Integracija adaptivnosti i igrifikacije u sustav za učenje programiranja

Ivanković, Toni

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 617

INTEGRACIJA ADAPTIVNOSTI I IGRIFIKACIJE U SUSTAV ZA UČENJE PROGRAMIRANJA

Toni Ivanković

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 617

INTEGRACIJA ADAPTIVNOSTI I IGRIFIKACIJE U SUSTAV ZA UČENJE PROGRAMIRANJA

Toni Ivanković

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

DIPLOMSKI ZADATAK br. 617

Pristupnik:	**Toni Ivanković (0036521641)**
Studij:	Računarstvo
Profil:	Znanost o podacima
Mentor:	doc. dr. sc. Tomislav Jagušt

Zadatak: **Integracija adaptivnosti i igrifikacije u sustav za učenje programiranja**

Opis zadatka:

Digitalne kompetencije, računalno i algoritamsko razmišljanje i vještine programiranja postaju sve važnije u suvremenom društvu, kako u obrazovanju tako i u radnom okruženju. S druge strane, sve je veći obrazovni izazov pronaći načine zadržavanja koncentracije i pažnje učenika, ali i stvaranja interesa i motivacije za savladavanje novih vještina. Cilj ovog diplomskog rada je razviti višeplatformski sustav za razvoj vještine algoritamskog razmišljanja kroz učenje računalnog programiranja. U sustav je potrebno ugraditi elemente igrifikacije kako bi se zadržali interes i pažnja učenika, a algoritmima adaptivnog učenja omogućiti prilagodbu procesa učenja različitim razinama sposobnosti učenika. Potrebno je istražiti postojeće tehnike igrifikacije i adaptivnog učenja i analizirati njihovu primjenjivost u predloženom sustavu. Također, potrebno je istražiti i neke od postojećih sustava za učenje programiranja i analizirati elemente koji se mogu integrirati u predloženi sustav. Razvijeni sustav potrebno je testirati sa stvarnim korisnicima te analizirati povratne informacije, radi identifikacije i implementacije potencijalnih poboljšanja.

Rok za predaju rada: 28. lipnja 2024.

Zahvaljujem se svom mentoru, doc. dr. sc. Tomislavu Jaguštu na savjetima pri istraživanju, na pruženom povjerenju i na inspiraciji za intenzivnije posvećivanje edukaciji i popularizaciji znanosti. Možda nisam ostao na zavodu, ali sigurno ću Vas se često sjetiti dok radim na edukaciji negdje na svojem putu.

Zahvaljujem se trima srednjoškolskim profesoricama – Moržan, Floreani i Šokčić. Vaš odnos prema učenicima i prema podučavanju zauvijek će mi biti inspiracija i nešto čemu težim.

Zahvaljujem se svim prijateljima koji su bili uz mene tijekom cijelog studija. Zbog vas mi ni Barcelona ne može zamijeniti Zagreb i Osijek. Posebno bih se zahvalio Nikši i Pauli, svojim najneumornijim alfa testerima.

Za kraj, zahvaljujem se svojoj obitelji. Od svakog od vas imao sam priliku naučiti nebrojeno vještina. Pobrinuli ste se da se uvijek mogu u potpunosti posvetiti učenju i od malena ste me poticali da budem najbolja verzija sebe.

I, realno, zahvaljujem sebi.

Sadržaj

Popis priloga

Slike

Tablice

Uvod

Kao što je osnovna pismenost bržim razvojem civilizacije postala neophodna za funkcioniranje u svakodnevnom životu, slična pojava danas se događa i s digitalnim kompetencijama. Sve bržim razvojem digitalnih tehnologija i njihovom sve većom isprepletenošću s nedigitalnim svijetom obrazovanje u području digitalnih kompetencija postaje sve neophodnije. Prema EU-ovom izvješću o informatičkom obrazovanju u školama u Europi [1], digitalne kompetencije su na čelu europskih nacionalnih edukacijskih programa. Generalni je stav da su digitalne kompetencije ključne za cjeloživotno učenje još od 2006. godine. Osim toga, EU digitalnim desetljećem (2021.-2030.) postavljen je i cilj da minimalno 80 % ljudi raspolaže osnovnim digitalnim kompetencijama.

U spomenutom izvješću fokus je stavljen na kurikulume informatike. Ipak, s obzirom na ograničeno trajanje nastave, ključne je vještine neophodno razvijati ne samo u školi nego i u slobodno vrijeme. I dok neki učenici motivaciju za samostalno učenje mogu pronaći sami, mnogima intrinzična motivacija nije dovoljna, već trebaju dodatan poticaj [2].

Digitalne kompetencije široko su područje. Dok je neke od njih moguće svladati češćim korištenjem (npr. vještina uporabe nekog digitalnog alata), za druge je potrebno razvijati određen način razmišljanja (npr. programiranje).

Programiranje i računalno razmišljanje ključni su za razvoj u području digitalnih kompetencija. Osim što se programiranje integrira u sve više struka, i kao takvo postaje korisna vještina već samo po sebi, razvoj računalnog razmišljanja koristan je i za brojne druge aspekte života. Pokazano je da razvoj računalnog razmišljanja povećava vještine kritičkog i analitičkog razmišljanja, koji su nužni ne samo za razvoj učenika u ostalim digitalnim kompetencijama nego i za razvoj u svim sferama života [3].

S obzirom na navedeno, cilj ovog rada bio je izraditi sustav namijenjen učenju programiranja. Kako bi učenje bilo dostupno svima, bilo je potrebno implementirati što nižu *donju granicu* za korištenje sustava, tj. što manju nužnu razinu predznanja. U klijentsku aplikaciju bilo je potrebno ugraditi elemente igrifikacije i adaptivnosti, kako bi korištenje aplikacije više angažiralo učenike, čija je koncentracija uslijed konstantne stimulacije i neprestanih distrakcija sve manja. Radi procjene koristi primjene elemenata igrifikacije, sustav je evaluiran eksperimentom u kojemu je sudjelovalo 48 učenika osnovne škole.

1. Osnovni koncepti

1.1. Motivacija iza digitalnog sustava

Pri razvoju novog digitalnog sustava za učenje, korisno je razmisliti o prednosti za izvannastavno podučavanje koje donosi digitalni sustav u odnosu na tradicionalne metode, tj. zašto uopće biramo digitalni sustav.

Postoje razni načini za podučavanje van učionice – udžbenici, snimke predavanja, obrazovna videa, virtualni laboratoriji, privatne poduke itd. Svi ti načini razlikuju se po potrebnoj samostalnoj motivaciji učenika te prema potrebnim resursima po učeniku, tj. skalabilnosti. Dok privatne poduke mogu najkvalitetnije ponuditi personaliziranu poduku učeniku, najmanje su skalabilan oblik podučavanja, jer je potreban jedan učitelj na jednog učenika (a čak i kad se radi o grupnim podukama, još uvijek je ovaj način nedovoljno skalabilan za velik broj učenika). S druge strane, udžbenici ili snimke predavanja vrlo su skalabilni, ali imaju ograničenu moć zadržavanja pozornosti učenika jer su uglavnom vrlo nepersonalizirani.

Značaj razvoja digitalnih tehnologija je u tome što omogućuje pristup koji je i personaliziran i skalabilan. Osim toga, nove tehnologije omogućuju i pristup učenju koji će smanjiti nužnu razinu motivacije učenika, čime se uključuje veći broj učenika i smanjuje napor pri učenju. U sljedećim potpoglavljima promotrit ćemo neke od koncepata koji će omogućiti takav pristup.

1.2. Adaptivno učenje

Adaptivno učenje (eng. *adaptive learning*) metoda je poučavanja kojom se sadržaj za učenje prilagođava učeniku tijekom procesa učenja. Prilagođavanje sadržaja omogućeno je prikupljanjem podataka o učeniku tijekom procesa učenja, a ti se podatci potom koriste za optimizaciju sadržaja. Tako svaki učenik dobiva individualni sadržaj u skladu sa svojim trenutnim sposobnostima i razinom znanja.

Često spominjan pojam je i personalizirano učenje. Ono je spoj adaptivnog učenja i prilagođenog učenja (eng. *customized learning*). Dakle, osim što sadržaj učenja prilagođava učeniku, također daje učeniku kontrolu da prilagođava iskustvo učenja prema svojim

preferencijama (npr. odabir sljedeće lekcije koju će naučiti) [4]. Iako je ovo vrlo koristan i široko primjenjiv koncept, u ovom radu fokus je na čistom adaptivnom učenju, tj. prilagođavanju sadržaja sposobnostima učenika.

Adaptivno učenje širok je pojam, a neki od primjera prilagođavanja procesa učenja su podešavanje puta učenja, učinkovite povratne informacije i dodatni sadržaji. Ovo je suprotno tradicionalnom učenju koje ima jedinstveni pristup za sve učenike (eng. *one-size-fits-all*) [5].

Korištenje adaptivnog učenja je očito rješenje problema navedenih u prethodnom potpoglavlju vezanih za personaliziranost sadržaja. Osim toga, prednosti korištenja adaptivnog učenja pokazane su brojnim istraživanjima. Učenici koji koriste adaptivne sustave imali su veće samopouzdanje zahvaljujući povratnim informacijama, a imali su i veći osjećaj autonomije. Učenici koji tradicionalnim metodama postižu manje uspjehe (učenici iz manjinskih skupina i *učenici prve generacije*, tj. učenici čiji roditelji nisu završili visoko obrazovanje) adaptivnim su metodama postizali uspjehe puno bliže ostalim vršnjacima. Također, zbog prilagođenog sadržaja, učenici su više uključeni u proces učenja i duže ustraju u učenju koje je prikladne težine, što naposljetku vodi do veće razine naučenog sadržaja [5].

Postavlja se pitanje kako implementirati sustav adaptivnog učenja. Umjetna inteligencija omogućuje obradu velikih količina podataka i ekstrakcije bitnih značajki koje se mogu koristiti za prilagođavanje procesa učenja (primjerice analiza vremena zadržavanja na pojedinom zadatku i konkretnih grešaka koje učenik napravi radi sugeriranja zadataka koje treba više vježbati).

Ipak, razvoj skalabilnih i prilagođenih sustava za učenje koji nadilaze tradicionalne alternative moguć je i bez korištenja umjetne inteligencije u užem smislu. Jednostavni sustavi bazirani na pravilima (*eng. rule-based*) mogu biti primjenjivi i učeniku prilagođavati iskustvo učenja bez korištenja složenih algoritama umjetne inteligencije. Neke od mogućih jednostavnih tehnika prilagođavanja procesa učenja učeniku su prikaz pogreške i točnog odgovora, ponavljanje pogrešno odgovorenih zadataka, prilagođavanje procijenjene razine znanja učenika po riješenim zadatcima i davanje zadataka primjerene težine. Sve te metode vrlo se lako mogu algoritamski implementirati, a daju sustav koji je znatno prilagođeniji učeniku nego tradicionalni pristupi.

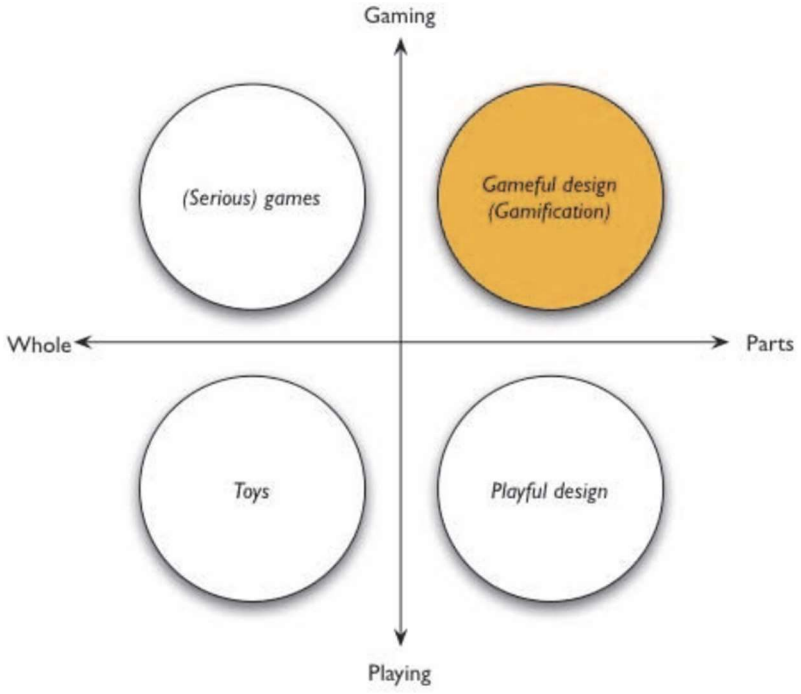
1.3. Igrifikacija

Igrifikacija (eng. *gamification*) podrazumijeva korištenje elemenata dizajna igara u kontekstu van igre [6]. U hrvatskom jeziku posjedujemo samo glagol „igrati“ koja se odnosi i na engleske glagole *game* i *play*. *Gaming* podrazumijeva taktike, eksplicitni skup pravila i kompetitivnost, dok *playing* podrazumijeva veću improvizaciju, ekspresivnost i slobodnu formu. Igrifikacija se odnosi upravo na *gaming*, s argumentom da su upravo svojstva *gaminga* ona koja su poželjna za primjene van igre.

