Web aplikacija studentskog rasporeda uz podršku Oracle baze podataka

Đuričković, Filip

Undergraduate thesis / Završni rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:875902

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-03-21**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

WEB APLIKACIJA STUDENTSKOG RASPOREDA UZ PODRŠKU ORACLE BAZE PODATAKA

Završni rad

Filip Đuričković

Osijek, 2020.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 01.09.2020.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime studenta:	Filip Đuričković
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3915, 23.09.2019.
OIB studenta:	69211231243
Mentor:	izv. prof. dr. sc. Alfonzo Baumgartner
Sumentor:	
Sumentor iz tvrtke:	
Naslov završnog rada:	Web aplikacija studentskog rasporeda uz podršku Oracle baze podataka
Znanstvena grana rada:	**Informacijski sustavi (zn. polje računarstvo)**
Predložena ocjena završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	01.09.2020.
Datum potvrde ocjene Odbora:	09.09.2020.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 09.09.2020.

Ime i prezime studenta:	Filip Đuričković
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3915, 23.09.2019.
Turnitin podudaranje [%]:	9

Ovom izjavom izjavljujem da je rad pod nazivom**: Web aplikacija studentskog rasporeda uz podršku Oracle baze podataka**

izrađen pod vodstvom mentora izv. prof. dr. sc. Alfonzo Baumgartner

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA

o odobrenju za pohranu i objavu ocjenskog rada

kojom ja _____Filip Đuričković_____, OIB: ____69211231243____, student Fakulteta elektrotehnike, računarstva i informacijskih tehnologija Osijek na studiju _Preddiplomski sveučilišni studij Računarstvo_, kao autor ocjenskog rada pod naslovom: _Web aplikacija studentskog rasporeda uz podršku Oracle baze podataka_, dajem odobrenje da se, bez naknade, trajno pohrani moj ocjenski rad u javno dostupnom digitalnom repozitoriju ustanove Fakulteta elektrotehnike, računarstva i informacijskih tehnologija Osijek i Sveučilišta te u javnoj internetskoj bazi radova Nacionalne i sveučilišne knjižnice u Zagrebu, sukladno obvezi iz odredbe članka 83. stavka 11. *Zakona o znanstvenoj djelatnosti i visokom obrazovanju* (NN 123/03, 198/03, 105/04, 174/04, 02/07, 46/07, 45/09, 63/11, 94/13, 139/13, 101/14, 60/15).

Potvrđujem da je za pohranu dostavljena završna verzija obranjenog i dovršenog ocjenskog rada. Ovom izjavom, kao autor ocjenskog rada dajem odobrenje i da se moj ocjenski rad, bez naknade, trajno javno objavi i besplatno učini dostupnim:

a) široj javnosti

b) studentima/icama i djelatnicima/ama ustanove

c) široj javnosti, ali nakon proteka 6 / 12 / 24 mjeseci (zaokružite odgovarajući broj mjeseci).

**U slučaju potrebe dodatnog ograničavanja pristupa Vašem ocjenskom radu, podnosi se obrazloženi zahtjev nadležnom tijelu Ustanove.*

_____Osijek, 09.09.2020._____ _____

(mjesto i datum) (vlastoručni potpis studenta/ice)

SADRŽAJ

1. Uvod 1
1.1. Zadatak završnog rada 1
2. Opis korištenih tehnologija i usporedba s postojećim rješenjima 2
2.1. Oracle sustav upravljanja bazom podataka 2
2.2. Oracle APEX razvojni okvir 2
2.2.1. Oracle REST podatkovni servisi 3
2.2.2. Apache Tomcat 3
2.2.3. Apache HTTP Server 3
2.2.3.1. Apache modul mod_auth_mellon 3
2.3. PL/SQL jezik 4
2.4. XML jezik 5
2.5. CSS 5
2.6. *iCalendar* format 5
2.7. Postojeća programska rješenja s istom namjenom 6
3. Modeliranje baze podataka 8
3.1. Analiza postojećih podataka i stvaranje konceptualnog modela 8
3.2. Stvaranje modela 11
3.3. Fizičko oblikovanje baze podataka 14
4. Izrada pozadinskih procesa u PL/SQL jeziku 21
4.1. Procedure za preuzimanje i obradu podataka iz MRKVE sustava 23
4.2. Vanjsko sučelje prema drugim platformama 26
4.3. Izračun zajedničkih slobodnih perioda između studenata 29
5. Izrada aplikacije u Oracle APEX razvojnom okviru 35
5.1. Konačan izgled aplikacije 40
6. Zaključak 46
Literatura 48
Popis kratica 52
Popis slika 53
Popis tablica 55
Sažetak 56
Abstract 57
Životopis 58

Prilozi 59

1. UVOD

Raspored nastave za studenta ili učenika jest tablica koja objedinjuje podatke o studentima, nastavnicima, prostorijama i nastavnim terminima na jednom mjestu. Na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek (FERIT) od 2013. godine se koristi aplikacija Digitalni raspored za praćenje nastavnih obaveza. FERIT-ov Digitalni raspored je web aplikacija koju Fakultet koristi za organizaciju nastave i ispita po pojedinim tjednima u akademskoj godini. Velika mana te aplikacije jest činjenica da ne nudi prikaz nastave pojedinom studentu tako da on vidi samo svoje grupe i termine, nego je u aplikaciji prikazana sva nastava pojedinog smjera i godine. To često dovodi do greški u čitanju rasporeda i dolasku u krive termine nastave i slično. Osim toga, aplikacija ne nudi mogućnost praćenja dodatnih grupa i kolegija (npr. za studente ponavljače ili demonstratore) ili prikaz u bilo kojem drugom obliku osim u tjednom prikazu. Zbog toga, cilj ovog završnog rada jest izraditi aplikaciju koja ispravlja te probleme uz dodavanje novih funkcionalnosti koje su praktične i korisne studentima. U ovom radu će biti opisan proces modeliranja baze podataka prema već dostupnim podatcima i opisati će se proces transformacije tih podataka u oblik pogodan za određenu svrhu različitu od originalne namjene. Procesi će biti objašnjeni na primjeru transformacije podataka iz FERIT-ovog Digitalnog rasporeda u oblik pogodan za laganu obradu i prikaz rasporeda za pojedinog studenta. Na kraju će biti prikazan postupak pisanja pozadinske poslovne logike, kao i postupak izrade aplikacije u Oracle APEX-u.

1.1. Zadatak završnog rada

Zadatak završnog rada jest izraditi web aplikaciju u Oracle APEX razvojnom okruženju koja za svrhu ima prikaz pojedinačnog rasporeda za svakog studenta FERIT-a. Osim pristupa putem web aplikacije, potrebno je osigurati pristup kalendarima iz drugih softverskih rješenja poput Microsoft Outlook-a, Google Calendar-a i sličnih platformi koristeći otvorene standarde za razmjenu kalendara. Aplikacija treba ponuditi studentima dodavanje dodatnih privatnih događaja u kalendar koji su vidljivi samo pojedinom studentu, kao i omogućiti računanje slobodnih termina između dva ili više studenata.

2. OPIS KORIŠTENIH TEHNOLOGIJA I USPOREDBA S POSTOJEĆIM RJEŠENJIMA

Prilikom izrade ovog završnog rada upotrijebljene su industrijski priznate i korištene tehnologije, jezici i programi koji su se tijekom svog dugogodišnjeg korištenja iskazali svojom stabilnošću, proširivosti i fleksibilnosti.

2.1. Oracle sustav upravljanja bazom podataka

Oracle baza podataka je višemodelna baza podataka koja podržava relacijske, nestrukturirane, objektno-orijentirane, XML i JSON strukturirane modele podataka. Ona predstavlja sustav za upravljanje bazama podataka (SUBP) koji korisnicima omogućava upravljanje, stvaranje i izmjenu relacijskih i drugih baza podataka. Prema *DB-Engine* ljestvici, Oracle baza je najkorišteniji SUBP na svijetu za *online* obradu upita i analizu podataka [1]. Korisnici ga mogu pokretati na 7 operacijskih sustava i direktno pristupati iz više desetaka programskih jezika koji uključuju Javu, COBOL, .NET obitelj jezika, C/C++, Ruby, PHP, Node.js, R i ostale [2]. Osim što nudi mogućnost pristupa iz navedenih jezika, Oracle SUBP ima vlastiti programski jezik PL/SQL namijenjen pisanju aplikativne logike koja je pohranjena i izvršava se unutar baze podataka zbog ostvarivanja boljih performansi i više razine sigurnosti pri obradi podataka.

2.2. Oracle APEX razvojni okvir

Oracle APEX (engl. *Oracle Application Express*) je „nisko-kodna“ platforma za brzi razvoj poslovnih web aplikacija unutar Oracle baze podataka. Poslužiteljski dio razvojnog okruženja i krajnjih aplikacija se pokreće unutar same baze podataka u PL/SQL jeziku, dok korisnici i programeri pristupaju sučeljima za razvoj i korištenje putem internetskog preglednika. APEX kao platforma je besplatna za korištenje i ugrađena je unutar svake instalacije baze počevši od verzije Oracle 11*g* [3]. Ona omogućuje brz i jednostavan razvoj modernih, responzivnih web aplikacija bez potrebe za pisanjem velike količine HTML, CSS i JavaScript koda jer se programeri mogu osloniti na postojeće teme, predloške i komponente razvojnog okvira i okruženja što omogućuje brzo stvaranje gotovih aplikacija i kasnije olakšava održavanje i nadogradnju aplikacija. Programeri osim ugrađenih komponenti i predložaka mogu stvarati svoja proširenja koristeći kombinaciju JavaScript-a, HTML-a i CSS-a za klijentski dio koda i PL/SQL za poslužiteljski dio proširenja.

2.2.1. Oracle REST podatkovni servisi

Oracle REST podatkovni servisi (engl. *Oracle REST Data Services* – ORDS) je Java EE aplikacija koja za ulogu ima povezivanje Oracle baze podataka s web klijentima na Internetu. Ona se može pokretati unutar Apache Tomcat aplikativnog poslužitelja, unutar Oracle WebLogic-a ili samostalno koristeći svoj unutarnji ugrađeni HTTP (engl. *Hypertext Transfer Protocol*) poslužitelj [4]. Osim što nudi mogućnost javnog izlaganja PL/SQL programskih jedinica prema Internetu, ORDS programerima i administratorima baza podataka nudi mogućnost da izlože pojedine tablice, poglede i ostale objekte unutar baze podataka kao REST servise koje druge aplikacije mogu koristiti uz minimalnu količinu truda. Često se koristi kao metoda za vanjski web pristup Oracle APEX-u koji je unutar Oracle baze.

2.2.2. Apache Tomcat

Apache Tomcat je Java *Enterprise Edition* aplikacijski poslužitelj otvorenog koda kojeg razvija *Apache Software Foundation* od 1999. godine [5]. Njegova svrha jest izvršavanje Java *servlet*-a i prikaz Java serverskih stranica (engl. *Java Server Pages* – JSP) [6]. U ovom završnom radu će se koristiti u kombinaciji s Apache HTTP poslužiteljem kako bi izložio aplikaciju putem ORDS-a javno na Internet.

2.2.3. Apache HTTP Server

Apache HTTP Server je web poslužitelj otvorenog koda kojeg razvija *Apache Software Foundation* [7]. On je jedan od najpopularnijih web poslužitelja [8] zahvaljujući svojoj otvorenosti, efikasnosti i proširivosti. Njegova prednost je što je vrlo skalabilan i uz minimalne promjene konfiguracije vrlo dobro podnosi veliki broj klijenata, kao i činjenica da je za njega napisano mnogo modula koji proširuju njegovu funkcionalnost s značajkama poput HTTPS šifrirane komunikacije, podrškom za poslužiteljske skriptne jezike poput PHP-a i sličnih [9]. U ovom završnom radu Apache HTTP Server će se koristiti kao *proxy* poslužitelj prema Tomcat-u koji osigurava šifriranu HTTPS vezu prema klijentima i izvršava sve poslove identifikacije korisnika pomoću modula mod_auth_mellon.

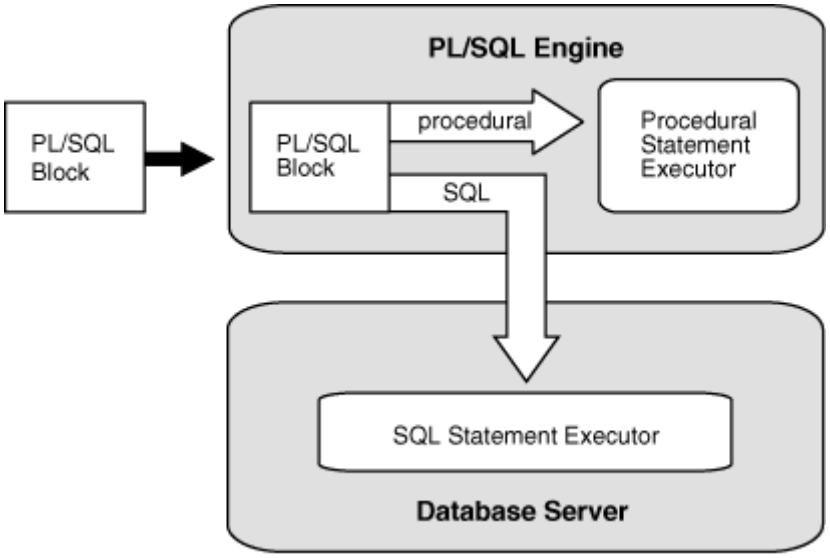
2.2.3.1. Apache modul mod_auth_mellon

Mod_auth_mellon je modul za Apache HTTP poslužitelj koji služi za identificiranje korisnika pomoću SAML 2.0 standarda, a razvila ga je Uninett tvrtka za potrebe norveške akademske i istraživačke mreže. On odobrava ili brani pristup određenim resursima na poslužitelju te po potrebi

prosljeđuje primljene podatke o korisniku daljnjim aplikacijama. U ovom završnom radu on se koristi za autentifikaciju korisnika web aplikacije pomoću njihovih AAI@EduHR korisničkih identiteta. AAI@EduHR infrastruktura omogućava vanjskim davateljima usluga (aplikacijama) autentifikaciju korisnika iz sustava znanosti i visokog obrazovanja putem SAML 2.0 standarda [10].

2.3. PL/SQL jezik

PL/SQL (engl. *Procedural Language extension for Structured Query Language*) je lako prenosiv i brz proceduralni jezik za obradu transakcija unutar Oracle baze podataka. On je usko povezan s SQL jezikom na način da se unutar jezika mogu koristiti svi SQL upiti i naredbe za manipulaciju podatcima i transakcijama, kao i sve ugrađene funkcije, operatori, tipovi podataka bez potrebe za konverzijom podataka. S obzirom da su PL/SQL programske jedinice većinu vremena pohranjene i izvršavane unutar same baze podataka, programi mogu ostvariti značajna ubrzanja zbog činjenice da podatci ne moraju biti razmjenjivani između klijenta i baze podataka putem mreže prilikom obrade [11]. Osim toga, PL/SQL je lako prenosiv zbog činjenice da se on može izvršavati na svakoj platformi na kojoj je dostupna Oracle baza podataka bez potrebe za zasebnim izdanjima koda ili ponovnim prevođenjem izvornog koda za novu platformu. Jezik je po svojoj sintaksi sličan Ada programskom jeziku jer dizajniran je prema njemu [11]. U ovom završnom radu se on koristi za pisanje sve aplikativne logike korištene unutar aplikacije i pozadinskih servisa.



Slika 2.1. Struktura izvođenja PL/SQL programa, izvor: [11]

2.4. XML jezik

XML (engl. *Extensible Markup Language*) je opisni jezik koji je nastao iz SGML (engl. *Standard Generalized Markup Language*, ISO 8879) jezika i definira skup pravila za kodiranje dokumenata i podataka u formatu pogodnom za strojnu i ljudsku obradu. Nastao je s ciljevima široke upotrebljivosti diljem Interneta i jednostavnosti obrade i stvaranja dokumenata [12]. XML je općeprihvaćen format za razmjenu podataka koji je uvelike standardiziran. Iako je prvotno bio osmišljen za opisivanje dokumenata, današnja glavna primjena je prikaz složenih struktura podataka, osobito u *web* servisima [13]. Svi podatci MRKVE sustava su dostupni putem standardiziranog programskog sučelja u XML formatu.

2.5. CSS

CSS (engl. *Cascading Style Sheets*) je stilski opisni jezik za stiliziranje elemenata označnih strukturiranih dokumenata poput HTML dokumenata [14]. Danas se ponajprije koristi za oblikovanje web stranica i predstavlja integralni dio svake web stranice. Direktive za oblikovanje se prilikom iscrtavanja dokumenta na ekranu vežu za pojedine elemente HTML dokumenta koji ispunjavaju uvjete zadane selektorom navedenim ispred direktiva u CSS datoteci [15]. U sklopu ovog završnog rada, CSS će se koristiti za dodatno oblikovanje elemenata unutar kalendara tako da svojim izgledom sliče u što većoj mjeri onima iz FERIT Digitalnog rasporeda kako ne bi korisnike dovodili u zabludu.

2.6. *iCalendar* format

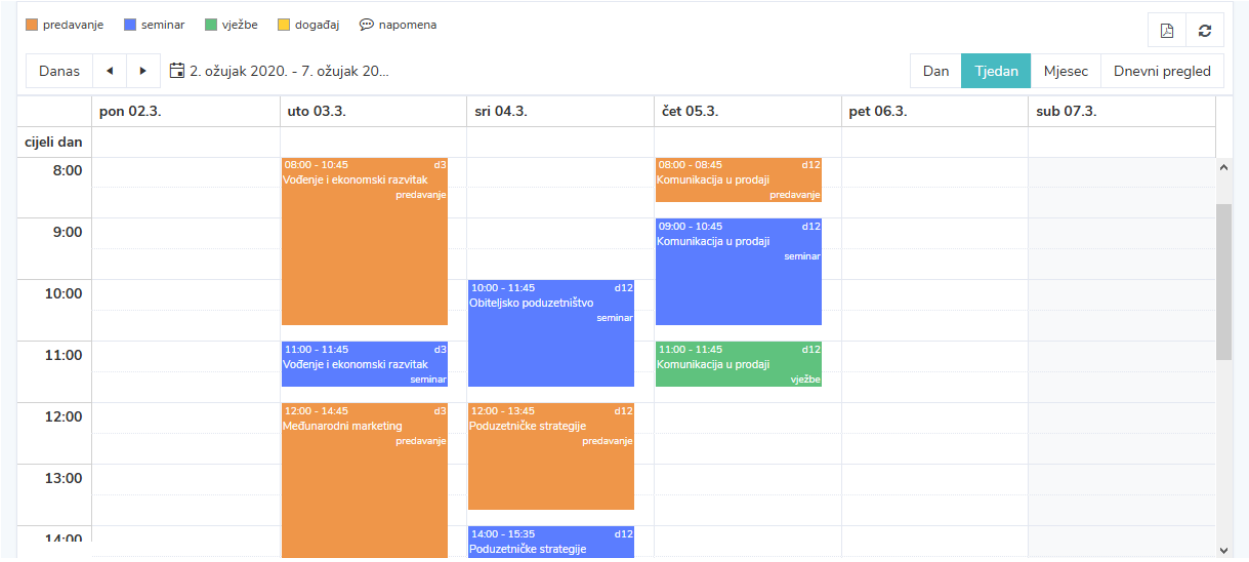
iCalendar (engl. *Internet Calendaring and Scheduling Core Object Specification* – iCal) je standard za razmjenu i pohranu raznih rasporeda i kalendara i događaja u jednostavnom tekstualnom obliku. Format tekstualne datoteke je opisan u predloženom internetskom standardu RFC 5545 [16] kojeg su implementirali mnogi proizvodi poput *Microsoft Outlook*-a [17], *Google Calendar*-a [18], *Apple Calendar*-a [19] i drugih. Standard u sebi ne propisuje određenu metodu za dostavu samih datoteka s informacijama, već je predviđeno da datoteke budu formatirane na način koji u što većoj mjeri omogućava siguran prijenos informacija bez grešaka neovisno o transportnom protokolu i mediju (HTTP resurs, prilog e-poruci i sl.). Svaka datoteka predstavlja jedan kalendar ili raspored, a u sebi u sebi može sadržavati više jedan ili više događaja, a svako svojstvo događaja je opisano u formatu „ključ : vrijednost“ [16].

