

**Diplomski studij**

**Informacijska i komunikacijska tehnologija**

**Računarstvo**

Telekomunikacije i informatika  
Obradba informacija  
Računalno inženjerstvo

**Internet stvari  
  
EVID3NT - Sustav za praćenje prisustva na nastavi**

**Projekt**

**Karlo Dimjašević (voditelj)**

**Fran Šribar**

**Marko Prosenjak**

**Lovro Gaćina**

**Barbara Kralj**

**Marin Pudić**

**Ak.g. 2023./2024.**

**Sadržaj**

[1. Uvod 3](#_Toc169299033)

[2. Opis rješenja 4](#_Toc169299034)

[3. Komunikacija sustava 6](#_Toc169299035)

[4. IoT platforma 10](#_Toc169299036)

[5. Korisničke aplikacije 12](#_Toc169299037)

[6. Dodatak 16](#_Toc169299038)

# Uvod

Sustav za praćenje prisustva na nastavi omogućuje kontrolu prisustva studenata na nastavi. Studenti pomoću posebnih kartica ili privjesaka RFID tehnologijom evidentiraju svoj dolazak na nastavu. Web-aplikacija omogućuje unos i pregled podataka o terminima predavanja i studentima te omogućuje pregled prisutnosti na pojedinim terminima predavanja.

RFID čitač postavili bi na ulaze u predavaonice i učionice. Svaka prostorija za održavanje nastave ima jedinstveni RFID čitač. Studenti dolaskom na predavanje prislone karticu ili privjesak na RFID čitač te time evidentiraju svoj dolazak. Također, bitno je prisloniti karticu ili privjesak i prilikom izlaska iz predavaonice/učionice, jer u suprotnom neće biti moguće evidentirati prisustvo.

Administrativni dio aplikacije omogućuje profesorima unos podataka o predmetima i terminima predavanja. Također, profesori imaju uvid u popis prisustva pojedinog predavanja.

Studenti mogu vidjeti termine predavanja kroz interaktivni kalendar te imati uvid u evidentirane dolaske. Studenti se prijavljuju u aplikaciju e-mail adresom i lozinkom.

Na FER-u je do prije nekoliko godina postojao sličan sustav koji je služio evidentiranju prisustva na nastavi. Moguće je kod ulaza u neke od predavaona vidjeti plave uređaje koji su korišteni u tom projektu.

# Opis rješenja

Kao senzor korišten je RFID RC522 čitač spojen na mikrokontroler ESP32. Čitač komunicira s privjescima ili karticama koji sadrže do 1 kB podataka. U ovom konkretnom slučaju, svaki privjesak (u stvarnoj upotrebi bile bi korištene studentske iskaznice) u memoriji ima ID koji se sastoji od 32 bita (u programima ga koristimo kao četiri 8-bitna cijela broja). Čitač radi na frekvenciji od 15,36 MHz, a ima domet od 10 cm.

Za izradu programskog rješenja korišten je Arduino IDE (korišten za pisanje programskog koda i prevođenje koda), a sam program napisan je u programskom jeziku Arduino (C/C++). U programu se koriste deklaracije varijabli navedene u config.h datoteci (ne nalazi se na Github repozitoriju jer se u njoj nalaze sigurnosno osjetljivi podaci, poput lozinke za lokalnu mrežu, no predana je datoteka example\_config.h koja sadrži sve deklaracije te može poslužiti kao primjer konfiguracije). Pri pokretanju programa, mikrokontroler se spaja na lokalnu Wi-Fi mrežu, koristeći definirane deklaracije navedene u config.h datoteci. Zatim se postavlja MQTT server te se pokušava povezati na njega (parametar MQTT\_CLIENT\_ID generiran je na ThingsBoard IoT platformi te je pohranjen u config.h datoteku). MQTT server o kojem je riječ je ThingsBoardov MQTT server, koji koristi Mosquitto kao radni okvir. Uređaj s ThingsBoard IoT platformom komunicira preko MQTT protokola. U petlji (funkcija loop) se na početku provjerava je li uređaj i dalje povezan na Wi-Fi te MQTT server, te ako nije, potrebno je provesti postupak povezivanja. Nakon toga, program čeka da senzor očita karticu ili privjesak. Navedeno se provjerava funkcijom rfid.PICC\_IsNewCardPresent – u slučaju da nije otkrivena kartica, izlazi se iz funkcije te se ne izvodi ostatak programskog koda ispod te naredbe. U trenutku kada je privjesak ili kartica skenirana, program provjerava je li ID prislonjene kartice (ili privjeska) jednak onome što je pohranjeno u nuidPICC varijabli. U nuidPICC se spremaju podaci o ID-u posljednje očitane kartice (ili privjeska). Ovaj uvjet je potreban kako se ne bi zabilježilo skeniranje iste osobe više uzastopnih puta. U slučaju da se ne radi o istoj kartici (ili privjesku) koja je korištena prilikom prijašnjeg očitanja (dovoljno je da se ID razlikuje u jednom byte-u), rfid.uid.uidByte se pretvara u String, te se zajedno s učionicom u kojoj bi se uređaj trebao nalaziti, formatira u JSON poruku.