Također, vrlo bitan aspekt definicije igrifikacije je da ona podrazumijeva dodavanje elemenata dizajna igara u kontekstu **van** igre. Dakle ključno je primijetiti da dodavanje elemenata dizajna igre u igru nije igrifikacija. Igrifikacija nužno podrazumijeva da se spomenuti elementi dodaju u neki sustav koji nije igra, jer bi taj sustav mogao imati koristi od korištenja elemenata dizajna igara.

Zadnji dio definicije koji je bitno istaknuti je da igrifikacija podrazumijeva dodavanje **elemenata** dizajna igara u kontekstu van igre. Za razliku od toga, cjelovita igra koja se koristi za neku svrhu osim zabave bila bi prikladnije nazvana ozbiljnom igrom (eng. *serious game*), ili čak samo igrom.

Razlika svih spomenutih koncepata dobro je prikazana na sljedećoj vizualizaciji:



Slika 1.1 – pozicija igrifikacije među sličnim konceptima [6]

Korist igrifikacije upravo je u tome što korisnici igrificirane aplikacije u idealnom slučaju nemaju dojam da rade naporan, koristan ili težak posao, već da igraju igru. Time se smanjuje nužna motivacija za korištenje aplikacije (u slučaju ovog rada – za učenje), a cjelokupno iskustvo je najčešće ugodnije i korištenje aplikacije je učinkovitije. U radu [7] napravljen je pregled literature radi ispitivanja spomenutih tvrdnji, te je utvrđeno da igrifikacija doista poboljšava iskustvo korištenja aplikacije, uz upozorenje da korist ovisi o kontekstu u kojemu se igrifikacija koristi te o samim korisnicima.

S druge strane, u radu [8] navodi se sumnja u efektivnost igrifikacije, pogotovo u *one-size-fits-all* pristup, tj. jednaku igrifikaciju za sve učenike, nasuprot personaliziranoj igrifikaciji. Kao razlog sumnje navedeni su šaroliki rezultati postojećih ispitivanja. Upravo zato je u poglavlju 8.2 provedeno istraživanje na aplikaciji razvijenoj u okviru ovog rada, kako bi se ispitala efektivnost igrifikacije u usporedbi s neigrificiranom aplikacijom.

Elementi dizajna igara koji se mogu koristiti pri igrifikaciji su brojni, ali najčešće viđeni su bodovi, ljestvica poretka, postignuća/značke, razine, priče, jasni ciljevi, povratna informacija, nagrade, napredak i izazovi. Još neki od važnih elemenata igara koji bi se mogli koristiti u kontekstu van igre navedeni su u radu [9], a to su: predstavljanje s pomoću avatara, trodimenzionalne okoline, narativni kontekst, povratna informacija, razine i činovi, trgovanje i ekonomije, natjecanje pod eksplicitnim pravilima koja se provode, timovi, paralelni sustavi za komunikaciju koji se lako postavljaju i mijenjaju, te vremenski pritisak.

1.4. Ideja rješenja

Ideja iza sustava razvijenog u ovom radu upravo je spoj rješenja navedenih izazova. Razvoj višeplatformske aplikacije omogućuje pristup učenju u bilo kojem trenutku, iskorištavajući tako nalete motivacije učenika. Algoritmi adaptivnog učenja omogućuju osjećaj učenja što sličniji privatnoj poduci, tj. učenje koje je usko krojeno po samom učeniku. Elementi igrifikacije učenika dodatno motiviraju, iskorištavajući npr. poriv za natjecanjem ili za skupljanjem bodova u svrhu učenja. Tako je, u ideji, proces učenja sveden na igru i on, u najmanju ruku, postaje manje naporan za učenika, a u idealnom slučaju učenik učenju pristupa kao igri i sve znanje upija pasivno.

U nastavku su navedeni primjeri postojećih sustava za učenje te su identificirani njihovi elementi koji bi se mogli iskoristiti u sustavu koji je bilo potrebno razviti u okviru ovog rada.

Osim toga, identificirani su i elementi koji tim sustavima nedostaju, a koji bi potencijalno mogli ponuditi bolje iskustvo učenja.

2. Postojeća rješenja

Kako bi se identificirala poželjna svojstva sustava razvijenog u sklopu ovog diplomskog rada, napravljena je analiza nekoliko postojećih sustava koji koriste igrifikaciju i adaptivno učenje, bilo u kontekstu učenja programiranja ili neke druge vještine.

2.1. Duolingo

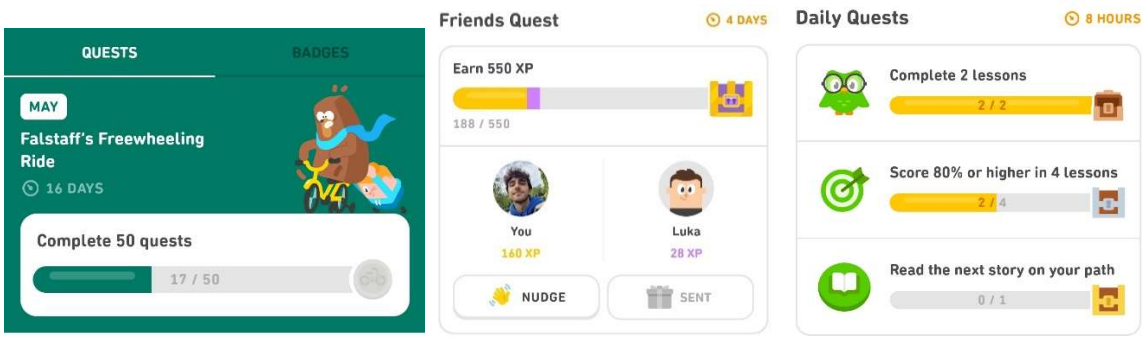
Duolingo je višeplatformska aplikacija, dostupna u web obliku i kao zasebna mobilna aplikacija. Cilj aplikacije je učenje stranih jezika, kojih ima 39 za govornike engleskog jezika, a postoji još mnogo drugih jezika dostupnih za učenje s nekim drugim polaznim jezikom. Aplikacija radi po *freemium* sustavu, točnije učenje je potpuno besplatno uz oglase, a u plaćenoj inačici korisnik uči neometano i ima pristup više značajki adaptivnog učenja.

Aplikacija je iznimno uspješna i popularna, te je već godinama praktički sinonim za samostalno učenje stranih jezika [10]. Osim kvalitetnog sadržaja, velik dio uspjeha aplikacija duguje upravo igrificiranosti i adaptivnosti učenja [11]. Osim igrificiranosti i adaptivnosti, glavna je teza aplikacije učenje kroz praksu (eng. *learn by doing*).

Vezano za igrifikaciju, Duolingo implementira sljedeće elemente:

- XP bodove
- *Streak* (broj uzastopnih dana tijekom kojih korisnik koristi aplikaciju)
- Ljestvicu poretka (eng. *leaderboard*)
- Predstavljanje s pomoću avatara
- Lekcije s vremenskim pritiskom
- Povratna informacija
- Natjecanje (tjedna ljestvica sa skupljenim XP bodovima)
- Smještanje u lige (ovisno o poziciji na tjednoj ljestvici)
- Ekonomiju (kupovanje dodataka s pomoću *gemova*)
- Timove i izazove (dnevni izazovi, tjedni izazov u paru s prijateljem, mjesečni izazov)
- Postignuća (značke za određene uspjehe)
- Nagrade (dupli XP bodovi navečer za rješavanje lekcije ujutro i obratno
- Kratke lekcije (eng. *bite-sized*)

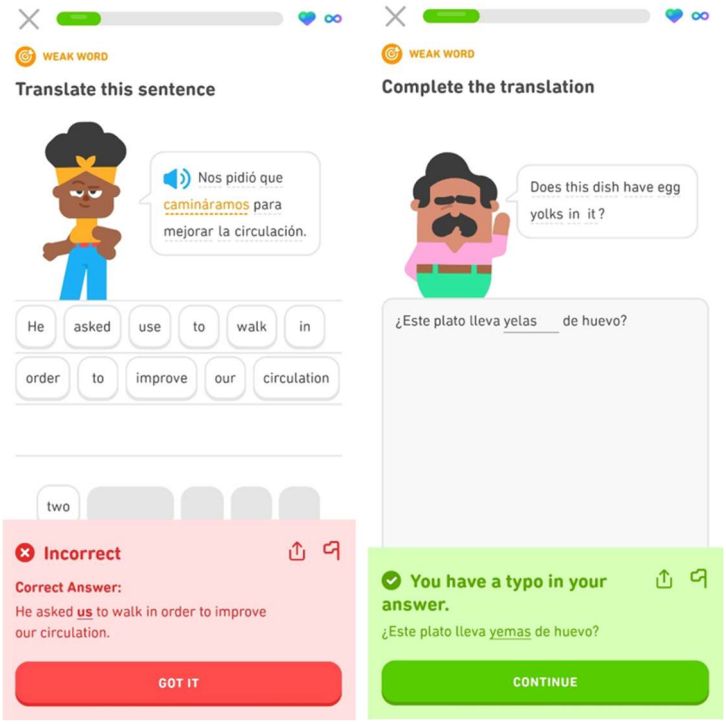
- Vježbe koje koriste slagalicu riječi umjesto samostalnog pisanja
- Interakcija s prijateljima (pregled međusobnih postignuća, čestitke za uspjehe)



Slika 2.1 – Duolingo elementi igrifikacije

Teško je identificirati koliko koji od tih elemenata utječe na uspjeh aplikacije. Ipak, aplikacija često ima blage izmjene zbog provođenja A/B testiranja, koje bi se moglo koristiti za identifikaciju koristi pojedinih značajki igrifikacije.

Vezano za adaptivnost učenja, Duolingo ima posebne lekcije zvane „Personalized practice“ (personalizirana vježba), u kojima učeniku daje vježbe u kojima su riječi koje je *najviše zaboravio*. Također, odgovorom na svaki zadatak učenik vidi je li ponudio točan odgovor, a, ako nije, dobiva informaciju o točnom odgovoru i o poziciji pogreške u rečenici. Čak i ako odgovori gotovo točno, ako se radi o *maloj* pogrešci, vježba se označava kao točno odgovorena, ali učenik dobiva informaciju o *točnijem* odgovoru.



Slika 2.2 – Duolingo adaptivna povratna informacija

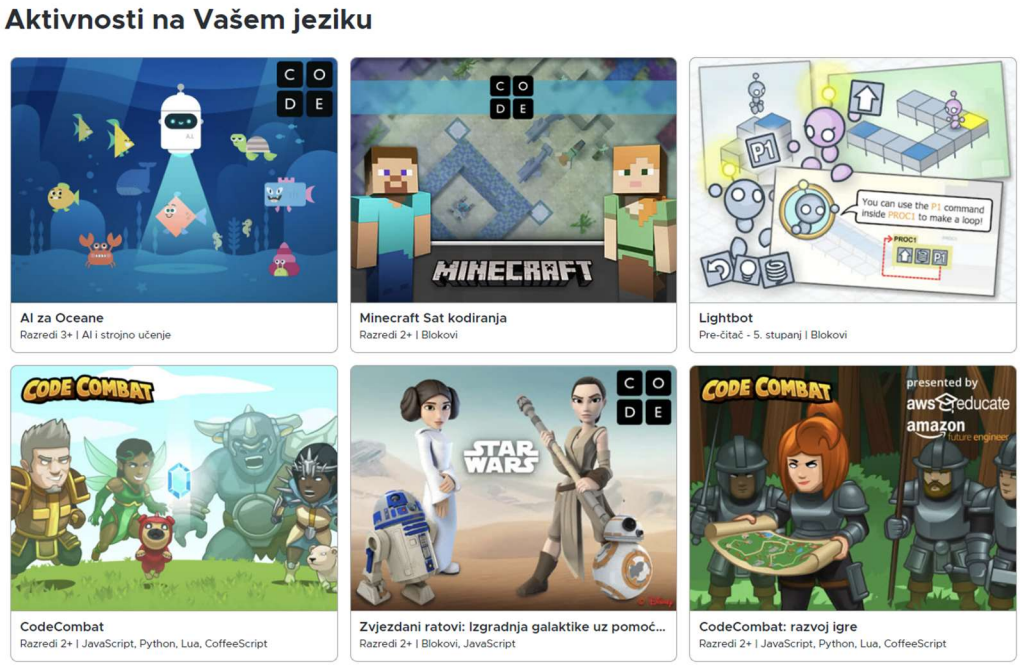
U plaćenoj inačici učenik ima pristup vježbama koje se fokusiraju na prošle pogreške, ima pristup vježbama s karticama za ponavljanje riječi (eng. *flashcards*) te specifičnim vježbama za slušanje i govorenje.