2.7. Postojeća programska rješenja s istom namjenom

Pregledom postojećih programskih rješenja, vidljivo je da na tržištu ne postoje programi koji bi imali sve značajke kao i ovaj završni rad. FERIT-ov Digitalni raspored u obliku kako je sad implementiran ne nudi pregled nastave za pojedinog studenta, već samo nudi tu mogućnost nastavnicima. Također ne nudi mogućnost dodavanja osobnih događaja u kalendar, kao ni automatsku razmjenu podataka s drugim platformama.

Slika 2.2. Prikaz nastavnog tjedna u Digitalnom rasporedu FERIT-a

Druga aplikacija sa sličnom funkcionalnosti jest Academio web aplikacija koja nudi sličnu osnovnu funkcionalnost poput pregleda nastave za pojedinog studenta i dodavanje dodatnih pretplata na kolegije u raspored, ali Academio podržava samo ISVU sustav kao izvor podataka o studentima, nastavnicima i kolegijima, dok se raspored sati stvara unutar same aplikacije. Također, Academio ne nudi opciju računanja slobodnih termina između studenata.



Slika 2.3. Prikaz nastavnog tjedna u Academio aplikaciji

3. MODELIRANJE BAZE PODATAKA

Modeliranje baze podataka predstavlja proces stvaranja konceptualne reprezentacije podataka unutar baze podataka i definiranja veza između određenih podatkovnih objekata temeljenih na apstraktnom poimanju stvarnog svijeta. Pravilno dizajniran model baze podataka pomaže osigurati točnost podataka pomoću definiranja ograničenja nad podatcima, osigurava integritet podataka pomoću veza i pomaže u pronalasku nedostajućih i redundantnih podataka. Proces modeliranja se može rastaviti na tri ključna dijela [20]:

- Utvrđivanje i analiza zahtjeva i postojećih podataka
- Stvaranje konceptualnih i logičkih modela podataka
- Prevođenje logičkih modela u fizički model za implementaciju u bazi podataka

S obzirom da su zahtjevi za aplikaciju Studentskog rasporeda već utvrđeni i analizirani, taj korak se u ovom slučaju može preskočiti i modeliranje se može započeti analizom postojećih podataka. Zbog olakšavanja cijelog procesa modeliranja i lakšeg dokumentiranja danas se koriste alati posebno predviđeni za modeliranje podataka poput *SAP PowerDesigner*, *Toad Data Modeler*, *erwin Data Modeler* i drugih, ovisno o tome koji će se sustav za upravljanje bazom podataka odabrati za fizičku implementaciju. Takvi alati korisnicima omogućuju da započnu proces modeliranja tako da izrade ER dijagram i sustav ga tada automatski pretvara u relacijski model i kasnije u SQL naredbe za stvaranje objekata u fizičkoj implementaciji, a neki nude mogućnost da se proces izvede u obratnom smjeru. U tom slučaju korisnik može spojiti alat za modeliranje na postojeću bazu podataka i alat će iz sistemskih informacija o objektima izraditi relacijski model koji se kasnije može pretvoriti u ER model po potrebi [21]. U ovom radu nisu korišteni takvi alati iako Oracle nudi *Oracle SQL Developer Data Modeler* programsko rješenje koje je besplatno i kompatibilno s Oracle, kao i IBM Db2 i *Microsoft SQL Server* sustavima za upravljanje bazama podataka.

3.1. Analiza postojećih podataka i stvaranje konceptualnog modela

FERIT za vođenje svog poslovanja koristi vlastiti informacijski sustav MRKVE koji ima nekoliko uloga [22]:

- Izrada rasporeda nastave i ispita
- Planiranje i realizacija nastave
- Evidencija održane nastave
- Matična knjiga djelatnika i vanjskih suradnika
- Evidencija radnog vremena djelatnika

S obzirom da MRKVE predstavlja centralnu bazu podataka o djelatnicima, prostorijama, radnom vremenu i slično, podatke iz tog sustava koriste i druge specifične aplikacije poput aplikacija za inventuru, evidenciju ulazaka i drugih. Iz tog razloga u sustav MRKVE je ugrađeno aplikacijsko programsko sučelje (engl. *Application Programming Interface* – API) koje pruža informacije o sljedećim podatcima u XML formatu:

Tablica 3.1. Popis resursa u MRKVE sustavu

Lokacija resursa (*HTTP endpoint*)	Opis resursa
djelatnici/ vanjski/	Popis djelatnika i vanjskih suradnika
predmeti	**Kolegiji**
predmeti/detaljno.php	Detalji o pojedinom kolegiju
prostorije/	**Popis i informacije o prostorijama**
raspored/	**Raspored nastave i ispita**
raspored/grupa.php	**Popis studenata u nastavnim grupama**
rokovi/	**Prijave ispitnih rokova**
student/	**Popis studenata i osnovni podatci**
tjedni/	**Popis nastavnih i ispitnih tjedana**
zavodi/	Popis organizacijskih jedinica Fakulteta

Od nabrojanih resursa, samo podebljani resursi su od interesa za potrebe izrade baze podataka za prikaz rasporeda. Osnovna polazišna točka za izgradnju modela jest resurs „Raspored nastave i ispita“ jer on predstavlja „središte“ baze podataka i većina potrebnih podataka će se dobivati iz njega. Problem s API resursima MRKVE sustava predstavlja činjenica da za njih nije objavljena takozvana XML Schema (engl. *XML Schema Document* – XSD) kojom se propisuje točna struktura i oblik svih elemenata unutar pojedinog XML dokumenta. Zbog toga vrlo je teško izraditi model sa stopostotnom točnošću jer se prilikom dizajna mogu raditi općenite pretpostavke koje vrijede u velikoj većini slučajeva, ali arhitekt modela, kao ni programeri, nemaju garanciju da su osigurali točnost modela i za rubne slučajeve. U Slici 3.1 je prikazan primjer djelića XML dokumenta dobivenog iz resursa „raspored“:

```xml
<?xml version="1.0" encoding="Windows-1250"?>
<raspored datum="2020-03-02" tjedan="10" tiptjedna="1" rednibrojtjedna="2">
    <!--...-->
    <stavkaRasporeda id="68774" idblok="13289">
        <smjer idsmjer="36">Sveučilišni diplomski studij Računarstvo, izborni blok
Računalno inženjerstvo</smjer>
        <predmet mrkveid="1920077" isvuid="149800" semestar="2" vrsta="0"
sifra="DR2-01">Računalni sustavi stvarnog vremena</predmet>
        <nastavnik mat_broj="[redacted]" domaci="[redacted]" aai="[redacted]@etfos.hr">[redacted]
[redacted]</nastavnik>
        <vrstanastave tip="1">PR - predavanja</vrstanastave>
        <pocetak>08:00</pocetak>
        <kraj>11:15</kraj>
        <prostorija idprostorije="8">K2-11</prostorija>
        <grupastudenata idgrupe="10840">DRA</grupastudenata>
        <dodatniopis/>
        <odradjeno>12</odradjeno>
        <planirano>15</planirano>
    </stavkaRasporeda>
    <!--...-->
</raspored>
```


Slika 3.1. Primjer dostupnih podataka iz resursa "raspored"

Iz gore prikazanog primjera je vidljivo da resurs „raspored“ nudi mnoštvo podataka od kojih neki nisu nužno potrebni poput atributa „idblok“, „semestar“, „tjedan“, „aai“, „vrsta“ itd. Od navedenih podataka može se stvoriti tablicu *EVENT_SCHEDULE* sa sljedećim atributima (stupcima):

Tablica 3.2. Popis stupaca tablice *EVENT_SCHEDULE*

Naziv stupca	Tip podatka	Ime podatka u XML
COURSE	NUMBER	idsmjer
MRKVE_ID	NUMBER	id (stavkaRasporeda)
SUBJECT	VARCHAR2(10 CHAR)	mrkveid
EMPLOYEE	VARCHAR2(5 CHAR)	mat_broj
LECTURE_TYPE	NUMBER	tip (nastave)
START_TIME	TIMESTAMP	datum i pocetak
END_TIME	TIMESTAMP	datum i kraj
GROUPID	NUMBER	idgrupe
ADDITIONAL_INFO	VARCHAR2(80 CHAR)	dodatniopis
DONE	NUMBER	odradjeno
PLANNED	NUMBER	planirano
ROOM	NUMBER	idprostorije
GUID	VARCHAR2(36 CHAR)	/
MIDTERM	NUMBER(1)	/

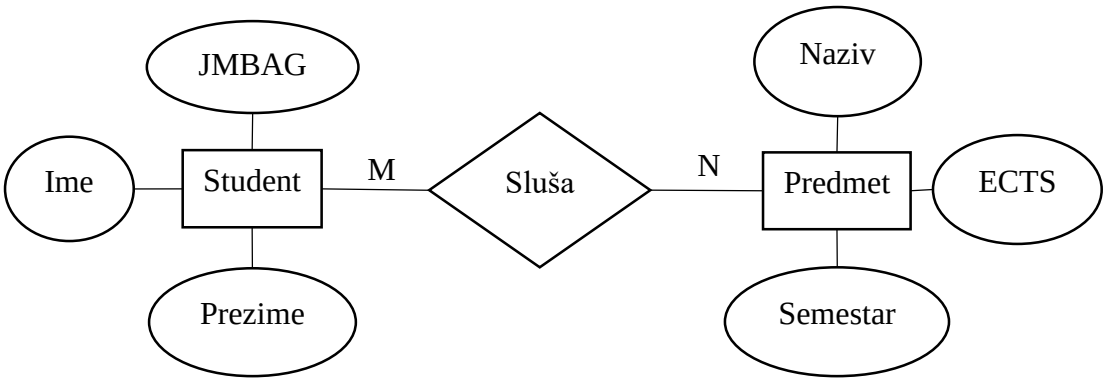
Osim podataka koji su dobiveni iz XML dokumenta, dodana su i dva stupca koji su potrebni radi lakšeg pristupa podatcima; to su *GUID* (engl. *Globally Unique Identification Number*) koji će predstavljati jedinstveni identifikator svakog događaja (i tako služiti kao primarni ključ tablice) i *MIDTERM* koji će imati vrijednost '1' ako je događaj prepoznat kao provjera znanja (kolokvij, kontrolna zadaća), a inače će imati vrijednost '0'. Značajna količina podataka koji se nalaze u resursu se mogu izdvojiti u procesu normalizacije u zasebne tablice koje će sadržavati podatke o nastavnicima, prostorijama, smjerovima, predmetima, tipovima nastave i slično. Proces normalizacije je bitan proces u oblikovanju baze podataka jer se tim procesom uklanjaju duplikati podataka i smanjuje se količina ponavljajućih podataka te se tako poboljšavaju performanse upita i smanjuje potreban prostor za pohranu pojedinih tablica. Cilj normalizacije podataka jest dobivanje tablica koje se nalaze barem u trećoj normalnoj formi (3NF). Tablica se nalazi u 3NF kada su ispunjeni svi sljedeći uvjeti [23]:

- Svi neključni atributi su funkcijski ovisni o primarnom ključu – postiže se uklanjanjem ponavljajućih vrijednosti iz tablice
- Svi neključni atributi su **potpuno** funkcijski ovisni o ključu – postiže se uklanjanjem stupaca (atributa) koji su ovisni samo o dijelu primarnog ključa
- Niti jedan neključni atribut nije tranzitivno ovisan o ključu – postiže se uklanjanjem stupaca (atributa) ovisnih o atributima koji nisu dio primarnog ključa

Ovakav proces analize, specifikacije podataka i normalizacije predloženih tablica je ponovljen za svaki resurs prije pristupanja izradi ER dijagrama.

3.2. Stvaranje modela

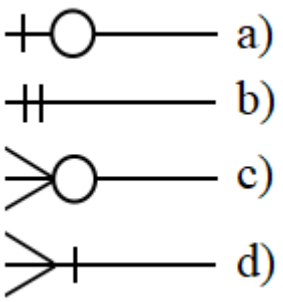
ER dijagram (engl. *Entity-relationship diagram*) jest grafički dijagram koji pokazuje relacije (odnose, veze) između entiteta (tablica) u bazi podataka [24]. On nastaje kao rezultat sustavne analize pojedinih procesa i sustava kako bi se definiralo koje su informacije ključne za praćenje procesa i funkcioniranje sustava. Najčešće se crta u Chenovoj notaciji gdje su entiteti predstavljeni pravokutnicima, veze rombovima, a atributi entiteta su predstavljeni elipsama koje su pridružene pojedinom atributu. Primjer ER dijagrama koji prati Chenovu notaciju je prikazan na Slici 3.2:



Slika 3.2. Primjer jednostavnog ER dijagrama

Gornji primjer je dobar za slučajeve kada broj entiteta i međusobnih veza među njima nije velik i kada entiteti nemaju puno atributa, dok se u ostalim slučajevima često pristupa izradi ER dijagrama u notaciji jedinstvenog jezika za modeliranje (engl. *Unified Modelling Language* – UML) jer ona omogućuje puno kompaktniji prikaz svih dijelova modela. U UML dijagramima svi atributi jednog entiteta su sadržani zajedno, izlistani ispod naziva entiteta, dok se veze predstavljaju običnim crtama koje na svojim krajevima imaju sa svake strane naznačen odnos omjera među entitetima (kardinalnost veze) putem brojeva i slovnih oznaka ili pomoću posebnih znakova na krajevima linija što je poznato kao notacija vranina stopala (engl. *Crow's foot notation*). U notaciji vranina stopala, koju je izmislio Gordon Everest [25], atributi koji pripadaju samoj vezi kao takvoj se ne označavaju na vezi nego se veza transformira u zasebni entitet koji ima veze sa originalnim entitetima, kao što je i slučaj za veze s omjerom više-na-više ($M:N$). U Oracleovim alatima te posebne oznake koje se koriste za označavanje kardinalnosti, prikazane na Slici 3.3 imaju svoja značenja kako slijedi:

- prsten i crtica – neobavezno članstvo; minimalno 0, maksimalno 1 entitet sudjeluje u vezi (Slika 3.3.a)
- crtica i crtica – obavezno članstvo; minimalno 1, maksimalno 1 entitet sudjeluje u vezi (Slika 3.3.b)
- prsten i vranino stopalo – neobavezno članstvo; minimalno 0, maksimalno N entiteta sudjeluje u vezi (Slika 3.3.c)
- crtica i vranino stopalo – obavezno članstvo; minimalno 1, maksimalno N entiteta sudjeluje u vezi (Slika 3.3.d)



Slika 3.3. Vizualizacija oznaka u notaciji vranina stopala

Osim oznaka na krajevima veze, nekada i sami tip crte na dijagramu govori o prirodi veze pa tako puna crta simbolizira da entitet s jedne strane ne može postojati bez drugog entiteta (postoji *NOT NULL* ograničenje nad stupcem koji je definiran kao strani ključ), dok isprekidana crta simbolizira da entitet može postojati bez drugog entiteta. Imajući sve navedene upute o stvaranju ER dijagrama, kao i informacije o svim dostupnim entitetima i njihovim atributima, idući korak jest izrada ER dijagrama. Na DVD-u koji je sastavni dio ovog završnog rada se nalazi potpuni ER dijagram Studentskog rasporeda u PDF formatu.

Nakon dizajniranja ER modela, slijedi pretvaranje ER modela u relacijski model. On je vrlo sličan ER modelu, ali u njemu neki detalji fizičke implementacije postaju vidljivi poput stupaca koji će biti strani ključevi, tipova podataka za atribute, svih vrsta ograničenja i sličnog. U procesu pretvorbe ER modela u relacijski model najveću pažnju je važno posvetiti pretvorbi veza u odgovarajuće strane ključeve i po potrebi nove entitete. Pravila za transformaciju veza su kako slijedi [26]:

- Svaki entitet postaje jedna tablica, a tablica ima isto ime kao i entitet
- Svaki atribut postaje jedan stupac tablice
- Za svaku tablicu potrebno je odabrati primarni ključ koji mora osigurati da su njegovi elementi jedinstveni i da su minimalan skup koji osigurava jedinstvenost
- Za veze s kardinalnosti 1:1 postoje različita pravila implementacije ovisno o obveznosti članstva:
    - Ako su oba članstva obavezna, dva entiteta se spajaju u jednu zajedničku tablicu i imaju zajednički primarni ključ
    - Ako nijedno članstvo nije obavezno, kreira se dodatna tablica s primarnim ključevima obje tablice
    - Ako je samo jedno članstvo obavezno, u entitetu u kojem se nalazi obavezno članstvo dodaje se strani ključ entiteta koji nema obavezno članstvo

- Kod veza s kardinalnosti 1:N na strani entiteta s N kardinalnosti se dodaje strani ključ koji se veže na primarni ključ entiteta s kardinalnosti 1
- Kod veza s kardinalnosti M:N stvara se dodatni entitet čiji kompozitni primarni ključ se sastoji od primarnih ključeva oba entiteta
    - U novom entitetu, osim kompozitnog primarnog ključa, stvaraju se i strani ključevi na primarne ključeve početnih entiteta
    - Ako sama veza ima dodatne atribute, oni se dodaju u novonastali entitet

Primjena ovih pravila osigurava da se podatci vjerodostojno repliciraju iz ER modela u fizičku implementaciju što u konačnici dovodi do osiguranja konzistentnosti podataka i normalizacije tablica. Na DVD-u koji je sastavni dio ovog završnog rada se nalazi u PDF formatu potpuni relacijski model koji je nastao kao rezultat primjene pravila transformacije na ER model.