Dobiveni podaci se u JSON formatu šalju na MQTT server na topic generičkog oblika v1/buildings/+/classrooms/# (npr. v1/buildings/b/classrooms/b4). U projektu nije omogućeno povezivanje na MQTT server koristeći dodatni sigurnosni protokol TLS, jer bi za ostvarivanje te mogućnosti bilo potrebno konfigurirati ThingsBoard, što nije moguće provesti bez pristupa računalu na kojem je pokrenut.

Osim fizičkog senzora, za isprobavanje skalabilnosti rješenja implementiran je i virtualni senzor u obliku skripte napisane u programskom jeziku Python. Pythonbiblioteka paho omogućuje programu da stvori virtualnog MQTT klijenta koji se spaja naintegrirani MQTT server ThingsBoard IoT platforme.

Skripta funkcionira na sljedeći način: pri pokretanju, korisnik kao argumente skripte zadaje naziv učionice u kojoj bi se senzor kojeg se simulira nalazio, te broj studenata koji bi se tamo evidentirao. Pahoklijent se stvara i zatim povezuje s MQTT serverom. Nakon toga, program ulazi u petlju u kojoj klijent šalje poruke serveru na topicdefiniran nazivom učionice. Npr. ako je učionica zadana kao a102, klijent šalje poruku na topic v1/buildings/a/classrooms/a102. Format poruke koja se šalje jednak je formatu one koju šalje fizički uređaj. Broj ponavljanja petlje jednak je zadanom broju studenata.

Nakon izvršavanja skripte, na *dashboardu* ThingsBoard IoT platforme mogu se vidjeti aktivnost i poslane MQTT poruke, kao da ih je poslao stvarni uređaj. Korištenje virtualnog senzora sa zadanim visokim brojem skeniranja u kratkom vremenu služi za demonstraciju skalabilnosti IoT platforme i razvijene korisničke aplikacije.

# Komunikacija sustava

Sustav se sastoji od nekoliko komponenti. Komponente, ovisno o ulozi u sustavu, komuniciraju priladnim protokolom. *Slika 1* prikazuje korištenu arhitekturu sustava te korištene tehnologije za pojedinu komponentu i komunikacijske protokole.

A screenshot of a computer

Description automatically generated

Slika 1 Arhitektura sustava i korišteni protokoli

Sekvencijski dijagram (vidi *Slika 2 Komunikacija sustava*) prikazuje primjer komunikacije cijelog sustava. Uređaj nakon očitanja privjeska ili kartice šalje JSON poruku na MQTT server. Format (engl. *payload*) JSON poruke izgleda kao na sljedećem primjeru:

{  
 'cardId': '28-39-09-213',  
 'classroom': 'b4'  
}

ThingsBoard je konfiguriran na način da pohranjuje sva senzorska očitanja u svoju PostgreSQL bazu podataka. Zbog domene primjene i problematike koju Sustav za praćenje prisustva na nastavi rješava, bilo je potrebno sinkronizirati podatke s ThingsBoard IoT platforme u vlastitu PostgreSQL bazu podataka. Ovo je ostvareno uvođenjem dodatnog servisa Synchronizer koji periodički (primjerice, svakih 15 sekundi) preko REST API sučelja dohvaća sve trenutno aktivne uređaje, te za svaki uređaj dohvaća posljednja senzorska očitanja. Bitno je napomenuti kako ovo nije idealno rješenje, no zbog ograničenosti pristupu računala na kojemu je pokrenuta ThingsBoard IoT platforma, ovo je bilo najbolje moguće rješenje. Bolji pristup sinkronizaciji podataka bio bi korištenjem MQTT servera i pretplate na definirane kanale, ili korištenje web socket-a. Primjer odgovora ThingsBoard IoT platforme na zahtjev za senzorskim očitanjima:

