Aplikacija za vježbu računalnog razmišljanja

Kuretić, Lana

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1617

APLIKACIJA ZA VJEŽBU RAČUNALNOG RAZMIŠLJANJA

Lana Kuretić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1617

APLIKACIJA ZA VJEŽBU RAČUNALNOG RAZMIŠLJANJA

Lana Kuretić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

ZAVRŠNI ZADATAK br. 1617

Pristupnica:	**Lana Kuretić (0036524206)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	doc. dr. sc. Tomislav Jagušt

Zadatak: **Aplikacija za vježbu računalnog razmišljanja**

Opis zadatka:

Računalno razmišljanje (eng. computational thinking) je način razmišljanja koji uključuje procese i strategije rješavanja problema slične onima koje koriste računala. Smatra se jednom od ključnih vještina 21. stoljeća i sve češće se uvodi u kurikulum već u osnovnim školama kako bi se učenicima omogućilo stjecanje temeljnih vještina potrebnih za uspješno funkcioniranje u digitalnom društvu. U sklopu ovog završnog rada potrebno je osmisliti i izraditi aplikaciju koja će učenicima pomoći da se upoznaju s osnovnim konceptima računalnog razmišljanja i logikom računalnog programiranja. Uz aplikaciju treba izraditi pripadajuću bazu podataka i materijale koji u nizu lekcija učenike uvode u područje računalnog i algoritamskog razmišljanja. Kako bi proces učenja bio zanimljiviji, svaka lekcija gradi priču koja korisnika motivira i vodi ka konačnom cilju. Osim aplikacije, potrebno je izraditi i popratne obrazovne videomaterijale, provesti testiranje sustava te analizirati i opisati dobivene rezultate.

Rok za predaju rada: 14. lipnja 2024.

Mom partneru, koji je tijekom cijelog mog studija imao negraničenu količinu strpljenja.

Hvala ti na svemu.

Sadržaj

1. Uvod

Desetljeće po desetljeće, računalna znanost napreduje sve intenzivnije i brže. Nakon utvrđivanja temeljnih znanja vezanih za teoriju računarstva, komunikacijske mreže, baze podataka, umjetnu inteligenciju i druga teorijski relevantna područja, struka se sada fokusira i na direktnu primjenu istih znanja u svakodnevnom životu. Nove tehnologije postaju sve pristupačnije i intuitivnije prosječnom korisniku, što znači da se polako, ali sigurno integriraju u razne dijelove društva.

Svakim danom sve je više aplikacija, sustava i alata koji korisniku pomažu i na radnom mjestu i u privatnom životu. Jedno od radnih mjesta koje računalna znanost planirano obogaćuje [1] upravo je prosvjeta, odnosno obrazovni sustav. Tu činjenicu potvrđuju i promjene u obrazovnom kurikulumu Republike Hrvatske [2] koji nastavnom predmetu Informatika iz godine u godinu pridaje sve veću pažnju.

Kontinuiranu integraciju računalne znanosti u obrazovni sustav možemo promatrati na dva načina. S jedne strane, jedan od glavnih proizvoda računarstva upravo je softver, a on kroz razne obrazovne tehnologije obogaćuje proces poučavanja i učenja te omogućava i nastavnicima i učenicima lakšu komunikaciju te efikasnije dijeljenje sadržaja. S druge strane, učenici se kroz školski predmet Informatika upoznaju s temeljnim idejama računalne znanosti te uče kako koristiti informacijske i komunikacijske tehnologije.

Teorijska znanja, unatoč tome što su bitna, danas nisu jedini fokus predmeta Informatika [2]. Redovitim izmjenama kurikuluma učenike se motivira u rješavanju problema i ispitivanju vlastitog kritičkog razmišljanja. Na taj način, iako potencijalno neće nastaviti obrazovanje u smjeru računalne znanosti, učenik svoja znanja može prenositi u druga životna polja.

Uvođenjem izbornog predmeta Informatike u niže razrede osnovne škole [3] dodatno

se naglašava koliko je ključno da učenik u što ranijoj dobi počne razvijati generičke kompetencije predmeta Informatika [2], kao što su: kreativnost i inovativnost stvaranjem digitalnih uradaka i algoritama, kritičko mišljenje i vrednovanje tehnologije i izvora znanja, rješavanje problema i donošenje odluka s pomoću informacijske i komunikacijske tehnologije, informacijska i digitalna pismenost razumijevanjem i konstruktivnim razgovorom o pojmovima iz područja informatike.

1.1. Odgojno-obrazovni ishodi predmeta Informatika

Kurikulum predmeta Informatika prezentira četiri domene [2] kroz koje se realiziraju ciljevi i generičke kompetencije spomenute u prethodnom ulomku. To su: *e-Društvo, Digitalna pismenost i komunikacija, Računalno razmišljanje i programiranje* te *Informacije i digitalna tehnologij*a.

Slika 1.1. Četiri domene kojima se realiziraju ciljevi nastavnog predmeta Informatika [2]

Predviđeno je da se navedene domene (Slika 1.1.) međusobno isprepliću i nadopunjavaju te da imaju brojne dodirne točke. U nastavku opisujemo svrhu svake od domena upravo onako kako ih prezentira sâm kurikulum. [2].

Informacije i digitalna tehnologija domena je koja naglašava snagu računala u smislu obrade i pohrane velikih količina podataka. Istaknuta je važnost poznavanja različitih izvora i oblika podataka, ali i načina pohrane te prijenosa istih. Učenik mora razviti sposobnost odabira ispravne tehnologije i načina uporabe za određeni tip informacije, odnosno podatka.

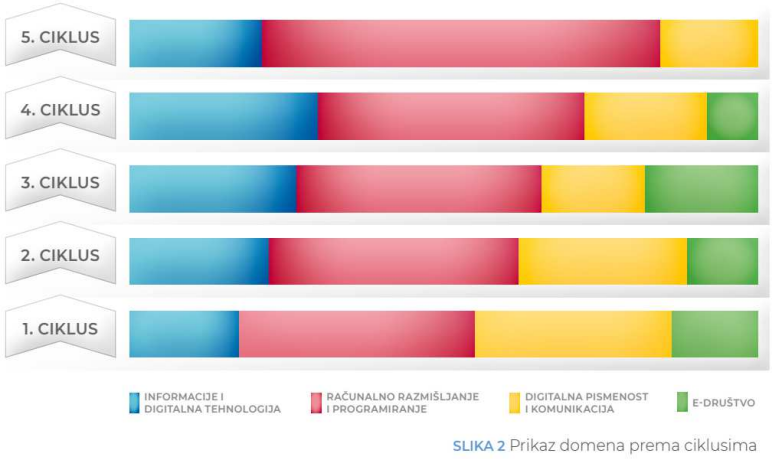
Digitalna pismenost i komunikacija domena je koja ističe važnost vještina suradnje i komunikacije u *online* okruženju koje je sve veće i prisutnije. Digitalna pismenost uključuje komunikacijske i društvene vještine svojstvene netehničkim granama društva, ali i odabir prikladnih digitalnih sadržaja te programa za komunikaciju. Pozitivan stav i odgovornost naglašeni su kao motivatori učenika i nastavnika.

E-društvo kao domena uključuje znanja o elektroničkim uslugama koje su dostupne pojedincu. Tehnologija danas olakšava pristup obrazovanju, ali i razonodi te potiče aktivno sudjelovanje u demokraciji. Učenicima se želi omogućiti odgovorno, sigurno i učinkovito upotrebljavanje komunikacijskih mreža te dobro poznavanje javno dostupnih usluga u kontekstu zdravlja i obrazovanja. Ova domena intenzivno se veže uz informacijske i komunikacijske tehnologije.

Računalno razmišljanje i programiranje, domena koja će nam biti u fokusu, njeguje proces rješavanja problema i utvrđuje kako je isti primjenjiv na računalu. Radi se o univerzalnoj vještini koja je primjenjiva u brojnim drugim životnim poljima. Vrednuju se inovativnost, upornost, preciznost i sposobnost jasne komunikacije s ostatkom grupe.

1.2. Računalno razmišljanje i programiranje

S obzirom na to da je tema završnog rada izrada aplikacije koja pomaže u vježbi računalnog razmišljanja, pobliže ćemo proučiti ovu domenu. Osnovnoškolsko obrazovanje podrazumijeva 70 sati Informatike godišnje u svim razredima [2].



Slika 1.2. Prikaz četiri domene odgojno-obrazovnih ciljeva prema obrazovnim ciklusima [2]

Crveni dio grafičkog prikaza (Slika 1.2.) koji prikazuje odnos nastavnih domena reprezentira *Računalno razmišljanje i programiranje*. Navedena domena u svih pet obrazovnih ciklusa zauzima najveći dio satnice, što ukazuje na važnost usvajanja određenih načina razmišljanja i kognitivnih alata koje nudi upravo rješavanje problema (engl. *problem solving*).

Računalno razmišljanje (engl. *computational thinking*) je misaona aktivnost formuliranja i rješavanja problema na način kako bi to izvelo računalo, iako ne nužno uz pomoć računala [4]. Postoje i druge definicije računalnog razmišljanja, no sve uključuju koncept univerzalnih procesa razmišljanja i prepoznavanja uzoraka tijekom rješavanja nekog problema.

Unutar domene računalnog razmišljanja prepoznate su sljedeće komponente: *dekompozicija, apstrakcija, algoritmi, otklanjanje pogrešaka, iteracija* i *generalizacija* [5]. U nastavku slijedi konkretizacija komponenti u kontekstu obrazovnog sustava [4].

Dekompozicija se bavi rastavljanjem kompleksnijih problema ili sustava na manje i lakše rješive dijelove. U sustavu obrazovanja, dekompozicija se često vježba kroz algoritme, karakteristike dobrih algoritama, koncept modularnosti i modeliranje funkcija.

Apstrakcija za zadatak ima prepoznavanje srži nekog problema ili sustava. Koncept apstrakcije dobro je poznat i u objektno-orijentiranoj paradigmi gdje se entiteti, odnosno podaci, moraju modelirati na način da su prikazana samo ona njihova svojstva koja su relevantna sustavu u kojem se entitet nalazi [6]. U obrazovnom sustavu, apstrakcija se poučava kroz obradu podataka i uočavanje veza između uzoraka.

Algoritmi su ključan pojam u računarstvu. Učenicima se prenosi znanje dizajniranja uređenih i pravilnih instrukcija koje postižu određeno rješenje, obično odgovor na neki problem. Osim dizajniranja tih algoritama, u obrazovnom sustavu učenici uočavaju svojstva dobrih algoritama te se upoznaju s konceptima efikasnosti i modularnosti.

Otklanjanje pogrešaka komponenta je koja motivira analiziranje i razumijevanje vlastitih, ali i tuđih algoritama. Tijekom nastave, učenici pišu i ispravljaju vlastite programe, ali i uočavaju česte greške koje ih poučavaju dobrim praksama u programiranju.

