**School mikroservis**

Katarina Stevanović

Fakultet tehničkih nauka

Univerzitet u Novom Sadu

[stevanovic.sr9.2022@uns.ac.rs](mailto:stevanovic.sr9.2022@uns.ac.rs)

Sažetak: U ovom radu je opisan mikroservis za evidenciju podataka o učenicima osnovnih i srednjih škola. Servis omogućava rad sa ocenama i izostancima učenika, kao i komunikaciju sa servisom za autentifikaciju korisnika i servisom za zdravstvo. Za realizaciju rešenja korišćen je programski jezik Go i biblioteka GORM za rad sa Postgres bazom podataka. Arhitektura mikroservisa obezbeđuje skalabilnost, lako održavanje i mogućnost integracije u veće sisteme namenjene obrazovanju. Demonstriran je scenario korišćenja i prikazane su prednosti primene mikroservisnog pristupa u školskim informacionim sistemima.

Ključne reči: mikroservisi, obrazovanje, ocene, izostanci, Go, Postgres

# Uvod

Obrazovni sistemi širom sveta se poslednjih decenija sve više oslanjaju na informaciono-komunikacione tehnologije. U prošlosti je evidencija o učenicima vođena isključivo u papirnim dnevnicima i matičnim knjigama. Takav način rada bio je spor, podložan greškama i teško prilagodljiv zahtevima savremenog obrazovanja. Pojavom digitalnih rešenja omogućeno je lakše praćenje ocena, izostanaka i uspeha učenika.

Problem koji se javlja jeste kompleksnost podataka i veliki broj aktera između osnovnih aktera školskog sistema-učenika i profesora. U monolitnim rešenjima svaka izmena u logici ocenjivanja ili evidenciji izostanaka zahteva intervenciju u celokupnom sistemu, što otežava održavanje i razvoj novih funkcionalnosti.

Mikroservisna arhitektura nudi mogućnost razlaganja složenog informacionog sistema na manje, nezavisne celine koje komuniciraju putem jasno definisanih interfejsa. Time se postiže fleksibilnost, lakše održavanje i mogućnost nezavisnog skaliranja pojedinih servisa.

Ostatak rada je organizovan na sledeći način. U drugom poglavlju dat je pregled postojećih rešenja i tehnologija koje se koriste u obrazovnim informacionim sistemima. Treće poglavlje sadrži specifikaciju zahteva za razvijeni sistem. U četvrtom poglavlju prikazana je specifikacija dizajna, dok peto poglavlje objašnjava implementaciju. Šesto poglavlje sadrži demonstraciju rada sistema kroz tipične scenarije. Sedmo poglavlje donosi zaključna razmatranja i ideje za dalji razvoj, dok su u osmom poglavlju navedene reference korišćene u radu.

# Srodna istraživanja

Jedno od najpoznatijih rešenja je **Moodle** [1], otvorena platforma koja se koristi za upravljanje učenjem. Moodle omogućava kreiranje kurseva, dodeljivanje zadataka i praćenje napretka učenika. Iako je veoma rasprostranjen, Moodle je pre svega usmeren na e-učenje, a manje na evidenciju podataka kao što su ocene i izostanci u formalnom obrazovanju.

**Google Classroom** [2] predstavlja rešenje zasnovano na cloud tehnologijama. Omogućava deljenje materijala, zadavanje domaćih zadataka i komunikaciju između nastavnika i učenika. Prednost ovog sistema je jednostavna integracija sa drugim Google servisima. Međutim, pristup je moguć samo uz Google nalog, što može biti ograničenje za određene škole.

U Srbiji je razvijen **e-Dnevnik** [3], nacionalni informacioni sistem za evidenciju ocena, izostanaka i napomene o učenicima. Sistem je centralizovan i koristi se u osnovnim i srednjim školama. Nedostatak je što se radi o monolitnom rešenju koje je teško prilagodljivo različitim školama i specifičnim potrebama nastavnika.

Na osnovu analize može se zaključiti da je mikroservisni pristup rešenje koje može prevazići nedostatke monolitnih sistema, jer omogućava modularnost i jednostavniju integraciju novih funkcionalnosti.

## 2.2. Korišćene tehnologije

Za razvoj mikroservisa korišćen je programski jezik **Go** [4], poznat po jednostavnosti, brzini i mogućnosti izgradnje visoko-performantnih aplikacija. Za rad sa relacijskim bazama podataka upotrebljena je biblioteka **GORM** [5], koja obezbeđuje ORM sloj i pojednostavljuje interakciju sa bazom.

Za komunikaciju između servisa korišćen je **REST API** [6], koji obezbeđuje standardizovan način razmene podataka putem HTTP protokola.

Kako bi se obezbedila portabilnost i lakše pokretanje servisa u različitim okruženjima, korišćen je **Docker**, dok je uz pomoć **Docker Compose** omogućena orkestracija više međusobno povezanih servisa i baza.

Na strani korisničkog interfejsa, za izgradnju frontend aplikacije koristi se **Angular**, koji omogućava jednostavno povezivanje sa mikroservisima preko REST API-ja i obezbeđuje interaktivno i responzivno korisničko iskustvo.