{  
 'cardId': [  
 {  
 'ts': 1718397195403,  
 'value': '28-39-09-213'  
 }  
 ],  
 'classroom': [  
 {  
 'ts': 1718397195403,  
 'value': 'b4'  
 }  
 ]  
}

Priložen primjer nam govori sljedeće: dogodilo se očitanje na senzoru u predavaoni b4 za studenta/profesora čiji je ID privjeska ili kartice '28-39-09-213', a vrijeme očitanja je petak, 14. lipnja 2024. u 20 sati, 33 minute i 15 sekundi.

Synchronizer servis dohvaćene podatke sa ThingsBoard IoT platforme pohranjuje u vlastitu bazu podataka (vidi *Slika 4*).

A diagram with text and words

Description automatically generated

Slika 2 Komunikacija sustava

Na zahtjev studenta za prikazom stranice prisustva, korišteni web-preglednik šalje zahtjev za dohvat podataka o prisustvu studenta. Podatak se prosljeđuje na backend servis koji komunicirajući s bazom podataka dohvaća prisustva studenata. Općenito govoreći, komunikacija između frontend servisa i backend servisa odvija se po REST načelu. Backend servis nudi prikaz dokumentacije sučelja (swagger ui), a *Slika 3* prikazuje primjer jednog zahtjeva i odgovora.

A screenshot of a computer

Description automatically generated

Slika 3 Primjer dokumentacije za backend API

Konačno, u nastavku je prikazan i model baze podataka, korišten za rješavanje problematike sustava.

A computer screen shot of a computer

Description automatically generated

Slika 4 Konceptualni model baze podataka

# IoT platforma

Pri izradi projekta koristila se IoT platforma ThingsBoard. ThingsBoard je platforma otvorenog koda dizajnirana za upravljanje, vizualizaciju i analizu podataka s povezanih uređaja. Pruža robusnu infrastrukturu za IoT aplikacije, omogućujući razvojnim programerima učinkovitu implementaciju i upravljanje IoT rješenjima. ThingsBoard podržava razne IoT protokole, integrira se s postojećim sustavima i nudi niz značajki koje ga čine popularnim izborom među programerima i tvrtkama. ThingsBoard pruža opsežne alate za upravljanje IoT uređajima. Podržava različite komunikacijske protokole kao što su MQTT (Message Queuing Telemetry Transport), CoAP (Constrained Application Protocol) i HTTP (Hypertext Transfer Protocol), što ga čini svestranim i prilagodljivim različitim scenarijima. Platforma ima mogućnost prikupljanja podataka s više senzora i uređaja u stvarnom vremenu. Omogućuje transformaciju podataka, filtriranje i obogaćivanje, osiguravajući da su prikupljeni podaci smisleni i spremni za analizu. Jedna od istaknutih značajki ThingsBoard-a je njegov ugrađeni 'Dashboard' za vizualizaciju podataka. Korisnici mogu stvoriti vrlo prilagodljive i interaktivne Dashboard-e za praćenje svojih IoT uređaja i podatkovnih trendova. To uključuje grafove u stvarnom vremenu, mape i karte koji pružaju jasan i intuitivan prikaz podataka. 'Rule Engine' je jednostavan framework za izgradnju toka rada temeljenog na događajima. Sadrži 3 glavne komponente: poruka (svaki dolazni događaj), 'Rule Node' (funkcija koja se izvršava nad dolaznom porukom) i 'Rule Chain' (čvorovi su međusobno povezani relacijama, tako da se izlazna poruka iz Rule Node-a šalje sljedećim povezanim čvorovima). Framework omogućuje automatizaciju radnji na temelju pristiglih podataka. Korisnici mogu definirati složena pravila za pokretanje upozorenja, obavijesti ili kontrolnih radnji. Ova značajka podržava složenu obradu događaja, što je bitno za naprednu analizu podataka i automatizirane odgovore. ThingsBoard pruža korisničko sučelje i REST API-je za upravljanje višestrukim tipovima entiteta i njihovim odnosima. Podržani entiteti su: Tenants (pojedinac ili organizacija koja posjeduje ili proizvodi uređaje), Customers (pojedinac ili organizacija koji kupuju ili koriste uređaje Tenant-a), Users (mogu pregledavati Dashboard-e i upravljati entitetima), Devices (osnovni IoT entiteti koji mogu proizvesti telemetrijske podatke), Assets (apstraktni IoT entiteti koji mogu biti povezani s drugim uređajima), Entity Views (korisni ako s korisnicima želite dijeliti samo dio podataka), Alarms (događaji koji identificiraju probleme s uređajima ili drugim entitetima), Dashboards, Rule Chain i Rule Node. Svaki entitet podržava atribute koji predstavljaju statičke i polustatičke parove ključ-vrijednost povezane s entitetima, 'Time-series' podatkovne točke dostupne za pohranjivanje, postavljanje upita i vizualizaciju, te svaki entitet također podržava usmjerene veze s drugim entitetima. Platforma osigurava integritet i privatnost podataka kroz snažne sigurnosne mjere, uključujući enkripciju i kontrolu pristupa.