Iteracija je komponenta računalnog razmišljanja koja motivira proces ponovnog pro-

laska kroz algoritam i ponovnog razmišljanja o problemu dok se ne dođe do funkcionalnog ili efikasnijeg rješenja [4].

Generalizacija je interdisciplinarna komponenta računalnog razmišljanja. Ona naglašava da proces dolaska do rješenja korišten u radu s računalom može biti primjenjiv i u drugim životnim sferama. Kroz fakultetski obrazovni sustav javlja se i koncept oblikovnih obrazaca koji prikazuju već generalizirane, dokazano efikasne metode modeliranja u kojima je smanjena međuovisnost komponenti [7].

1.3. Motivacija za izradu aplikacije

Računalno razmišljanje smatra se jednom od ključnih vještina 21. stoljeća. S obzirom na to da je jedna od domena u odgojno-obrazovnim ishodima kurikuluma za predmet Informatika [2] u Republici Hrvatskoj, i nastavnici i inženjeri razvijaju pristupe i alate koji će olakšati učenicima proces usvajanja gradiva. Proces učenja možemo obogatiti i učiniti zanimljivijim uvođenjem elemenata običnih i računalnih igara.

Igrifikacija (engl. *gamification*) svodi se na korištenje elemenenata igara izvan konteksta u kojem se igra obično provodi [8]. Kao koncept se u obrazovnom sustavu koristi već dugo, najčešće kroz društvene igre, slagalice i izazove [9], a napredak računalne znanosti i ubrzanje razvoja softvera otvorili su velik broj novih mogućnosti.

Obrazovne aplikacije inspirirane igrifikacijom dodatno se obogaćuju igraćim mehanizmima koji uključuju misije, razine, virtualne diplome, tablice poretka, praćenje napretka i stvaranje avatara [8]. U aplikaciji koja je tema ovog završnog rada implementirano je praćenje napretka i preuzimanje virtualne diplome. Iz pedagoških razloga, kako bi se motiviralo kvalitetno učenje, a ne brzina rješavanja, tablica poretka dostupna je samo učitelju koji je povezan s određenom skupinom učenika.

U ovom završnom radu bit će opisana aplikacija za vježbanje računalnog razmišljanja pod nazivom *Šumska škola*. Radi se o web aplikaciji predviđenoj za korištenje na računalima te se razlikuju dvije uloge: uloga učitelja i uloga učenika. Svaki učitelj može stvoriti proizvoljan broj grupa koji korespondira s razredima ili predmetima. Učenika u grupu pozivamo putem poveznice specifične za grupu, generirane u web aplikaciji.

Web aplikacija kroz svoje glavno sučelje nudi niz autorskih lekcija koje pokrivaju odgojno-obrazovne ishode zadane kurikulumom predmeta Informatika [2] te istovremeno uvode ključne termine u programiranju. Svaka lekcija sadrži obrazovni videozapis, odnosno videolekciju (Slika 1.3.) koja kroz priču o stanovnicima šume (*Šumske škole*) uvodi termine iz domene programiranja i računalnog razmišljanja. Uz obrazovni videozapis, svaka lekcija ima i zadatak koji provjerava razumijevanje naučenog.

Slika 1.3. Isječak obrazovnog videozapisa iz prve lekcije

Iako aplikacija pokriva gradivo od prvog do osmog razreda osnovne škole, složenost lekcija i način izražavanja krojeni su za peti razred osnovne škole. U nastavku su prikazane obrazovne lekcije dostupne kroz aplikaciju, a redom su navedeni: zadatak koji lekcija provjerava (korespondira ishodu učenja opisanom u kurikulumu [2]), razred osnovne škole u kojem se ishod očekuje (po kriterijima u kurikulumu [2], no razredi su grupirani u parove), komponenta računalnog razmišljanja (ili više njih) koju zadatak provjerava i koncept u domeni programiranja (ili više njih) koji se uvodi u lekciji.

ZADATAK	RAZRED	KOMPONENTE	KONCEPTI
uočiti točne upute	1./2.	algoritmi	algoritam
uočiti korak koji nedostaje	1./2.	algoritmi, dekompozicija	beskonačni algoritmi
vrednovati značajke algoritma	1./2.	algoritmi	značajke algoritma
koristiti odgovarajuću strukturu podataka	3./4.	algoritmi, apstrakcija	varijabla, konstanta
koristiti odgovarajuću strukturu podataka	3./4.	algoritmi, dekompozicija	lista
uočavanje potrebe za grananjem kôda	3./4.	algoritmi, dekompozicija	grananje
modeliranje *if-else* grananja	5./6.	algoritmi, apstrakcija	*if-else* grananje
slijedne *if* selekcije	5./6.	algoritmi, apstrakcija	*if* selekcija
upotreba prikladnog grananja	5./6.	algoritmi, apstrakcija	tôk kôda
uočiti ponavljajući kôd	5./6.	algoritmi, dekompozicija	petlja
modelirati *for* petlju	5./6.	algoritmi, uzorci	*for* petlja
modelirati *while* petlju	5./6.	algoritmi, uzorci	*while* petlja
upotreba prikladnog mehanizma ponavljanja	5./6.	algoritmi, apstrakcija	tôk kôda
modelirati funkciju	5./6.	dekompozicija, apstrakcija	funkcija
pretraživati listu podataka	7./8.	algoritmi	slijedna pretraga
koristiti strukturu reda	7./8.	apstrakcija	red
koristiti strukturu stoga	7./8.	apstrakcija	stog

Tablica 1.1. Autorske lekcije u sklopu *Šumske škole*

Na ovaj način, učenik može kroz konkretne lekcije dobiti upravo ona znanja koja se po obrazovnom kurikulumu Republike Hrvatske [2] smatraju ishodima učenja u domeni računalnog razmišljanja, a uz to i ključnu terminologiju koja se proteže kroz sve programske jezike te ne ovisi o implementaciji.

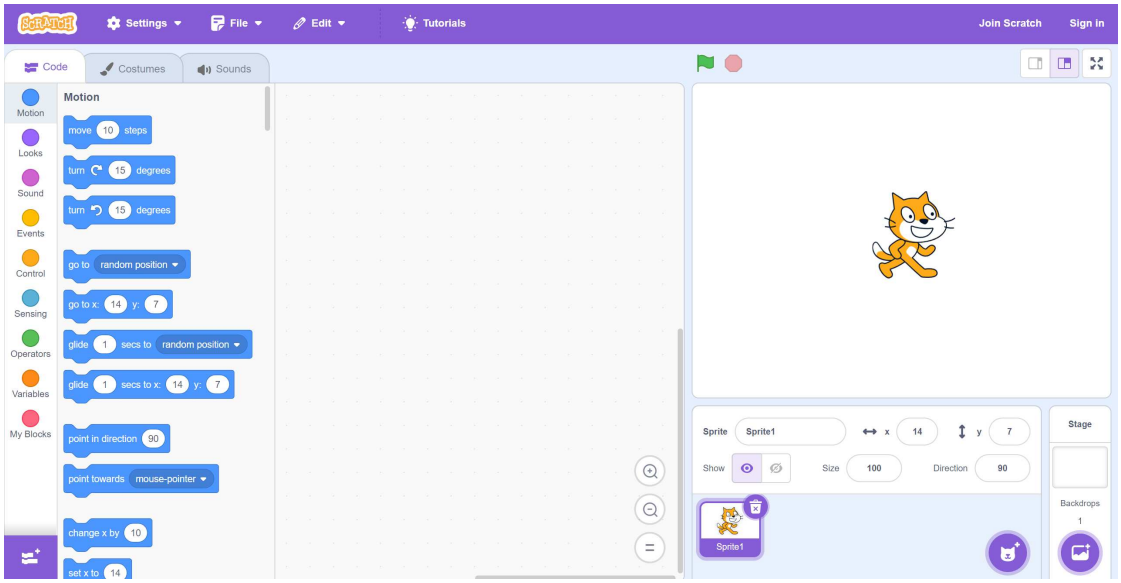
Ideja web aplikacije je pružanje zaigranog, a obrazovnog prostora za upoznavanje s računalnim razmišljanjem. Učenik samostalno prolazi kroz ponuđene lekcije koje se sastoje od obrazovnih videozapisa i zadataka, a uz to ima i mogućnost kontaktiranja nastavnika u slučaju dileme. Na taj način svaki učenik može raditi svojim tempom i istovremeno dobiti adekvatnu podršku u procesu učenja. Više o funkcionalnostima sustava i korištenim tehnologijama slijedi u narednim poglavljima.

2. Slične obrazovne platforme

Prije raspisivanja funkcionalnih i nefunkcionalnih zahtjeva aplikacije izrađene u sklopu završnog rada bilo je potrebno proučiti već postojeće obrazovne platforme, alate i igre. U sklopu tog proučavanja istaknule su se prednosti i mane određenih programskih sustava na koje je kasnije, u razvoju, usmjerena veća količina pažnje.

2.1. Programski jezik Scratch

Programski jezik *Scratch* [10] jezik je visoke razine čija ciljna publika uključuje djecu od osam do šesnaest godina. Koristi se kao alat u obrazovanju i smatra se vizualnim programskim jezikom zbog toga što koristi gotove, funkcionalne blokove koji se razmještaju pomoću računalnog miša. Razvijen je na sveučilištu MIT.

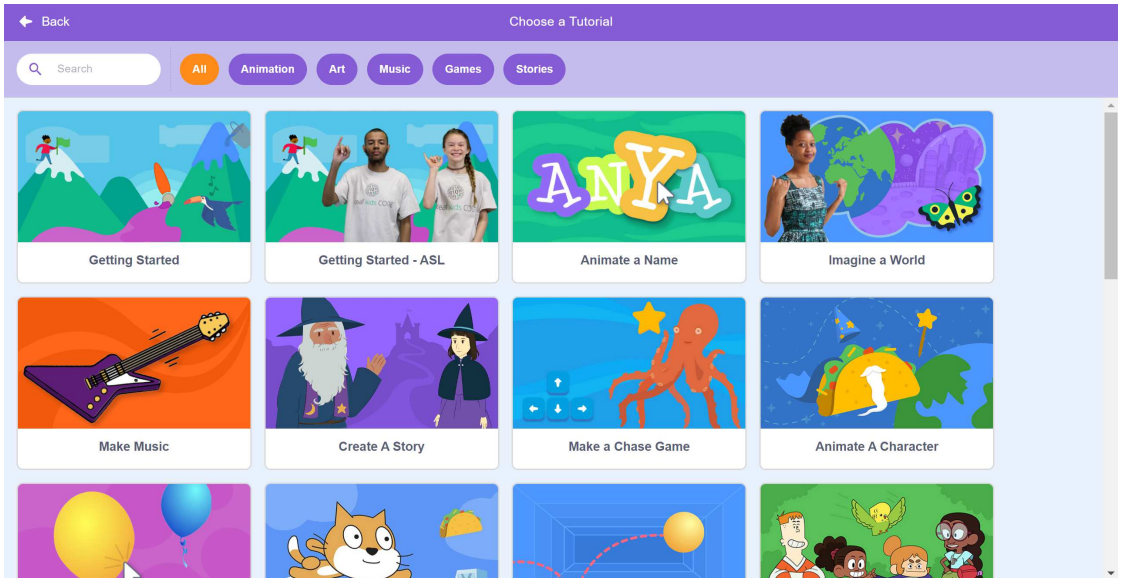


Slika 2.1. Prikaz korisničkog sučelja aplikacije *Scratch* [10]

Korisnike se potiče da sami stvaraju svoje projekte korištenjem gotovih funkcija podijeljenih u kategorije: *motion, looks, sound, events, control, sending, operators* i *variables*.