Jačina aplikacije leži i u angažiranju više načina učenja – čitanje, pisanje, slušanje, govorenje, prevođenje, odgovaranje na pitanja o tekstu.

2.2. Hour of Code

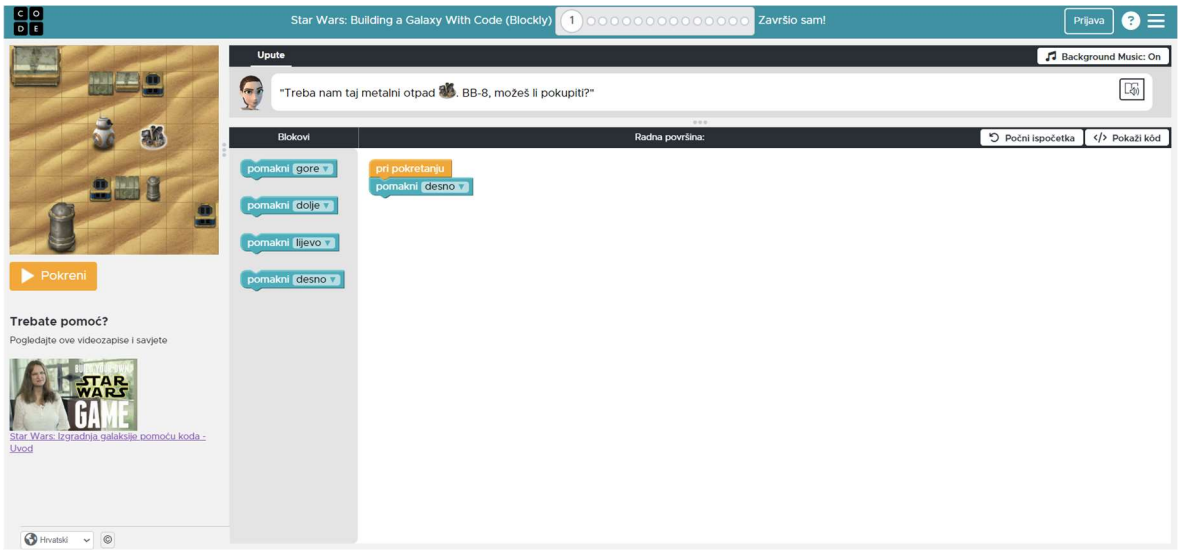
The Hour of Code besplatni je uvod u računarsku znanost kroz zabavne aktivnosti i videa za učenike svih razina predznanja. Dostupan je za učenje na više od 45 jezika i isprobalo ga je preko 100 milijuna učenika. Često se koristi tako da nastavnik prijavi događaj, te onda s razredom održi sat kodiranja, gdje učenici koriste aplikaciju tijekom jednog školskog sata [12].

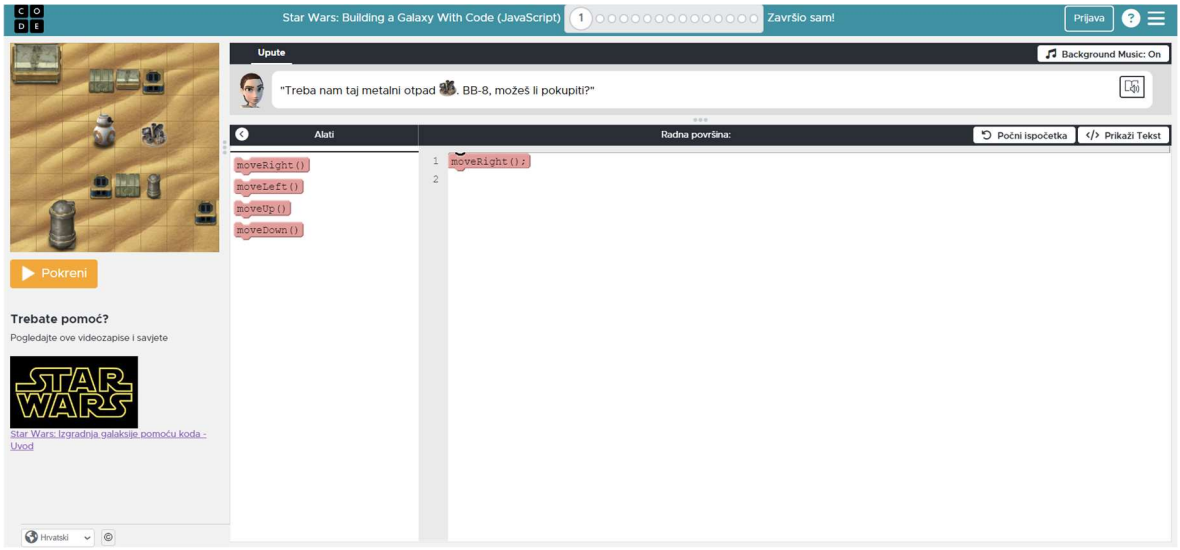
Proces učenja odvija se kroz igru. Učenik može odabrati temu po želji, a teme su najčešće povezane s nekim poznatim igrama ili filmovima.



Slika 2.3 – Hour of Code tematski sadržaji

U aplikaciji učenici mogu odabrati između čistog blokovskog programiranja, ili mješavine blokovskog programiranja i JavaScripta. U svakom slučaju, na početku dobiju uvodni video koji objašnjava kako koristiti aplikaciju, a potom dobiju priliku sami riješiti traženi zadatak.





Slika 2.4 – Hour of Code zadatak vezan za temu

Ova je aplikacija po definiciji bi više spadala u edukativnu igru nego u igrificiranu aplikaciju. Zadatak je lika dovesti do cilja, a edukativni dio aplikacije samo je sredstvo.

Potencijalna stvar koja bi mogla dati pozitivan utjecaj na motivaciju je korištenje postojećih igara i filmova koje učenici već poznaju. S druge strane, korištenje postojećih igara i filmova uvodi problem autorskih prava i, u komercijalnom kontekstu, potencijalno čini aplikaciju značajno skupljom.

Druga prepoznata pozitivna karakteristika je izbor između *lakšeg* i *težeg* načina, tj. blokovskog programiranja i JavaScript programiranja (doduše i JavaScript inačica je također blokovska). Na taj način, ako su spremni za nešto teže, učenici mogu naučiti konkretan programski jezik koji će moći koristiti i van igre. Time je aplikacija prikladna za primjenu i u srednjim školama ili fakultetima koji žele popularizirati programiranje, ali i onima koji ovu aplikaciju žele iskoristiti kao prvi korak pri podučavanju programiranja.

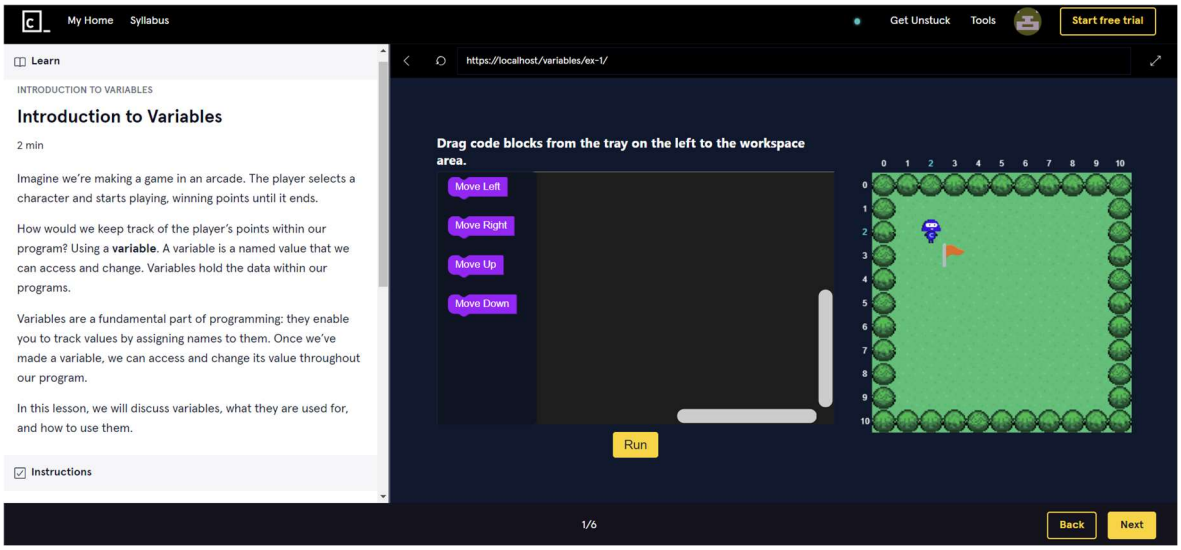
Kao i Duolingo, Hour of Code također prakticira *učenje kroz praksu*.

2.3. Codecademy

Codecademy [13] je američka interaktivna platforma namijenjena učenju 12 programskih i označnih jezika (Python, Java, Go, JavaScript, Ruby, SQL, C++, C#, Swift, HTML i CSS, R i Bash). Osim toga, nude i tečajeve iz umjetne inteligencije, računarstva u oblaku, podatkovne znanosti, web dizajna i web razvoja te kibernetičke sigurnost. Platforma je

besplatna za korištenje, ali radi na *freemium* principu, tj. u *Pro* paketu nudi personalizirane kvizove, plan učenja te realistične projekte [14].

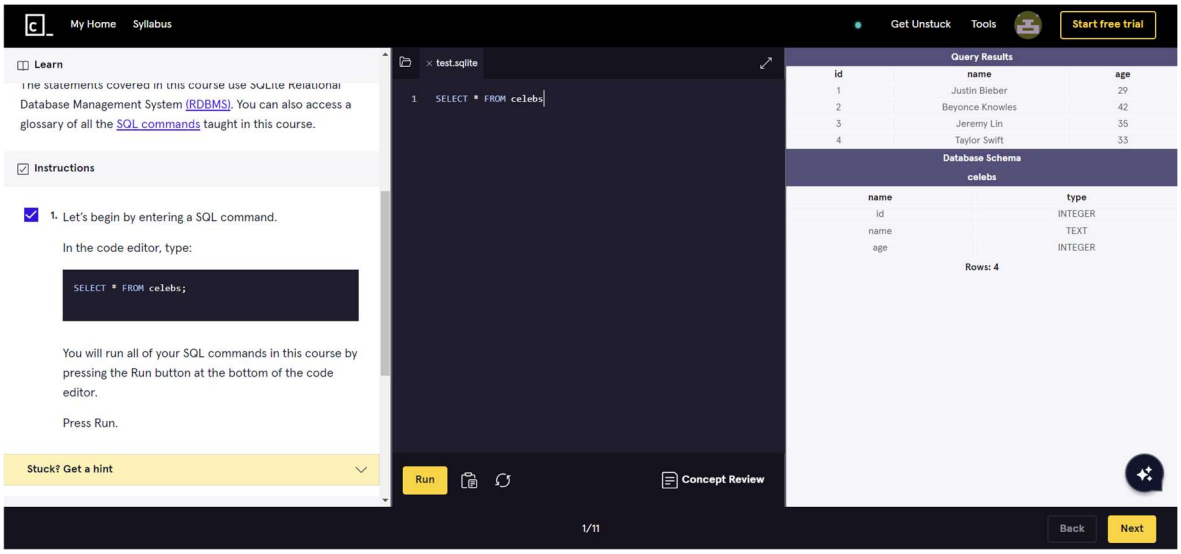
Codecademy također prakticira učenje kroz praksu, doduše uz nešto više čitanja nego prethodne dvije aplikacije.



Slika 2.5 – Codecademy zadatak vezan uz igru uz teoretsko objašnjenje

U odnosu na prethodne dvije aplikacije, Codecademy više je fokusiran na eksplicitno učenje uz vježbu na primjerima nego što je fokusiran na samostalno zaključivanje s pomoću primjera.

Snaga Codecademyja je u velikom broju raznolikih tečaja te u razini tih tečaja. S obzirom na to da je *više fokusiran* na učenje nego na igru te dopušta veću količinu teksta u tečajevima, moguće je pohađati i napredne tečajeve koji traju dulje.