3.3. Fizičko oblikovanje baze podataka

Nakon dizajniranja relacijskom modela, jedino što preostaje jest napisati SQL naredbe za definiciju objekata unutar baze podataka (engl. *Data definition language* – DDL). Taj postupak uvelike ovisi o mogućnostima i ograničenjima pojedinog SUBP-a jer nemaju svi sustavi iste mogućnosti niti istu sintaksu za iste mogućnosti. Prilikom fizičkog oblikovanja najčešće korištene SQL DDL naredbe su *CREATE TABLE* i *ALTER TABLE*. One korisniku omogućuju da stvori novu tablicu (entitet) i da ju po potrebi kasnije izmjeni na način da doda, ukloni ili promijeni ograničenja ili stupce (atribute). Važno je naglasiti da te dvije naredbe ne mijenjaju sadržaj podataka unutar tablice, nego samo strukturu pojedine tablice. Općenita sintaksa za *CREATE TABLE* naredbu je prikazana u Slici 3.4 dok je primjer *CREATE TABLE* naredbe prikazan u Slici 3.5 na primjeru tablice *EVENT_SCHEDULE*.

```
create_table ::= CREATE TABLE [schema.] table_name (relational_properties);

relational_properties ::= {column_definition|out_of_line_constraint} [,...]

column_definition ::= column_name data_type [VISIBLE|INVISIBLE] [DEFAULT expr]
[{inline_constraint}...]

inline_constraint ::= [CONSTRAINT constraint_name] {[NOT] NULL|UNIQUE|PRIMARY
KEY|references_clause|CHECK (condition)}

references_clause ::= REFERENCES [schema.] object [(column [,column]...)] [ON
DELETE {CASCADE|SET NULL}]

out_of_line_constraint ::= [CONSTRAINT constraint_name] {UNIQUE (column
[,column]...) |PRIMARY KEY (column [,column]...)|FOREIGN KEY (column [,column]...)
references_clause|CHECK (condition)}
```

Slika 3.4. Sintaksa *CREATE TABLE* naredbe u Backus-Naurovoj formi, izvor: [27]

```
CREATE TABLE EVENT_SCHEDULE
   (COURSE           NUMBER
                     NOT NULL,
    SUBJECT          VARCHAR2(10)
                     NOT NULL,
    EMPLOYEE         VARCHAR2(5),
    LECTURE_TYPE     NUMBER
                     NOT NULL,
    START_TIME       TIMESTAMP
                     NOT NULL,
    END_TIME         TIMESTAMP
                     NOT NULL,
    GROUPID          NUMBER
                     NOT NULL,
    ADDITIONAL_INFO VARCHAR2(80),
    DONE             NUMBER,
    PLANNED          NUMBER,
    ROOM             NUMBER
                     NOT NULL,
    GUID             VARCHAR2(36)
                     DEFAULT REGEXP_REPLACE(RAWTOHEX(SYS_GUID()),
                                            '([:xdigit:]{8})'
                                             ||'([:xdigit:]{4})'||'([:xdigit:]{4})'
                                             ||'([:xdigit:]{4})'||'([:xdigit:]{12})',
                                            '\1-\2-\3-\4-\5')
                     PRIMARY KEY,
    MIDTERM          NUMBER(1)
                     DEFAULT 0
                     NOT NULL,
    MRKVE_ID         NUMBER
                     NOT NULL,
    CONSTRAINT EVENT_SCHEDULE_CHK1
        CHECK (DONE >= 0),
    CONSTRAINT EVENT_SCHEDULE_CHK2
        CHECK (PLANNED >= 0),
    CONSTRAINT EVENT_SCHEDULE_CHK3
        CHECK (MIDTERM IN (0, 1)),
    CONSTRAINT EVENT_SCHEDULE_CHK4
        CHECK (START TIME < END TIME),
    CONSTRAINT EVENT_SCHEDULE_UQ
        UNIQUE (COURSE, MRKVE_ID, GROUPID);
    CONSTRAINT EVENT_SCHEDULE_FK1_COURSES
        FOREIGN KEY (COURSE)
        REFERENCES COURSES (ID)
        ON DELETE SET NULL,
    CONSTRAINT EVENT_SCHEDULE_FK2_EMPLOYEES
        FOREIGN KEY (EMPLOYEE)
        REFERENCES EMPLOYEES (ID)
        ON DELETE SET NULL,
    CONSTRAINT EVENT_SCHEDULE_FK3_SUBJECTS
        FOREIGN KEY (SUBJECT, COURSE)
        REFERENCES SUBJECTS (SUBJECT, COURSE)
        ON DELETE SET NULL,
    CONSTRAINT EVENT_SCHEDULE_FK4_LECTURE_TYPE
        FOREIGN KEY (LECTURE TYPE)
        REFERENCES LECTURE_TYPES (ID)
        ON DELETE SET NULL,
    CONSTRAINT EVENT_SCHEDULE_FK5_GROUPS
        FOREIGN KEY (GROUPID)
        REFERENCES GROUPS (GROUPID)
        ON DELETE SET NULL,
    CONSTRAINT EVENT_SCHEDULE_FK6_ROOMS
        FOREIGN KEY (ROOM)
        REFERENCES ROOMS (ROOM)
        ON DELETE SET NULL);
```

Slika 3.5. SQL naredba za stvaranje tablice *EVENT_SCHEDULE*

Kao što je vidljivo u Slici 3.5, prilikom navođenja definicija stupaca i ograničenja programer ne mora upotrijebiti sve dijelove sintakse (navedene unutar uglatih zagrada u Slici 3.4), dok neke može ponoviti onoliko puta koliko su mu potrebni. Osim gore navedenih dijelova sintakse, programeri mogu iskoristiti i mnoge dijelove sintakse koji nisu navedeni u Slici 3.4 kako bi preciznije odredili svojstva i ponašanje tablica. Analogno primjeru za kreiranje tablice *EVENT_SCHEDULE*, postupak se ponavlja za sve tablice iz relacijskog modela s odgovarajućim stupcima i ograničenjima. Osim stvaranja tablica i pratećih ograničenja, često je potrebno stvoriti i dodatne objekte u bazi podataka koji na neki način olakšavaju ili ubrzavaju rad s podatcima poput pogleda, indeksa i okidača. Pogled jest pohranjeni SQL upit koji vraća određeni skup podataka iz jedne ili više tablica na jednostavan način. Prednost pogleda jest to što skrivaju složenost podataka od drugih dijelova sustava, programeri im mogu pristupati kao i svim drugim običnim tablicama u bazi podataka, iako se u pozadini pogleda može nalaziti *SELECT* upit koji pristupa mnoštvu tablica i radi različite vrste sortiranja, filtriranja i agregacija nad podatcima [28]. Sintaksa za stvaranje pogleda pomoću naredbe *CREATE VIEW* dana je u Slici 3.6:

```
create_view ::= CREATE [OR REPLACE] VIEW [schema.] view_name [({alias
[VISIBLE|INVISIBLE] [inline_constraint...]|out_of_line_constraint}[,...])] AS
select_query ;
```

Slika 3.6. Sintaksa naredbe *CREATE VIEW* naredbe u Backus-Naurovoj formi, izvor: [27]

Prilikom stvaranja pogleda programer može, a i ne mora unaprijed definirati stupce koje će pogled prikazivati. Ako ih programer ne navede, SUBP će automatski odrediti tip i naziv stupaca. Greška će nastati ako se u pogledu vraća stupac za kojeg nije definiran alias unutar *SELECT* upita, a ni unutar zagrada u naredbi *CREATE VIEW*. Primjer *CREATE VIEW* naredbe za stvaranje pogleda koji prikazuje sve ne-ispitne događaje u rasporedu za sve studente u bazi je dan u Slici 3.7:

```
CREATE OR REPLACE VIEW CLASSES_FOR_STUDENT AS
SELECT A.GUID                       AS EVENT_GUID,
       A.START_TIME                 AS START_TIME,
       A.END_TIME                   AS END_TIME,
       E.NAME                       AS LECTURE_TYPE,
       D.NAME                       AS TEACHER_NAME,
       DECODE(A.MIDTERM,
              1, 'YES',
              0, 'NO')              AS MIDTERM,
       F.NAME                       AS ROOM_NAME,
       A.DONE || '/' || A.PLANNED AS
DONE_PLANNED_HOURS,
       G.NAME                       AS GROUP_NAME,
       A.ADDITIONAL_INFO            AS ADDITIONAL_INFO,
       C.NAME                       AS SUBJECT_NAME,
       B.JMBAG
  FROM EVENT_SCHEDULE A
       INNER JOIN STUDENTS_GROUPS B
               ON A.GROUPID = B.GROUPID
       INNER JOIN SUBJECTS C
               ON A.SUBJECT = C.SUBJECT
              AND A.COURSE = C.COURSE
       LEFT  JOIN EMPLOYEES D
               ON A.EMPLOYEE = D.ID
       INNER JOIN LECTURE_TYPES E
               ON A.LECTURE_TYPE = E.ID
       INNER JOIN ROOMS F
               ON A.ROOM = F.ROOM
       INNER JOIN GROUPS G
               ON A.GROUPID = G.GROUPID
 WHERE A.LECTURE_TYPE <> 7;
```

Slika 3.7. SQL naredba za stvaranje pogleda *CLASSES_FOR_STUDENT*

Pogled koji je kreiran s naredbom iz Slike 3.7 programer kasnije može koristiti u svim *SELECT* upitima i raditi sve vrste sortiranja, filtriranja i agregacija nad dobivenim podatcima iz pogleda kao da je riječ o običnoj tablici. Ovaj pogled će vratiti individualizirane rasporede za sve studente pa tako programer može dodatno filtrirati podatke u novom upitu tako da dobije individualni raspored za jednog studenta:

```
   SELECT *
     FROM CLASSES_FOR_STUDENT
    WHERE JMBAG = '[redacted]'
 ORDER BY START_TIME DESC
    FETCH FIRST 10 ROWS ONLY;
```

Slika 3.8. *SELECT* upit za dobivanje dijela individualnog rasporeda za pojedinog studenta

Izvođenjem upita iz Slike 3.8 SUBP će pokrenuti upit zapisan u definiciji pogleda *CLASSES_FOR_STUDENT*, zatim izdvojiti retke koji zadovoljavaju uvjet da je JMBAG istovjetan određenoj vrijednosti, potom sortirati filtrirane retke tako da najnoviji nastavni sati budu na vrhu i na kraju korisniku vratiti samo prvih 10 redaka. Krajnji rezultat je prikazan u Tablici 3.3:

	SUBJECT	START_TIME	END_TIME	LECTURE_TYPE	ROOM
1	Engleski jezik III	05.06.2020 11:00	05.06.2020 13:00	auditorne vježbe	2-31
2	Ekonomika poduzeća	01.06.2020 08:00	01.06.2020 10:00	auditorne vježbe	2-31
3	Modeliranje i simulacija	25.05.2020 08:00	25.05.2020 09:30	laboratorijske vježbe	K1-1
4	Modeliranje i simulacija	11.05.2020 08:00	11.05.2020 09:30	laboratorijske vježbe	K+1
5	Ekonomika poduzeća	30.04.2020 09:45	30.04.2020 11:15	auditorne vježbe	2-31
6	Ekonomika poduzeća	30.04.2020 08:00	30.04.2020 09:30	predavanja	2-31
7	Modeliranje i simulacija	29.04.2020 08:00	29.04.2020 09:30	laboratorijske vježbe	K3-1
8	Ekonomika poduzeća	28.04.2020 13:15	28.04.2020 14:45	auditorne vježbe	2-31
9	Ekonomika poduzeća	28.04.2020 11:30	28.04.2020 13:00	predavanja	2-31
10	Ekonomika poduzeća	23.04.2020 09:45	23.04.2020 11:15	auditorne vježbe	2-31

Slika 3.9. Rezultati upita iz Slike 3.8

Osim pogleda, prilikom fizičkog i relacijskog modeliranja programeri i arhitekti trebaju posvetiti pažnju stvaranju odgovarajućih indeksa nad stupcima tablica. Indeks je podatkovna struktura koja služi bržem dohvatu i pretrazi podataka u tablici. Oni u radnoj memoriji drže pohranjenu kopiju podataka iz pojedinih stupaca za koje arhitekti i programeri procjene da se često koriste u pretrazi podataka i na taj način indeksi čitanjem podataka iz RAM-a umjesto s tvrdog diska uvelike ubrzavaju vrijeme potrebno za izvođenje upita. Prilikom stvaranja indeksa potrebno je voditi o računa o tome hoće li indeks doprinijeti ubrzavanju upita. To znači da arhitekti ne trebaju stvarati indekse nad stupcima koji se brzo pretražuju bez indeksa ili se rijetko koriste, kao niti nad stupcima koji ne mogu biti u cijelosti pohranjeni u memoriji jer ako se indeks nad velikim stupcem zbog svoje veličine pohrani i čita s diska sve uštede na vremenu izvođenja se anuliraju [29]. Sintaksa za stvaranje indeksa je prikazana u Slici 3.10 dok je primjer naredbe za stvaranje indeksa dan u Slici 3.11.

```
create_index ::= CREATE [UNIQUE] INDEX [schema.] index_name ON table_index_clause;

table_index_clause ::= [schema.] table_name [table_alias]
({column|column_expression} [ASC|DESC] [, {column|column_expression}
[ASC|DESC]]...)
```

Slika 3.10. Sintaksa naredbe *CREATE INDEX* naredbe u Backus-Naurovoj formi, izvor: [27]

```
CREATE INDEX EVENT_SCHEDULE_IX1
    ON EVENT_SCHEDULE(GROUPID, SYS_EXTRACT_UTC(START_TIME));
```

Slika 3.11. Naredba za stvaranje indeksa nad stupcem i izrazom nad stupcem iz *EVENT_SCHEDULE* tablice

Većina SUBP-a automatski prilikom kreiranja tablice stvori i indeks nad stupcima koji tvore primarni ključ jer jedinstveni indeks osigurava jedinstvenost primarnog ključa i pretpostavka je da će se primarni ključ najčešće koristiti kao uvjet u upitima. Neki SUBP-i novih generacija (tzv. autonomne baze podataka) mogu automatski samostalno s vremenom procijeniti koje dodatne indekse bi trebalo kreirati kako bi se poboljšalo vrijeme izvođenja dugotrajnih upita [30].

Prilikom analize procesa i sustava često se može dogoditi da pojedini korak u procesu ili događaj u sustavu zahtijeva složenije postupanje od onog što se može dobiti s SQL DML (engl. *Data Manipulation Language*) naredbama poput *SELECT*, *INSERT*, *UPDATE* i *DELETE*. Nekada izmjena podataka u jednoj tablici treba automatski povlačiti izmjenu podataka u drugim tablicama ili ponovni izračun i evaluaciju određenih vrijednosti koje se ne mogu izračunati unutar SQL upita. Tada se koriste objekti baze podataka poznati kao okidači. Okidači su programske jedinice koje se pokreću svaki puta kada se dogodi određeni događaj nad nekim objektom u bazi podataka, najčešće je to tablica ili pogled, dok su događaji uglavnom umetanje, izmjena ili brisanje redaka unutar tablica, iako mogu biti i drugi događaji kao prijava i odjava korisnika ili izmjena strukture tablica i slično. Arhitekti i programeri imaju mogućnost birati hoće li se programski kod unutar okidača izvršiti prije, poslije ili umjesto događaja koji ga je pozvao i unutar programskog koda mogu na taj način izmijeniti vrijednosti prije nego se zapišu u bazu podataka ili ih iskoristiti za neke druge postupke [28]. Zbog svojih mogućnosti, okidači se najčešće koriste za očuvanje integriteta podataka i pišu se u proceduralnom jeziku kojeg nudi pojedini SUBP. Sintaksa za stvaranje okidača ovisi od sustava do sustava jer ne nude svi SUBP-i jednake mogućnosti niti isti proceduralni jezik. Osnovna sintaksa za stvaranje okidača nad tablicom u Oracle bazi podataka je prikazana na Slici 3.12:

```
CREATE [OR REPLACE] TRIGGER [schema.] trigger_name
    {BEFORE|AFTER} {DELETE|INSERT|UPDATE [OF column [, column]...]}
    [OR {DELETE|INSERT|UPDATE [OF column [, column]...]}...]
    ON [schema.] table_name
    [REFERENCING { OLD [ AS ] old | NEW [ AS ] new | PARENT [ AS ]
parent}...]
    [FOR EACH ROW]
    [WHEN (condition)]
    plsql_block
```

Slika 3.12. Sintaksa za *CREATE TRIGGER* naredbu u Backus-Naurovoj formi, izvor: [27]

Ako se okidač izvršava prije same izmjene podataka (umetanja, promjene ili brisanja) programski kod može pročitati i po potrebi izmijeniti vrijednosti koje će se na kraju upisati u tablice. Osim toga, kako bi se spriječilo nepotrebno pokretanje okidača, osim što se navodi za koje događaje se okidač pokreće, može se navesti i koji uvjeti za podatke moraju vrijediti kako bi se okidač

pokrenuo (npr. okidač se pokreće samo ako je povećanje neke vrijednosti veće od određene granice i sl.). Primjer jednostavnog okidača koji se izvršava prije umetanja ili promjene podataka i po potrebi mijenja podatke prije zapisa u tablicu je dan u Slici 3.13:

```
CREATE OR REPLACE TRIGGER TRG_EVENT_SCHEDULE_ADD_MIDTERM
    BEFORE INSERT OR UPDATE
    ON EVENT_SCHEDULE
    FOR EACH ROW
BEGIN
    :NEW.MIDTERM := CASE
                        WHEN UPPER(:NEW.ADDITIONAL_INFO) LIKE '%ISPIT%'
                            THEN 1
                        WHEN UPPER(:NEW.ADDITIONAL_INFO) LIKE '%K%L%K%V%'
                            THEN 1
                        WHEN UPPER(:NEW.ADDITIONAL_INFO) LIKE '%K%T%R%L%Z%'
                            THEN 1
                        WHEN UPPER(:NEW.ADDITIONAL_INFO) LIKE '%PREDROK%'
                            THEN 1
                        ELSE 0
                    END;
END;
```

Slika 3.13. Okidač nad *EVENT_SCHEDULE* tablicom za određivanje provjera znanja

Funkcionalnost okidača iz Slike 3.13 se mogla implementirati unutar procesa koji popunjavaju podatke u tablicu *EVENT_SCHEDULE*, ali korištenjem okidača postiže se veći stupanj pouzdanosti da će se vrijednost ispravno ažurirati čak i ako dođe do ručne ili izmjene podataka koju je napravio neki drugi proces.

4. IZRADA POZADINSKIH PROCESA U PL/SQL JEZIKU

Nakon modeliranja baze podataka, potrebno je osigurati da u bazu podataka budu spremljeni aktualni i točni podatci za kasniji prikaz i obradu. U pravilu podatke u bazu spremaju korisnici dok pristupaju bazi podataka direktno ili putem klijentskih aplikacija ili se podatci određenim automatizmom uvoze iz vanjskih izvora koji mogu biti druge baze podataka, datoteke, sučelja, sustavi, vanjski resursi, senzori i slično. Za uvoz podataka automatizmom u pravilu su zaduženi procesi ili servisi na poslužiteljima koji mogu biti pisani u bilo kojem jeziku za kojeg postoje pogonski programi za komunikaciju s odredišnom bazom podataka. Oni se mogu pokretati periodički u pravilnim vremenskim intervalima ili kao reakcija na neki vanjski događaj. Česti problem prilikom automatiziranog uvoza podataka jest sporost uvoza uzrokovana velikom količinom podataka. Sporost često nastaje zbog loše napisanih upita prema bazi podataka i mrežne ili međuprocesne latencije u komunikaciji između procesa i baze podataka. Zbog toga često je uputno sve dijelove aplikativne i poslovne logike koji obrađuju velike količine podataka napisati i izvršavati direktno unutar baze podataka kako bi ugrađeni alat za optimizaciju upita unutar SUBP-a maksimalno poboljšao upite i kako bi se skratilo vrijeme potrebno za prijenos podataka izvan baze podataka u logičkim međukoracima.

Neki SUBP-i imaju visoko razvijene proceduralne jezike ili nude mogućnost pokretanja programskih jedinica pisanih u drugim popularnim programskim jezicima poput Jave ili C i C++ direktno unutar istog procesa kao i ostatak baze podataka kako bi se postigle uštede vremena i resursa pri obradi podataka. Oracle je u tu svrhu razvio PL/SQL proceduralni programski jezik koji ima visoke performanse uz visok stupanj prenosivosti između hardverskih platformi. Jezik je po svojoj sintaksi i strukturi sličan jeziku Ada prema kojem je i dizajniran. Osim toga, kao Ada i Pascal, PL/SQL ne razlikuje velika i mala slova, ima podršku za iznimke, podržava klasične petlje i naredbe za grananje programa, ima ugrađenu podršku za varijable i konstante i programski kod je strukturiran u blokove. Također PL/SQL nudi visoku razinu povezanosti s SQL-om tako da programer može koristiti sve SQL tipove podataka bez potrebe za pretvorbom i svi SQL upiti se izvršavaju direktno unutar koda bez potrebe za posebnim pozivima funkcija za komunikaciju s bazom podataka. Rezultati pojedinih upita se mogu obrađivati direktno unutar PL/SQL blokova, a PL/SQL funkcije se mogu koristiti unutar SQL upita dok god povratna vrijednost i argumenti funkcije su tip podatka koji SQL podržava [11].