Kategorije su vidljive na lijevoj strani korisničkog sučelja (Slika 2.1.). Neki od blokova korespondiraju funkcijskim ulazima, drugi manipuliraju podacima, a treći se svode na neki oblik ispisa ili drugi tip obavještavanja korisnika.

Zapravo se motivira izrada algoritama bez poznavanja sintakse nekog programskog jezika, oslanjanjem na vizualne komponente. *Scratch* također nudi velik broj ideja i uputa za izradu specifičnih projekata (Slika 2.2.), a same upute se prikažu u centru korisničkog sučelja (Slika 2.1.) i kroz njih se prolazi korak po korak, što je vrlo praktično.



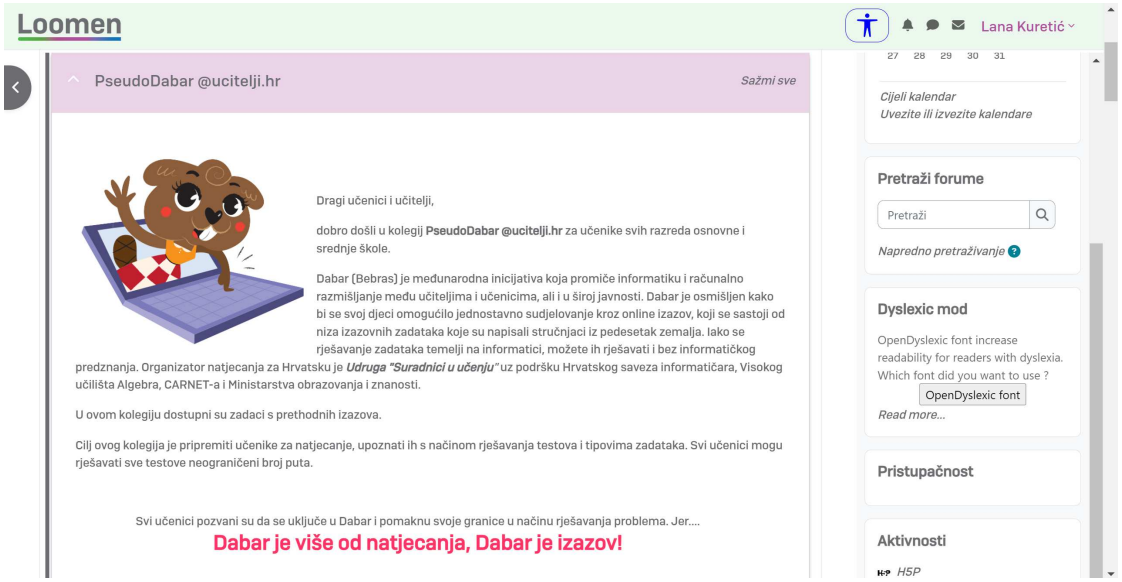
Slika 2.2. Različiti projekti s pripadajućim uputama

Ovaj besplatan softver odličan je alat za poučavanje računalnog razmišljanja zbog toga što ne opterećuje učenika sintaksom programskog jezika, već koncepte poučava na razini iznad. Iako tema ovog završnog rada nije mogla ostati isključivo na konceptualnoj razini, naglasak je i dalje stavljen na konceptualno razumijevanje algoritama, a ne prepoznavanje programskih kôdova. Ukoliko se u lekciji koristi programski kôd, korištena sintaksa odgovara programskom jeziku *Python*.

2.2. Virtualna učionica PseudoDabar

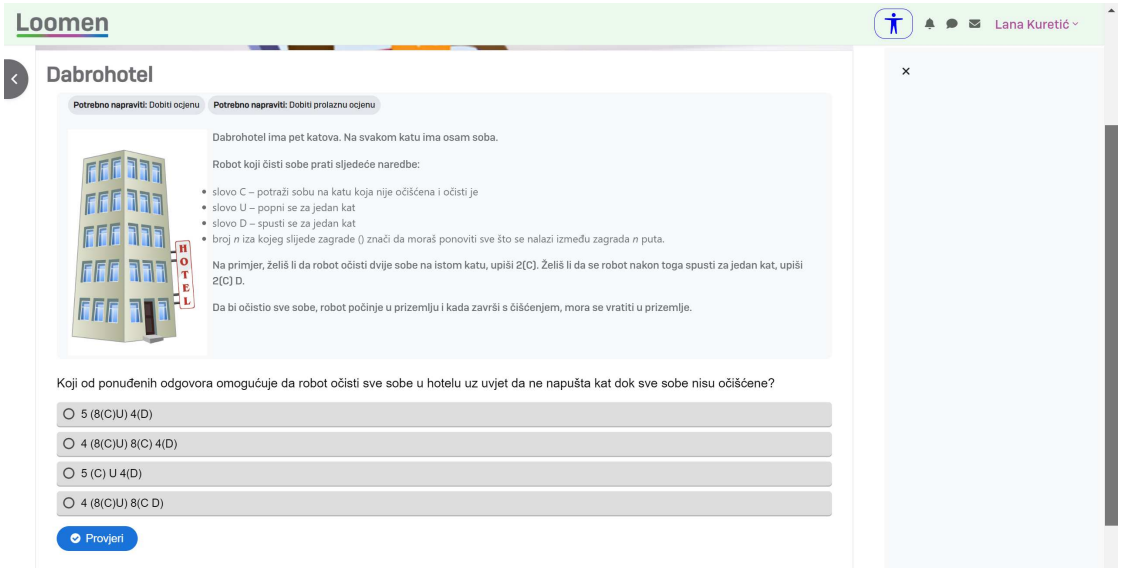
PseudoDabar[11] je virtualni kolegij za učenike svih razreda osnovne i sredne škole, a nastao je kao međunarodna inicijativa koja promiče računalno razmišljanje. Svake godine održava se natjecanje pod imenom Dabar te je ova virtualna učionica (Slika 2.3.) upravo priprema za navedeno natjecanje. Bilo tko u sustavu CARNET (čak i studenti te

osoblje fakulteta) mogu pristupiti ovom portalu.



Slika 2.3. Početna stranica virtualne učionice

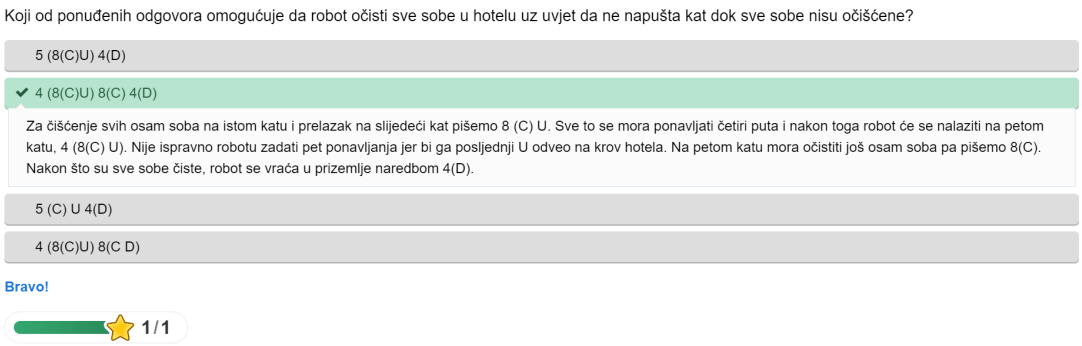
Nakon prijavljivanja s *AAI@EduHr* računom, učenici i nastavnici mogu pristupiti interaktivnim zadacima u kojima se vježbaju logika i računalno razmišljanje. Prikazan je (Slika 2.4.) jedan od zadataka koji vježba prepoznavanje bitnih elemenata priče (apstrakcija), razvijanje algoritma koji zadovoljava uvjet zadatka (algoritmi), ali i pseudozapis rješenja (generalizacija).



Slika 2.4. Primjer zadatka: Dabrohotel

Posebno koristan koncept u samoj virtualnoj učionici jest dostupnost povratne informacije nakon odgovaranja na pitanje (Slika 2.5.). Netočan odgovor obojan je crvenom

bojom te je argumentirano zašto taj odgovor nije točan. Točan odgovor obojan je zelenom bojom te je još jednom obrazloženo zašto je on ispravan. Ovaj koncept svakako je uključen i u aplikaciju koja je tema ovog završnog rada.

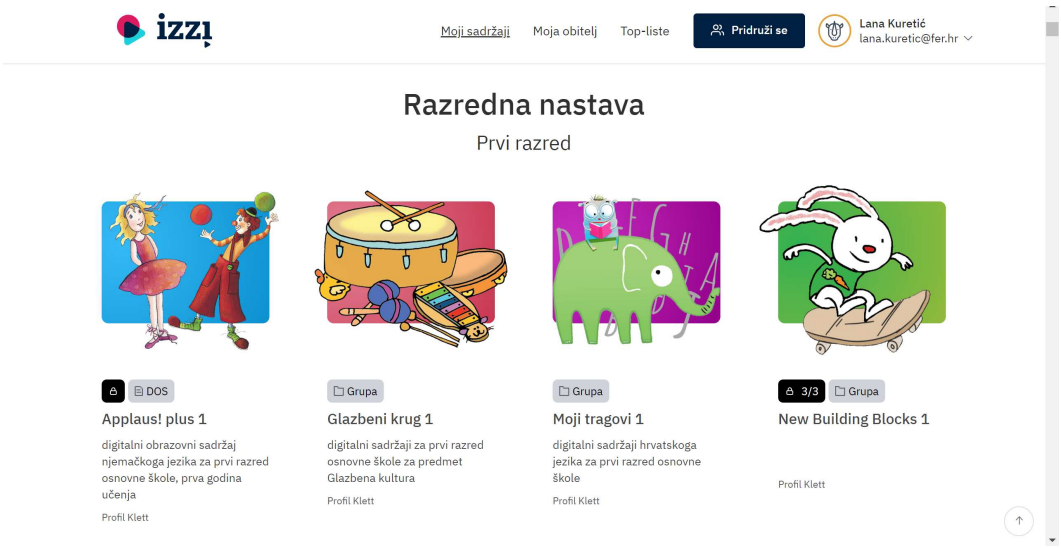


Slika 2.5. Povratna informacija za točno odgovoreno pitanje

PseudoDabar [11] je besplatan i dostupan na materinjem jeziku te kombinira znanja matematike, logike i informatike na jednom mjestu. Portal je prilagođen i osobama s invaliditetom te implementira brojne asistivne tehnologije.