Slika 2.6 – Codecademy zadatak na naprednijem tečaju

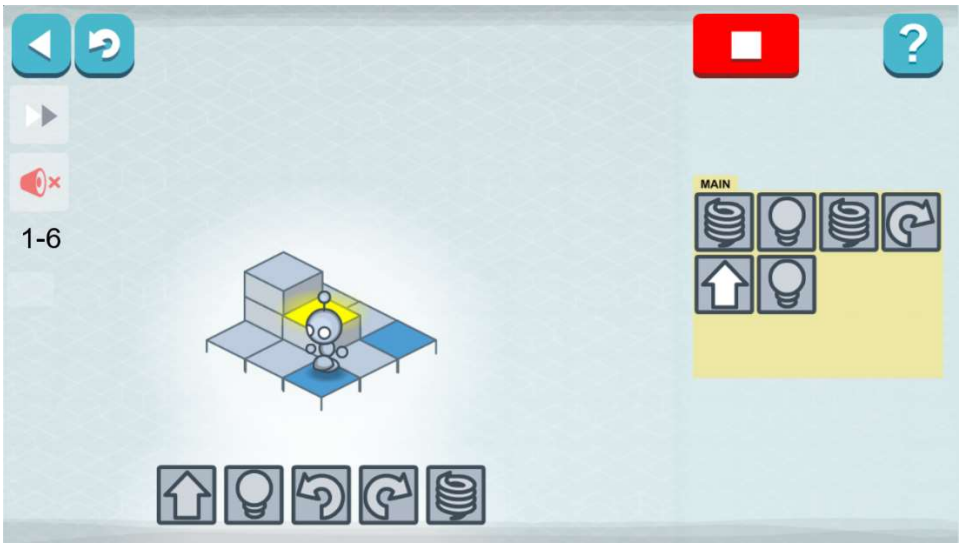
Codecademy nema mnogo elemenata igrifikacije. U eventualne elemente igrifikacije mogli bi se svrstati kratke lekcije i zadatci koji uključuju prostorno kretanje.

Vezano za adaptivnost učenja, korisnik može birati svoj put učenja i postavljati vlastite ciljeve [15], što bi se umjesto u adaptivno učenje moglo svrstati u prilagođeno učenje jer se sustav ne prilagođava korisniku, već korisnik sustav prilagođava sebi. Savjeti (eng. *hints*) su najčešće općeniti, a ne prilagođeni konkretnom korisniku. U plaćenoj inačici korisnik ima pristup AI-asistentu, što bi se moglo smatrati specifičnom pomoći, pa bi to mogla biti jedna adaptivna značajka.

U zaključku, Codecademy daje izvrsno iskustvo učenja s velikim izborom kvalitetnih tečajeva, ali nije toliko prilagođen motiviranju korisnika igrifikacijom i povećanju kvalitete iskustva učenja s pomoću adaptivnosti.

2.4. Lightbot

Lightbot je obrazovna videoigra namijenjena učenju koncepata programiranja. Cilj igre je navoditi robot kroz polje za igru i upaliti svjetla u podu [16]. Koncepte programiranja igrač uči „potajno“ dok igra igru [17]. Aplikacija je dostupna na mobilnim platformama (App Store, Google Play, Amazon Apps).



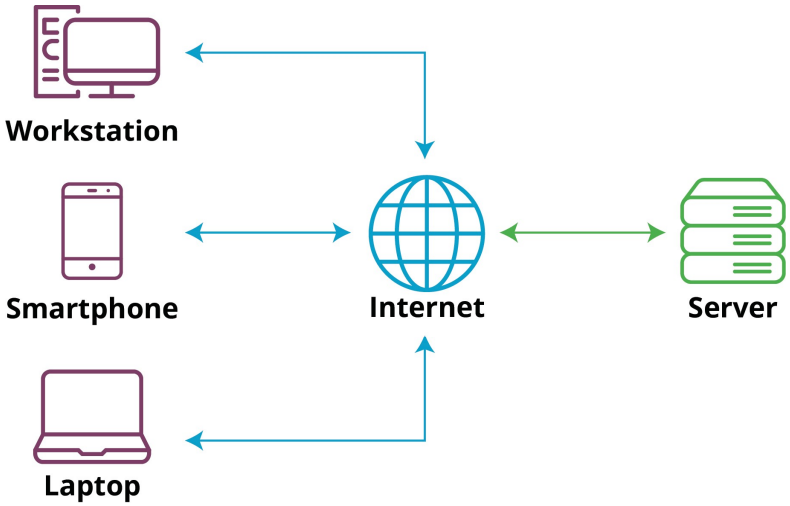
Slika 2.7 – Lightbot učenje kroz igru

Ova je aplikacija sigurno obrazovna igra, tako da se ne može govoriti o elementima igrifikacije u njoj (jer je igrifikacija po definiciji dodavanje elemenata dizajna igara van konteksta igre). Ipak, Lightbot je dobar primjer kako se osnovni koncepti programiranja mogu učiti na jednostavan način bez eksplicitnog učenja programiranja.

S obzirom na to da koristi pristup učenju kroz igru, ova aplikacija primjerenija je za korištenje razvoja algoritamskog razmišljanja općenito, ali ne bi bila primjerena za korištenje pri savladavanju nastavnog gradiva propisanog kurikulumom jer ne podučava nijedan konkretan programski jezik.

3. Arhitektura i elementi sustava

Sustav koji je razvijen u okviru zadatka ovog rada logički je odvojen u tri cjeline – podatkovni sloj (baza podataka), poslužiteljski sloj (eng. *server*) i klijentski sloj (eng. *client*). Formalno, arhitektura sustava je *klijent-poslužitelj* arhitektura. U nastavku poglavlja bit će opisana arhitektura na višoj razini apstrakcije, interakcija slojeva te implementirani aspekti igrifikacije i adaptivnosti. U sljedećim će poglavljima biti dan detaljan pregled svakog od navedenih slojeva.



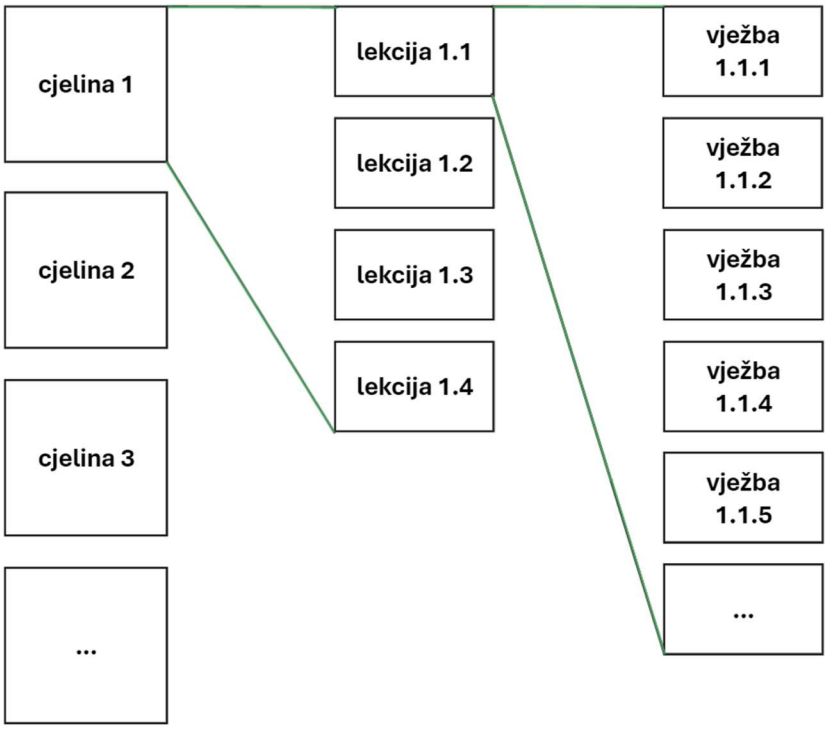
Slika 3.1 - Klijent-poslužitelj arhitektura [18]

Jedini sloj s kojim korisnici imaju izravan dodir je klijentski sloj. Klijentski sloj čini ili web ili mobilna aplikacija. Bez obzira koristi li korisnik web ili mobilnu inačicu aplikacije, korisničko sučelje izgleda i cjelokupno iskustvo u osnovi je jednako. Klijentska aplikacija nema znanje o sadržaju tečaja, ali ima znanje o strukturi entiteta i kako ih treba predstaviti korisniku. Svi podatci potrebni za funkcioniranje aplikacije nalaze se u podatkovnom sloju, tj. u bazi podataka. Kao posrednik između ta dva sloja služi poslužiteljska aplikacija. Ona na zahtjev klijentske aplikacije dohvaća podatke iz baze podataka, provjerava ih, formatira u dogovoreni oblik, te šalje klijentskoj aplikaciji. Ovako je postignuta *raspodjela odgovornosti* (eng. *separation of concerns*), gdje se svaki sloj fokusira na svoj konkretni zadatak.

3.1. Elementi aplikacije

Aplikacija poznaje četiri vrste korisnika: učenik, učitelj, kreator i administrator. Većina funkcionalnosti aplikacije prilagođena je učeniku, koji je primarni korisnik aplikacije.

Ideja aplikacije je učeniku omogućiti stjecanje znanja programiranja rješavanjem niza lekcija. Lekcije su sastavljene od kratkih vježbi, kojih je najčešće 10-15 u svakoj lekciji. Tek točnim rješavanjem svih vježbi u nekoj lekciji učenik može prijeći na rješavanje sljedeće lekcije. Detaljna struktura tečaja navedena je u poglavlju 7 – Struktura tečajadolje. Radi preglednosti, lekcije su po sadržaju grupirane u cjeline. Svaka cjelina sastoji se od nekoliko lekcija, najčešće 3-5. Tek dovršetkom svih lekcija u jednoj cjelini učenik može prijeći na sljedeću cjelinu, i tako dalje, sve do kraja tečaja.



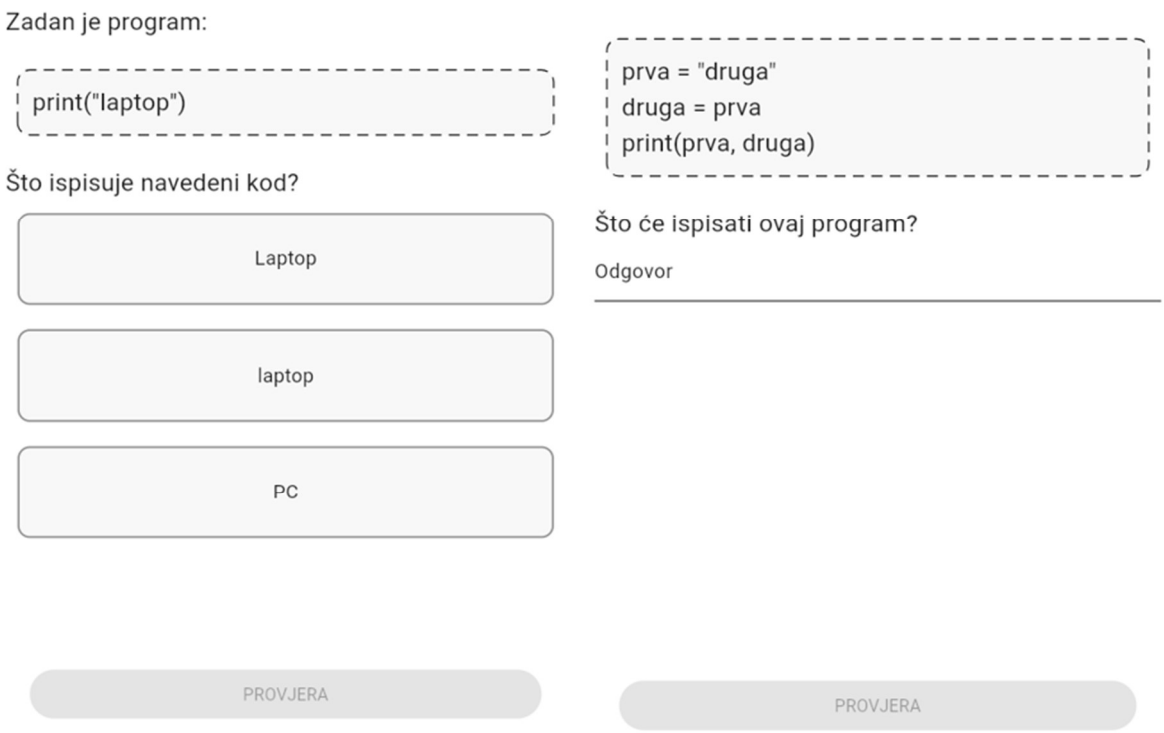
Slika 3.2 - Struktura tečaja (jedinice grupiranja)

