kalendara nastave

Klikom na nastavni sat u kalendaru, otvara se dijalog s detaljima koji uključuje naziv i opis nastavnog sata, odabranu aktivnost (podaktivnost) i vrijeme održavanja.



Slika 5.3.34 – Raspored - pregled detalja o nastavnom satu

Odabirom opcije *"Profil"* na navigacijskoj traci, otvara se stranica s profilom učenika. Upute o profilu učenika su već navedene u potpoglavlju *5.3.3 Pregled detalja o učeniku*.

6. Upute za instalaciju i lokalno pokretanje

Klijent

Izvorni kod klijenta može se pronaći na sljedećoj poveznici: https://github.com/mdulibic/tzk-dnevnik-front .

Ako Node.js nije instaliran, potrebno je prvo instalirati Node.js (https://nodejs.org/). Nakon uspješne instalacije, potrebno je smjestiti se u root direktorij projekta i instalirati sve vanjske ovisnosti pokretanjem komande *npm install*. Aplikacija se zatim pokreće sa *npm run dev* i vrti se na portu 5173.

Baza podataka

Za korištenje PostgreSQL baze potrebno je instalirati i konfigurirati *PostgreSQL* server. Detalje o instalaciji mogu se pronaći na službenoj stranici PostgreSQL-a. Server baze podataka se vrti na portu 5432.

Nakon uspješne instalacije, potrebno je kreirati potrebne baze podataka izvedbom sljedećih naredbi u terminalu:

```
createdb userService
createdb schoolService
createdb activityService
createdb eventsService
createdb resultsService
```

Kôd 6.1 – Naredbe za kreiranje baza podataka

Za inicijalizaciju svake baze podataka potrebno je pokrenuti odgovarajuće SQL skripte koje se nalaze u direktoriju `/sql`. Ove skripte će kreirati tablice i unijeti inicijalne podatke:

```
psql -d userService -f users.sql
psql -d schoolService -f school.sql
psql -d activityService -f activity.sql
psql -d eventsService -f event.sql
psql -d resultsService -f result.sql
```

Kôd 6.2 – Naredbe za kreiranje tablica i njihovu prepopulaciju unutar pojedine baze podataka

activity.sql

Users > mdulibic > Documents > diplomski > tzk-dnevnik-server > sql > activity.sql

```sql
CREATE TABLE activity (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL
);

CREATE TABLE subactivity (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    parent_activity_id INT NOT NULL,
    FOREIGN KEY (parent_activity_id) REFERENCES activity(id)
);

-- Insert Activities
INSERT INTO activity (name) VALUES
                                ('Košarka'),
                                ('Odbojka'),
                                ('Atletika'),
                                ('Badminton'),
                                ('Rukomet'),
                                ('Nogomet'),
                                ('Motoricke i funkcionalne sposobnosti');

-- Insert SubActivities for Košarka
INSERT INTO subactivity (name, parent_activity_id) VALUES
                                                    ('Kretanje u vođenje i zaustavljanje', 1),
                                                    ('Šut jednom rukom s grudi iz mjesta', 1),
                                                    ('Skok šut nakon vođenja ili dodane lopte', 1),
                                                    ('Imitacija dvokoraka s izbačajem lopte', 1);

-- Insert SubActivities for Odbojka
INSERT INTO subactivity (name, parent_activity_id) VALUES
                                                    ('Servis', 2),
                                                    ('Podlaktično odbijanje', 2),
                                                    ('Vršno odbijanje', 2),
                                                    ('Smeč', 2);

-- Insert SubActivities for Atletika
INSERT INTO subactivity (name, parent_activity_id) VALUES
                                                    ('Imitacija bacanje kugle tehnikom O' Brien', 3),
                                                    ('Sunožni skokovi preko niskih prepona', 3),
                                                    ('Tehnika trčanja preko markacija prostora', 3);
```

Slika 6.1. - Primjer skripte za kreiranje i prepopuliranje tablica *activity* i *subactivity*

Poslužitelj

Izvorni kod poslužitelja može se pronaći na sljedećoj poveznici: GitHub - tzk-dnevnik-back.