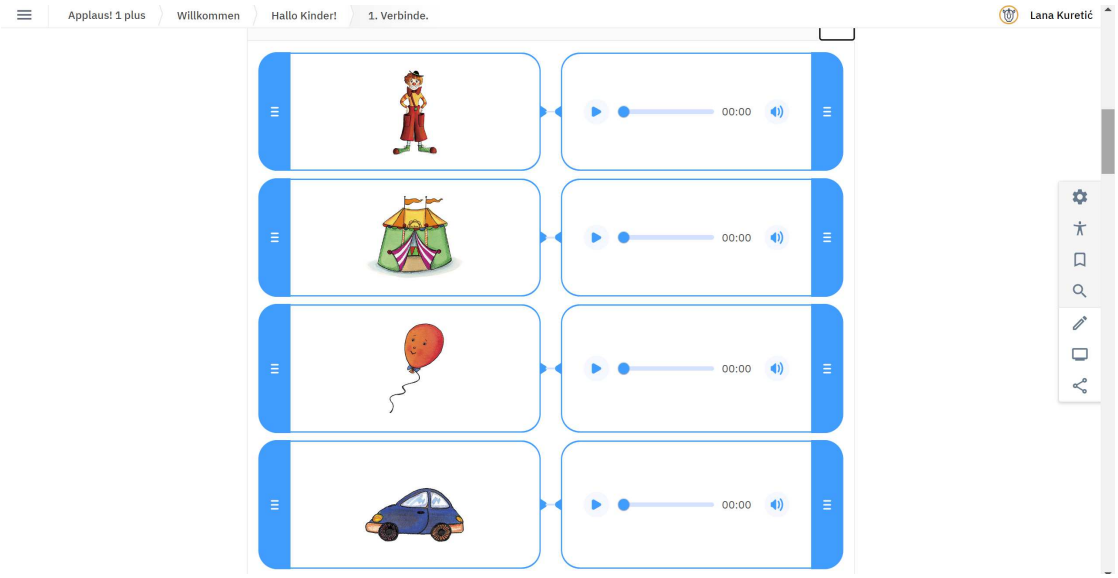
2.3. Platforma IZZI

Platforma *IZZI* [12] projekt je nakladničke kuće *Profil Klett* koji obogaćuje njihove udžbenike interaktivnim igrama i dodatnim digitalnim materijalima. Kad se osoba prijavi u sustav, prikazane su joj brojne lekcije (Slika 2.6.), no samo je mala količina otključana. Ostatak sadržaja otključava se šiframa koje se dobivaju uz udžbenike.



Slika 2.6. Naslovna stranica portala IZZI [12]

U primjeru (Slika 2.7.) otvorena je lekcija koja obogaćuje udžbenik njemačkog jezika i ponuđene su interaktivne igre koje omogućavaju djetetu slušanje izgovora stranih riječi i spajanje istih s točnim prikazima. S obzirom na to da je otključan sadržaj bio onaj za prvi razred osnovne škole, teško je reći bilo što o kompleksnosti interaktivnih zadataka za učenike viših razreda i koliko oni zaista motiviraju računalno razmišljanje.



Slika 2.7. Interaktivan zadatak za lekciju Willkomen

Vizualni aspekt portala obogaćen je ilustracijama iz udžbenika te privlači pažnju i malih i velikih, ali je također i dosta tekstualnog sadržaja. U aplikaciji koja je tema ovog završnog rada izrađen je vizualni identitet koji zelenim nijansama izaziva smirujuće i ugodne osjećaje te su umjesto tekstualnih objašnjenja korištene autorske snimke u kojima se gradivo tumači detaljnije.

3. Opis sustava

Šumska škola je web aplikacija za vježbanje računalnog razmišljanja namijenjena djeci u višim razredima osnovne škole. Lekcije su kompleksnošću i gradivom orijentirane primarno na peti razred, no zapravo obuhvaćaju većinu ključnih osnovnoškolskih obrazovnih ishoda specificiranih u kurikulumu [2].

Primarni zadatak web aplikacije jest upoznavanje učenika s temeljnim konceptima programiranja, a naglasak jest na svrhovitosti svakog koncepta. Na taj način učenik razmišlja o tome zašto je neki koncept potreban i gdje se koristi, odnosno prolazi kroz misaoni proces koji će mu kasnije koristiti u rješavanju zadataka. *Šumska škola* formatirana je kao platforma na kojoj se sluša tečaj o računalnom razmišljanju.

Sama aplikacija dozvoljava učenicima registraciju isključivo putem posebnih poveznica koje stvaraju učitelji. Učitelji se također mogu prijaviti u aplikaciju, no ne mogu slušati lekcije, već prate napredak učenika koji su svrstani u grupe. Navedene grupe učitelji stvaraju i imenuju sami, a potom generiraju poveznice putem kojih se učenici registriraju i automatski se povezuju s grupom.

Jednom kad se učenik prijavi u sustav, prikazuju mu se podaci o napretku, popis svih lekcija, ali i mogućnost da ispiše svoju diplomu jednom kad prođe sve lekcije. Svaka od lekcija nudi učeniku i teoriju i praksu jer se sastoji od autorske videolekcije u kojoj autorica završnog rada poučava neki koncept iz programiranja te zadatka koji provjerava je li učenik poslušao videolekciju i razumije li koncept.

Videolekcije (Slika 1.3.) snimane su uz autorsku prezentaciju koja se uvijek sastoji od tri ista dijela: neka problematika iz šumskog svijeta, rješenje koje bi moglo riješiti tu problematiku i zaključak koji ukazuje na ključni koncept koji se uveo kroz rješenje problema. Ukoliko učenik zadatak ispod videolekcije riješi točno, dobiva potvrdu zašto

je upravo taj odgovor točan, a ukoliko ga ne riješi točno, dobiva smjernicu zašto odabrani odgovor nije rješenje zadatka. U pedagoške svrhe uvedeni su i vremenski penali od deset sekundi nakon netočnog odgovora.

Učenici koji zadatak riješe pogrešno dobivaju takozvanu *SOS* opciju koja šalje obavijest učitelju koji je povezan s učenikom. Učitelj ima informaciju na kojoj lekciji je učenik, ali i u koje vrijeme je poslao zahtjev. Osim toga, učitelj ima uvid i u napredak svakog učenika svake grupe koju je stvorio.

4. Model podataka

Poslužiteljska strana aplikacije putem objektno-relacijskog mapiranja komunicira sa PostgreSQL bazom podataka, ali koristi i dodatnu inicijalizacijsku *TypeScript* datoteku (engl. *seed*) koja u procesu generiranja, migracije i punjenja baze u nju unosi početne informacije. U nastavku su prikazane strukture najbitnijih komponenata aplikacije.

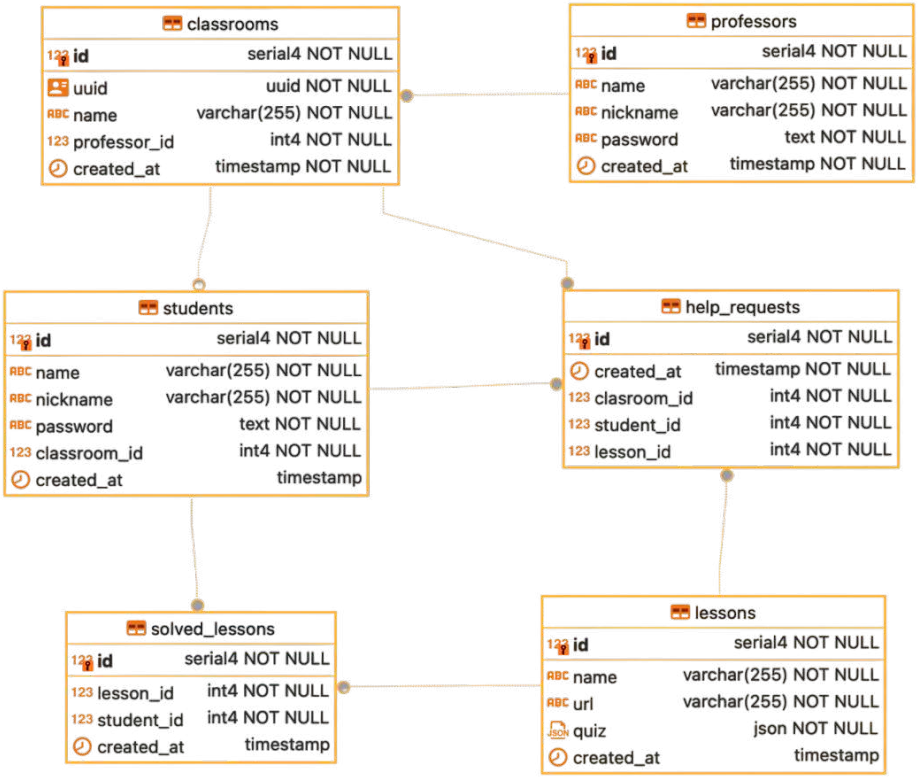
4.1. Model baze podataka

Aplikacija koristi relacijsku bazu podataka, što znači da su različiti entiteti međusobno povezani preko stranih ključeva (engl. *foreign key*). Ovakav pristup razvoju bio je intuitivan iz razloga što su svi entiteti sustava međusobno dobro povezani. U nastavku je prikazan *ER* dijagram (engl. *entity-relationship diagram*) izvezen iz alata *DBeaver* [13] te su objašnjeni glavni entiteti (Slika 4.1.).

Relacija *professors* sadrži podatke o učiteljima koji su trenutno uneseni u sustav aplikacije. Inicijalizacijska *TypeScript* datoteka unosi jednog učitelja, a administrator ih može dodati još ili kroz njenu modifikaciju ili kroz modifikaciju same baze podataka. Svaki učitelj opisan je jedinstvenim identifikatorom, imenom, nadimkom i lozinkom kojima se prijavljuje, a također ima i vremensku oznaku stvaranja entiteta zbog boljih kontrolnih ispisa u aplikaciji.

Relacija *classrooms* sadrži podatke o grupama koje su učitelji stvorili. Grupa je opisana jedinstvenim identifikatorom, opisnim imenom, vremenskom oznakom stvaranja, ali i stranim ključem koji ukazuje na učitelja koji ju je stvorio. Na taj način relacija *classrooms* povezana je s relacijom *professors*.