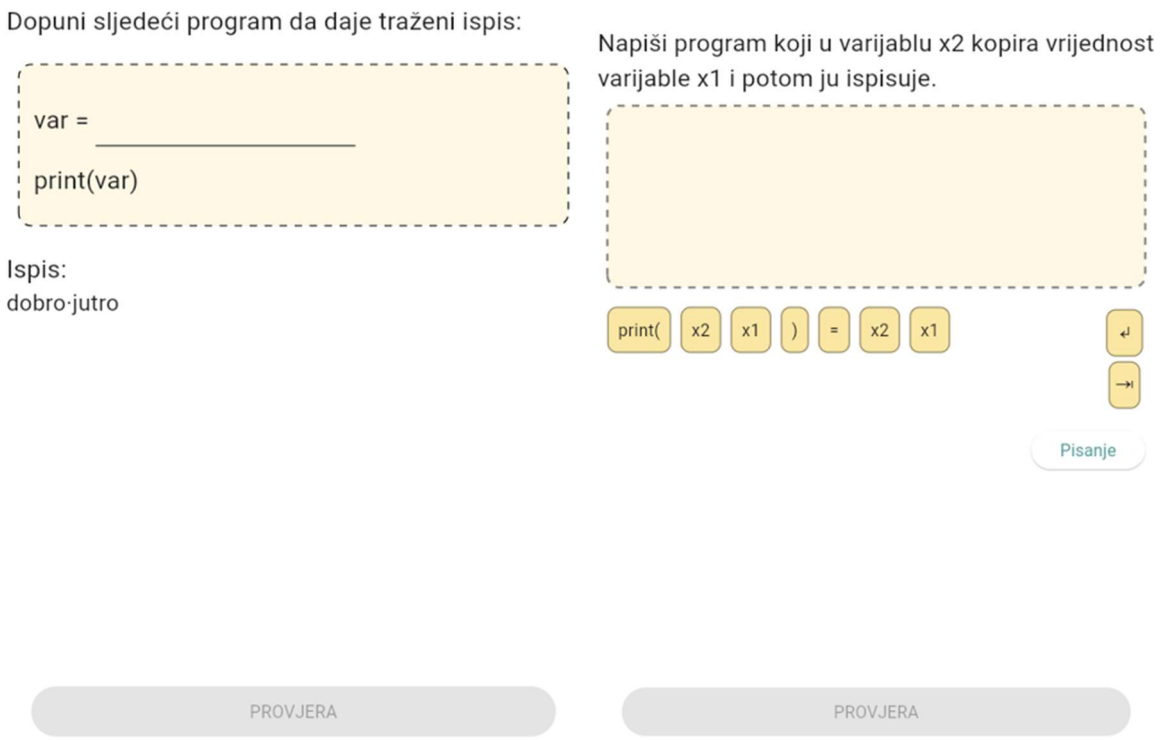
Cjeline i lekcije jedinice su grupiranja te najčešće sadrže savjete za rješavanje, točnije mogu se smatrati formalnim gradivom koje se očekuje usvojiti u ovoj cjelini/lekciji. Ipak, savjeti služe samo kao pomoć (eng. *hint*). Čitanje savjeta nije neophodno za rješavanje lekcije. Lekcije započinju s lakšim vježbama, tako da bi učenici koji vole samostalno zaključivati i učiti kroz praksu mogli riješiti lekciju bez poteškoća. Čak i ako nešto ne mogu samostalno

zaključiti, nakon davanja pogrešnog odgovora predstavljen im je točan odgovor, pa na primjeru mogu steći očekivano znanje i na sljedećim vježbama ponuditi točan odgovor. Ovakav pristup bilo je moguće vidjeti u postojećim aplikacijama. Također, ovaj pristup poznat je u teoriji učenja kao pojam *učenje temeljeno na problemu* (eng. *problem-based learning*) [19].

3.2. Vježbe

Najmanje jedinice tečaja, su vježbe. Implementirane su četiri osnovne vrste vježbi: višestruki izbor (eng. *multiple choice* – MC), kratki odgovor (eng. *short answer* – SA), dopunjavanje koda (eng. *short code writing* – SCW) i dugi odgovor (eng. *long answer* – LA). Iako težina vježbe ovisi o sadržaju, generalno su navedeni tipovi vježbi poredani rastuće po težini.





Slika 3.3 - Vrste vježbi

Vježbe višestrukog izbora zahtijevaju prepoznavanje te učenik vidi točan odgovor, koji se može više ili manje očito razlikovati od pogrešnih odgovora. Stoga lekcije najčešće započinju s *očitim* pitanjima višestrukog izbora (MC), na kojima učenik može lako naučiti očekivani ishod, a potom znanje primijeniti na težim vrstama pitanja. Sva ostala pitanja zahtijevaju samostalnu produkciju odgovora.

U vježbama kratkog odgovora (SA) najčešće se traži ispis programa ili sadržaj varijable nakon izvođenja isječka programa.

U vježbama dopunjavanja koda (SCW) potrebno je u jednu ili više praznina (*crta*) upisati riječi i znakove tako da cjelokupni program bude ispravan i daje traženi ispis ili sadržaj varijable. Iako je potrebno producirati relativno kratak odgovor, kao i kod SA vježbi, u SA vježbama generalno se traži *unaprijedno* razmišljanje, tj. što će se dogoditi izvođenjem zadanog programa. U vježbama dopunjavanja koda generalno se traži *unazadno* razmišljanje – što treba napisati kako bi se dobio traženi rezultat. Prema teoriji kognitivnog opterećenja [20], unazadno razmišljanje generalno je teže od unaprijednog pa su tako vježbe dopunjavanja koda generalno teže od vježbi kratkog odgovora.

U pravilu, najteže vježbe su one dugog odgovora (LA). Zadatak tih vježbi je napisati cjelokupni isječak programa koji će dati traženi ispis ili sadržaj varijable. Te vježbe dolaze u dvije inačice – slagalica i pisanje. U inačici slagalice, učenik od ponuđenih dijelova treba složiti cjelokupni kôd. U inačici pisanja, učenik potpuno samostalno treba napisati cjelokupni traženi kôd. Inačica slagalice generalno je lakša jer učeniku nudi dijelove točnog odgovora, koje on treba prepoznati i složiti u ispravnom poretku.

3.3. Korisnici

Prije korištenja aplikacije, korisnici se moraju prijaviti. Ako nemaju račun, stvaraju ga unošenjem traženih podataka, te se nakon toga prijavljuju. Autentifikacija se vrši na poslužitelju s pomoću JSON web tokena (JWT). Radi jednostavnosti korištenja, tj. smanjenja potrebe svakodnevne prijave, token je dugog vijeka trajanja (oko jedne godine). Takva odluka smanjuje sigurnost (krađa tokena omogućuje neovlašteni pristup aplikaciji), ali je donesena radi poboljšanja korisničkog iskustva. JWT autentikacija izabrana je umjesto autentifikacije kolačićima sjednice radi veće skalabilnosti.

JWT sadrži najbitnije informacije o korisniku (identitet i uloga), dok se sve detaljnije informacije mogu dobiti slanjem autentificiranog zahtjeva na poslužitelj.

Postoje četiri vrste korisnika u sustavu – učenik, učitelj, kreator i administrator. Učenik je primarna vrsta korisnika aplikacije, a njegova je uloga pohađanje tečaja u aplikaciji i usvajanje sadržaja. Uloga učitelja je definiranje razreda radi grupiranja učenika i nadzor napretka učenika. Uloga kreatora je stvaranje sadržaja – novih cjelina, lekcija i vježbi te preraspoređivanje sadržaja unutar tečaja. Uloga administratora je samo kreacija računa učitelja i kreatora.

3.4. Elementi igrifikacije

Iskustvo korištenja aplikacije obogaćeno je elementima igrifikacije. Implementirani elementi igrifikacije navedeni su i opisani u nastavku.

3.4.1. XP bodovi

Rješavanjem lekcija korisnik skuplja XP bodove (eng. *experience* – iskustvo). XP bodovi česti su element igrificiranih aplikacija. Iako sami po sebi nemaju izravno značenje, mogu

koristiti kao motivator učenicima (skupljanje bodova može biti više motivirajuće od rješavanja lekcije), ali i kao sredstvo za međusobno natjecanje. Učenici za rješavanje nove lekcije dobivaju 100 XP bodova, a za rješavanje bilo koje lekcije koju su već riješili dobivaju 40 XP bodova.

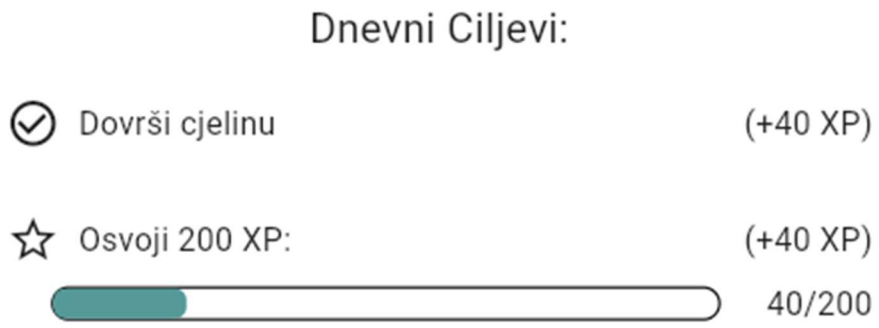
3.4.2. Dnevni ciljevi

Svaki dan korisnik dobiva po dva cilja koje treba ispuniti. Trenutno implementirane vrste ciljeva su:

- Visoka točnost (riješiti 2 lekcije, svaku s točnosti > 90 %)
- Velika brzina (riješiti 2 lekcije, svaku za manje od 45 sekundi)
- Dovrši cjelinu
- Osvoji 150 XP bodova

Osim što učeniku daju jasan zadatak, daju mu i nagradu. Rješavanje jednog dnevnog cilja učeniku nosi dodatnih 50 XP bodova.

Ako korisnik ne ispuni sve ciljeve u nekom danu, neće imati nikakve posljedice (osim što neće dobiti potencijalne XP bodove).



Slika 3.4 - Primjeri dnevnih ciljeva

3.4.3. Ljestvica poretka

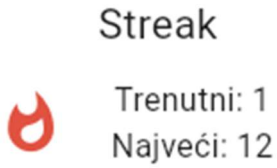
Učenici koji su dio razreda mogu se uspoređivati s ostalim učenicima koji su dio istog razreda. Nakon svake riješene lekcije te na ekranu profila mogu vidjeti ljestvicu poretka, na kojoj su poredani po ukupno osvojenim XP bodovima. Ova funkcionalnost može služiti i za vanjsko nagrađivanje (npr. nastavnik nagrađuje prva 3 učenika), ali i za međusobno natjecanje bez eksplicitne nagrade.

Kako bi se povećala kompetitivnost , moguće je izmijeniti funkcionalnosti tako da ljestvica poretka uzima u obzir samo XP bodove stečene u trenutnom tjednu/mjesecu. Tako bi se spriječilo pojedinog korisnika da u kratkom vremenu nagomila velik broj bodova i da ostali učenici steknu dojam kako nikada ne mogu dostići tu razinu.

Slika 3.5 - Primjer ljestvice poretka

3.4.4. Streak

Svakim danom u kojem učenik riješi lekciju (bilo novu ili neku već riješenu lekciju), produžuje mu se *streak* (niz dana). Ako učenik propusti dan rješavanja, *streak* mu se resetira na nulu. Ova funkcionalnost može potaknuti učenike na svakodnevno korištenje (i, posredno, učenje) bez eksplicitne nagrade. Osim što učenik ima uvid u svoj *streak*, na ljestvici poretka osim XP bodova prikazuje se i *streak* drugih korisnika. Tako se, osim osobnog mjerila uspjeha, *streak* koristi i kao sredstvo za natjecanje.



Slika 3.6 - Streak (niz dana rješavanja)

3.4.5. Slagalica kôda

U zadatcima dugog odgovora (LA), osim ručnog pisanja odgovora postoji i mogućnost slaganja odgovora korištenjem manjih dijelova. Zamorni zadatak pisanja kôda tako idejno postaje zabavniji jer nalikuje igri, a također je i olakšan jer gledanjem ponuđenih dijelova učenik može dobiti ideju odakle započeti pisanje. Ipak, kako ovo svojstvo ne bi previše olakšalo rješavanje vježbi, često su ponuđeni i pogrešni dijelovi kôda.

Slika 3.7 - Primjer slagalice kôda

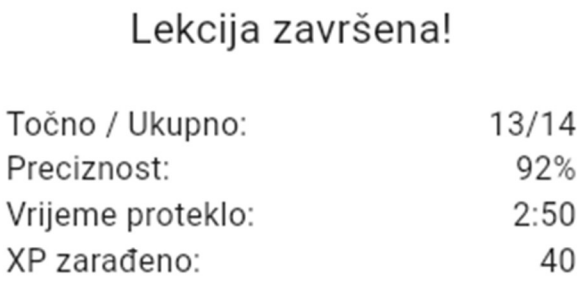
3.5. Elementi adaptivnosti

Osim igrifikacije, u sustav je bilo potrebno implementirati i elemente adaptivnog učenja. U nastavku su navedeni i opisani implementirani elementi.