Kod se može strukturirati u anonimne blokove koji se ne pohranjuju i služe za jednokratnu upotrebu ili u obliku funkcija i procedura. Razlika između procedura i funkcija je u tome što funkcije imaju povratnu vrijednost, dok procedure nemaju povratnu vrijednost (ekvivalent

PL/SQL procedurama u C programskom jeziku su funkcije s povratnim tipom *void*). Osim funkcija i procedura kod se može pohraniti u prethodno spominjanim okidačima i u objektima zvanim paketi [11].

Paketi su objekti unutar baze podataka koji služe za logičko grupiranje potprograma, konstanti i varijabli, a sastoje se od specifikacije i tijela paketa. Specifikacija paketa predstavlja sučelje prema paketu koje samo deklarira (bez implementiranja koda) sve vrste korisničkih tipova podataka, iznimki, kursora, varijabli, konstanti i potprograma (funkcija i procedura). Svi objekti deklarirani unutar specifikacije su javni jer im svatko može pristupiti van paketa. Tijelo paketa sadrži sami programski kod za sve objekte navedene u specifikaciji i može sadržavati privatne objekte koje mogu pozivati i koristiti samo objekti unutar tog istog tijela paketa [11].

Najbliži ekvivalent paketima u drugim programskim jezicima jest princip razdvajanja koda u C programskom jeziku u zaglavnu (engl. *header*, nastavak „.h“) i programsku datoteku (nastavak „.c“). U tom slučaju specifikacija paketa se može poistovjetiti s zaglavnom datotekom, a tijelo paketa s programskom datotekom koja sadrži implementacije svih deklaracija iz zaglavlja. Drugi koncept sličan paketima jesu klase u C++ jeziku. U takvoj usporedbi, specifikacija paketa je jednaka klasi koja u početnoj deklaraciji ne sadrži implementacije potprograma već samo deklaracije. Tada sve što je sadržano u specifikaciji paketa bi bilo deklarirano kao javna statična metoda ili varijabla unutar C++ klase, dok bi objekti koji se nalaze samo u tijelu paketa bili deklarirani kao privatni. Programski kod iz tijela paketa bi u C++ bio sadržan izvan deklaracije klase.

Za razliku od drugih programskih jezika, PL/SQL nije objektni jezik iako ima neke funkcionalnosti koje su slične objektnim jezicima. Sve varijable koje se koriste u PL/SQL se deklariraju prije izvršnog dijela bloka u *DECLARE* odjeljku. Zatim slijedi izvršni dio koji započinje ključnom riječju *BEGIN*, a završava riječju *END*. Izvršni dio u sebi može sadržavati blok za obradu iznimki koji dolazi na kraju prije ključne riječi *END* i počinje s riječi *EXCEPTION*. Naredbe se u PL/SQL-u završavaju znakom točka-zarez, a blokovi se mogu ugnježđivati [11]. Struktura PL/SQL bloka je prikazana na Slici 4.1:

```
<< oznaka>> (neobavezno)
DECLARE    -- Odjeljak za deklaracije (neobavezno)
  -- Deklaracije lokalnih korisničkih tipova podataka, varijabli i
potprograma

BEGIN      -- Izvršni odjeljak (obavezan)
  -- Naredbe i izjave (mogu koristiti varijable iz prethodnog odjeljka)

EXCEPTION -- Odjeljak za rukovanje iznimkama (neobavezno)
  -- Rukovatelji za iznimke koje mogu nastati izvršavanjem naredbi u
izvršnom dijelu bloka
END;
```

Slika 4.1. Opis dijelova PL/SQL programskog bloka, izvor: [11]

4.1. Procedure za preuzimanje i obradu podataka iz MRKVE sustava

Za razliku od proceduralnih SQL proširenja za druge SUBP-e poput PostgreSQL, MySQL i MariaDB, PL/SQL ima mnogo ugrađenih knjižnica (engl. *library*) za izvršavanje različitih često korištenih funkcionalnosti poput rukovanja datotekama van baze podataka, slanje e-mailova, komunikaciju putem HTTP(S) protokola, obradu XML i JSON (engl. *JavaScript Object Notation*) datoteka i slično [31]. Zbog toga programeri mogu u PL/SQL-u pisati i dijelove aplikativne logike koji bi se inače morali pisati u drugim višim programskim jezicima poput Jave ili C#. U ovom završnom radu svi pozadinski servisi koji periodički preuzimaju podatke iz drugih sustava i izlažu aplikacijska programska sučelja prema drugim korisnicima su napisani u PL/SQL kako bi se pojednostavio dizajn sustava i povećala razina prenosivosti između platformi. U Slici 4.2 i Slici 4.3 su prikazane procedure koje preuzimaju podatke s HTTP poslužitelja i obrađuju primljene podatke o predmetima i pohranjuju ih u tablicu *SUBJECTS*. Na analogan način su stvorene i procedure za obradu ostalih resursa koje se pokreću periodički svaki dan u ponoć putem ugrađenog raspoređivača (engl. *scheduler*) unutar SUBP-a.

```
FUNCTION DO_HTTP_REQUEST (P_URL IN VARCHAR2) RETURN BLOB IS
    L_RESPONSE_BODY     BLOB;
    L_HTTP_REQUEST      UTL_HTTP.REQ;
    L_HTTP_RESPONSE     UTL_HTTP.RESP;
    L_RAW               RAW(32767);
    L_AMOUNT            NUMBER := 32767;
BEGIN
    DBMS_LOB.CREATETEMPORARY(L_RESPONSE_BODY, FALSE);

    UTL_HTTP.SET_TRANSFER_TIMEOUT(600);

    L_HTTP_REQUEST  := UTL_HTTP.BEGIN_REQUEST(P_URL);
    L_HTTP_RESPONSE := UTL_HTTP.GET_RESPONSE(L_HTTP_REQUEST);

    BEGIN
        LOOP
            UTL_HTTP.READ_RAW(L_HTTP_RESPONSE, DATA => L_RAW, L_AMOUNT);

            DBMS_LOB.WRITEAPPEND(L_RESPONSE_BODY, UTL_RAW.LENGTH(L_RAW), L_RAW);
        END LOOP;
    EXCEPTION
        WHEN UTL_HTTP.END_OF_BODY THEN
            UTL_HTTP.END_RESPONSE(L_HTTP_RESPONSE);
    END;

    RETURN L_RESPONSE_BODY;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(UTL_HTTP.GET_DETAILED_SQLERRM);
        DBMS_LOB.FREETEMPORARY(L_RESPONSE_BODY);
        UTL_HTTP.END_RESPONSE(L_HTTP_RESPONSE);
        RAISE;
END DO_HTTP_REQUEST;
```

Slika 4.2. Programski kod funkcije za dohvat HTTP resursa

U liniji 8 funkcije se zasebno zauzima prostor za BLOB (engl. *Binary large object*) varijablu jer zbog njihove maksimalne veličine nije ih dovoljno deklarirati u *DECLARE* bloku. Prilikom preuzimanja velikih datoteka može doći do nenamjernog *timeout*-a prilikom čekanja poslužiteljskog odgovora pa se zato on u liniji 10 postavlja na 600 sekundi (5 minuta). Zatim se otvara veza prema udaljenom poslužitelju i čeka se na njegov odgovor. Kada se dobije odgovor od druge strane pokreće se petlja koja preuzima podatke u manjim komadima od 32 767 bajtova i dopisuje ih na kraj *L_RESPONSE_BODY* varijable sve dok ne dođe do iznimke da je dosegnut kraj odgovora i da nema više podataka. Na kraju unutar *EXCEPTION* bloka je dodano rukovanje za sve iznimke osim *END_OF_BODY* iznimke. Ako dođe do nepredviđene iznimke, ispisat će se tekst greške na konzolu, osloboditi *L_RESPONSE_BODY* varijabla i prekinuti veza prema poslužitelju prije prosljeđivanja iznimke pozivajućem programu ili krajnjem korisniku.

```
PROCEDURE PARSE_SUBJECTS IS
    L_URL       VARCHAR2(200);
    L_HTTP_BODY BLOB;
    XMLDATA     XMLTYPE;
BEGIN
    L_URL := FILIP_UTIL.GET_SETTING('MRKVE_API_ROOT_URI')
             || 'predmeti';
    L_HTTP_BODY := DO_HTTP_REQUEST(L_URL);
    L_XML_DOC   := XMLTYPE(L_HTTP_BODY, 0);

    MERGE INTO SUBJECTS S
        USING (SELECT DISTINCT
                       SUBJECT.SUBJECT  AS SUBJECT,
                       SUBJECT.SEMESTER AS SEMESTER,
                       SUBJECT.TYPE     AS TYPE,
                       SUBJECT.CODE     AS CODE,
                       SUBJECT.NAME     AS NAME,
                       SUBJECT.COURSE   AS COURSE
               FROM XMLTABLE('/raspored/stavkaRasporeda/predmet'
                             PASSING XMLDATA
                             COLUMNS
                              SUBJECT  VARCHAR2(10)  PATH '@mrkveid',
                              SEMESTER NUMBER        PATH '@semestar',
                              TYPE     NUMBER        PATH '@vrsta',
                              CODE     VARCHAR2(20)  PATH '@sifra',
                              NAME     VARCHAR2(100) PATH '/',
                              COURSE   NUMBER        PATH './../smjer/@idsmjer')
                       SUBJECT) X
        ON (S.SUBJECT    = X.SUBJECT
            AND S.COURSE = X.COURSE)
        WHEN NOT MATCHED THEN
            INSERT (SUBJECT, CODE, NAME, TYPE, COURSE, SEMESTER)
            VALUES (X.SUBJECT, X.CODE, X.NAME, X.TYPE, X.COURSE, X.SEMESTER)
        LOG ERRORS (TO_CHAR(SYSDATE, 'YYYY-MM-DD'))
        REJECT LIMIT UNLIMITED;
    COMMIT;
END PARSE_SUBJECTS;
```

Slika 4.3. Programski kod procedure za obradu XML dokumenta o kolegijima

Procedura PARSE_SUBJECTS prvo konstruira URL (engl. *Uniform Resource Locato*r) na kojem se nalazi odgovarajući resurs MRKVE sustava. Zbog olakšavanja izmjene i zaštite lokacije korijenski dio, uključujući ime poslužitelja i putanju, je spremljen u zasebnu tablicu kojoj samo administratori imaju pristup. Nakon konstruiranja URL-a, on se koristi u pozivu prethodno objašnjene funkcije DO_HTTP_REQUEST. Odgovor dobiven od te funkcije se predaje konstruktoru *XMLType* objekta. *XMLType* je poseban tip podatka u Oracle SUBP-u koji služi za jednostavno rukovanje XML podatcima unutar konteksta relacijske baze podataka. On omogućava korištenje XML dokumenata u relacijskim upitima, indeksiranje, transformaciju i validaciju dokumenata i jednostavno čitanje vrijednosti pojedinih elemenata unutar pojedinog dokumenta [32]. Nakon stvaranja apstraktne reprezentacije XML dokumenta poziva se *MERGE* naredba. *MERGE* naredba, kolokvijalno poznata kao i *UPSERT* (engl. *UPDATE or INSERT*) naredba, omogućava osvježavanje sloga u tablici s novim podatcima iz druge tablice ili podupita ako redak postoji u odredišnoj tablici. Postojanje ili poklapanja redaka se utvrđuje ovisno o ispunjenosti

uvjeta navedenog unutar „*ON* (...)“ dijela naredbe. Ako redak ne postoji u odredišnoj tablici, bit će stvoren koristeći parametre navedene unutar *INSERT* podnaredbe, a ako postoji bit će izmijenjen prema parametrima *UPDATE* podnaredbe [27]. Na taj način je smanjena složenost koda i povećana brzina izvođenja jer se podatci skupno obrađuju, a ne jedan po jedan redak. Unutar podupita za dohvat novih podataka unutar naredbe *MERGE*, korištena je i funkcija *XMLTable* koja transformira *XMLType* varijablu u relaciju na način da svaki rezultat XQuery upita nad proslijeđenim XML dokumentom postane novi slog koji se sastoji od atributa koji su definirani unutar parametara funkcije. Također, unutar *MERGE* naredbe je korištena klauzula „*LOG ERRORS*“ koja dozvoljava da se pogreške prilikom DML operacija zabilježe u posebnu tablicu za pogreške skupa s podatcima koji su izazvali grešku. „*REJECT LIMIT UNLIMITED*“ klauzula dozvoljava beskonačan broj grešaka prilikom skupne obrade podataka bez prekida obrade i DML operacija.

4.2. Vanjsko sučelje prema drugim platformama

S obzirom da puno studenata danas koristi razne digitalne platforme za upravljanje vremenom i organizaciju osobnih rasporeda, iz praktičnih razloga je dobro da aplikacija ponudi studentima jednostavan način za prikaz nastavnih obaveza na jednom mjestu. Iz tog razloga se korisnicima nudi mogućnost da izvezu svoj nastavni kalendar u općeprihvaćenom *iCalendar* formatu kojeg podržavaju mnoge mobilne, desktop kao i web aplikacije za upravljanje vremenom. Alternativa *iCalendar* formatu jest povezivanje aplikacije Studentskog rasporeda s programskim sučeljem svake pojedine platforme što bi zasigurno povećalo složenost aplikacije i otvorilo velik prostor za pogreške prilikom implementacije. Osim toga, svako programsko sučelje s vremenom doživljava promjene i dužnost je programera čiji proizvod koristi ta sučelja da prati te promjene i pravovremeno prilagodi svoju aplikaciju. S druge strane, *iCalendar* format je univerzalni tekstualan format dizajniran za jednostavno strojno generiranje i obradu. Format tekstualne datoteke je opisan u predloženom internetskom standardu RFC 5545, a standard je pisan tako da prepušta korisnicima kako će dostavljati datoteke s informacijama. Svaka datoteka predstavlja jedan kalendar ili raspored, a u sebi u sebi može sadržavati više jedan ili više događaja, a svako svojstvo događaja je opisano u formatu „ključ : vrijednost“ i razdvojeno u zasebnu liniju. Sadržaj datoteke s jednim kalendarom koji u sebi ima jedan događaj je prikazan na Slici 4.4:

```
BEGIN:VCALENDAR
VERSION:2.0
PRODID://Studrasp
METHOD:PUBLISH
BEGIN:VEVENT
UID:deadbeef-dead-beef-dead-beef00000075
SUMMARY:Dan Fakulteta
LOCATION:FERIT Osijek
DTSTART:20200423T220000Z
DTEND:20200424T215959Z
DTSTAMP:20200612T113400Z
DESCRIPTION: Dan Fakulteta se slavi na godišnjicu Rješenja za izvođenje nastave
CLASS:PUBLIC
END:VEVENT
END:VCALENDAR
```

Slika 4.4. Primjer kalendara s jednim neponavljajućim događajem

U gornjem primjeru je vidljivo da svaki objekt započinje s „BEGIN:“ i navođenjem tipa objekta, a završava s „END:“ i imenom tipa objekta. Između se navode atributi čija imena se pišu velikim slovom, a svaki događaj treba imati svoj jedinstveni nepromjenjivi „UID“ (engl. *universal identifier*) preko kojeg se može pratiti promjena svojstava događaja između različitih verzija kalendara. „PROIDID“ svojstvo se navodi na početku kalendara i ono služi za identificiranje programa koji je generirao datoteku. „DTSTART“ i „DTEND“ svojstva navode kada svaki događaj počinje i završava, a navode se u formatu „GGGGMMDD'T'HHMISS'Z'“. Vrijeme događaja mora biti zapisano u UTC (engl. *Coordinated Universal Time*) vremenskoj zoni kako ne bi došlo do pogreške u pretvorbi na lokalno vrijeme korisnika. DTSTAMP polje navodi kada je došlo do zadnje promjene događaja što aplikaciji omogućava da korisniku prikaže promjene kako su se vremenski dogodile i da uvijek ima najnoviju verziju događaja u kalendaru. U Slici 4.5 je prikazan programski kod procedure koja je javno izložena putem Interneta svim korisnicima radi preuzimanja rasporeda u druge platforme.

```
CREATE OR REPLACE PROCEDURE GETICAL(P_INPUT IN VARCHAR2) AS
    L_JMBAG STUDENTS.JMBAG%TYPE;

    CURSOR C_USER_EVENTS IS
        SELECT ICAL_UID, SUMMARY, LOCATION, DTSTART, DTEND, DTSTAMP, DESCRIPTION
          FROM ICAL$EVENT_DATA
         WHERE JMBAG = L_JMBAG;
BEGIN
    BEGIN
        SELECT JMBAG
          INTO L_JMBAG
          FROM STUDENTS
         WHERE UPPER(P_INPUT) =
STANDARD_HASH(FILIP_UTIL.GET_SETTING('ICAL_HASH_SALT')
                  || NVL(JMBAG, AAI)
                  || FILIP_UTIL.GET_SETTING('ICAL_HASH_SALT'),
              'MD5');
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            OWA_UTIL.STATUS_LINE(400, 'Bad Request: Invalid user');
            RETURN;
    END;

    OWA_UTIL.MIME_HEADER('text/calendar', FALSE);
    OWA_UTIL.HTTP_HEADER_CLOSE;

    HTP.P('BEGIN:VCALENDAR');
    HTP.P('VERSION:2.0');
    HTP.P('PRODID://Studrasp');
    HTP.P('METHOD:PUBLISH');

    FOR I IN C_USER_EVENTS LOOP
        HTP.P('BEGIN:VEVENT');
        HTP.P('UID:'          || I.ICAL_UID);
        HTP.P('SUMMARY:'      || I.SUMMARY);
        HTP.P('LOCATION:'     || I.LOCATION);
        HTP.P('DTSTART:'      || I.DTSTART);
        HTP.P('DTEND:'        || I.DTEND);
        HTP.P('DTSTAMP:'      || I.DTSTAMP);
        HTP.P('DESCRIPTION:'  || I.DESCRIPTION);
        HTP.P('CLASS:PUBLIC');
        HTP.P('END:VEVENT');
    END LOOP;

    HTP.P('END:VCALENDAR');
END GETICAL;
```

Slika 4.5. Procedura za generiranje *iCalendar* kalendara za pojedinog studenta

Proceduri GETICAL se predaje jedan parametar koji jednoznačno identificira korisnika i njegova vrijednost je generirana na način tako da nije jednostavno izračunati ili pretpostaviti vrijednost za drugog korisnika. Identifikacijski ključ korisnika se stvara na način da se korisnikovom JMBAG-u pridaju dva stringa koji su pohranjeni u bazi podataka na siguran način i nedostupni krajnjim korisnicima i zatim se ta vrijednost predaje kao parametar MD5 funkciji (engl. *Message Digest 5*). MD5 je jednosmjerna *hash* funkcija koja predanu vrijednost pretvara jedinstveni niz heksadekadskih znakova iz kojih se ne može utvrditi originalni podatak. Kada korisnik pozove putem interneta proceduru sa svojim identifikacijskom ključem koji mu je dostupan unutar aplikacije, procedura prvo provjerava postoji li student s jednakim identifikacijskim ključem u

bazi podataka i koji je njegov JMBAG. Ako prilikom upita dođe do iznimke, to jedino može značiti da ključ ne postoji u bazi podataka i zato se klijentu šalje HTTP odgovor s statusom „400: *Bad Request*“ i odmah se prekida izvršavanje funkcije. Ako je korisnik pronađen, baza podataka šalje HTTP zaglavlje koje govori o kojem tipu podatka je riječ putem MIME (engl. *Multipurpose Internet Mail Extensions*) zaglavlja. Vrijednost „*text/calendar*“ govori da je riječ o *iCalendar* datoteci. Nakon slanja zaglavlja s tipom datoteke šalje se znak za kraj zaglavlja i prelazi se na slanje iCalendar datoteke, koja se generira trenutno tijekom izvođenja bez pohrane. Zaglavlje je uvijek jednako za sve korisnike dok događaji ovise o svakom korisniku. Popis događaja se formira putem FOR petlje koja, umjesto ponavljanja određeni broj puta definiran razlikom granica petlje, iterira po redcima predefiniranog upita kao granice (slično kao *foreach* petlja u drugim jezicima). Taj predefinirani upit je definiran u *DECLARE* dijelu kao kursor *C_USER_EVENTS* koji dohvaća unaprijed formatirane podatke iz pogleda *ICAL$EVENT_DATA* za pojedinog korisnika. Za svaki redak koji je dobiven izvršavanjem kursora, izvršava se niz naredbi unutar FOR petlje, dok su podatci iz trenutnog retka dostupni korištenjem indeksne varijable I. Prilikom ispisa na internetsko sučelje mora se koristiti posebna *HTP.P* procedura koja dani string sprema u poseban međuspremnik, različit od onog koji se koristi za ispis na konzolu putem *DBMS_OUTPUT.PUT_LINE* procedure.