Nakon uspješnog kloniranja repozitorija, potrebno je pokrenuti sljedeće komponente redoslijedom: registracijski poslužitelj (Eureka server), API gateway, te nakon toga svaki od Spring Boot servisa pronađenih u direktoriju */microservices*.

```
cd putanja/do/tzk-dnevnik-server/eureka-server
./gradlew bootRun

cd putanja/do/tzk-dnevnik-server/api-gateway
./gradlew bootRun

// Primjer pokretanja Spring Boot mikroservisa
cd putanja/do/tzk-dnevnik-server/microservices/spring-boot/activity-service
./gradlew bootRun
```

Kôd 6.3 – Naredbe za pokretanje Spring Boot projekata

Flask mikroservis se pokreće na drugačiji način unutar virtualnog okruženja.

```
// Pozicioniranje unutar virtualnog okruženja
cd data-clustering-service-env

// Pokretanje virtualnog okruženja
data-clustering-service-env/Scripts/activate.bat

// Pokretanje mikroservisa
python3 /path/to/flask_app.py
```

Kôd 6.4 – Naredbe za pokretanje Flask mikroservisa

Poslužitelj se vrti na portu 8080 te je ulazna putanja u aplikaciju */login*.

Zaključak

U okviru ovog diplomskog rada razvijena je web aplikacija e-TZK Dnevnik, usmjerena na olakšavanje vođenja evidencije o učenicima i organizaciju nastave u području tjelesnog odgoja u školama.

Na početku rada provedena je analiza u kojoj smo detaljno istražili povezanost između morfoloških obilježja, motoričkih i funkcionalnih sposobnosti učenika, koristeći stvarne podatke učenika srednje škole. Kroz provedenu analizu, koja je uključivala metode poput korelacijske analize, Principal Component Analysis (PCA) i K-means klasteriranja, dobili smo uvid u korelaciju među parametrima te grupiranje razreda u klastere. Također smo istražili anomalije i njihove uzroke kako bismo bolje razumjeli njihovu pojavu. Ovi rezultati poslužili su kao temelj za razvoj funkcionalnosti aplikacije e-TZK Dnevnik.

Aplikacija pruža korisnicima, kako nastavnicima tako i učenicima, niz funkcionalnosti koje uključuju evidenciju podataka o učenicima, organizaciju rasporeda nastave te analizu postignutih rezultata u obliku deskriptivnih statistika i provođenja klasterske analize. Korisnicima je za pristup aplikaciji potrebna autentifikacija putem korisničkog imena i lozinke ili putem Google računa. Administratori imaju ovlasti registracije korisnika (nastavnika i učenika) te pregleda i upravljanja korisnicima. Nastavnici mogu pregledavati sve razrede i učenike škole, dodavati nove aktivnosti i podaktivnosti te upravljati nastavnim satima. Aplikacija omogućuje detaljnu evidenciju o učenicima, uključujući osnovne informacije, morfološka obilježja (težina, visina, ITM), rezultate po školskim godinama te statističke prikaze u obliku grafova i histograma. Također, omogućena je klasterska analiza radi boljeg razumijevanja karakteristika razreda.

Ova aplikacija je razvijena kao prototip s ciljem unapređenja kvalitete nastave tjelesnog odgoja te naglašavanja važnosti tog predmeta koji je često podcijenjen i zanemaren od strane učenika, ali i školstva općenito. Od velike je važnosti otpočetka pratiti napredak u razvoju djece te promicati među njima važnost tjelesnog kretanja i zdravog načina života.

Literatura

[1] Wikipedia, *Korelacija*, Poveznica: https://hr.wikipedia.org/wiki/Korelacija#Pearsonov_koeficijent_korelacije; pristupljeno 20.travnja 2024.

[2] Zakaria Jaadi, Built in, *A step-by-step explanation of Principal Component Analysis (PCA)*, Poveznica: https://builtin.com/data-science/step-step-explanation-principal-component-analysis; pristupljeno 20.travnja 2024.

[3] Pulkit Sharma, Analytics Vidhya, *The Ultimate Guide to K-Means Clustering: Definition, Methods, and Applications*, Poveznica: https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/; pristupljeno 20.travnja 2024.

[4] Geeks for geeks, *Elbow method for the optimal value of k in KMeans*, https://www.geeksforgeeks.org/elbow-method-for-optimal-value-of-k-in-kmeans/ ; pristupljeno 20.travnja 2024.

[5] Wikipedia, *React*, Poveznica: https://en.wikipedia.org/wiki/React_(JavaScript_library) ; pristupljeno 15.lipnja 2024.

[6] Nivetha Krishnanm, Medium, *What is Vite and why did it replace the Create React app?;* Poveznica: https://nivethakrishnan72.medium.com/what-is-vite-and-why-has-it-replaced-create-react-app-ac792c7388d1 ; pristupljeno 15.lipnja 2024.

[7] Tailwindcss documentation; Poveznica: https://tailwindcss.com/docs/installation; pristupljeno 15.lipnja 2024.

[8] Span, FER predavanje, *Agilni razvoj digitalnih platformi pomoću mikroservisa*; Poveznica: https://www.fer.unizg.hr/_download/repository/03.1-Uvod_u_mikroservise.pdf; pristupljeno 16.lipnja 2024.