3.5.1. Povratna informacija

Osnovno svojstvo adaptivnosti je prikaz povratne informacije učeniku. Nakon svake riješene vježbe, učenik dobiva povratnu informaciju – ako je vježbu riješio točno, dobiva zeleni oblačić i poruku da je vježbu riješio točno, a ako ju je riješio pogrešno, dobiva crveni oblačić i naveden točan odgovor.

Osim povratne informacije za pojedine vježbe, nakon svake lekcije učenik dobiva izvještaj o točnosti (eng. *accuracy*) i o proteklom vremenu. Tako učenik nakon dovršetka lekcije može imati predodžbu o generalnoj uspješnosti rješavanja.

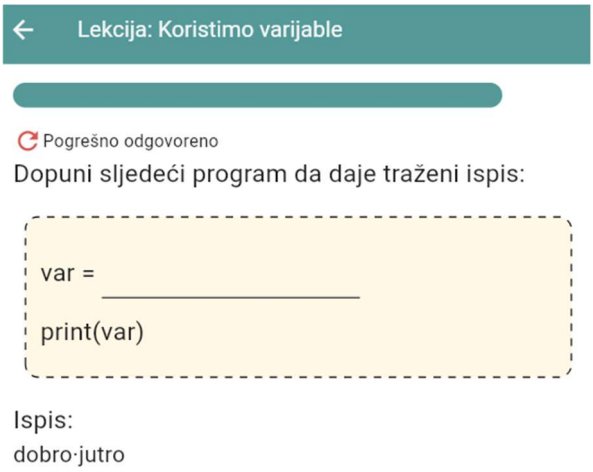


Slika 3.8 - Primjer povratne informacije

3.5.2. Ponavljanje pogrešno odgovorenih vježbi

S obzirom na to da je ideja aplikacije *učenje kroz praksu*, očekivano je da će učenik raditi greške za vrijeme rješavanja. Kad učenik pogriješi, prikaže mu se točan odgovor. Ipak, očekivano je da, ako je već jednom pogriješio na određenom pitanju, da nije sigurno je li usvojio zamišljeni koncept.

Zbog toga, pogrešno odgovorene vježbe ponavljaju se na kraju lekcije. Na taj je način moguće provjeriti je li učenik stekao neko znanje pogrešnim odgovorom i uvidom u točan odgovor. Ponavljanjem na kraju lekcije dobiva se i vremenski odmak pa se bolje može provjeriti je li učenik stvarno usvojio zamišljeni koncept, a ne samo naučio rješenje napamet.



Slika 3.9 - Ponavljanje pogrešno odgovorene vježbe

Vježba se učeniku prikazuje sve dok na nju ne ponudi točan odgovor. Moguće je da će učenik vježbu ponavljati puno puta, sve dok odgovor ne nauči napamet, ali višestrukim pogrešnim rješavanjem vježbe smanjivat će mu se razina, što je opisano u sljedećem potpoglavlju.

3.5.3. Adaptivne lekcije

Među najtežim je izazovima adaptivnosti učeniku dati zadatak primjerene težine u primjereno vrijeme. Neke lekcije obrađuju nove koncepte i u njima nema previše smisla regulirati težinu u ovisnosti o učeniku – svaki učenik treba proći lekcije s novim konceptima kako bi ih uopće imao priliku usvojiti. Ipak, za vježbu i ponavljanje korisno je prilagoditi težinu zadataka pojedinom učeniku kako bi se omogućilo učenje primjerenom brzinom prije nego što nastavi dalje. Takvim pristupom mogli bi se zadržati interes i motivacija učenika, zadržavajući izazov učenja na primjerenoj razini.

Adaptivne lekcije, koje su sadržane u adaptivnim cjelinama, rade upravo to – učeniku daju vježbe prilagođene njegovoj razini znanja. Algoritam koji predlaže primjerene vježbe učeniku nije jednoznačno određen. Način na koji funkcionira algoritam implementiran u ovom radu opisan je u nastavku:

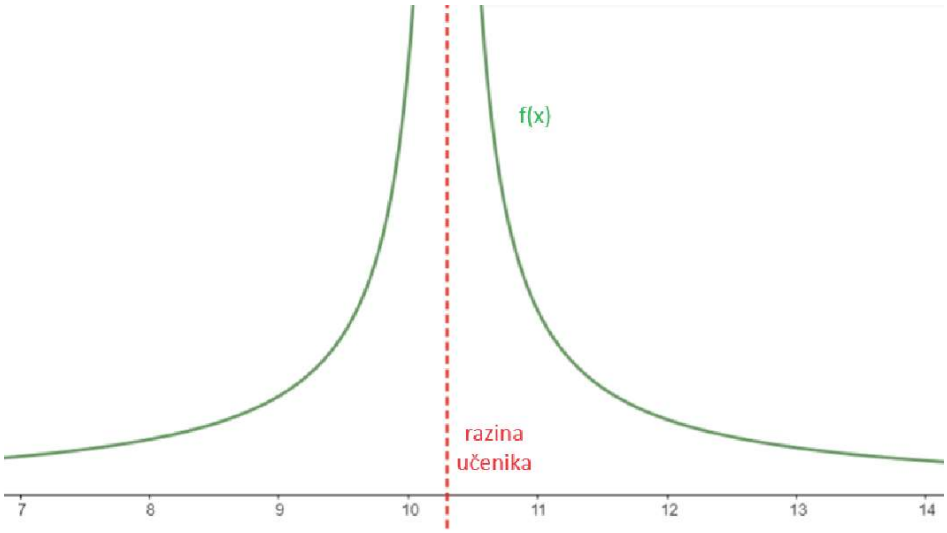
1. Svaka vježba ima određenu težinu (parametar *difficulty*, cijeli broj)
2. Svaki učenik ima trenutnu razinu znanja (parametar *score*, realan broj) – razina nema direktne veze s XP bodovima, ona služi samo u svrhu adaptivnih lekcija
3. Pokretanjem adaptivne lekcije iz skupa vježbi bira se 15 vježbi s težinom bliskom učenikovoj razini

Ideja ovog algoritma je da će učenik rješavanjem vježbi povećavati svoju razinu te da će mu u adaptivnim lekcijama dolaziti sve teže i teže vježbe. *Vježba s težinom bliskom učenikovoj razini* određuje se s dozom nasumičnosti – vježbe s težinom bliskom učenikovoj razini imaju veću vjerojatnost da će biti odabrane nego vježbe koje su značajno lakše/teže od učenikove razine. Vjerojatnost odabira vježbe proporcionalna je recipročnoj udaljenosti težine vježbe od razine učenika, tj. formalno:

$$d_{total} = \sum_{vježba \in vježbe} \frac{1}{|težina(vježba) - razina(učenik)|}$$

$$P_{odabira}(vježba) = \frac{1}{|težina(vježba) - razina(učenik)| * d_{total}}$$

Zbroj svih vjerojatnosti dat će 1, a vježbe koje su *dalje* s obzirom na težinu imat će manju vjerojatnost odabira. Ova metoda odabira naziva se i odabir kotača ruleta (eng. *roulette-wheel selection*). Distribucija vjerojatnosti odabira vježbe ovisno o njenoj težini i o razini učenika prikazana je na sljedećem grafu:



Slika 3.10 - Distribucija vjerojatnosti odabira vježbe ovisno o njenoj težini i o razini učenika

Ipak, ideja je pri odabiru vježbi preferirati vježbe teže od učenikove razine kako bi se postiglo učenje, tj. kako bi učenik dobio izazov. Zato je od N odabranih vježbi unaprijed određeno da je 70 % težih od učenikove razine, a 30 % lakših. Potom se od težih vježbi bira 0,7N vježbi po odabiru kotača ruleta, a od lakših 0,3N istom metodom.

Primjerice, učenik ima razinu 5,5 i pokreće adaptivnu lekciju. Bit će odabrano 5 lekcija s težinom < 5,5 i 10 lekcija s težinom > 5,5. Od težih lekcija, one s težinom 6 bit će

najvjerojatnije za odabir, dok će npr. neka lekcija težine 10 moći biti odabrana, ali s vrlo malom vjerojatnosti. Analogno vrijedi i za lakše lekcije.

Kako bi se spriječila pojava vježbi s gradivom koje učenik još nije obradio, skup svih vježbi koje mogu biti izabrane u adaptivnoj lekciji sadrži samo vježbe koje su bile prisutne u nekoj od već savladanih lekcija.

Za ispravno funkcioniranje ovog algoritma nužna su dva preduvjeta – točnost učenikove razine i točnost težine vježbi. Zato su razvijeni algoritmi koji će adaptirati te parametre, a opisani su u nastavku.

3.5.4. Algoritam prilagodbe razine učenika

Učenikova razina prilagođava se automatski, svakim davanjem odgovora za neku vježbu. Njegova razina ostat će ista ako točno riješi vježbu s težinom manjom od svoje razine (točno riješi *lakšu* vježbu). Također, razina će mu ostati ista i ako pogrešno riješi vježbu s težinom većom od svoje razine (pogrešno riješi *težu* vježbu).

Izmjena razine događa se u preostala dva slučaja – ako učenik pogrešno riješi vježbu s težinom manjom od svoje razine (pogrešno riješi *lakšu* vježbu), kad će mu se razina smanjiti te ako učenik točno riješi vježbu s težinom većom od svoje razine (točno riješi *težu* vježbu).

Naime, očekuje se da učenik zna riješiti lakše vježbe i da smije ne znati riješiti teže vježbe (propušta priliku za učenje). S druge strane, ako ne zna riješiti lakšu vježbu, očito mu je razina prevelika. Također, ako zna riješiti težu vježbu, očito je naučio nešto teže od svoje razine i ona se treba podesiti.

Podešavanje razine odvija se po sljedećoj formuli:

$$
\begin{aligned}
&razina(učenik) \\
&\coloneqq razina(učenik) \\
&+ \begin{cases}
0, & točna\ lakša\ vježba \\
0, & pogrešna\ teža\ vježba \\
\big(težina(vježba) - razina(učenik)\big) * \eta^{+}, & točna\ teža\ vježba \\
-\big(razina(učenik) - težina(vježba)\big) * \eta^{-}, & pogrešna\ lakša\ vježba
\end{cases}
\end{aligned}
$$

Stope učenja η^{+} i η^{-} mogu se izmjenjivati, a obje trenutno implementirane vrijednosti iznose 0,1.

Iz osmišljene formule vidljivo je još jedno svojstvo ovog algoritma – što je težina vježbe udaljenija od razine učenika, promjena razine (kazna/nagrada) je veća. U svakom slučaju, učenikova razina se *približava* težini vježbe u navedena dva slučaja.

3.5.5. Algoritam prilagodbe težine za vježbe

Pri izradi vježbe, kreator predlaže razinu vježbe, i to predstavlja njezinu inicijalnu vrijednost. Ipak, puštanjem vježbe u sustav gdje ga koriste učenici, može se pokazati da je vježba značajno lakša/teža od očekivane težine. Kreator u bilo kojem trenutku može analizirati sve postojeće vježbe. Za sve će izračunati nekoliko indikatora – prosječnu razinu učenika koji su točno riješili zadanu vježbu, prosječnu razinu učenika koji su pogrešno riješili zadanu vježbu te predloženu težinu.

Predložena težina određuje se kao najmanja težina na kojoj bi točnost rješavanja bila veća od 60 % za učenike s razinom iznad te težine. Primjerice, ako svi učenici s razinom >= 3 u prosjeku rješavaju vježbu s 52 % točnosti, svi učenici s razinom >= 4 s 57 % točnosti, a svi učenici s razinom >= 5 sa 65 % točnosti, onda će predložena težina biti 5.

Potrebno je uočiti kako će zbog ponavljanja pogrešno odgovorenih vježbi vrlo često učenici zadatak koji jednom odgovore pogrešno jednom odgovoriti i ispravno, stoga će prosječna točnost rješavanja najčešće biti iznad 50 % za bilo koju vježbu. Naravno, moguće je i da netko puno puta pogrešno odgovori na vježbu sve dok jednom ne ponudi točan odgovor. Zbog te je činjenice prag točnosti stavljen na 60 %.

Postoje i neki rubni slučajevi. Ako ne postoje učenici koji su vježbu riješili pogrešno, predložena će težina biti prosječna razina svih učenika koji su ju riješili točno. Nadalje, ako ne postoje učenici koji su vježbu riješili točno (očito je preteška), tada će predložena težina biti najviša razina od svih učenika koji su pogrešno riješili taj zadatak. Za kraj, ako niti s jednom *predloženom* težinom točnost ne iznosi više od 60 %, tada će predložena težina biti najviša razina od svih učenika koji su ikad pokušali riješiti zadatak.