4.3. Izračun zajedničkih slobodnih perioda između studenata

Mogućnost izračuna zajedničkih perioda kada je grupa od dva ili više studenata je vrlo praktična i korisna funkcija koju studenti mogu iskoristiti za lakše dogovaranje zajedničkog učenja, druženja ili sličnih dogovora. Iako je izračun zajedničkih slobodnih termina vrlo jednostavan za čovjeka, prilikom korištenja relacijskih upita taj postupak je vrlo vremenski zahtjevan. Zbog toga se posao izračuna zajedničkih slobodnih termina razdvojio u dva dijela: prvi dio se izvršava svaku noć nakon preuzimanja novog rasporeda i on izračunava i sprema u tablice kada je pojedini student slobodan, dok drugi dio izračuna se događa tek kada student putem web sučelja odabere s kojim studentima želi izračunati slobodno vrijeme. Prvi dio procesa za izračun slobodnog vremena se sastoji od četiri dijela:

- dodavanje studentove nastave u zasebne tablice za izračun
- dodavanja tzv. univerzalnih „blokirajućih“ termina
- spajanje preklapajućih i dodirujućih termina
- izračun slobodnih termina za svakog studenta

Nakon uvoza novog rasporeda, u posebne inscenacijske (engl. *staging*) tablice se privremeno pohranjuju sva nastava (uključujući i događaje koje je student dodao u osobni raspored) za svakog

pojedinog studenta od početka akademske godine, ali samo vrijeme početka i kraja, bez detalja o nastavi. Nakon toga se dodaju univerzalni „blokirajući“ termini koji sprječavaju sustav da izračuna slobodne termine između 20 sati i 8 sati ujutro idućeg dana i termine koji su tijekom vikenda. Upiti za izdvajanje studentske nastave i dodavanje blokirajućih termina preko noći su prikazani u Slikama 4.6 i 4.7.

```
INSERT INTO STG$FP_ALL_LECTURE_EVENTS
        (JMBAG, START_TIME, END_TIME)
    SELECT DISTINCT
            JMBAG,
            START_TIME,
            END_TIME
      FROM EVENTS_BY_USER
     WHERE START_TIME >= GET_ACADEMIC_YEAR_START_DATE()
       AND JMBAG IS NOT NULL
    UNION
    SELECT DISTINCT
            JMBAG,
            START_TIME,
            END_TIME
      FROM EVENT_SCHEDULE_CUSTOM
     WHERE START_TIME >= GET_ACADEMIC_YEAR_START_DATE();
```

Slika 4.6. *INSERT* upit za kopiranje studentske nastave u inscenacijsku tablicu

Korištenjem ključne riječi *DISTINCT* i operatora *UNION* umjesto *UNION ALL*, upit će umetnuti samo jedinstvene retke koji nemaju jednaka vremena početka i vremena završetka za pojedinog studenta.

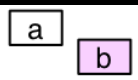
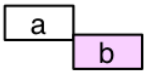
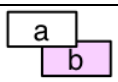
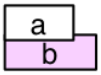
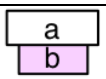
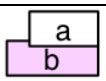
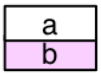
```
INSERT INTO STG$FP_BLOCK_WKDAY_NIGHT (START_TIME, END_TIME, JMBAG)
    SELECT DISTINCT
            GREATEST(END_TIME, START_TIME + 20 / 24) AS START_TIME,
            TRUNC(END_TIME, 'DD') + 1 + 8 / 24 AS END_TIME,
            JMBAG
      FROM (SELECT DISTINCT
                    JMBAG,
                    TRUNC(START_TIME, 'DD') AS START_TIME,
                    END_TIME
              FROM STG$FP_ALL_LECTURE_EVENTS
             WHERE TO_CHAR(START_TIME, 'D') <= 5);
```

Slika 4.7. *INSERT* upit za umetanje blokirajućih vremenskih termina u inscenacijsku tablicu

Upit kao izvor podataka koristi prethodno izdvojene podatke o nastavi za svakog studenta, ali samo za nastavu koja se odvija radnim danima (od ponedjeljka do petka). Prilikom stvaranja blokirajućeg termina, vrijeme početka nije od interesa jer blokirajući termin započinje od 20 sati ili kraja zadnjeg nastavnog događaja u danu, ovisno što dolazi kasnije, i traje do idućeg dana u 8 sati ujutro.

Prilikom izračuna slobodnog vremena, intervali od interesa koji se proučavaju su nastavni sati u rasporedu svakog pojedinog studenta. Prije izračunavanja slobodnih termina, radi olakšavanja i ubrzavanja izračuna uputno je vremenske intervale koji se međusobno preklapaju ili susreću (odnosi 2, 3, 4, 5, 6 i 7 u Tablici 4.1) spojiti u jedan termin koji započinje u trenutku početka prvog termina, a završava u trenutku završetka drugog termina. Na taj način će se smanjiti broj intervala koji se moraju obraditi. Za rad s vremenskim rasponima, James Allen je u svom radu definirao 13 različitih odnosa između dva vremenska intervala od iz kojih se može utvrditi postojanje sedam jedinstvenih odnosa koji su prikazani u Tablici 4.1 i koji olakšavaju izračun i poimanje odnosa između intervala [33].

Tablica 4.1. Odnosi među vremenskim intervalima, izvor: [34]

	Odnos	Vizualizacija
1.	A prethodi B	
2.	A susreće B	
3.	A preklapa B	
4.	A započinje kad i B	
5.	A je tijekom B	
6.	A završava kad i B	
7.	A je jednak B	

Za otkrivanje preklapanja i spajanje preklapajućih termina koristit će se SQL analitička funkcija *MATCH_RECOGNIZE* kojom se mogu prepoznati i transformirati uzorci u redcima koristeći regularne izraze (engl. *regular expression*) i logičke predikate [35]. Primjer korištenja *MATCH_RECOGNIZE* funkcije za pronalazak preklapajućih termina je dan u Slici 4.8.

```
SELECT *
  FROM table
MATCH_RECOGNIZE(
    ORDER BY start_time,
             end_time
    MEASURES FIRST(start_time) start_time,
             MAX(end_time)     end_time
     PATTERN (A B*)
      DEFINE B AS start_time <= PREV(end_time)
);
```

Slika 4.8. Primjer korištenja *MATCH_RECOGNIZE* funkcije za pronalaženje uzorka

Funkcija se koristi tako da u *ORDER BY* dijelu se prvo navede po kojem uvjetu retke treba sortirati prije traženja uzoraka, zatim putem *MEASURES* klauzule se definiraju izraze koje će funkcija vratiti kao stupce. *FIRST* funkcija će vratiti traženu vrijednost stupca iz prvog retka koji je dio traženog uzorka, dok će *MAX* funkcija vratiti najveću vrijednost stupca u pronađenom uzorku. *PATTERN* klauzula služi za navođenje regularnog izraza kojim se definira uzorak kojeg se želi pronaći [35]. U ovom slučaju potrebno je naći bilo koji redak (A uvjet) kojeg slijedi nula ili više redaka koji zadovoljavaju B uvjet. S obzirom da A uvjet nije definiran, za njega se implicitno uzima da taj uvjet uvijek vrijedi. B uvjet je definiran tako da je logički uvjet zadovoljen kada trenutni događaj (redak) počinje prije ili u istom trenutku kada prethodni događaj završava.

Nakon spajanja preklapajućih i dodirujućih termina može se pristupiti izračunu praznina između nastave odnosno slobodnih perioda. Upit za izračun treba izdvojiti zadnje vrijeme završetka do i uključujući trenutni redak kao početak slobodnog termina i početno vrijeme idućeg retka kao završetak slobodnog termina. Oba uvjeta se trebaju gledati odvojeno za svakog studenta, a ne globalno na razini čitave tablice. Ogledni primjer takvog upita je dan u Slici 4.9.

```
INSERT INTO FREE_PERIODS(JMBAG, START_TIME, END_TIME)
    SELECT JMBAG,
           MAX(END_TIME)     OVER (PARTITION BY JMBAG
                                   ORDER BY START_TIME) START_TIME,
           LEAD(START_TIME) OVER (PARTITION BY JMBAG
                                   ORDER BY START_TIME) END_TIME
      FROM STG$FREE_PERIOD_DATA;
```

Slika 4.9. Upit za izračun i pohranu slobodnih termina za svakog studenta

Zbog dugotrajnog izvođenja prethodno prikazanih upita, izračunate slobodne termine je dobro pohraniti kako bi kasniji dohvat bio brz prilikom izračuna slobodnih termina između više studenata. Na taj način kada student odluči izračunati slobodne termine između sebe i svojih kolega, sustav jedino mora dohvatiti prethodno izračunate slobodne termine svakog od odabranih studenata i utvrditi kada se oni preklapaju.

Izračun slobodnih termina između studenata nije moguće unaprijed odraditi jer postoji prevelik broj kombinacija studenata što bi dovelo do toga da proces izračuna traje beskonačno dugo. Stoga zadnji dio funkcionalnosti se može izvesti tek kada korisnik putem web sučelja odabere jednog ili više kolega za kojeg ga zanima zajedničko slobodno vrijeme. Taj popis studenata se sprema u APEX-ov skup podataka koji se čuva za svaku aktivnu korisničku sjednicu (engl. *session*) i njemu se pristupa putem ugrađenog pogleda *APEX$COFFEE_TIME_ALL_USERS* čija je definicija prikazana na Slici 4.10.

```
CREATE OR REPLACE VIEW APEX$COFFEE_TIME_ALL_USERS AS
       SELECT C001 AS OIB
         FROM APEX_COLLECTIONS
        WHERE COLLECTION_NAME = 'APEX_FREE_TIME_STUDENT_LIST_NAME'
    UNION ALL
       SELECT V('APP_USER') AS OIB
         FROM DUAL
```

Slika 4.10. Definicija pogleda APEX$COFFEE_TIME_ALL_USERS

Za izračun vremena, osim popisa studenata koje je korisnik odabrao, potreban je i identifikator samog studenta kao jednog od elemenata za izračun. Pristup podatcima je moguć jedino iz aktivne korisničke sjednice kako bi se zaštitila privatnost korisnika.

```
  WITH UNPIVOTED AS
       (SELECT START_TIME,
               SEG_TYPE
          FROM APEX$COFFEE_TIME_ALL_USERS ACTAU
               JOIN FREE_PERIODS
                    UNPIVOT (START_TIME FOR SEG_TYPE IN (START_TIME AS  1,
                                                         END_TIME   AS -1)) FP
                 ON FP.JMBAG = ACTAU.JMBAG
         WHERE FREE = 1),
        FREE_TIMES AS
        (SELECT START_TIME,
                 LEAD(START_TIME) OVER (ORDER BY START_TIME) END_TIME,
                 SUM(SEG_TYPE)    OVER (ORDER BY START_TIME) MATCHES
            FROM UNPIVOTED)
SELECT START_TIME,
       END_TIME
  FROM FREE_TIMES
 WHERE MATCHES = (SELECT COUNT(1)
                    FROM APEX$COFFEE_TIME_ALL_USERS)
   AND START_TIME < END_TIME
```

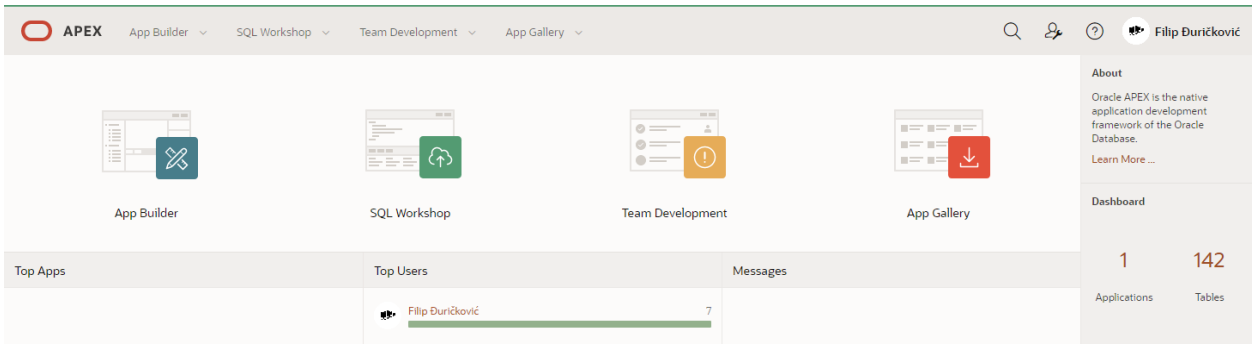
Slika 4.11. Upit za dohvat zajedničkih slobodnih termina grupe studenata

Upit prikazan Slikom 4.11 daje podatke o preklapajućim slobodnim terminima svih ljudi koje je student prethodno izabrao. Sastoji se od tri podupita od kojih su dva unutar *WITH* klauzule zbog lakše čitljivosti i razumljivosti. Prvi podupit služi za dohvaćanje slobodnih termina samo

odabranih studenata što se postiže povezivanjem pogleda *APEX$COFFEE_TIME_ALL_USERS* i *FREE_PERIODS* tablice prema uvjetu jednakosti JMBAG-a u obje relacije. Prije povezivanja relacija, izvodi se *UNPIVOT* operacija nad *FREE_PERIODS* tablicom. *UNPIVOT* transponira podatke iz stupaca u retke i stvara novi stupac koji poprima vrijednost u ovisnosti iz kojeg stupca je originalno translatirana vrijednost u retku. U ovom slučaju, *UNPIVOT* operacija će stupce *START_TIME* i *END_TIME* spojiti u zajednički stupac *START_TIME* i u novom stupcu *SEG_START* će se nalaziti vrijednost 1 ako je podatak originalno bio iz stupca *START_TIME*, a ako je bio iz stupca *END_TIME* imat će vrijednost -1. Nakon izvršavanja tog upita, unutar *FREE_TIMES* se izračunavaju krajevi slobodnog vremena pomoću *LEAD* analitičke funkcije koja vraća vrijednost iz idućeg retka i *SUM* funkcije koja koristi vrijednost *SEG_START* kako bi izračunala za koliko studenata je određeni termin slobodan. Nakon toga, u zadnjem upitu, filtriraju se svi termini tako da ostanu samo oni koji imaju trajanje strogo veće od 0 sekundi i koji su zajednički svim studentima na popisu.

5. IZRADA APLIKACIJE U ORACLE APEX RAZVOJNOM OKVIRU

Oracle APEX je nisko-kodna platforma za razvoj responzivnih i interaktivnih aplikacija koje se u potpunosti izvršavaju unutar Oracle baze podataka. APEX je nasljednik uspješnog sustava *Oracle Forms & Reports* koji omogućava na sličan intuitivan i grafički orijentiran način izradu aplikacija uz malu količinu koda potrebnu za razvoj često korištenih značajki. Kontinuitet u tehnologiji, skupa s jednostavnim načinom razvoja aplikacija i prenosivost aplikacija skupa uz podatke omogućava brz i kvalitetan razvoj aplikacija za profesionalne potrebe. Oracle APEX dolazi ugrađen uz svaku instalaciju Oracle SUBP-a i u potpunosti je besplatan. Za korisnike koji žele isprobati funkcionalnost bez instalacije Oracle SUBP-a, Oracle kompanija nudi besplatno razvojno okruženje u oblaku s ograničenjem prostora i ograničenim trajanjem. S obzirom da je APEX u potpunosti web-baziran, tehnološki preduvjeti za korisnike i programere su vrlo niski, jer je jedino potreban moderan internetski preglednik. Nakon prijave u razvojno okruženje, korisnicima je dostupna naslovna stranica vidljiva na Slici 5.1 s četiri poveznice na primarne alate.