ThingsBoard nudi dvije glavne verzije svoje platforme: Community Edition i Professional Edition. Obje verzije zadovoljavaju različite potrebe korisnika. Community Edition verzija platforme je otvorenog koda. Dostupna je besplatno za preuzimanje i korištenje, što ju čini idealnom za programere i male tvrtke koje žele raditi na IoT rješenjima bez financijskih ulaganja. Professional Edition je komercijalna verzija koja uključuje poboljšane značajke, podršku i opcije skalabilnosti dizajnirane za implementacije na razini poduzeća. Nudi dodatne alate i usluge koji nisu dostupni u Community Edition verziji, što ju čini prikladnom za tvrtke s kritičnim IoT potrebama.

Za izradu projekta bilo je potrebno iskoristiti većinu mogućnosti koje pruža platforma Thingsboard. Korišten je podržani MQTT protokol za pouzdanu i učinkovitu komunikaciju između ESP uređaja i ThingsBoard platforme. Navedeni protokol je prikladan za scenarije koji uključuju brojne uređaje i čest prijenos podataka. Potrebni uređaji registrirani su na platformi, osiguravajući da se svaki uređaj može jedinstveno identificirati i upravljati njime. Sposobnost platforme za prikupljanje telemetrijskih podataka iskorištena je za prikupljanje potrebnih poslanih podataka s uređaja. Za obradu dolaznih tokova podataka korišten je Rule Engine. Definirana su pravila za filtriranje, transformaciju i obogaćivanje telemetrijskih podataka na temelju unaprijed definiranih uvjeta. Dolazni podaci od uređaja su transformirani kako bi uključili varijablu pomoću koje je omogućeno praćenje aktivnosti uređaja kroz vrijeme. Značajke Dashboard-a korištene su za stvaranje vizualizacija određenih parametara vezanih uz rad i korištenje uređaja. Pomoću njega moguće je pratiti vremenski period u kojem je određeni uređaj bio aktivan, te je također moguće dobiti uvid u informaciju o vremenu i mjestu gdje je i kada određena kartica skenirana.

# Korisničke aplikacije

U sklopu projekta razvijena je jedna glavna korisnička aplikacija koja podržava tri različite uloge: admin, student i profesor. Svaka uloga ima specifične funkcionalnosti i pristup različitim dijelovima aplikacije, čime se osigurava prilagođeno korisničko iskustvo prema potrebama svakog korisnika.

**Studenti** koriste aplikaciju za pregled termina predavanja i uvid u svoje prisustvo. Prilikom dolaska na predavanje, studenti prislone svoje RFID kartice ili privjeske na čitače postavljene na ulazima u predavaonice. Ova evidencija se automatski bilježi u sustavu. Studenti se prijavljuju u aplikaciju putem e-mail adrese i lozinke, što im omogućuje siguran i personaliziran pristup njihovim podacima.

**Profesori** imaju pristup podacima o prisustvu studenata na svojim predavanjima. Oni mogu pregledavati evidenciju prisustva za svaki predmet i termin, što im pomaže u praćenju i evaluaciji prisutnosti studenata. Također, profesori mogu unositi i mijenjati termine predavanja.