Relacija *students* sadrži podatke o učenicima koji su registrirani. Svaki učenik opisan je jedinstvenim identifikatorom, imenom, nadimkom, lozinkom, vremenskom oznakom



Slika 4.1. Dijagram relacijske baze podataka

stvaranja, ali i stranim ključem koji ukazuje na grupu iz relacije *classrooms* kojoj učenik pripada na temelju poveznice putem koje se registrirao.

Relacija *lessons* sadrži podatke o lekcijama koje su dostupne kroz sustav. Ova se relacija puni pomoću inicijalizacijske *TypeScript* datoteke koja sadrži informacije o samim lekcijama te posljedično sadrži jedinstveni identifikator lekcije, ime lekcije, poveznicu prema videolekciji, zadatak koji je zadan uz lekciju (formatiran u *JSON* obliku) te vremensku oznaku stvaranja.

Relacija *solved_lessons* most je između relacije *lessons* i relacije *students*. Ona prati učenikov napredak te sadrži lekcije koje su uspješno odslušane i riješene. Takve su lekcije opisane jedinstvenim identifikatorom, vremenskom oznakom stvaranja, informacijom o točnosti rješenja (formatiranu u *boolean* obliku) te stranim ključevima prema rješavanoj lekciji (iz relacije *lessons*) i učeniku koji rješava (iz relacije *students*).

Za kraj, relacija *help_requests* sprema učeničke upite za pomoć prema učiteljima. Svaki je zahtjev za pomoć opisan jedinstvenim identifikatorom, vremenskom oznakom stvaranja te sa tri strana ključa. Prvi strani ključ pokazuje na grupu kojoj učenik pri-

pada (iz relacije *classrooms*) te posljedično povezuje učenika i učitelja, drugi pokazuje na samog učenika koji traži pomoć (iz relacije *students*), dok treći pokazuje na lekciju u vezi koje se traži pomoć (iz relacije *lessons*). Na taj način možemo uspješno formatirati obavijest (Slika 6.9.) koju učitelj prima kad učenik zatraži pomoć.

4.2. Inicijalizacijska *TypeScript* datoteka

Već spomenuta inicijalizacijska *TypeScript* datoteka zapravo sadrži sve početne informacije kojima se baza podataka puni pri prvom pokretanju aplikacije. Ona sadrži dva tipa objekta: objekti koji opisuju lekcije i objekti koji opisuju učitelje.

Lekcije su opisane imenom, poveznicom prema videolekciji (koja je prenesena na *YouTube*) te kvizom koji se sastoji od pitanja i liste odgovora. Svaki odgovor sastoji se od tri komponente: ponuđeni odgovor, točnost odgovora i smjernica koja se pokazuje kad učenik odgovori na pitanje klikom na baš taj odgovor (ili upozorenje zašto je neki odgovor netočan ili potvrda zašto je neki odgovor točan).

```
{
      name: 'Puž hvata (algo)ritam',
      url: 'https://www.youtube.com/embed/TKAscZrxzXY?si=-
   uDn5xu9yRpy3AtP',
      quiz: {
        question: 'Koju izmjenu u algoritmu smijem napraviti?',
        answers: [
          {
            answer: 'Smijem maknuti prvi korak.',
            correct: false,
            hint: 'Kada bismo maknuli prvi korak, puž se uopće ne bi
   probudio pa se ne bi mogao niti spremiti za putovanje. Dakle, ovo
   nije točan odgovor.'
          },
         ...
          {
            answer: 'Smijem između trećeg i četvrtog koraka dodati
   pranje zubića.',
            correct: true,
```

```
            hint: 'Puž je u drugom koraku pojeo smrdljivi doručak pa ima smisla da opere zube tek nakon! Točno!'
        },
      ]
    }
  },
```

Listing 4..1: Primjer objekta koji opisuje lekciju

Učitelji uneseni u inicijalizacijsku datoteku opisani su imenom, nadimkom i lozinkom koja je pohranjena u obliku kriptografskog sažetka (engl. *hash*), a generirana je pomoću funkcije *bcrypt* [14].

```
{
    name: 'Lana Kuretić',
    nickname: 'lmk',
    password: '$2b$10$4Sy5vw3IMELgcQghDO.l1Ov33Eb2UQBvLxxyBE6759UvTvgDnweRe',
  }
```

Listing 4..2: Primjer objekta koji opisuje učitelja

5. Korištene tehnologije i alati

Prije no što detaljno objasnimo uloge pojedinih tehnologija u razvoju web aplikacije *Šumska škola*, promotrit ćemo krupnu sliku. Aplikacija koristi arhitekturu klijent-poslužitelj (engl. *client-server architecture*) pa posljedično korištene tehnologije možemo podijeliti u dvije skupine: tehnologije na klijentskoj strani i tehnologije na poslužiteljskoj strani.

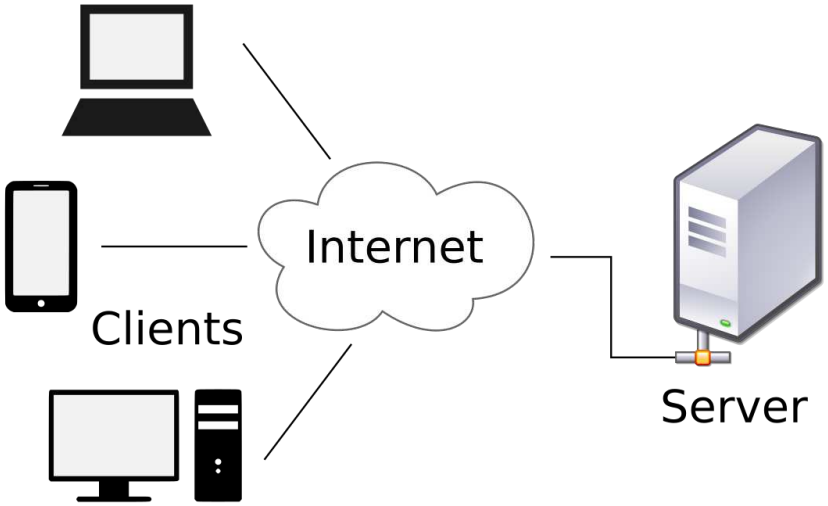
Na klijentskoj strani koristimo lokalni razvojni poslužitelj *Vite* [15] i biblioteku *React* [16] te razvoj klijentske strane temeljimo na komponentama. Dodatno, na klijentskoj je strani i *Tailwind* [17], razvojni okvir za CSS koji olakšava stiliziranje aplikacije.

Na poslužiteljskoj strani koristimo *Node.js* koji je okruženje za izvođenje JavaScript kôda, *Fastify* [18] koji se instalira unutar tehnologije *Node.js* te služi kao razvojna tehnologija za poslužiteljsku stranu i *drizzle* [19], sustav za objektno-relacijsko mapiranje koji nam omogućuje komunikaciju poslužitelja s bazom podataka. Dodatno korišteni paketi uključuju *axios* koji služi za obradu *HTTP* zahtjeva na klijentskoj strani te *postgres* koji omogućuje povezivanje baze podataka s poslužiteljem.

I klijentska i poslužiteljska strana koriste *TypeScript* [20] u svrhu obvezne tipizacije svih varijabli i komponenti. Na taj način povećava se čitkost programskog kôda i smanjuje se šansa za greškom, što svakako utječe na sigurnost aplikacije.

5.1. Klijent-poslužitelj arhitektura

Arhitektura klijent-poslužitelj primjer je raspodijeljenog sustava i jedna je od najrasprostranjenijih arhitektura u razvoju programskih rješenja. S jedne strane imamo poslužitelja (engl. *server*) koji dostavlja uslugu, a s druge klijenta (engl. *client*) koji pristupa poslužitelju i traži uslugu. Jedan poslužitelj u pravilu poslužuje veliki broj klijenata te klijenti znaju informacije poslužitelja [21].



Slika 5.1. Arhitektura klijent-poslužitelj [22]

Klijenti putem preglednika pristupaju željenom resursu na Internetu te ga traže od poslužitelja specificirajući njegov URL (engl. *Uniform Resource Locator*). Taj se zahtjev šalje prema pregledniku *HTTP*-om (engl. *HyperText Transfer Protocol)* koji omogućuje komunikaciju diljem Interneta. Ta se komunikacija zapravo svodi na razmjenu *HTTP* zahtjeva i *HTTP* odgovora u čijim se zaglavljima nalaze bitne informacije koje jedinstveno identificiraju komunikaciju između klijenta i poslužitelja.

U *HTTP* odgovoru poslužitelja nalazi se resurs koji je klijent specificirao te se isti šalje natrag prema klijentu. Identifikatori u procesu slanja zapravo su *IP* (engl. *Internet Protocol*) adrese koje se tijekom usmjeravanja neke informacije ne mijenjaju. Pomoćni protokoli u ovoj komunikaciju uključuju *DNS* (engl. *Domain Name Server*) koji služi za povezivanje simboličke i numeričke *IP* adrese nekog entiteta te *TLS* (engl. *Transport Layer Security*) koji osigurava uspješnu isporuku informacija u točnom redoslijedu [23].

U web aplikaciji koja je tema završnog rada predviđeno je da klijenti budu preglednici stolnih i prijenosnih računala koji će slati *HTTP* zahtjeve poslužitelju koji se nalazi na računalu autorice završnog rada čija je odgovornost pokrenuti poslužitelj u trenutku korištenja aplikacije.

5.2. *TypeScript*

S obzirom na to da brojne poslužiteljske i klijentske tehnologije koriste programski jezik *JavaScript*, intuitivno je bilo izabrati ili njega ili njegovu tipiziranu verziju, *TypeScript*. Kako bi sigurnost aplikacije bila na višoj razini i proces programiranja bio stroži, ali i jasniji, izabran je *TypeScript*, a u nastavku je istaknuta njegova ključna značajka.

Za razliku od programskog jezika *JavaScript*, njegov "omotač" *TypeScript* nudi već gotove tipove podataka, ali i mogućnost izrade vlastitih. Bitno je napomenuti da *TypeScript* ima sve funkcionalnosti svog "pretka", a dodatno nudi i provjeru ispravnosti pridruživanja vrijednosti "tipiziranim" varijablama, argumentima, parametrima i genericima [20].

Na taj način razvojni proces postaje lakši i razvojno okruženje u kojem radimo može nas upozoriti na potencijalne greške u formatiranju, pretvorbi, unosu ili ispisu podataka, što je od velike važnosti kod opširnijih i/ili skalabilnih aplikacija.

5.3. *Vite*

Za razvoj klijentske strane bio je potreban lokalni razvojni poslužitelj, a s obzirom na to da ima gotove predloške projekata za *React* i podršku za *TypeScript*, izabran je *Vite*. On, između ostalog, omogućuje brži i jednostavniji razvoj te automatsko ažuriranje novih unosa u programskom kôdu.

5.4. *React*

React je razvojna biblioteka za programski jezik *JavaScript* (posljedično i *TypeScript*) koja se svodi na korištenje komponenti. Osim što možemo koristiti vanjske biblioteke koje nude gotove komponente i nadomjestive funkcije (mi smo koristili *react-router-dom*, recimo), omogućena nam je i opcija razvijanja vlastitih komponenti.