[9] Medium, *Get started with microservices using Spring Boot*; Poveznica: https://medium.com/ms-club-of-sliit/lets-build-a-microservice-with-spring-boot-faf39b968857 ; pristupljeno 16.lipnja 2024.

[10] Baeldung, Introduction to Spring Cloud Netflix - Eureka; Poveznica: https://www.baeldung.com/spring-cloud-netflix-eureka ; pristupljeno 16.lipnja 2024.

[11] Spring.io, *Spring Boot*; Poveznica: https://docs.spring.io/spring-boot/index.html; pristupljeno 16.lipnja 2024.

[12] Medium, Introduction to Spring MVC; Poveznica: https://laukikrupne.medium.com/introduction-to-spring-mvc-a2fbb65b3a9d ; pristupljeno 16.lipnja 2024.

[13] Wikipedia, *Flask*, Poveznica: https://en.wikipedia.org/wiki/Flask_(web_framework) ; pristupljeno 16.lipnja 2024.

[14] Chris Richardson, Pattern: *Database per service*; Poveznica: https://microservices.io/patterns/data/database-per-service.html ; pristupljeno 16.lipnja 2024.

[15] Javatpoint, *Spring Boot Architecture*; Poveznica: https://www.javatpoint.com/spring-boot-architecture ; pristupljeno 16.lipnja 2024.

[16] Dinesh on Java, *Microservice Registry Discovery using Netflix Eureka*; Poveznica: https://www.dineshonjava.com/microservice-registry-discovery-using-netflix-eureka/ ; pristupljeno 16.lipnja 2024.

[17] Medium, *Exploring Microservices: Understanding Service Registry and Discovery with Spring Cloud Netflix Eureka*; Poveznica: https://medium.com/@pathirage/exploring-microservices-understanding-service-registry-and-discovery-with-spring-cloud-netflix-950b45fe7608; pristupljeno 16.lipnja 2024.

[18] Learncsdesign, *Microservices Data Design Patterns*; Poveznica: https://www.learncsdesign.com/microservices-data-design-patterns/ ; pristupljeno 16.lipnja 2024.

Sažetak

Ovaj rad opisuje razvoj web aplikacije e-TZK Dnevnik, namijenjene vođenju evidencije o učenicima i organizaciji nastave u području tjelesnog odgoja u školama. Provedena je detaljna analiza motoričkih i funkcionalnih sposobnosti 120 učenika prvog razreda srednje škole, uključujući njihovo klasteriranje radi boljeg razumijevanja korelacija među učenicima.

Klijentska aplikacija je implementirana u Reactu, dok je poslužiteljska strana izgrađena kao mikroservisna arhitektura koristeći razvoje okvire Spring Boot i Flask. Cilj aplikacije je unaprijediti kvalitetu nastave tjelesnog odgoja, promicati zdrav način života te pružiti pravovremene intervencije za poboljšanje učeničkog razvoja. Funkcionalnosti uključuju vođenje evidencije o učenicima, organizaciju nastave te analizu rezultata u obliku deskriptivnih statistika i klasterske analize. Administratori imaju ovlasti registracije korisnika i upravljanja njihovim podacima, dok nastavnici mogu upravljati rasporedom nastave, dodati nove učeničke rezultate i školske aktivnosti te pratiti napredak učenika putem detaljne evidencije, uključujući morfološka obilježja, školske rezultate i statistiku postignutih rezultata. Učenici imaju pristup rasporedu nastave te detaljima svog profila, koji obuhvaćaju školske rezultate i statistiku postignutih rezultata.

Ključne riječi: web aplikacija, tjelesni odgoj, analiza podataka, klasteriranje, mikroservisna arhitektura

Summary

This paper describes the development of the web application e-TZK Dnevnik, intended for keeping records on students and organizing classes in the field of physical education in schools. A detailed analysis of the motor and functional abilities of 120 first grade high school students was conducted, including data clustering for a better understanding of correlations among students.

The client application is implemented in React, while the server side is built as a microservice architecture using the Spring Boot and Flask development frameworks. The goal of the application is to improve the quality of physical education classes, promote a healthy lifestyle, and provide timely interventions to improve student development. Functionalities include keeping records of students, organization of classes and analysis of results in the form of descriptive statistics and cluster analysis. Administrators have the authority to register users and manage their data, while teachers can manage class schedules, add new student results and school activities, and track student progress through detailed records, including morphological features, school results, and achievement statistics.Students have access to the class schedule and the details of their profile, which includes school results and statistics of achieved results.

Keywords: web application, physical education, data analysis, clustering, microservice architecture