# Specifikacija zahteva

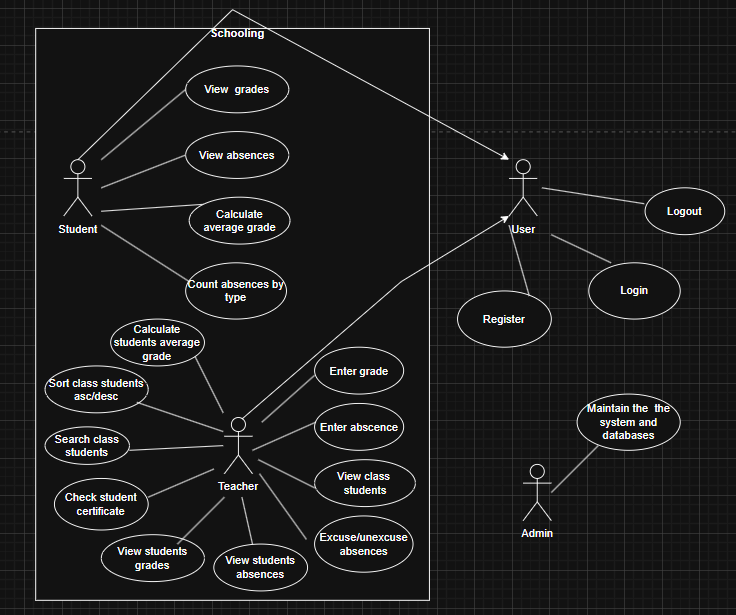
Specifikacija zahteva definiše funkcionalnosti koje sistem mora da podrži.

* Evidencija učenika, nastavnika i predmeta
* Evidencija ocena učenika
* Evidencija izostanaka (opravdani, neopravdani, izostanak na čekanju)
* Izračunavanje prosečne ocene po učeniku i po predmetu
* Mogućnost pregleda svih izostanaka učenika
* Ažuriranje tipa izostanka od strane predmetnog profesora
* Komunikacija sa servisom za autentifikaciju radi provere identiteta korisnika
* Komunikacija sa servisom za zdravstvo radi dobavljanja informacija o zdravstvenom sertifikatu korisnika

## Akteri sistema

* Korisnik – registruje se, prijavljuje na sistem i odjavljuje (login, register, logout)
* Učenik – ima mogućnost pregleda svojih ocena i izostanaka
* Profesor– unosi ocene, evidentira izostanke i status izostanaka
* Administrator – održava sistem i baze podataka

## Use case dijagram



# Specifikacija dizajna

Sistem je razvijen kao mikroservis koji komunicira sa ostalim servisima u okviru šireg sistema za upravljanje obrazovanjem. Arhitektura je organizovana u slojeve:

* Modeli – definicija entiteta (Student, Teacher, Grade, Absence, Subject, Class)
* Servisi – poslovna logika, rad sa bazom, izračunavanja
* Kontroleri (Handlers) – REST API rute koje omogućavaju komunikaciju sa korisnicima i drugim servisima
* Baze podataka – relaciona baza za podatke

# Implementacija

Implementacija je realizovana korišćenjem programskog jezika Go. Struktura projekta podeljena je na module:

* models – definisani entiteti kao Go strukture
* services – implementacija poslovne logike
* handlers – definisane HTTP rute
* database – konekcija sa bazama podataka

Primer API ruta:

* /students/{id}/absences – vraća listu izostanaka učenika
* /grades/student/{id} – vraća ocene učenika
* /grades/average/student/{id} – izračunava prosek ocena učenika
* /grades/average/subject/{id} – izračunava prosek po predmetu

Za rad sa bazom podataka korišćen je GORM.

Primer implementacije metode:

func(s \*SchoolService) GetStudentAbsences(studentID uint) ([]models.Absence, error) {

var absences []models.Absence

if err := s.DB.Where("student\_id = ?", studentID).Find(&absences).Error; err != nil {

return nil, err

}

return absences, nil

}

# Demonstracija

Demonstracija sistema prikazuje tipične scenarije korišćenja.

* Scenario 1: Unos ocene

Profesor unosi ocenu za učenika putem API poziva. Ocena se čuva u bazi i postaje dostupna učeniku prilikom sledećeg pregleda svojih podataka.

* Scenario 2: Evidencija izostanka

Profesor beleži izostanak učenika. Izostanak se vodi kao „na čekanju“. Nakon toga profesor ažurira status u opravdan ili neopravdan.

* Scenario 3: Pregled proseka

Učenik može da pristupi podacima o svojoj prosečnoj oceni po svim predmetima. Sistem koristi agregaciju podataka i vraća izračunatu vrednost.

# Zaključak

Razvijeni školski mikroservis omogućava efikasno upravljanje ocenama i izostancima učenika, uz podršku za integraciju sa širim obrazovnim sistemom. Prednost mikroservisnog pristupa ogleda se u modularnosti, lakšem održavanju i mogućnosti proširenja funkcionalnosti bez uticaja na ceo sistem.

Dalji razvoj može obuhvatiti razvoj korisničkog interfejsa, integraciju sa mobilnim aplikacijama i dodavanje analitičkih modula za predviđanje uspeha učenika.

# Reference

[1] Moodle. About Moodle. Preuzeto 02.10.2025. sa https://moodle.org  
[2] Google. Google Classroom. Preuzeto 02.10.2025. sa https://edu.google.com/classroom  
[3] Ministarstvo prosvete. e-Dnevnik. Preuzeto 02.10.2025. sa https://esdnevnik.rs  
[4] The Go Programming Language. Preuzeto 02.10.2025. sa https://go.dev  
[5] GORM. The fantastic ORM library for Golang. Preuzeto 02.10.2025. sa [https://gorm.io](https://gorm.io" \t "C:\\Users\\Katarina\\AppData\\Local\\Temp\\_new)  
[6] Fielding, R. (2000). Architectural Styles and the Design of Network-based Software Architectures. University of California, Irvine.