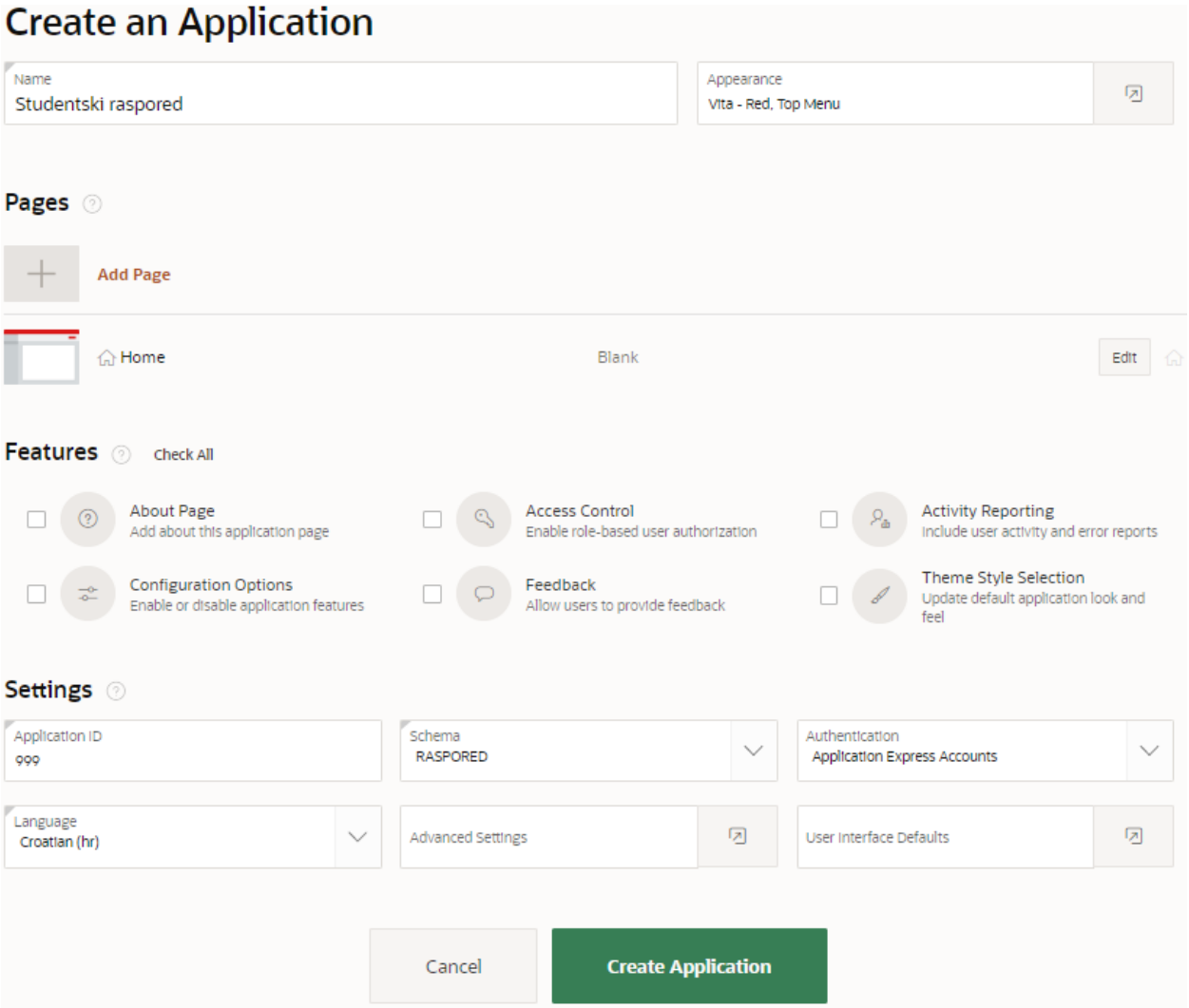
Slika 5.1. Prikaz naslovne stranice Oracle APEX programerskog sučelja

Odmah pri prijavi programeri mogu odabrati jednu od četiri funkcionalnosti:

- *App Builder* – alat za razvoj aplikacija za krajnje korisnike
- *SQL Workshop* – okruženje za razvoj baze podataka koje omogućava izvođenje SQL naredbi i promjene nad objektima u bazi podataka
- *Team Development* – alati za praćenje napretka aplikacija koji omogućuju raspodjelu zadataka kao i analizu uspješnosti
- *App Gallery* – kolekcija unaprijed ugrađenih gotovih i probnih aplikacija za jednostavne primjene

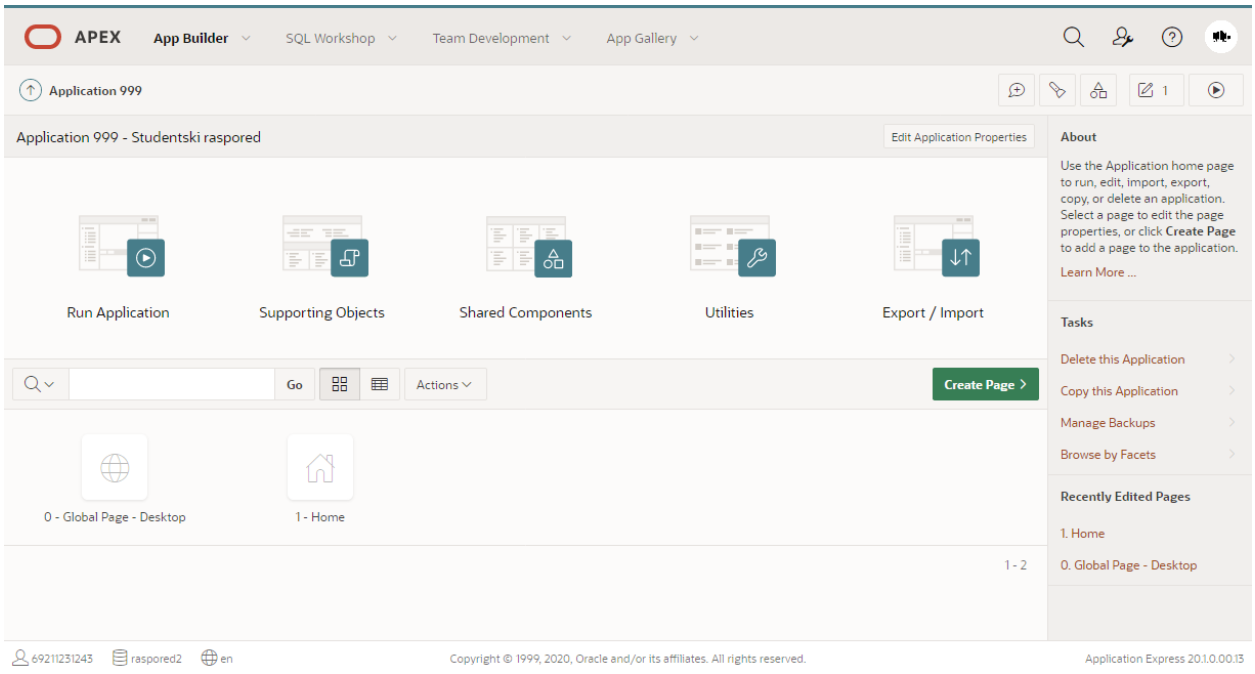
Najveća količina zadataka vezanih za razvoj na APEX platformi se upravo odvija u *App Builder*-u u kojem na početku nije kreirana ni jedna aplikacija, pa jedino korisnici mogu kreirati novu aplikaciju ili uvesti i instalirati postojeću. Osim toga, korisnicima su unutar *App Builder*-a na

raspolaganju razni alati za nadzor i upravljanje radom APEX platforme. Razvoj aplikacije se započinje s pokretanjem čarobnjaka za izradu aplikacije, prikazanog na Slici 5.2, koji od programera prikuplja osnovne informacije o aplikaciji, poput naziva, izgleda teme aplikacije, popisa stranica, jezika i slično.



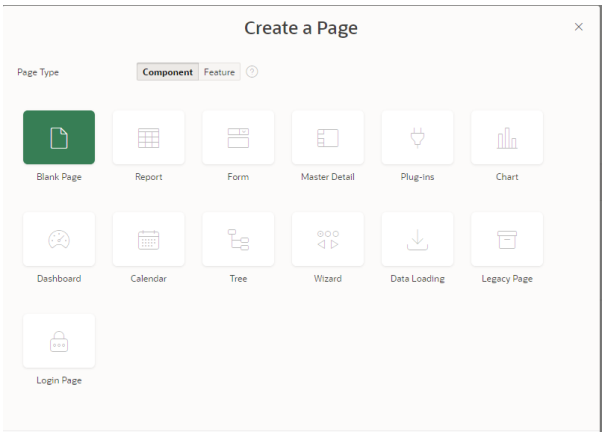
Slika 5.2. Čarobnjak za izradu aplikacija

Odmah unutar čarobnjaka programeri mogu odabrati aktivaciju nekih unaprijed ugrađenih značajki poput izvještavanja o aktivnosti, slanja povratnih informacija od strane korisnika i slično. Nakon kreiranja aplikacije, unutar *App Builder*-a se korisniku prikazuje popis stranica unutar aplikacije, skupa s opcijama za pokretanje aplikacije, upravljanje potpornim objektima u bazi podataka (tablice, pogledi i sl.), pregled i izmjenu postavki same aplikacije, raznim alatima za dijagnostiku i opciju za uvoz i izvoz aplikacije.

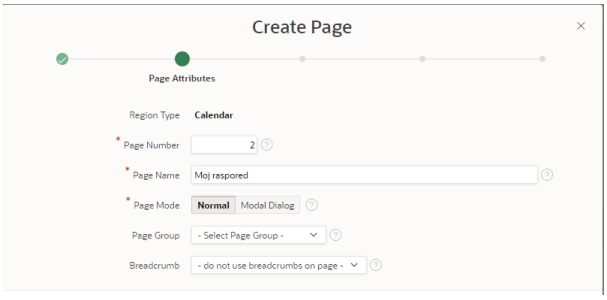


Slika 5.3. Pregled novonastale aplikacije u *App Builderu*

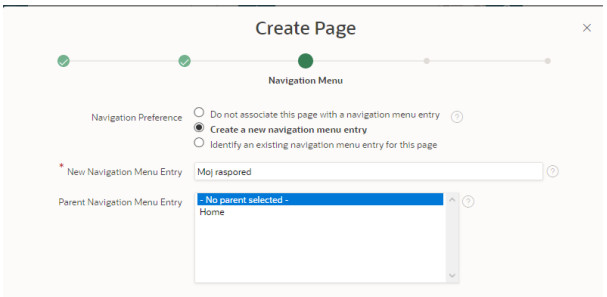
Klikom na *Create Page*, pokreće se čarobnjak za izradu stranice, prikazan na Slikama 5.4, 5.5, 5.6, 5.7 i 5.8 koji u nekoliko koraka prikuplja sve potrebne podatke za stvaranje stranice bazirane na SQL upitu, pogledu ili tablici.



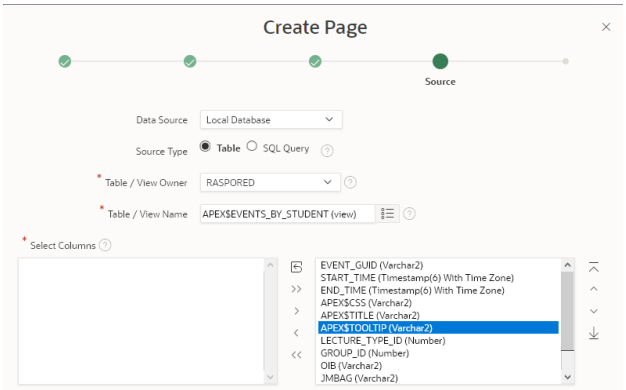
Slika 5.4. Odabir tipa stranice



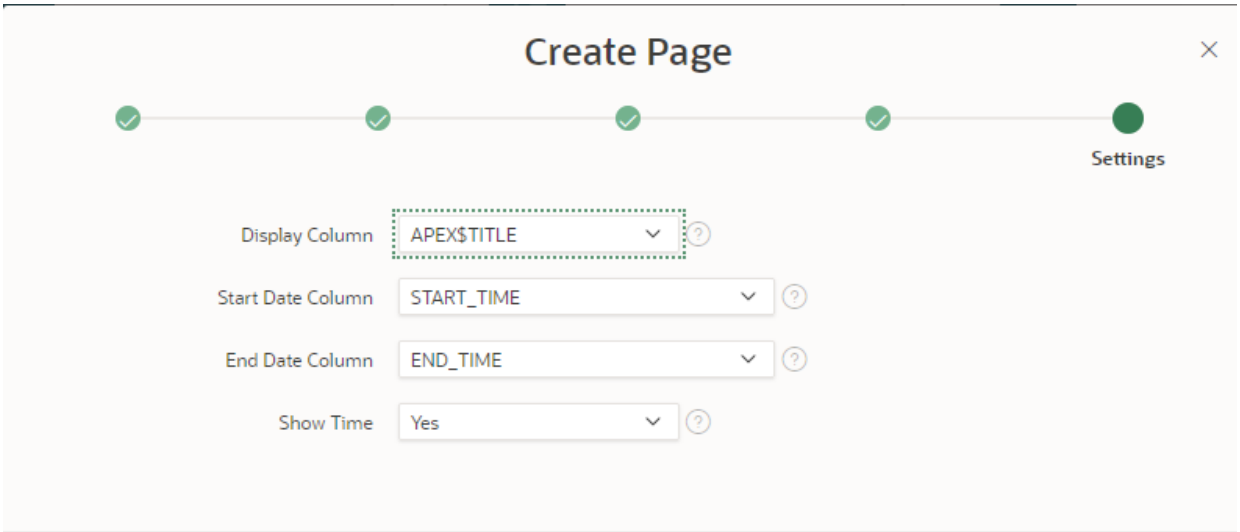
Slika 5.5. Unos osnovnih podataka o stranici



Slika 5.6. Podešavanje navigacije

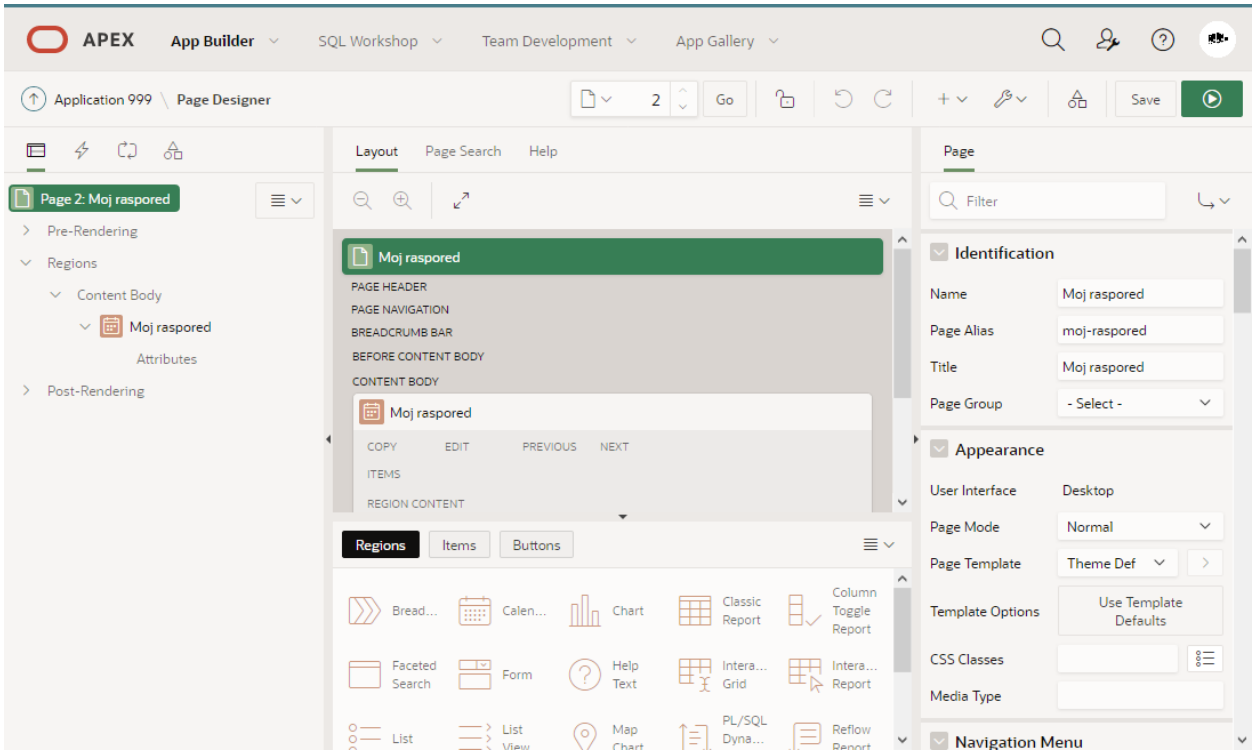


Slika 5.7. Odabir izvora podataka



Slika 5.8. Konfiguracija stupaca za prikaz

Nakon završnog koraka čarobnjaka stranica se kreira i automatski se otvara u pregledniku mogućnost za izmjenu novonastale stranice, prikazana na Slici 5.9. Tu programer može promijeniti raspored stranice, dodati nove elemente ili ih ukloniti, unijeti postavke i dodatno konfigurirati ponašanje stranice i podelemenata.



Slika 5.9. Prikaz komponenti pojedine stranice

Klikom na opciju *Attributes* lociranu pod čvorom *Moj raspored* u izborniku s lijeve strane, ponašanje i svojstva kalendara se mogu dodatno promijeniti, pa se tako mogu konfigurirati prikazi *tooltip*-a prelaskom kursora preko događaja, stiliziranje različitih događaja, format prikaza vremena i slično.

Dodavanje osobnih događaja je izvedeno na način da student klikom na gumb „Dodaj novi osobni događaj“ otvara novi dijaloški prostor u kojem definira vrijeme i datum početka i završetka događaja, kao i u kojoj predavaonici se o

Funkcionalnost izračuna slobodnog vremena je implementirana na način da na inicijalnoj stranici student krene unositi dio imena i prezimena kolege, na što sustav automatski nudi odgovarajuća imena i prezimena i student ih može odabrati i potvrditi. Kada je student zadovoljan odabirom kolega, pritiskom na gumb „Nađi slobodni period“ student se preusmjerava na iduću stranicu gdje mu je prikazan kalendar u kojem su grafički prikazani svi slobodni periodi.

Autentifikacija korisnika je izvedena tako da u Apache konfiguracijskoj datoteci za lokaciju „/apex“ (na kojoj Apache Tomcat poslužuje ORDS Java *servlet*) se aktivira mod_auth_mellon modul koji ne dozvoljava posluživanje resursa korisniku ako prethodno nije uspješno identificiran. Za svakog korisnika koji je identificiran, njegovi podatci dobiveni od AAI@EduHR sustava se prosljeđuju Oracle APEX-u putem dodatnih zaglavlja u HTTP zahtjevu koje korisnik ne može ni

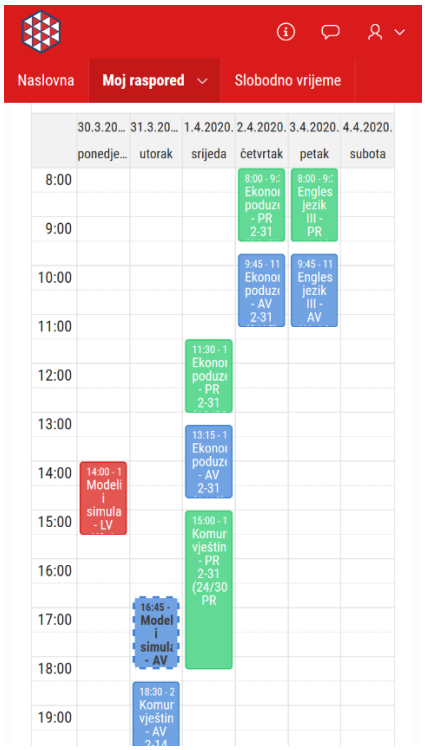
na koji način pročitati ili izmijeniti. U Oracle APEX-u je konfigurirano da se za korisničko ime uzima vrijednost koja odgovara hrEduPersonUniqueID atributu koji u sebi sadrži JMBAG studenta.

```
<Location "/apex">
    MellonEnable "auth"
    MellonUser "hrEduPersonUniqueID"
    MellonSetEnvNoPrefix "AAI_UNIQUE_ID" "hrEduPersonUniqueID"
    MellonSetEnvNoPrefix "AAI_GIVEN_NAME" "givenName"
    MellonSetEnvNoPrefix "AAI_SN" "sn"
    MellonSetEnvNoPrefix "AAI_UNIQUE_NUMBER" "hrEduPersonUniqueNumber"
    MellonSetEnvNoPrefix "AAI_AFFILIATION" "hrEduPersonPrimaryAffiliation"
    MellonSetEnvNoPrefix "AAI_ORGANIZATION" "hrEduPersonHomeOrg"
    MellonSetEnvNoPrefix "AAI_EMAIL" "mail"
    RequestHeader set AAI-USER %{AAI_UNIQUE_ID}e
    RequestHeader set AAI-IDNum-Original %{AAI_UNIQUE_NUMBER}e
    RequestHeader set AAI-IDNum-JMBAG %{AAI_UNIQUE_NUMBER}e
    RequestHeader edit AAI-IDNum-JMBAG "(.*)(JMBAG)([ :]*)(\d{10})(.*)" "$4"
    RequestHeader set AAI-FirstName %{AAI_GIVEN_NAME}e
    RequestHeader set AAI-LastName %{AAI_SN}e
    RequestHeader set AAI-Affiliation %{AAI_AFFILIATION}e
    RequestHeader set AAI-Organization %{AAI_ORGANIZATION}e
    RequestHeader set AAI-EMail %{AAI_EMAIL}e
    RequestHeader set AAI-UNIQUE-ID %{AAI_UNIQUE_ID}e
    RequestHeader unset Origin
    ProxyPreserveHost On
    ProxyPass "ajp://localhost:8009/apex"
    ProxyPassReverse "ajp://localhost:8009/apex"
</Location>
```

Slika 5.10. Isječak postavki za autentifikaciju iz Apache konfiguracijske datoteke

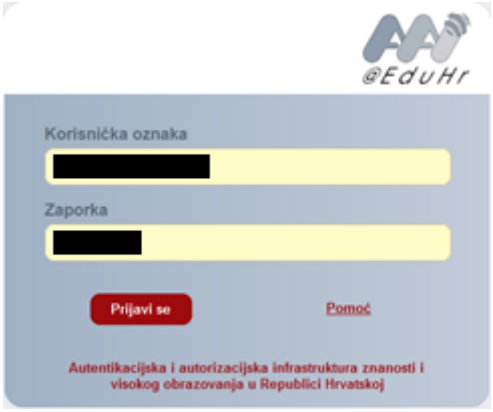
5.1. Konačan izgled aplikacije

Aplikacija se sastoji od ukupno četiri obične stranice i dvije stranice implementirane kao modalni dijalog. Detaljan pregled, kao i obrazac za unos pojedinog događaja su izvedeni u obliku posebnih skočnih prozora koje korisnici moraju kako bi mogli se kretati po ostatku stranice. Koristeći ugrađenu funkcionalnost Oracle APEX-a postignuta je responzivnost web stranica tako da se sadržajno isti HTML dokumenti različito iscrtavaju u pregledniku u ovisnosti o veličini ekrana.