Iste formiraju stablo komponenti u čijem je korijenu *App* komponenta te se to stablo uz pomoć *react-dom* paketa pretvara u virtualni *DOM* (engl. *Document Object Model*) poznat pregledniku. Komponente su formirane u obliku funkcija te vraćaju *HTML* kôd.

```
import LoginPage from "./modules/auth/LoginPage";
import RegisterPage from "./modules/auth/RegisterPage";
import AuthLayout from "./modules/auth/AuthLayout";
import AccountHelpPage from "./modules/auth/AccountHelpPage";
import ProfessorLayout from "./modules/professor/ProfessorLayout";
import ClassroomsPage from "./modules/professor/ClassroomsPage";
import authLoader from "./modules/auth/AuthLoader";
import Root from "./modules/Root";
import StudentLayout from "./modules/student/StudentLayout";
import LessonsPage from "./modules/student/LessonsPage";
import LessonQuizPage from "./modules/student/LessonQuizPage";
import HelpRequestsPage from "./modules/professor/HelpRequstsPage";
import DiplomaPage from "./modules/student/DiplomaPage";
```

Listing 5..1: Većina komponenti korištenih u aplikaciji

Od ključne važnosti u ovoj biblioteci su i nadomjestive funkcije (engl. *hooks*) koje nam omogućavaju manipulacije komponentama. Najčešće korištene nadomjestive funkcije u našoj aplikaciji uključuju funkciju *useState()* dostupnu kroz samu biblioteku te funkcije *useQuery()*, *useMutation()* i *useQueryClient()* dostupne kroz *TanStack Query* (*react-query*) [24], biblioteku koja pomaže u dohvaćanju, sinkronizaciji i ažuriranju podataka između klijentske i poslužiteljske strane koristeći lokalni spremnik (engl. *cache*).

```
function App() {
  const router = createRouter(queryClient)
  return (
    <QueryClientProvider client={queryClient}>
      <RouterProvider router={router} />
      <ReactQueryDevtools initialIsOpen={false} />
    </QueryClientProvider>
  )
}
```

Listing 5..2: Postavljanje TanStack biblioteke u korijensku komponentu

Nakon što u korijensku komponentu dodamo *queryClient*, možemo koristiti njegove nadomjestive funkcije za provjeru lokalnog spremnika. Na taj način ne moramo iznova dohvaćati podatke koji su već dohvaćeni (i vezani uz neki ključ), a ako se neki podaci mijenjaju, možemo ručno javiti alatu da informacije koje on sadrži više nisu najnovije.

```
export default function HelpRequestsPage() {
  const queryClient = useQueryClient();
  const user =  queryClient.getQueryData<User>(['user'])!;

  const { data: helpRequests, isPending: isHelpRequestsPending } =
   useQuery({
    queryFn: () => getHelpRequests({ professorId: user.id }),
    queryKey: ['help-requests', user.id],
    enabled: !!user
  });

  const { mutate: deleteHelpRequestMutation, isPaused:
   isHelpRequestDeleting } = useMutation({
    mutationFn: deleteHelpRequest,
    onSuccess: () => {
      queryClient.invalidateQueries({ queryKey: ['help-requests', user
   ] });
    },
  });
  return (...)
}
```

Listing 5..3: Korištenje nadomjestivih funkcija biblioteke react-query

Zadnja stavka vrijedna izdvajanja unutar razvojne biblioteke *React* jest paket *react-router-dom* koji nam je poslužio za usmjeravanje kroz komponente aplikacija. Korištenjem predviđene funkcije *createRouter* i definiranjem liste objekata u kojima su upareni putevi i komponente stvorili smo sustav za navigaciju s klijentske strane.

```
const ROUTES: RouteObject[] = [
    {
      path: '/',
      Component: Root,
    },
    {
      Component: AuthLayout,
      children: [
        {
          path: '/login',
          Component: LoginPage,
```

```
        },
        {
          path: '/register',
          Component: RegisterPage,
        }
      ]
    },
    ...
}
```

Listing 5..4: Isječak funkcije za usmjeravanje

Navedena funkcija samo uparuje određene komponente i puteve u aplikaciji kako bi se na klijentskoj strani mogla ostvariti navigacija, a stvarne *HTTP* zahtjeve naše arhitekture klijent-poslužitelj (Slika 5.1.) obrađuje *HTTP* klijent pod imenom *axios*.

5.5. *axios*

Axios se instalira u okruženje *Node.js* i služi za obradu naših *HTTP* zahtjeva i odgovora. Njegove predviđene funkcije *post*, *get* i *delete* u aplikaciji su korištene kako bismo stvarne zahtjeve uz odgovarajuće tipizirane podatke poslali na određene puteve te njihove odgovore pravilno obradili (u poslužiteljskoj tehnologiji *Fastify*).

```
type CreateClassroomRequest = {
  name: string;
  professorId: number;
}
type CreateClasroomResponse = {
  id: number;
}
export function createClassroom(info: CreateClassroomRequest) {
  return axios.post<CreateClasroomResponse>('/classrooms', info).then(
   res => res.data);
}
```

Listing 5..5: Obrada HTTP zahtjeva za stvaranje učionice

5.6. *Tailwind*

Kako bi aplikacija bila vizualno atraktivna, klijentska strana morala je biti stilizirana tehnologijom *CSS* (engl. *Cascading Stylesheets*) ili nekim razvojnim okvirom koji ga koristi. Izabran je jedan od najpopularnijih razvojnih okvira za *CSS*, *Tailwind*. Njegova glavna karakteristika jest stiliziranje putem već gotovih klasa, ali i onih personaliziranih [17].

```
/** @type {import('tailwindcss').Config} */
export default {
  theme: {
    fontFamily: {
      'normalni': ["Nunito Sans"],
      'ukrasni': ["Madimi One"]
    },
    ...
    extend: {
      colors: {
        primary: '#adc178',
        secondary: '#835c45',
        background: '#fffbed'
      },
      }
    },
  },
  plugins: [],
}
```

Listing 5..6: Isječak konfiguracijske Tailwind datoteke

Iznad prikazana konfiguracijska datoteka specificira korištenu paletu boja i tipografiju, a potom se definirane boje, tipografija i fotografije koriste u drugim *TypeScript* datotekama koje poslužuju natrag *HTML* (engl. *HyperText Markup Language*).

```
<div className="flex flex-col gap-4 w-full items-center">
          <span className="text-lg text-background">
            Hej {user.name}, na {percentSolved}% si!
          </span>
          <div className="w-full border border-secondary bg-background
    p-0.5 rounded-full">
            <div className={`z-10 p-2 bg-secondary rounded-full`}
```

```
    style={{width: `${percentSolved}%`}}>
          </div>
        </div>
      </div>
```

Listing 5..7: Primjer korištenja Tailwind klasa

5.7. *Node.js*

Poslužiteljska strana aplikacije prvenstveno je realizirana uz *Node.js*, okruženje u kojem se *JavaScript* (pa tako i *TypeScript* koji se u njega prevodi) kôd može izvoditi izvan preglednika. Izabran je, između ostalog, kako bismo mogli koristiti razvojni okvir *Fastify* te kako bismo mogli upravljati potrebnim modulima putem datoteke *package.json*.

5.8. *Fastify*

Ključna poslužiteljska tehnologija upravo je *Fastify*, poslužiteljski razvojni okvir koji se instalira u *Node.js*. On je bio idealan izbor za našu aplikaciju zbog podrške za *TypeScript* i već gotovog predloška za pokretanje poslužitelja.

Obrada *HTTP* zahtjeva u poslužiteljskom okviru *Fastify* svodi se na rad s dodacima (engl. *plugins*) koji nude registriranje puteva u aplikaciji te povezivanje istih puteva s funkcijama obrade (engl. *handlers*). U primjeru niže jest put karakterističan za stvaranje nove grupe tijekom koje se šalje *POST* zahtjev prema poslužitelju s učiteljskog klijenta. Funkcija obrade komunicira s bazom podataka te podatke primljene od učitelja prosljeđuje na trajnu pohranu. U pozadini se koristi i tehnologija *axios*.

```
type CreateClasroom = {
  name: string;
  professorId: number;
};

const clasroomsRoutes: FastifyPluginAsync = async (server:
   FastifyInstance, options: FastifyPluginOptions) => {
  const db = server.db;

```

```
  server.post<{ Body: CreateClasroom }>('/classrooms',  { preHandler:
   [server.prof] }, async (request, reply) => {
    const res = await db.insert(classrooms)
      .values(request.body)
      .returning({ id: classrooms.id });

    return reply.code(201).send(res[0].id)
  });
};
```

Listing 5..8: Primjer obrade klijentskog zahtjeva na poslužitelju

5.9. *Drizzle*

U aplikaciji je korišten i alat za objektno-relacijsko mapiranje. To je koncept koji povezuje *SQL* bazu podataka, odnosno njene relacije sa zapisima objektno-orijentirane paradigme, obično putem klasa ili funkcija koje nude *ORM* biblioteke.

Izabran je *Drizzle*, alat koji pomoću ručno zadanih shema relacija i međusobnih odnosa između relacija (putem *npm run generate*) generira prikladnu migracijsku skriptu za stvaranje, modifikaciju i brisanje podataka unutar baze podataka, a ista se kasnije koristi u upitima. Svaka od relacija definirana je predviđenom funkcijom *pgTable* koja definira stupce svake relacije te njihove karakteristike i poveznice između relacija.

```
export const helpRequests = pgTable('help_requests', {
  id: serial('id').primaryKey(),
  createdAt: timestamp('created_at').defaultNow().notNull(),
  classroomId: integer('clasroom_id').references(() => classrooms.id).
   notNull(),
  studentId: integer('student_id').references(() => students.id).
   notNull(),
  lessonId: integer('lesson_id').references(() => lessons.id).notNull
   (),
});
```

Listing 5..9: Stvaranje relacije putem biblioteke drizzle

Dodatno možemo pojasniti međusobne odnose relacija tako što ćemo definirati spojnost. Ista će nam biti od velike koristi pri slanju upita u bazu podataka.

```
export const helpRequestsRelations = relations(helpRequests, ({ one }) => ({
  classroom: one(classrooms, {
    fields: [helpRequests.classroomId],
    references: [classrooms.id]
  }),
  ...
}));
```

Listing 5..10: Isječak iz pojašnjenja odnosa dvije relacije

Kad definiramo shemu (Kôd 5.9.), kroz ostatak programskog kôda možemo jednostavno slati upite u bazu podataka asinkronim funkcijama i sintaksom samog *ORM* alata.

```
    const classroom = (await db.query.classrooms.findFirst({
      where: eq(classrooms.id, student.classroomId)
    }))!;
```

Listing 5..11: Primjer upita u bazu podataka

5.10. *postgres*

Biblioteka *node-postgres* u aplikaciji nam služi za komuniciranje između poslužitelja i baze podataka, ali je bitno napomenuti da je u njihovu komunikaciju uključen i *Drizzle ORM* koji zapravo služi kao njihov zajednički jezik.

```
import { drizzle } from 'drizzle-orm/postgres-js';
import fp from 'fastify-plugin';
import postgres from 'postgres';
import * as schema from '../schema.js';
const DB = async (fastify) => {
    if (!process.env.DATABASE_URL) {
        throw new Error('DATABASE_URL must be specified');
    }
    const client = postgres(process.env.DATABASE_URL);
    const db = drizzle(client, { schema, logger: true });
    fastify.decorate('db', db);
};
export default fp(DB);
```

Listing 5..12: Pristup PostgreSQL bazi podataka uz ORM

5.11. *DBeaver*

Jedan od korištenih alata u procesu razvoja bio je i alat za prikaz i upravljanje bazom podataka, *Dbeaver*[13]. Radi se o besplatnom alatu koji se koristi za popularne SQL baze podataka, pa tako i za našu PostgreSQL bazu podataka. Omogućio nam je pregled baze podataka po shemama, ali i slanje testnih upita (engl. *query*) koje smo kasnije koristili u aplikaciji.