Bitno je istaknuti i da se prilagodba težine vježbe ne događa automatski, već ju kreator mora napraviti ručno. Ova odluka donesena je iz nekoliko razloga. Učenici bi namjernim pogrešnim rješavanjem vježbe mogli smanjiti njezinu predloženu težinu a da to ne govori ništa o stvarnoj težini vježbe. Također, vježba može doista biti teška, ali ju učenici s manjom razinom mogu točno rješavati jer su dobro usvojili gradivo. I kao dodatan razlog, mjere koje se prikazuju kreatoru služe samo kao vodilja/upozorenje, ali same po sebi ne sadrže

informaciju o stvarnoj težini vježbe. Zbog toga je finalna odluka o težini vježbe prepuštena kreatoru i unosi se ručno.

4. Strukture podataka

U ovom će poglavlju biti objašnjena struktura baze podataka te detalji entiteta korišteni u aplikaciji.

4.1. Baza podataka

Za pohranu podataka korištena je NoSQL baza podataka. Konkretna implementacija je MongoDB.

Razlog odabira NoSQL baze u odnosu na relacijsku bazu je izostanak potrebe za shemom baze i veća skalabilnost. Iako bi baza podataka za ovu potrebu mogla biti implementirana i kao relacijska baza, postoji nekoliko vrsta entiteta sličnih atributa, za koje ima više smisla pohranjivati ih kao JSON objekte nego kao instance u relaciji. Također, korištenje ovakvog pristupa dopušta i denormalizaciju koja je u nekim primjenama potrebna (npr. bilježenje dnevničkih zapisa – eng. *logs*), kao i spremanje ugniježđenih podataka.

Osim toga, korištenje NoSQL pristupa omogućuje i pohranu složenijih tipova podataka (liste, objekti…) što nije ustaljena praksa s relacijskim bazama. Osim same strukture podataka, skalabilnost u ovakvoj aplikaciji je od iznimne važnosti – brzina pristupa i mogućnost gradnje raspodijeljenih sustava je iznimno važna radi osiguranja dostave informacija korisniku u dovoljno kratkom vremenu.

NoSQL baza podataka sastoji se od kolekcija koje su navedene u nastavku, a za svaku su kolekciju navedeni njezino značenje i opis atributa. Obavezni atributi označeni su zvjezdicom (*). Prilikom rada aplikacije, često se šalju neki podskupi ili kombinacije navedenih entiteta nužnih za neku funkciju. Ovdje su navedeni atributi koji se pohranjuju u bazi podataka.

4.2. Users (korisnici)

Svi korisnici aplikacije pohranjuju se u kolekciju *users*. Glavni atributi, koje imaju svi korisnici, i njihovi opisi su:

Tablica 4.1 - Atributi kolekcije *users*

Naziv atributa	Tip podatka	Opis atributa
UserName *	string	Korisničko ime – jedinstveni identifikator korisnika
PasswordHash *	string	Sažetak lozinke (eng. *hash*) – vrijednost kojom se provjerava lozinka bez pohrane sadržaja lozinke
Roles *	List<string>	Popis uloga korisnika
FirstName	string	Ime korisnika
LastName	string	Prezime korisnika
DateOfBirth	DateOnly	Datum rođenja korisnika
HighestLessonId	int	Id najviše dovršene lekcije
HighestLessonGroupId	int	Id najviše dovršene cjeline
NextLessonId *	int	Id lekcije koju treba dovršiti za nastavak
NextLessonGroupId *	int	Id cjeline koju treba dovršiti za nastavak
TotalXP *	int	Ukupan broj XP bodova
XPachieved *	List<Pair<DateTime, int>>	Lista osvojenih XP bodova s trenutkom osvajanja
School	string	Škola kojoj korisnik pripada
Quests	List<Pair<DateOnly, List<Quest>>>	Lista dnevnih ciljeva za svaki datum kad se korisnik prijavio u sustav
Score *	double	Razina učenika (korištena za adaptivnost)

UserGroup	int	Indikator verzije aplikacije koju će dobiti učenik (puna/kontrolna verzija)

Moguće uloge korisnika su STUDENT, TEACHER, CREATOR i ADMIN.

Svaki *Quest* (cilj) sastoji se od tipa cilja (string), datuma, ograničenja (int), broja lekcija (int), napretka (int) i indikatora završenosti (bool). Vrste ciljeva detaljnije su prethodno opisane u potpoglavlju 3.4.2.

4.3. Lesson_groups (cjeline)

Podatci o cjelinama pohranjuju se u kolekciju *lesson_groups*. Atributi koji čine cjelinu su:

Tablica 4.2 - Atributi kolekcije *lesson_groups*

Naziv atributa	Tip podatka	Opis atributa
id *	int	Jedinstveni identifikator cjeline
name *	string	Naziv cjeline
tips	string	Tekst sa savjetima za cjelinu
lessons *	List<int>	Lista identifikatora lekcija koje su sadržane u cjelini
order *	int	Redni broj cjeline u tečaju
adaptive	bool	Indikator radi li se o adaptivnoj lekciji

Radi štednje prostora i očuvanja integriteta pri promjenama, popis lekcija sadrži samo identifikatore, a ne cijele lekcije (tj. nije provedena denormalizacija).

4.4. Lessons (lekcije)

Podatci o lekcijama pohranjuju se u kolekciju *lessons*. Atributi koji čine lekciju su:

Tablica 4.3 - Atributi kolekcije *lessons*

Naziv atributa	Tip podatka	Opis atributa

id *	int	Jedinstveni identifikator lekcije
name *	string	Naziv lekcije
specificTips	string	Tekst sa savjetima za konkretnu lekciju
exercises *	List<int>	Lista identifikatora vježbi koje su sadržane u lekciji

Ponovno, radi štednje prostora i očuvanja integriteta pri promjenama, popis vježbi sadrži samo identifikatore vježbi, a ne cijele vježbe.

4.5. Exercises (vježbe)

Podatci o konkretnim vježbama nalaze se u kolekciji *exercises*. S obzirom na to da postoji 4 vrste vježbi, prvo su navedeni zajednički atributi, a potom specijalni. Sve vježbe sastoje se od:

Tablica 4.4 - Atributi kolekcije *exercises*

Naziv atributa	Tip podatka	Opis atributa
id *	int	Jedinstveni identifikator lekcije
type *	string (enum)	Tip vježbe
difficulty *	int	Težina vježbe
statement	string	Uvodni tekst vježbe
statementOutput	string	Ispis kôda zadanog u zadatku ili ispis koji treba producirati uneseni kôd
specificTip	string	Tekst sa savjetima za konkretnu vježbu

Tipovi vježbi su MC, SA, LA i SCW (detaljnije opisani u potpoglavlju 3.2).

U nastavku su popisani i specifični atributi za svaki od tipova vježbi. Te atribute vježbe mogu, ali ne moraju sadržavati (osim u slučaju obaveznih atributa označenih zvjezdicom):

Tablica 4.5 - Specifični atributi MC vježbi

Naziv atributa	Tip podatka	Opis atributa
statementCode	string	Kôd zadan uz tekst zadatka
question	string	Pitanje vezano uz tekst zadatka i kôd
answerOptions *	Map<string, string>	Ponuđeni odgovori (točnije parovi slovne oznake i ponuđenog odgovora)
correctAnswer *	string	Slovna oznaka točnog odgovora

Tablica 4.6 - Specifični atributi SA vježbi

Naziv atributa	Tip podatka	Opis atributa
statementCode	string	Kôd zadan uz tekst zadatka
question	string	Pitanje vezano uz tekst zadatka i kôd
correctAnswers	List<string>	Odgovori koji se prihvaćaju kao točni

Tablica 4.7 - Specifični atributi LA vježbi

Naziv atributa	Tip podatka	Opis atributa
answerOptions	Map<string, string>	Dijelovi kôda koji se koriste u načinu slagalice (točnije parovi rednog broja dijela kôda i sadržaja kôda)
correctAnswers *	List<string>	Odgovori koji se prihvaćaju kao točni

Tablica 4.8 - Specifični atributi SCW vježbi

Naziv atributa	Tip podatka	Opis atributa
statementCode *	string	Kôd zadan u zadatku koji sadrži praznine

correctAnswers	List<List<string>>	Odgovori koji se prihvaćaju kao točni za svaku prazninu
defaultGapLengths *	List<int>	Inicijalne duljine praznina za nadopunjavanje kôda

4.6. Classes (razredi)

Podatci o razredima učenika pohranjuju se u kolekciji *classes*, a sadržana su sljedeća svojstva:

Tablica 4.9 - Atributi kolekcije *classes*

Naziv atributa	Tip podatka	Opis atributa
id *	int	Jedinstveni identifikator razreda
name *	string	Ime razreda
school *	string	Ime škole kojoj razred pripada
teacherUsername *	string	Korisničko ime učitelja koji je stvorio razred
students *	List<string>	Lista korisničkih imena učenika koji su dio razreda

4.7. Logs (dnevnički zapisi)

Zadnja kolekcija sadrži podatke o dnevničkim zapisima (eng. *logs*). Oni služe za analize – analizu težine vježbi, analizu uspješnosti učenika, analizu vremena provedenog učeći, analizu broja riješenih lekcija…

Postoje tri vrste zapisa – LogStartLesson (zapis početka lekcije), LogEndLesson (zapis kraja lekcije) i LogExerciseAnswer (zapis ponuđenog odgovora na vježbu). Oni dijele sljedeća svojstva:

Tablica 4.10 - Atributi kolekcije *logs*

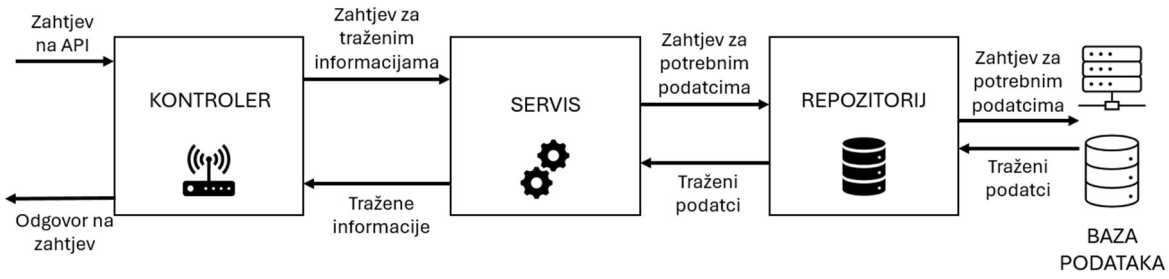
Naziv atributa	Tip podatka	Opis atributa
id *	ObjectId	Jedinstveni identifikator zapisa
userId *	string	Identifikator korisnika (korisničko ime)
userGroup	string	Indikator verzije aplikacije koju korisnik koristi
timestamp	DateTime	Trenutak kad je zapis stvoren

Osim zajedničkih svojstava, LogStartLesson sadrži identifikator lekcije koja je započeta. LogEndLesson sadrži identifikator lekcije koja je završena, ukupan broj ponuđenih odgovora, broj točnih ponuđenih odgovora, trajanje rješavanja lekcije i točnost rješavanja. LogExerciseAnswer sadrži identifikator vježbe, točan odgovor, ponuđen odgovor, je li pitanje odgovoreno točno i učenikovu razinu u trenutku zapisa.

5. Arhitektura poslužiteljske aplikacije

Poslužiteljski sloj izgrađen je u radnom okviru .NET. Radni okvir .NET (*dot net*) razvio je Microsoft. Obilježava ga Common Language Runtime (CLR), koji pokreće aplikacije koje se izvršavaju, a nudi upravljanje memorijom i dretvama, sigurnost te *garbage collection*. Glavni programski jezik kôda u .NET-u je C# [21].

Poslužiteljski sloj spaja se na bazu podataka, iz nje dohvaća podatke, obrađuje ih te ih šalje klijentskoj aplikaciji. Radi raspodjele odgovornosti unutar poslužiteljske aplikacije, ona je logički organizirana prema Controller-Service-Repository obrascu [22].



Slika 5.1 - Obrazac kontroler-servis-repozitorij

Konkretna implementacija sadrži i dva dodatna sloja – ugovorni (eng. *Contracts*) i konfiguracijski (eng. *Configuration*), ali oni služe spomenutim logičkim slojevima, tj. bitni su za unutarnju implementaciju. Pojedini logički slojevi opisani su u nastavku:

5.1. Kontroleri

Kontroleri (eng. c*ontrollers*) razredi su koji služe za otkrivanje funkcionalnosti aplikacije drugim aplikacijama preko aplikacijsko-programskog sučelja, tj. API-ja (eng. application programming interface). Svaki od kontrolera ima svoju adresu te prima zahtjeve po protokolu HTTP te šalje odgovore na njih.

Operacije koje kontroleri nude oni ne obavljaju sami. Gotovo sva logika aplikacije odvija se u drugim dijelovima (primarno u servisima). Funkcija kontrolera je dobivene podatke uobličiti u prikladan HTTP odgovor s prikladnim statusnim kôdom.