Slika 5.11: Prikaz personaliziranog tjednog rasporeda na mobitelu

Kada korisnik posjeti web sjedište aplikacije, prvo se mora prijaviti putem AAI@EduHR sustava. AAI@EduHR portal od svakog korisnika zahtijeva unos korisničke oznake i zaporke koju mu je izdala njegova matična ustanova, kao što je prikazano na Slici 5.12.

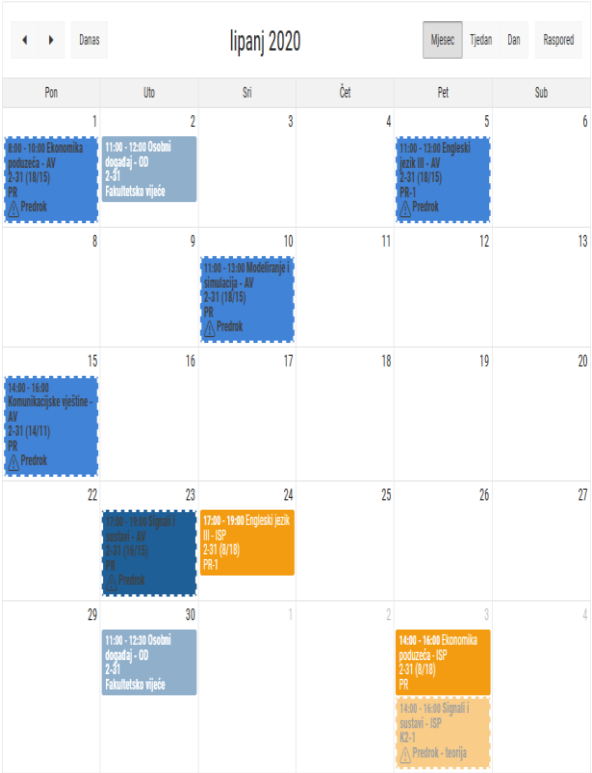


Slika 5.12. AAI@EduHR prijavni prozor

Nakon prijave, korisniku se prezentira njegov osobni raspored za trenutni tjedan u tjednom prikazu kao što je prikazano na Slici 5.13, a ako to želi, korisnik može uključiti i neki od drugih oblika prikaza poput mjesečnog kalendara (Slika 5.14), prikaza za pojedini dan ili prikaza u obliku popisa. Prilikom prelaska mišem preko pojedinog događaja u kalendaru, korisniku se prikazuju osnovne

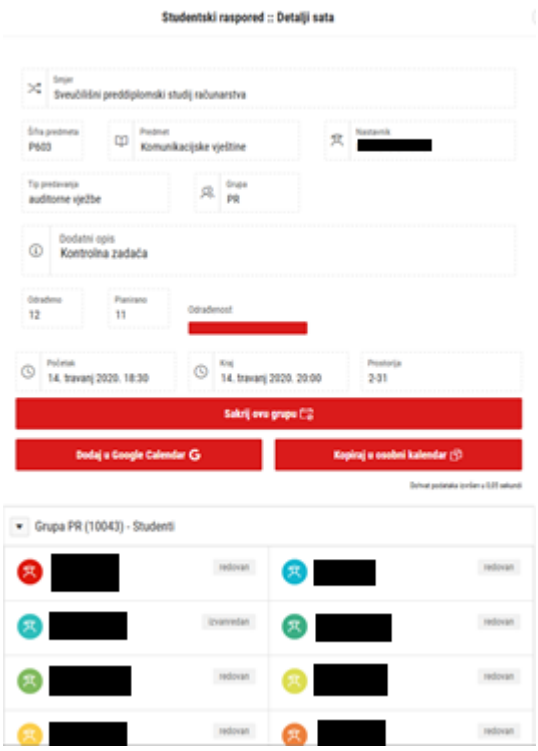
informacije o pojedinom događaju poput predmeta, dvorane, vremena početka i završetka, vrste nastave i nastavnika.

Slika 5.13. Prikaz početne stranice

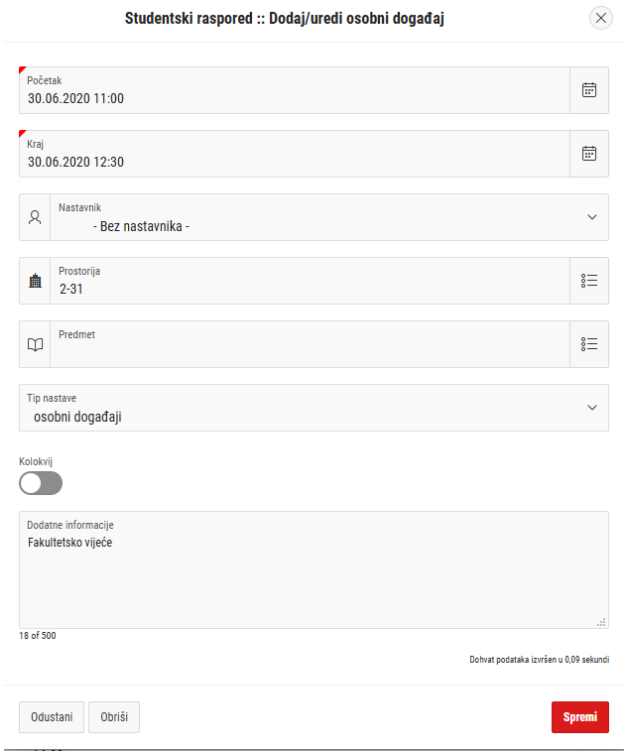


Slika 5.14. Mjesečni prikaz osobnog rasporeda

Klikom na pojedini događaj korisniku se prezentiraju sve informacije o poejdinom događaju, uz dodatne mogućnosti kopiranja događaja u osobni kalendar, skrivanja sve nastave vezane uz tu grupu i predmet i prikaz drugih strudenata iz te nastavne grupe, kao što je prikazano na Slici 5.15. Klikom na gumb „Dodaj osobni događaj“ (Slika 5.16) otvara se dijaloški prozor sličan onomu za prikaz detalja u nastavi gdje student unosi podatke poput vremena početka i završetka osobnog događaja, je li on vezan uz nekog nastavnika ili predmet i u kojoj će se od predavaonica eventualno održati. Osim toga korisnik može označiti da je riječ o kolokviju ili nekoj vrsti provjere znanja kako bi se događaj više istaknuo u kalendaru i može unijeti kratke dodatne informacije i bilješke vezane uz događaj. Neovisno o unosu nastavnika, predmeta i/ili predavaonice, događaj se prikazuje isključivo korisniku koji ga je unio u svoj osobni raspored.

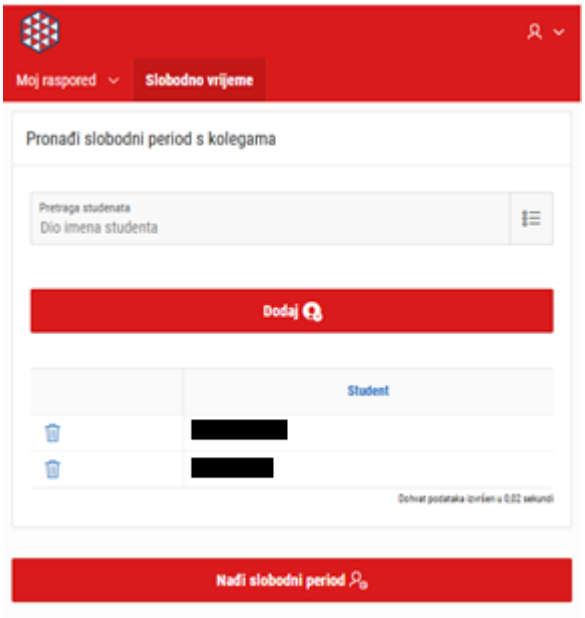


Slika 5.15. Prikaz detalja događaja

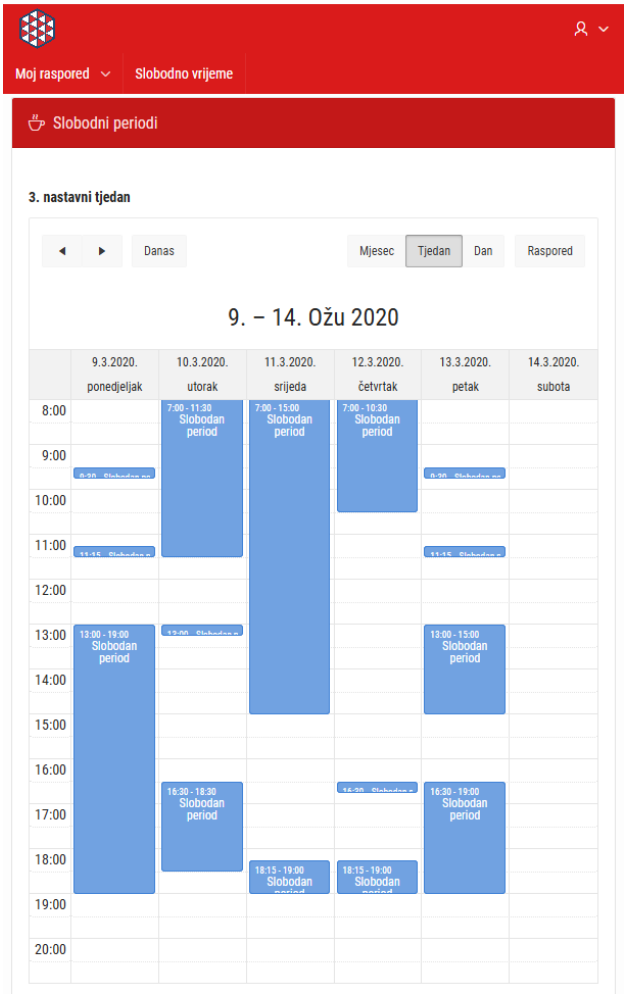


Slika 5.16. Obrazac za unos osobnog događaja

Osim navedene osnovne funkcionalnosti, korisnici također mogu koristiti funkcionalnost računanja slobodnog vremena s drugim korisnicima klikom na opciju „Slobodno vrijeme“ u gornjoj izbornoj traci. Nakon klika korisniku se otvara stranica prikazana na Slici 5.17, na kojoj može unijeti dio imena i prezimena kolege i nakon toga mu se prikazuje popis ljudi čija imena odgovaraju unesenom tekstu. Nakon odabira odgovarajuće osobe i pritiska na gumb dodaj korisniku se prikazuju svi dotad odabrani kolege koje može kasnije po volji uklanjati ili dodati novog kolegu u popis. Kada je korisnik zadovoljan s popisom, klikom na gumb „Nađi slobodni period“ preusmjerava se na novu stranicu, prikazanu na Slici 5.18, u kojoj je prikazan kalendar s naznačenim slobodnim terminima. Unutar tog kalendara korisnik se može kretati kao i na glavnom kalendaru i mijenjati prikaze kao i pomicati se na prethodne ili buduće dane i tjedne. Zbog zaštite privatnosti drugih kolega, korisnik ne može vidjeti njihova eventualna zauzeća, samo činjenicu da ona postoje.

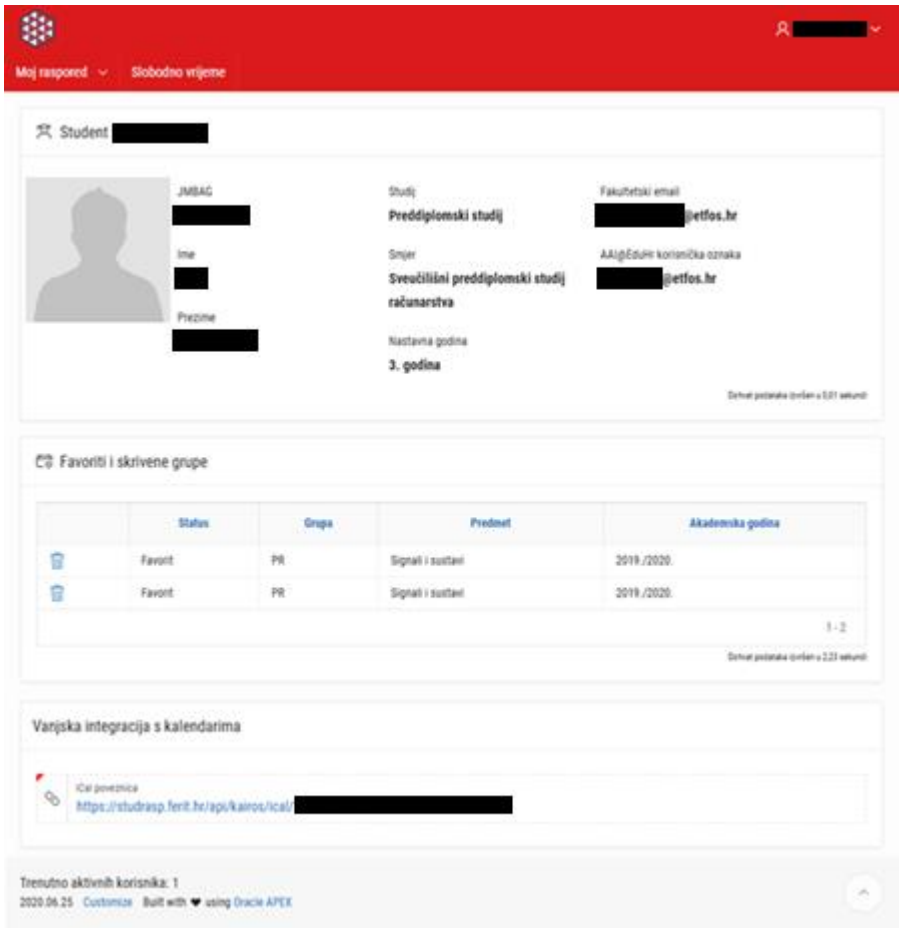


Slika 5.17. Unos ostalih kolega za računanje slobodnog vremena



Slika 5.18. Prikaz slobodnog vremena među kolegama

Posljednja funkcionalnost koju korisnik u aplikaciji može koristiti jest sinkronizacija kalendara s drugim platformama. Ona je dostupna klikom na korisnikovo ime u gornjem izborniku i odabirom opcije Postavke. Na toj stranici, prikazanoj na Slici 5.19, korisnik može na jednom mjestu vidjeti sve podatke koje sustav posjeduje o korisniku i po potrebi ispraviti neke od njih poput skrivenih i naknadno pretplaćenih grupa. Na dnu stranice se nalazi *iCalendar* poveznica koju korisnik može otvoriti ili kopirati u kalendarski program po svom izboru kako bi ih mogao drugdje prikazati i koristiti. U datoteci posluženoj na toj lokaciji se nalaze svi događaji koje bi korisnik pronašao u kalendaru Moj raspored, uključujući i osobne događaje.



Slika 5.19. Prikaz informacija o korisniku

6. ZAKLJUČAK

Zadatak završnog rada bio je napraviti web aplikaciju personaliziranog rasporeda nastave za svakog studenta Fakulteta elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. Aplikacija je izrađena koristeći Oracle APEX razvojni okvir i nisko-kodnu platformu uz programsku potporu Oracle PL/SQL programskog jezika. Veći dio rada je bio posvećen izradi i modeliranju novog modela prilagođenog aplikaciji koji se temelji na već dostupnim informacijama i obliku predodređenom iz fakultetskih informacijskih sustava.

Na početku rada spomenute su korištene tehnologije, standardi i programski jezici kao i alati te je napravljena usporedba s dva postojeća programska rješenja. Prilikom modeliranja baze podataka objašnjeni su koncepti ER i relacijskih dijagrama te su objašnjene sintakse odnosno pravila crtanja i dizajniranja ER dijagrama te pravila transformacije ER u relacijske dijagrame. Osim toga, objašnjena je i osnovna sintaksa za stvaranje objekata u bazi podataka, kao i sintaksa PL/SQL programskog jezika koji je uz SQL korišten unutar Oracle baze podataka za pisanje ukupne aplikativne logike poput dohvaćanja i obrade preuzetih podataka u oblik pogodan za korištenje u aplikaciji, kao i za posluživanje *iCalendar* podataka drugim platformama. Zaključno, prikazan je postupak kreiranja jednostavne aplikacije u Oracle APEX razvojnom okviru i demonstrirana je funkcionalnost aplikacije kako na osobnim računalima, tako i na mobilnim uređajima. Zbog zaštite osobnih podataka, imena i prezimena te ostali identificirajući podatci su uklonjeni ili na odgovarajući način skriveni na svim slikama. Svi studenti putem prijave s AAI@EduHR korisničkom računom mogu pristupiti aplikaciji i posljedično svom osobnom rasporedu.

Aplikacija se može proširiti i doraditi tako da se omogući djelatnicima zaduženima za kreiranje rasporeda nastave unos rasporeda i grupa kroz aplikaciju. Osim toga, aplikacija se uz minimalne preinake može iskoristiti za prikaz rasporeda nastave za nastavnike i pregled rasporeda po prostorijama. Vezano uz studentsku funkcionalnost, aplikacija bi se mogla proširiti na način da studentima ponudi opciju praćenja svog studija (pregled ocjena, upisanih kolegija i slično) te prijave i odjave ispita direktno kroz aplikaciju koristeći ISVU programsko sučelje. Osim toga, pojedinim studentima koji su demonstratori mogla bi se stvoriti funkcionalnost da vide svoje demonstrature i odabiru termine istih ili da ih aplikacija automatizirano dodjeljuje ovisno o demonstratorovom rasporedu nastave i broju dodijeljenih sati demonstrature. Isto tako, slična funkcionalnost bi mogla omogućiti kako nastavnicima, tako i pojedinim studentima koji imaju opravdan razlog za korištenje laboratorija i/ili dvorana da upišu rezervaciju dvorane koja je kasnije vidljiva svima.

Sve navedene nove funkcionalnosti se mogu jednostavno dodati ili unutar postojeće aplikacije ili putem drugih srodnih aplikacija kako bi se postigla modularnost dizajna arhitekture sustava, ali prije svega ključna je voljnost Uprave Fakulteta za uvođenje takve digitalizacije u već postojeće procese.

LITERATURA

[1] solid IT gmbh, »DB-Engines Ranking - popularity ranking of database management systems,« solid IT gmbh, srpanj 2020. [Mrežno]. Available: https://db-engines.com/en/ranking. [Pokušaj pristupa 04. 07. 2020.].