6. Demonstracija aplikacije

U nastavku je opisano korisničko iskustvo aplikacije iz dvije različite perspektive: prvo prolazimo kroz iskustvo učenika, a onda iskustvo učitelja. Iako su te dvije uloge neizbježno povezane, ovakva sistematizacija olakšava demonstraciju.

6.1. Demonstracija iz uloge učenika

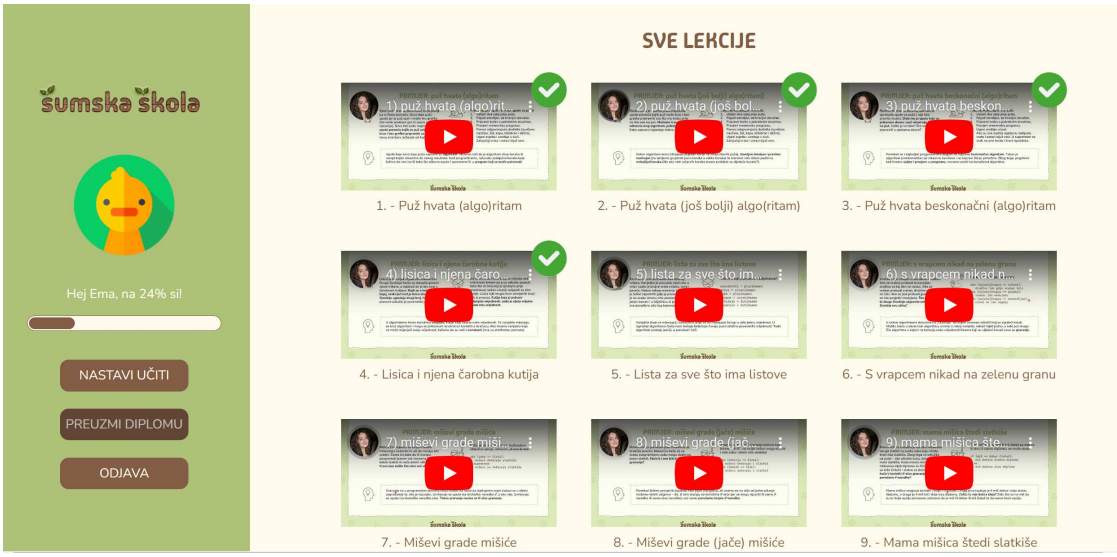
Učenici web aplikaciji *Šumska škola* pristupaju putem poveznice koju im šalju učitelji. Ta poveznica u sebi sadrži indikator grupe kojoj će učenici posljedično registraciji pripadati. Učenici se registriraju svojim imenom i prezimenom, nadimkom te lozinkom od barem osam znakova.

Slika 6.1. Registriranje u web aplikaciji *Šumska škola*

U procesu prijavljivanja učenici unose svoj nadimak i lozinku te na taj način pristupaju početnoj stranici web aplikacije. Za sada nije implementiran niz upozorenja koji bi upućivao na greške tijekom prijave pa je to napomenuto u poglavlju vezanom za nadogradnje.

Slika 6.2. Prijava u aplikaciju *Šumska škola*

Na početnoj stranici web aplikacije (engl. *dashboard*) učenik ima uvid u lekcije koje su za njega pripremljene, a kvačicom su označene one koje je već uspješno prošao, odnosno one čiji je zadatak točno riješio. S lijeve strane učenik može vidjeti svoj nasumično dodijeljen *avatar* te napredak kroz lekcije prikazan kroz postotak u svrhu igrifikacije. Gumb za preuzimanje diplome onemogućen je dok sve lekcije nisu uspješno dovršene.



Slika 6.3. Početna stranica aplikacije *Šumska škola*

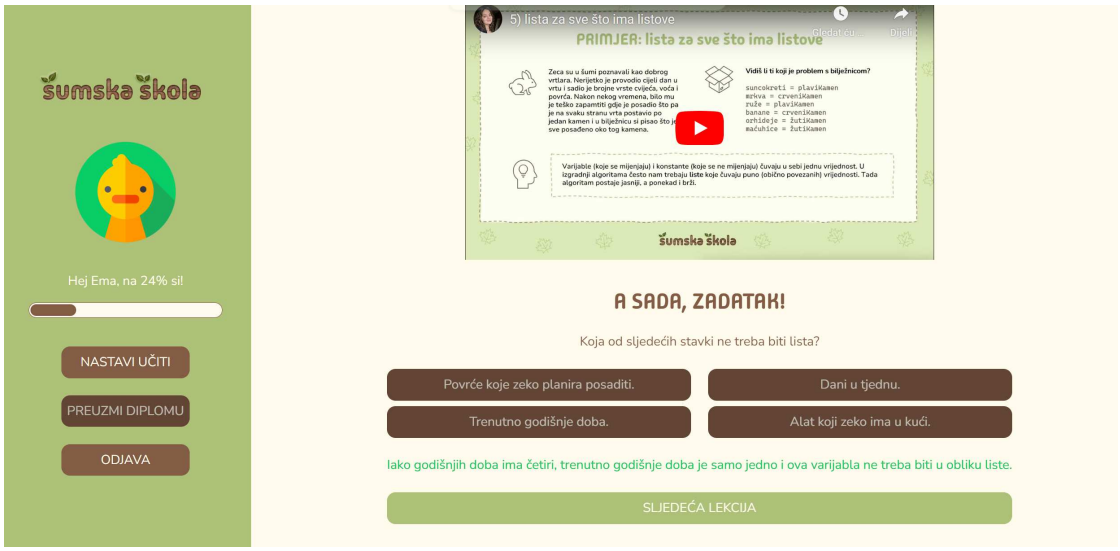
Svaka od lekcija sastoji se od videolekcije i zadatka. Videolekcija je u obliku *iframe* elementa uvezena s platforme *YouTube* na koju su videolekcije postavljene nakon snimanja i obrade u obliku popisa za reprodukciju [25]. Ispod videolekcije nalazi se zadatak koji ispituje upravo ispričano gradivo.

Slika 6.4. Primjer lekcije u aplikaciji *Šumska škola*

Ako učenik odabere pogrešan odgovor, prikazuju mu se dvije nove funkcionalnosti. Prva jest traženje pomoći od učitelja putem *SOS* gumba, a druga jest prikaz pomoćne smjernice koja se za netočne odgovore prikazuje u crvenoj boji, a za točne u zelenoj. Iz pedagoških razloga, kako ne bi samo nasumično isprobavali odgovore, učenicima je na deset sekundi onemogućeno ponovno odgovaranje na pitanje.

Slika 6.5. Primjer netočno riješene lekcije u aplikaciji *Šumska škola*

Kod točnih odgovora, pomoćna smjernica zapravo je potvrda koja dodatno utvrđuje gradivo. Na taj način učenici još jednom na raspolaganju imaju ključno znanje iz lekcije te odmah mogu preći na sljedeće pitanje. Ovaj je koncept preuzet iz virtualne učionice Dabar (Slika 2.4.).



Slika 6.6. Primjer točno riješene lekcije u aplikaciji *Šumska škola*

Točnim rješavanjem zadnje lekcije otključava se gumb za preuzimanje diplome. Ona se može ispisati pomoću gumba za ispis koji pokreće dijalog preglednika te je zapravo još jedan od elemenata igrifikacije. Diploma je uvedena umjesto tablice poretka koja uspoređuje učenike međusobno jer je naglasak na osobnom procesu svakog učenika.

Slika 6.7. Uspješno osvojena diploma u aplikaciji *Šumska škola*

6.2. Demonstracija iz uloge učitelja

Učitelji nemaju opciju registracije, već se prijavljuju u sustav podacima koje dobivaju od administratora. Za njih prijava izgleda kao i za učenike (Slika 6.2.), a početna stranica koju vide ima dvije funkcionalnosti: pregled grupa učenika i pregled *SOS* zahtjeva.

Prva funkcionalnost (Slika 6.8.) je stvaranje grupa koje mogu, ali ne moraju korespondirati razredima. Učitelj potom može kopirati poveznicu koju šalje učenicima u svrhu registriranja i povezivanja s grupom te posljedično ima uvid u učenički napredak.

Slika 6.8. Početna stranica za učitelje u aplikaciji *Šumska škola*

U sekciji *SOS* zahtjevi, učitelj proučava zahtjeve koje su učenici poslali u procesu učenja. On vidi ime učenika, grupu kojoj učenik pripada, datum i vrijeme zahtjeva, ali i lekciju koja učenika zbunjuje. S obzirom na to da je svrha *Šumske škole* ta da učenici zaista nauče gradivo, a ne samo da prođu lekcije, nudi se i ova vrsta suradnje sa školskim osobljem. Time zaključujemo funkcionalnosti koje su dostupne učiteljima i učenicima unutar aplikacije *Šumska škola*.

Slika 6.9. SOS zahtjevi u aplikaciji *Šumska škola*

7. Revizija razvojnog procesa

Aplikacija *Šumska škola* u sebi sadrži sve funkcionalnosti koje su u fazi specifikacije i raspisa funkcionalnih zahtjeva za nju i predviđene. Aplikacija nije testirana u obrazovnom okruženju, no interno testiranje pokazalo je da sve implementirane funkcionalnosti rade onako kako je to zamišljeno.

Prvi od koraka koji bi svakako omogućio nadogradnju i ispravljanje do sada skrivenih problema u aplikaciji bilo bi testiranje iste u raznim osnovnim školama. Na taj način, osim grešaka u samoj aplikaciji, dobili bismo dojam treba li aplikacija neke pedagoške preinake, odnosno nalaze li učenici neke "rupe u sustavu" koje im omogućuju da aplikaciju koriste na nepredviđene ili nejasne načine.