**Admin** uloga namijenjena je administratorima sustava, omogućujući im unos i pregled podataka o predmetima, terminima predavanja i studentima. Administratori mogu upravljati svim aspektima sustava, uključujući dodavanje novih predmeta, definiranje termina predavanja te pregled i upravljanje evidencijom prisustva studenata.

Izgled aplikacije pažljivo je dizajniran kako bi bio intuitivan i jednostavan za korištenje. Pri samom učitavanju aplikacije prikazuje se sučelje za prijavljivanje, prikazano na slici 1, odnosno registraciju, prikazano na slici 2, u sustav.

A screenshot of a login form

Description automatically generated

Slika 5: Prikaz sučelja za prijavu

A screenshot of a login form

Description automatically generated

Slika 6: Prikaz sučelja za registraciju

Nakon uspješne prijave korisnik može vidjeti pregled postojećih predavanja u kalendaru (slika 3). Klikom na određeni datum i kalendaru studentu se prikazuje popis predavanja koje ima toga datuma te sve popratne informacije vezane uz predavanje (ime kolegija, učionica u kojoj se predavanje održava, kada počinje i kada završava). Dodatno uz te informacije, profesor ima mogućnost dodati novo predavanje klikom na stavku *Add lecture* gdje može, u formu, unijeti potrebne informacije za novo predavanje kako je prikazano na slici 4. Također svako postojeće predavanje može i obrisati klikom na gumb *Obriši* koji se nalazi uz predavanje na koje se odnosi.

A screenshot of a computer

Description automatically generated

Slika 7: Prikaz kalendara s popisom predavanja

A screenshot of a computer screen

Description automatically generated

Slika 8: Prikaz forme za unos novog predavanja

Klikom na stavku *Atteendance* omogućuje mu se prikaz svih predavanja za kolegije u koje je upisan kao što je vidljivo na slici 5. Uz svako predavanje se nalaze i informacije o datumu održavanja, učionici u kojoj je predavanje održano, kada je predavanje počelo i završilo te informacija je li student bio prisutan. S druge strane, profesor ima padajući izbornik gdje odabire studenta za kojeg želi vidjeti sve zabilježene prisutnosti. Nakon pronalaska željenog studenta, prikazuje se popis svi predavanja za tog student zajedno s informacijom je li studen bio prisutan na predavanjima ili ne.

A screenshot of a computer

Description automatically generated

Slika 9: Prikaz prisutnosti za odabranog studenta

Za sve korisnike aplikacija sadrži gumb za odjavu (logout).

Tehnologije korištene u razvoju ove aplikacije uključuju ASP.NET (C#) za backend, Angular za frontend i PostgreSQL za bazu podataka.

**Pokretanje i postavljanje projekta na backendu**

Da bi se projekt pokrenuo i postavio preduvjet je imati instaliran Visual Studio s podrškom za .NET Core i ASP.NET. Nakon toga potrebno je klonirati repozitorij s GitHuba na vlastito računalo. Sada je potrebno učitati projekt u Visual Studiou na način da se u početnim izborniku odabere *Open a project or solution* ili *File* -> *Open* -> *Project / Solution...* . Navigiramo se do projekta te odabiremo dokument *be-evid3nt.snl*. Sada kada je projekt postavljen, pokreće se klikom na gumb be\_evid3nt (slika 6).



Slika 10: Pokretanje projekta

**Implementacijski detalj**

Jedan od najizazovnijih dijelova sustava bilo je napisati upit prema bazi podataka koji će dohvatiti sve termine predavanja za pojedinog studenta te odrediti na temelju sinkroniziranih očitanja senzora je li student bio prisutan na pojedinom predavanju. Poslovno pravilo je takvo da studenta smatramo prisutnim ako je evidentirao dolazak do 15 minuta prije ili nakon početka termina predavanja i ako je evidentirao odlazak do 15 minuta prije ili nakon kraja termina predavanja. Detaljni implementacije dostupni su na javnom GitHub repozitoriju (vidi *Dodatak*) u metodi backend/be-evid3nt/Controllers/TelemetryController.cs::GetStudentAttendanceAsync.

# Dodatak

Poveznica na javno dostupan GitHub repozitorij: <https://github.com/evid3nt/System-For-Monitoring-Class-Attendance/tree/develop>  
Ovdje je sadržan **izvorni kod** sustava i upute za pokretanje pojedine komponente sustava (README.md datoteka u mapi pojedine komponente).