[2] Oracle Corporation, Oracle Database Express Edition Installation Guide, 18c for Linux x86-64, Redwood, CA, SAD: Oracle, 2018.

[3] »Oracle APEX,« Oracle Corporation, 20. 05. 2020. [Mrežno]. Available: https://apex.oracle.com/en/. [Pokušaj pristupa 01. 07. 2020.].

[4] Oracle Corporation, »FAQ,« Oracle Corporation, [Mrežno]. Available: https://www.oracle.com/tools/technologies/faq-rest-data-services.html. [Pokušaj pristupa 01. 07. 2020.].

[5] Apache Software Foundation, »Apache Tomcat® - Heritage,« Apache Software Foundation, [Mrežno]. Available: https://tomcat.apache.org/heritage.html. [Pokušaj pristupa 01. 07. 2020.].

[6] Apache Software Foundation, »Apache Tomcat® - Welcome!,« Apache Software Foundation, [Mrežno]. Available: http://tomcat.apache.org/index.html. [Pokušaj pristupa 01. 07. 2020.].

[7] »About the Apache HTTP Server Project - The Apache HTTP Server Project,« Apache Software Foundation, [Mrežno]. Available: https://httpd.apache.org/ABOUT_APACHE.html. [Pokušaj pristupa 01. 07. 2020.].

[8] Netcraft Ltd, »June 2020 Web Server Survey | Netcraft News,« Netcraft Ltd, 25. 05. 2020. [Mrežno]. Available: https://news.netcraft.com/archives/2020/06/25/june-2020-web-server-survey.html. [Pokušaj pristupa 01. 07. 2020.].

[9] V. Beal, »What is Apache Web Server? Webopedia Definition,« TechnologyAdvice, [Mrežno]. Available: https://www.webopedia.com/TERM/A/Apache_Web_server.html. [Pokušaj pristupa 01. 07. 2020.].

[10] M. Lovrić, »Sustav jedinstvene autentikacije korisnika | AAI@EduHr,« Srce - Sveučilišni računski centar Sveučilišta u Zagrebu, [Mrežno]. Available: https://www.aaiedu.hr/za-davatelje-usluga/za-web-aplikacije/sustav-jedinstvene-autentikacije-korisnika. [Pokušaj pristupa 01. 07. 2020.].

[11] L. Jayapalan i L. Morin, Oracle Database Database PL/SQL Language Reference, 18c, Redwood, CA, SAD: Oracle Corporation, 2019.

[12] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler i F. Yergeau, »Extensible Markup Language (XML) 1.0 (Fifth Edition),« World Wide Web Consortium (W3C), 26. 11. 2008. [Mrežno]. Available: https://www.w3.org/TR/xml/. [Pokušaj pristupa 01. 07. 2020.].

[13] P. Fennell, »Extremes of XML,« u *XML London 2013 Conference Proceedings*, London, 2013.

[14] Mozilla Corporation, »Learn to style HTML using CSS - Learn web development | MDN,« Mozilla Foundation, 01. 07. 2020. [Mrežno]. Available: https://developer.mozilla.org/en-US/docs/Learn/CSS. [Pokušaj pristupa 01. 07. 2020.].

[15] »What is CSS? - Learn web development | MDN,« Mozilla Foundation, 01. 07. 2020. [Mrežno]. Available: https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/What_is_CSS. [Pokušaj pristupa 01. 07. 2020.].

[16] B. Desruisseaux, »RFC 5545 - Internet Calendaring and Scheduling Core Object Specification (iCalendar),« Internet Engineering Task Force, Montreal, 2009.

[17] Microsoft Corporation, »See your Google Calendar in Outlook - Outlook,« Microsoft Corporation, [Mrežno]. Available: https://support.microsoft.com/en-us/office/see-your-google-calendar-in-outlook-c1dab514-0ad4-4811-824a-7d02c5e77126. [Pokušaj pristupa 01. 07. 2020.].

[18] Google, »Import events to Google Calendar - Computer - Calendar Help,« Google, [Mrežno]. Available: https://support.google.com/calendar/answer/37118. [Pokušaj pristupa 01. 07. 2020.].

[19] Apple Inc., »Subscribe to calendars on Mac - Apple Support,« Apple Inc., [Mrežno]. Available: https://support.apple.com/guide/calendar/subscribe-to-calendars-icl1022/mac. [Pokušaj pristupa 01. 07. 2020.].

[20] I. Lukić, »Baze podataka - Uvodno predavanje,« 28. 09. 2015. [Mrežno]. Available: https://loomen.carnet.hr/pluginfile.php/327491/mod_resource/content/5/BP%2001%20Uvodno%20predavanje.pdf. [Pokušaj pristupa 01. 07. 2020.].

[21] C. Cherian i C. Murray, Oracle SQL Developer Data Modeler User's Guide, Release 20.2, Redwood, CA, SAD: Oracle Corporation, 2020.

[22] »MRKVE - login,« Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek, [Mrežno]. Available: http://mrkve.etfos.hr/. [Pokušaj pristupa 01. 07. 2020.].

[23] I. Lukić, »Baze podataka - Normalizacija podataka,« 21. 10. 2015. [Mrežno]. Available: https://loomen.carnet.hr/pluginfile.php/338772/mod_resource/content/3/BP%2004%20Normalizacija%20podataka.pdf. [Pokušaj pristupa 01. 07. 2020.].

[24] S. Davila i T. Kaštelan, Modeliranje podataka, Zagreb: Algebra d.o.o., 2010.

[25] G. C. Everest, »Basic data structure models explained with a common example,« u *Fifth Texas Conference on Computing Systems*, Austin, TX, SAD, 1976.

[26] I. Lukić, »Baze podataka - Modeliranje podataka,« 12. 10. 2015. [Mrežno]. Available: https://loomen.carnet.hr/pluginfile.php/330877/mod_resource/content/3/BP 03 Modeliranje podataka.pdf. [Pokušaj pristupa 01. 07. 2020.].

[27] U. Krishnamurthy i M. B. Roeser, Oracle Database SQL Language Reference, 18c, Redwood, CA, SAD: Oracle Corporation, 2020.

[28] I. Lukić, »Baze podataka - Procedure, funkcije i okidači,« 26. 10. 2015. [Mrežno]. Available: https://loomen.carnet.hr/pluginfile.php/342183/mod_resource/content/2/BP%2007%20Procedure%2C%20funkcije%20i%20okida%C4%8Di.pdf. [Pokušaj pristupa 01. 07. 2020.].

[29] R. Manger, Baze podataka - Skripta, Zagreb: Matematički odsjek Prirodoslovno-matematičkog fakulteta Sveučilišta u Zagrebu, 2011.

[30] A. Nanda, »Autonomous Indexing,« Oracle Corporation, 15. 05. 2019. [Mrežno]. Available: https://blogs.oracle.com/oraclemagazine/autonomous-indexing. [Pokušaj pristupa 01. 07. 2020.].

[31] P. Kannan, L. Morin, D. Raphaely, L. Ashdown, D. Baker, D. Carver, M. Chaliha, B. Cheng, R. Day, S. Fogel, B. Llewellyn, P. Lane, D. McDermid, T. Morales, A. Murphy, C. Murray, S. Pelski, K. Rich, A. Romero, V. Schupmann, C. Shea, M. Taft, K. Taylor, R. Urbano, P. Huey, D. Adams, S. Surumpudi, R. Ward, R. Bhatiya, T. Choudhury, R. A. Kumar i A. Chaudhry, Oracle Database PL/SQL Packages and Types Reference, 18c, Redwood, CA, SAD: Oracle Corporation, 2019.

[32] S. Higgins, N. Agarwal, A. Agrawal, O. Alonso, S. Banerjee, M. Bauer, R. Booreddy, Y. Chan, S. Chandrasekar, V. Chao, M. Drake, F. Ge, W. He, T. Hoang, S. Idicula, N. Jalali, B. Khaladkar, V. Krishnamurthy, M. Krishnaprasad, W. Lin, A. Liu, A. Manikutty, J. Melnick, N. Montoya, S. Muench, R. Murthy, E. Paapanen, S. Pannala, J. Russell, E. Sedlar, V. Shah, C. Shea, T. Singh, S. Slack, M. Subramanian, A. Tarachandani, R. Urbano, P. Vennapusa i J.

Warner, Oracle9i XML Database Developer’s Guide - Oracle XML DB, Release 2 (9.2), Redwood, CA, SAD: Oracle Corporation, 2002.

[33] J. F. Allen, »Maintaining knowledge about temporal intervals,« *Communications of the ACM*, svez. 26, br. 11, p. 832–843, 1983.

[34] T. A. Alspaugh, »Allen's Interval Algebra,« 24. 01. 2019. [Mrežno]. Available: https://thomasalspaugh.org/pub/fnd/allen.html. [Pokušaj pristupa 01. 07. 2020.].

[35] P. Potineni, Oracle Database Data Warehousing Guide, 18c, Redwood, CA, USA: Oracle Corporation, 2019.

POPIS KRATICA

3NF	Treća normalna forma
APEX	Application Express
API	Application Programming Interface
BLOB	Binary Large Object
COBOL	Common Business Oriented Language
CSS	Cascading Style Sheets
DBMS	Database Management System
DDL	Data Definition Language
DML	Data Manipulation Language
DVD	Digital Video Disc
ER	Entity-Relationship
FERIT	Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek
GUID	Globally Unique Identifier
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ISO	International Organization for Standardization
ISVU	Informacijski Sustav Visokih Učilišta
Java EE	Java Enterprise Edition
JMBAG	Jedinstveni matični broj akademskog građanina
JSON	JavaScript Object Notation
JSP	Java Server Pages
MD5	Message Digest 5
MIME	Multipurpose Internet Mail Extension
ORDS	Oracle REST Data Services
PDF	Portable Document Format
PHP	PHP: Hypertext Preprocessor
PL/SQL	Procedural Language for SQL
RAM	Random Access Memory
REST	Representational state transfer
RFC	Request for Comments
SAML	Security Assertion Markup Language
SGML	Standard Generalized Markup Language
SQL	Structured Query Language
SUBP	Sustav upravljanja bazom podataka
UID	Universal Identifier
UML	Unified Modelling Language
URL	Uniform Resource Locator
UTC	Coordinated Universal Time
XML	Extended Markup Language
XSD	XML Schema Document

POPIS SLIKA

Slika 2.1. Struktura izvođenja PL/SQL programa, izvor: [11] 4
Slika 2.2. Prikaz nastavnog tjedna u Digitalnom rasporedu FERIT-a 6
Slika 2.3. Prikaz nastavnog tjedna u Academio aplikaciji 7
Slika 3.1. Primjer dostupnih podataka iz resursa "raspored" 10
Slika 3.2. Primjer jednostavnog ER dijagrama 12
Slika 3.3. Vizualizacija oznaka u notaciji vranina stopala 13
Slika 3.4. Sintaksa *CREATE TABLE* naredbe u Backus-Naurovoj formi, izvor: [27] 14
Slika 3.5. SQL naredba za stvaranje tablice *EVENT_SCHEDULE* 15
Slika 3.6. Sintaksa naredbe *CREATE VIEW* naredbe u Backus-Naurovoj formi, izvor: [27] 16
Slika 3.7. SQL naredba za stvaranje pogleda *CLASSES_FOR_STUDENT* 17
Slika 3.8. *SELECT* upit za dobivanje dijela individualnog rasporeda za pojedinog studenta 17
Slika 3.9. Rezultati upita iz Slike 3.8 18
Slika 3.10. Sintaksa naredbe *CREATE INDEX* naredbe u Backus-Naurovoj formi, izvor: [27] .. 18
Slika 3.11. Naredba za stvaranje indeksa nad stupcem i izrazom nad stupcem iz *EVENT_SCHEDULE* tablice 18
Slika 3.12. Sintaksa za *CREATE TRIGGER* naredbu u Backus-Naurovoj formi, izvor: [27] 19
Slika 3.13. Okidač nad *EVENT_SCHEDULE* tablicom za određivanje provjera znanja 20
Slika 4.1. Opis dijelova PL/SQL programskog bloka, izvor: [11] 23
Slika 4.2. Programski kod funkcije za dohvat HTTP resursa 24
Slika 4.3. Programski kod procedure za obradu XML dokumenta o kolegijima 25
Slika 4.4. Primjer kalendara s jednim neponavljajućim događajem 27
Slika 4.5. Procedura za generiranje *iCalendar* kalendara za pojedinog studenta 28
Slika 4.6. *INSERT* upit za kopiranje studentske nastave u inscenacijsku tablicu 30
Slika 4.7. *INSERT* upit za umetanje blokirajućih vremenskih termina u inscenacijsku tablicu ... 30
Slika 4.8. Primjer korištenja *MATCH_RECOGNIZE* funkcije za pronalaženje uzorka 32
Slika 4.9. Upit za izračun i pohranu slobodnih termina za svakog studenta 32
Slika 4.10. Definicija pogleda APEX$COFFEE_TIME_ALL_USERS 33
Slika 4.11. Upit za dohvat zajedničkih slobodnih termina grupe studenata 33
Slika 5.1. Prikaz naslovne stranice Oracle APEX programerskog sučelja 35
Slika 5.2. Čarobnjak za izradu aplikacija 36
Slika 5.3. Pregled novonastale aplikacije u *App Builderu* 37
Slika 5.4. Odabir tipa stranice 38

Slika 5.5. Unos osnovnih podataka o stranici 38
Slika 5.6. Podešavanje navigacije 38
Slika 5.7. Odabir izvora podataka 38
Slika 5.8. Konfiguracija stupaca za prikaz 38
Slika 5.9. Prikaz komponenti pojedine stranice 39
Slika 5.10. Isječak postavki za autentifikaciju iz Apache konfiguracijske datoteke 40
Slika 5.11: Prikaz personaliziranog tjednog rasporeda na mobitelu 41
Slika 5.12. AAI@EduHR prijavni prozor 41
Slika 5.13. Prikaz početne stranice 42
Slika 5.14. Mjesečni prikaz osobnog rasporeda 42
Slika 5.15. Prikaz detalja događaja 43
Slika 5.16. Obrazac za unos osobnog događaja 43
Slika 5.17. Unos ostalih kolega za računanje slobodnog vremena 44
Slika 5.18. Prikaz slobodnog vremena među kolegama 44
Slika 5.19. Prikaz informacija o korisniku 45

POPIS TABLICA

Tablica 3.1. Popis resursa u MRKVE sustavu 9
Tablica 3.2. Popis stupaca tablice *EVENT_SCHEDULE* 10
Tablica 4.1. Odnosi među vremenskim intervalima, izvor: [34] 31

SAŽETAK

Web aplikacija studentskog rasporeda uz podršku Oracle baze podataka

S pomakom tehnologije nastaje povećanje stupnja digitaliziranosti našeg svakodnevnog života. Cilj ovog rada jest razvoj web aplikacije koja svakom pojedinačnom studentu prikazuje raspored s njegovim nastavnim grupama i nudi mogućnost izvoza u druge aplikacije i platforme. Aplikacija je izrađena u Oracle APEX razvojnom okviru i platformi za razvoj aplikacija s malom količinom programskog koda. Prije toga je demonstriran postupak dizajniranja ER i relacijskog modela baze podataka kao i postupak dizajniranja i pisanja pogleda unutar baze podataka. Osim toga objašnjene su i pozadinske tehnologije i standardi na kojima su temeljene značajke aplikacije i pozadinskih servisa za dohvat i posluživanje podataka korisnicima. Na kraju je demonstriran rad responzivne web aplikacije za osobna računala i pametne telefone koja studentima omogućuje izračun i prikaz slobodnog vremena s njihovim kolegama, dodavanje osobnih događaja vezanih uz nastavu i studiranje u kalendar kao i prikaz i izvoz njihovih rasporeda u druge aplikacije.

Ključne riječi: APEX, baza podataka, Oracle, PL/SQL, Studentski raspored

ABSTRACT

Student schedule web application with Oracle database support

With the shift in technology comes an increase in the degree of digitization of our daily lives. The aim of this paper is to develop a web application that shows each individual student a schedule with their teaching groups and offers the possibility of exporting the timetable to other applications and platforms. The application is built in the Oracle APEX framework and "low-code" application development platform. Prior to that, the process of designing an ER and a relational database model, as well as the process of designing and writing views within a database were demonstrated. In addition, the background technologies, and standards on which the features of the application and background services for retrieving and serving data to users are based on are explained. Finally, a responsive web application for personal computers and smartphones was demonstrated that allows students to calculate and display free time with their colleagues, add personal study-related events to the calendar, as well as display and export their timetables to other applications.

Keywords: APEX, database, Oracle, PL/SQL, Student schedule

ŽIVOTOPIS

Filip Đuričković je rođen 1998. godine u Osijeku. Pohađao je Osnovnu školu Tenja u prigradskom naselju Tenji pokraj Osijeka. Nakon završene osnovne škole u Tenji, 2012. godine upisuje se u I. Gimnaziju Osijek, smjer opća gimnazija. Tijekom srednjoškolskog obrazovanja aktivno sudjeluje na *Infokup* natjecanjima iz informatike. Nakon završetka gimnazije, upisuje se na preddiplomski sveučilišni studij Računarstvo pri Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek. Područja interesa su mu razvoj bankarskih i ERP sustava i relacijskih baza podataka (posebice ORACLE i IBM Db2) kao i održavanje i programiranje *mainframe* sustava.

Sudjelovao je na Elektrijadi 2017. godine gdje je osvojio ekipno drugo mjesto iz Engleskog jezika, a na natjecanjima *STEM Games* 2018. godine osvojio je ekipno četvrto mjesto, dok je 2019. godine osvojio ekipno sedmo mjesto u *Technology* areni. Osim akademskih natjecanja, sudjelovao je i na IBM *Master the Mainframe* 2019 natjecanju. Također, tijekom studiranja, radi na raznim projektima i tijekom 2019. godine prijavljuje se na fakultetski natječaj za izradu projekata „Pro-student“ u kategoriji ZAPRO s projektom: „*FERIT.mobile* – maketa mobilne mreže druge generacije“. Projekt je odabran i financiran od strane Fakulteta, a završen je iste godine za što je i dobio priznanje Fakulteta. Izrada makete je rezultirala seminarskim radom „Izrada makete GSM mreže koristeći SDR radio“ iz kolegija Komunikacijske mreže za kojeg je dobio Rektorovu nagradu u akademskoj godini 2018./2019..

Predsjednik je studentske udruge IEEE studentski ogranak Sveučilišta J. J. Strossmayera u Osijeku i zamjenik je predsjednika Studentskog zbora FERIT-a te član predstavnik studenata u Povjerenstvu za unaprjeđenje i osiguranje kvalitete visokog obrazovanja na FERIT-u. Također, tijekom studija je povremeno zaposlen kao demonstrator na kolegijima Komunikacijske mreže i Programiranje II.

Tijekom ljeta 2019. godine odradio je stručnu praksu u Hrvatskoj narodnoj banci, a od 2020. godine je i stipendist Hrvatske narodne banke i zaposlen je u tvrtki CROZ na mjestu *mainframe* inženjera – pripravnika.

Filip Đuričković

PRILOZI

Na priloženom DVD disku su sadržani:

- Kopija baze podataka u *Oracle Data Pump* formatu
- SQL i PL/SQL skripte za stvaranje svih objekata u bazi podataka
- Kopija aplikacije Studentski raspored namijenjena uvozu u Oracle APEX okruženje
- Originalne datoteke dijagrama baze podataka izrađene u Oracle SQL Developer Data Modeler alatu
- PDF kopije ER dijagrama i relacijskog dijagrama