Osim toga, moguće je dodati i nove tipove zadataka. Svaka lekcija mogla bi biti provjeravana kroz nekoliko tipova zadataka, a osim trenutnog tipa koji je klasični "jedno pitanje, četiri odgovora" oblik upitnika, opcije su i kartice (engl. *flashcards*) koje motiviraju samopoučavanje te razna pitanja otvorenog tipa koja su u ovoj fazi izbjegavana zbog validacije unosa.

Još jedna od mogućnosti jest proširivanje ispitnog gradiva s većim fokusom na razred koji će se ciljano testirati ili pak s fokusom na neko gradivo koje se planira ispitivati uskoro. Potencijalno bi se mogli stvoriti i zasebni tečajevi koji su zapravo skup određenih lekcija, a učitelj odlučuje kojoj je grupi dostupna koja skupina tečajeva.

Komunikacija između učenika mogla bi se dodatno motivirati uvođenjem "džoker zovi" opcije koja je slična SOS funkcionalnosti za učitelje (Slika 6.9.), ali je zapravo dostupna učenicima koji pripadaju istoj grupi. Na taj način učenici koji su već prošli određenu lekciju mogu dobiti zahtjeve za pomoć vezane uz te istu.

Jednostavnija, a bitna funkcionalnost koja poboljšava korisničko iskustvo bilo bi uvođenje većeg broja obavijesti u sklopu samog sustava. Od obavijesti koja pri prijavi objašnjava u čemu je problem s unesenim podacima do obavijesti koja učeniku javlja da učitelj uskoro stiže, opcija je mnogo, a komunikacija između sustava i korisnika uvijek je dobra i praktična ideja.

8. Zaključak

Završni zadatak čiji je rezultat ovaj projekt bio je razvoj aplikacije za vježbu računalnog razmišljanja. Preciznije, zadatak je bio osmisliti i izraditi aplikaciju koja će učenicima približiti logiku računalnog programiranja. Osim same aplikacije, obvezna je bila i izrada pripadajuće baze podataka te lekcija koje će učenicima biti materijali za učenje.

Procesu je pristupljeno cjelovito pa je prva faza bila razvoj vizualnog identiteta, osmišljavanje potrebnih lekcija te snimanje videolekcija koje će biti dostupne unutar sustava. Nakon toga je uslijedila implementacija uz već navedene tehnologije.

Sustav je uspješno projektiran, no testiranje je napravljeno interno. Analiza podataka koje bismo dobili iz testiranja u obrazovnim ustanovama svakako bi bila od pomoći u nadogradnji aplikacije, no taj dio procesa ostaje kao potencijalna nadogradnja.

Aplikacija uspješno učenike poučava načinima razmišljanja koji će im kasnije poslužiti u računalnom programiranju, ali i drugim životnim poljima. Iako je računalno razmišljanje kompleksno polje koje se mora poučavati s posebnom pažnjom i trudom, završni zadatak uspješno približava učenicima one osnovne koncepte kroz ilustrativne primjere.

Literatura

[1] DEAP: Akcijski plan za digitalno obrazovanje (europska inicijativa 2021-2027) , https://education.ec.europa.eu/hr/focus-topics/digital-education/action-plan, [mrežno; stranica posjećena: 16. svibnja 2024.].

[2] Ministarstvo znanosti i obrazovanja: Kurikulum nastavnog predmeta Informatika za osnovne škole i gimnazije (kurikulum u PDF obliku), https://mzo.gov.hr/UserDocsImages/dokumenti/Publikacije/Predmetni/Kurikulum%20nastavnog%20predmeta%20Informatika%20za%20osnovne%20skole%20i%20gimnazije.pdf, [mrežno; stranica posjećena: 12. ožujka 2024.].

[3] Ministarstvo znanosti i obrazovanja: Izborna informatika od iduće školske godine za učenike razredne nastave (službena obavijest), https://mzo.gov.hr/vijesti/izborna-informatika-od-iduce-skolske-godine-za-ucenike-razredne-nastave/3726, [mrežno; stranica posjećena: 12. ožujka 2024.].

[4] Krešo Tomljenović: Računalno razmišljanje i uloga učenja pomoću igre na njegov razvoj (kvalifikacijski rad na Učiteljskom fakultetu u Zagrebu), https://inf.uniri.hr/images/studiji/poslijediplomski/kvalifikacijski/Kvalifikacijski_rad_Kreso_Tomljenovic.pdf, [mrežno; stranica posjećena: 16. svibnja 2024.].

[5] Valerie J. Shute, Chen Sun, Jodi Asbell-Clarke: Demystifying computational thinking, Volume 22, 2017 (pregled obrazovnih istraživanja), https://doi.org/10.1016/j.edurev.2017.09.003., [mrežno; stranica posjećena: 16. svibnja 2024.].

[6] Fakultet elektrotehnike i računarstva: 3. Definiranje klasa i prava pristupa. Konstruktori. Statičke metode i varijable (edukativni materijal kolegija Objektno

orijentirano programiranje), https://www.fer.unizg.hr/_download/repository/03-Konstruktori[2].pdf, [mrežno; stranica posjećena: 16. svibnja 2024.].

[7] Fakultet elektrotehnike i računarstva: Predavanje 1 (edukativni materijal kolegija Oblikovni obrasci u programiranju), https://www.zemris.fer.hr/~ssegvic/ooup/pubs/ooup0Intro.pdf, [mrežno; stranica posjećena: 16. svibnja 2024.].

[8] Klara Lovrečki, Ivan Moharić: Igrifikacija (elementi videoigara) u nastavi: pogled iz pedagoško-didaktičke perspektive, UDK 316.7:004 (pregledni rad), https://hrcak.srce.hr/file/396308, [mrežno; stranica posjećena: 16. svibnja 2024.].

[9] PCOM: Gamification in the Classroom (članak na portalu fakulteta), https://www.pcom.edu/academics/programs-and-degrees/physical-therapy/news/analog-gamification-in-the-classroom.html, [mrežno; stranica posjećena: 16. svibnja 2024.].

[10] MIT: Aplikacija Scratch (interaktivna aplikacija), https://scratch.mit.edu/projects/editor/?tutorial=getStarted, [mrežno; stranica posjećena: 16. svibnja 2024.].

[11] CARNET: Virtualna učionica PseudoDabar @ ucitelji.hr (online tečaj) , https://natjecanja.loomen.carnet.hr/course/view.php?id=119, [mrežno; stranica posjećena: 16. svibnja 2024.].

[12] Profil Klett: IZZI (edukativni portal) , https://hr.izzi.digital/#/, [mrežno; stranica posjećena: 16. svibnja 2024.].

[13] DBeaver: službena stranica alata, https://dbeaver.io/, [mrežno; stranica posjećena: 27. ožujka 2024.].

[14] bcrypt: kriptografska biblioteka, službena stranica, https://www.npmjs.com/package/bcrypt, [mrežno; stranica posjećena: 27. ožujka 2024.].

[15] Vite: lokalni poslužitelj, službena stranica alata, https://vitejs.dev/, [mrežno; stranica posjećena: 27. ožujka 2024.].

[16] React: besplatna front-end biblioteka, službena stranica alata, https://react.dev/, [mrežno; stranica posjećena: 27. ožujka 2024.].

[17] React: razvojno okruženje za CSS, službena stranica alata, https://tailwindcss.com/, [mrežno; stranica posjećena: 27. ožujka 2024.].

[18] Fastify: poslužiteljsko razvojno okruženje u sklopu tehnologije Node.js, službena stranica alata, https://fastify.dev/, [mrežno; stranica posjećena: 27. ožujka 2024.].

[19] Drizzle: sustav za objektno-relacijsko mapiranje, službena stranica alata, https://fastify.dev/, [mrežno; stranica posjećena: 27. ožujka 2024.].

[20] TypeScript: tipizirani programski jezik temeljen na jeziku JavaScript, službena stranica, https://www.typescriptlang.org/, [mrežno; stranica posjećena: 27. ožujka 2024.].

[21] Fakultet elektrotehnike i računarstva: Predavanje 8 (edukativni materijal kolegija Programsko inženjerstvo), https://moodle.fer.hr/pluginfile.php/92023/mod_resource/content/10/PROINZ_08_ARH_21.pdf, [mrežno; stranica posjećena: 31. svibnja 2024.].

[22] Wikipedia: Client–server model, wiki stranica, https://en.wikipedia.org/wiki/Client%E2%80%93server_model, [mrežno; stranica posjećena: 31. svibnja 2024.].

[23] Wikipedia: HTTP, wiki stranica, https://en.wikipedia.org/wiki/HTTP, [mrežno; stranica posjećena: 31. svibnja 2024.].

[24] TanStack: službeno web-sjedište alata, https://tanstack.com/query/v3, [mrežno; stranica posjećena: 27. ožujka 2024.].

[25] YouTube: Šumska škola, popis za reprodukciju, https://www.youtube.com/playlist?list=PLM7B8W7Jik-qgeCb1lCfxwgZuRx3Ul0nS, [mrežno; stranica posjećena: 31. svibnja 2024.].

Sažetak

Aplikacija za vježbu računalnog razmišljanja

Lana Kuretić

Tema ovog završnog rada je izrada aplikacije koja pomaže učenicima osnovne škole usvojiti računalno razmišljanje. Promatranjem trenutno korištenog kurikuluma za nastavni predmet Informatika razvijen je mali tečaj kroz čije lekcije učenici usvajaju programske koncepte i vježbaju rad s algoritmima, prepoznavanje uzoraka te primjene koncepata apstrakcije i dekompozicije. Aplikacija učenicima nudi registraciju, prijavu, prolazak kroz tečaj, traženje pomoći od učitelja te preuzimanje diplome po završetku učenja. Učiteljima nudi izradu grupa učenika, prikaz napretka u svakoj grupi te pregled svih zahtjeva za pomoć od strane učenika. Web aplikacija razvijena je uz pomoć biblioteke *React*, poslužiteljske tehnologije *Fastify* i *PostgreSQL* baze podataka.

Ključne riječi: računalno razmišljanje; igrifikacija; obrazovna aplikacija; React; Fastify; Typescript

Abstract

A web application for practicing computational thinking

Lana Kuretić

The topic of this final paper is the creation of an application that helps elementary school students to adopt computational thinking. By observing the currently used curriculum for IT-related subjects, a small course was developed. It teaches the students programming concepts and helps them practice working with algorithms, pattern recognition, as well as abstraction and decomposition. The application offers students the options to register, login, study the course, ask for the teacher's help, as well as downloading their diploma upon completion. The teachers can create groups of students, display progress of each student in each group, and view all requests for help from their students. The web application was developed with the help of *React* on the frontend, *Fastify* on the backend and an *PostgreSQL* database.

Keywords: computational thinking; gamification; educational application; React; Fastify; Typescript