U aplikaciji razvijenoj u ovom radu postoji pet kontrolera: *ExercisesController*, *LessonsController*, *LessonGroupsController*, *UserController* i *InteractionController*.

Funkcije koje nudi *ExercisesController* povezane su s vježbama, a to su: CRUD operacije za vježbe (eng. *create*, *read*, *update* i *delete*), validacija ponuđenih odgovora, te dohvat predložene težine za tražene vježbe.

Funkcije koje nudi *LessonsController* povezane su s lekcijama, a to su CRUD operacije za lekcije.

Funkcije koje nudi *LessonGroupsController* povezane su s cjelinama, a to su CRUD operacije za cjeline.

Funkcije koje nudi *UserController* povezane su s korisnicima: registracija korisnika, prijava u sustav, dohvat podataka o trenutnom korisniku, završetak lekcije, promjena lozinke, registracija učitelja, registracija kreatora te izmjena osobnih podataka o korisniku. Završetak lekcije jedina je funkcija koja naizgled više pripada *LessonsControlleru*, ali gotovo sva logika koja se treba odviti završetkom lekcije zapravo se tiče podataka o korisniku, a ne podataka o lekciji pa, zapravo, ta funkcija puno prikladnije spada u *UserController*.

Funkcije koje nudi *InteractionController* povezane su s interakcijom među korisnicima (najčešće vezano uz igrifikaciju), a to su: CRUD operacije za razred, dohvat škola, dohvat i filtriranje učenika od strane učitelja te dohvat ljestvice poretka (za sebe ili za neki razred).

5.2. Servisi

Servisi se bave poslovnom logikom aplikacije, tj. nude funkcije za dohvat, obradu i pohranu podataka. Sam dohvat i pohrana podataka ne obavljaju servisi, već taj posao delegiraju dalje repozitorijima, ali dohvaćene podatke mogu obogatiti podatcima iz više izvora, raditi provjere nad dobivenim podatcima i računati neke izvedene vrijednosti.

Aplikacija implementira sedam servisa. Za svaki kontroler spomenut u prethodnom poglavlju postoji i servis koji omogućuje funkcije koje su navedene. Osim tih pet servisa, postoji i *LogsService* te *TokenGeneratorService*.

LogsService poziva više kontrolera, a zadaća mu je pohrana te dohvat dnevničkih zapisa. Iako se sama pohrana i dohvat delegiraju repozitoriju, servis iz dohvaćenih podataka o dnevničkim zapisima bira one relevantne (konkretno, vezane za odgovore na pojedinu vježbu) i prosljeđuje ih pozivatelju. Isto tako servis formira strukture podataka dogovorene

u ugovorima kako bi ih repozitorij mogao jednostavno spremiti, bez potrebe za tumačenjem gdje se neki podatak nalazi.

TokenGeneratorService ima jednu funkciju, a to je stvaranje JWT. Postavke JWT čita iz postavki aplikacije (trajanje tokena postavljeno je na oko jednu godinu) te token potpisuje SHA256 algoritmom. Taj token šalje se uz svaki autentificirani zahtjev poslužitelju, a radni okvir .NET automatski uz pomoć tokena autentificira i autorizira zahtjeve korisnika i tako dopušta/brani pristup pojedinim funkcijama kontrolera.

5.3. Repozitoriji

Repozitoriji su najniži sloj u kontekstu rukovanja s podatcima. Svaki repozitorij povezan je na jednu kolekciju u bazi podatka. Repozitoriji generalno nude CRUD operacije za tip entiteta koji se pohranjuje u odgovarajućoj kolekciji, ali često nude i specifičnije funkcije za dohvat podataka koje, primjerice, filtriraju podatke po nekom predanom argumentu (npr. ID entiteta ili neki ne ključni atribut).

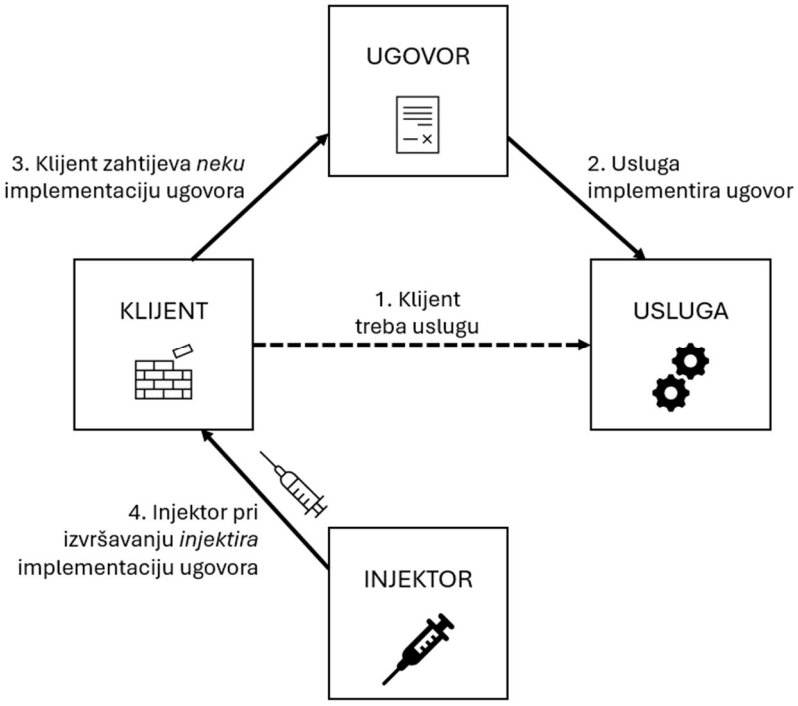
U razvijenoj aplikaciji implementirano je pet repozitorija: *ClassesRepository*, *ExercisesRepository*, *LessonGroupsRepository*, *LessonsRepository* i *LogsRepository*. Svaki od njih povezan je na kolekciju u bazi podataka s odgovarajućim imenom.

Jedina kolekcija koja nema svoj repozitorij je *users*. Razlog tomu je što .NET nudi ugrađeni repozitorij *UserManager*, koji se podešava u konfiguraciji aplikacije i ima implementirane sve potrebne funkcije za rad s korisnicima. Tako nije bilo potrebno ručno implementirati sažimanje lozinke (eng. *hashing*), već je ta funkcija automatski spremna za korištenje, uz niz dodatnih funkcija koje nisu korištene u ovoj aplikaciji.

5.4. Ugovori

Radi postizanja inverzije ovisnosti (eng. *Dependency Inversion*), servisi i repozitoriji implementiraju odgovarajuća sučelja (eng. *interface*) koja su definirana u ugovornom sloju. Na taj način, moduli više razine (kontroleri i servisi) ne ovise o modulima niže razine (servisima i repozitorijima), već ovise o njihovim apstrakcijama, tj. sučeljima.

Konkretne implementacije sučelja definiraju se u konfiguracijskom dijelu, a radni okvir potom automatski pruža te implementacije svim klasama koje ih traže u svojim konstruktorima.



Slika 5.2 - princip inverzije upravljanja (ubacivanja ovisnosti)

Osim sučelja, ugovorni sloj definira i strukturu entiteta koji se koriste u aplikaciji. To su entiteti koji se pohranjuju u bazi, ali i neki izvedeni entiteti (npr. specifični tipovi vježbi, ljestvica poretka, dnevni cilj, izvještaj s kraja lekcije…). Osim entiteta, definirane su enumeracije, iznimke i objekti za prijenos podataka (DTO, eng. *Data Transfer Object*). DTO-ovi se najčešće koriste u kontrolerima, a služe kao odgovor na korisnički zahtjev na API. Eksplicitnim definiranjem objekata koji se šalju kao odgovor na zahtjeve lakše je uskladiti poslužiteljsku i klijentsku aplikaciju u pogledu podataka koji se razmjenjuju.

5.5. Postavljanje aplikacije

Kako bi se klijentska aplikacija mogla spojiti na API koji nudi poslužiteljska aplikacija, poslužiteljsku aplikaciju bilo je potrebno postaviti u rad (eng. *deploy*). Zbog prilagođenosti .NET aplikacijama za objavu je odabran portal Azure.

Kako bi izmjene na izvornom kodu aplikacije mogle biti automatski dostupne na produkcijskoj verziji aplikacije, stvoren je *CI/CD pipeline* (cjevovod) s pomoću *GitHub Actions Workflowa*.

Workflow (tijek rada) implementiran za ovu svrhu sastoji se od dvije faze – izgradnja i objava. Izgradnja aplikacije kompajlira izvorni kôd aplikacije, dok faza objave prenosi dobiveni izvršni kôd na *Azure WebApp* servis, gdje se on pokreće i postaje javno dostupan.

S obzirom na to da su neki podatci u aplikaciji tajni (npr. ključ za spajanje na bazu podataka ili ključ kojim se potpisuju JWT), oni nisu uneseni u izvorni kod aplikacije (eng. *hard-coded*), već se čitaju iz okoline (eng. *environment*) i tako su vidljivi samo aplikaciji dok je pokrenuta, ali ne i prije nego što se pokrene. Na taj je način izvorni kod siguran za javnu objavu i aplikacija može funkcionirati kao aplikacija otvorenog koda (eng. *open-source*).

6. Struktura klijentske aplikacije

6.1. Flutter

Klijentska aplikacija izrađena je u radnom okviru Flutter. Flutter je radni okvir otvorenog kôda koji je razvio Google, a namijenjen je izgradnji izvorno kompajliranih višeplatformskih aplikacija koristeći jedinstveni izvorni kôd. Izvorni kôd piše se u programskom jeziku Dart [23]. Aplikacije razvijene u Flutteru mogu se kompajlirati za web, Windows, macOS, iOS, Android, Linux i Fuchsia OS.

Osnovni elementi programa u Flutteru su *widgeti*, koji mogu sadržavati druge *widgete*. Flutter ne koristi izvorne komponente platforme na kojoj se nalazi, već renderira *widgete* piksel po piksel. Kôd u Flutteru češće je strukturiran kao kompozicija *widgeta* nego kao nasljeđivanje *widgeta* ili dodavanje atributa svakom *widgetu*. Takav pristup omogućuje veću modularnost [24].

```
body: SingleChildScrollView(
  child: Padding(
    padding: const EdgeInsets.symmetric(vertical: 30),
    child: Container(
      constraints: const BoxConstraints(minWidth: 500.0),
      child: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Padding(
              padding: const EdgeInsets.symmetric(
                  vertical: 20.0, horizontal: 50.0), // EdgeInsets.symmetric
              child: RichTextMarkdown(
                text: lesson.specificTips!,
                style: TextStyle(
                  color: Theme.of(context).colorScheme.onSurface,
                  fontSize: 16,
                ), // TextStyle
              ), // RichTextMarkdown
            ), // Padding
            nextButton,
            const SizedBox(height: 50.0)
          ],
        ), // Column
      ), // Center
    ), // Container
  ), // Padding
), // SingleChildScrollView
```

Slika 6.1 - Struktura kôda kao kompozicije

U nastavku je dan pregled klijentske aplikacije i proces njenog korištenja.

6.2. Struktura kôda

Kao i poslužiteljska aplikacija, i klijentska aplikacija implementira obrazac modela, repozitorija i servisa. Modeli definiraju razrede s atributima i pomoćnim metodama koji će se koristiti pri dohvatu podataka s poslužitelja i razmjeni podataka unutar aplikacije. Repozitoriji se bave samim dohvatom podataka – brinu se o spajanju na API te o deserijalizaciji objekata definiranih u modelima. Servisi nude sve aplikaciji potrebne metode za koje je potrebna poslovna logika. Dakle, u servisima se nalaze metode za dohvat podataka (koje se delegiraju repozitorijima), ali i za autentifikaciju, za pokretanje sjednice učenja, dohvat podataka o trenutnom korisniku itd.

Drugi dio kôda posvećen je *widgetima*. On se dijeli u pet kategorija – *widgeti* za učenika, za učitelja, za kreatora, za administratora i za autentifikaciju. Generalno je aplikaciju moguće logički podijeliti i u četiri odvojene *podaplikacije* (tj. korisnička sučelja), koja dijele samo proces prijave. Ipak, aplikacija čini jednu cjelinu zbog dijeljenih struktura podataka, repozitorija i servisa, a i jer se pojedini *widgeti* namijenjeni učeniku koriste i u pogledu učitelja i kreatora, s namjerom uvida u učeničko iskustvo. Primjerice, kreator pri pregledu vježbi može vidjeti točno kako će pojedina vježba izgledati učeniku, kako bi mogao bolje prilagoditi sadržaj vježbe (npr. skratiti predugi tekst ili ga prilagoditi gumbima na ekranu).

Svako od sučelja prikazano je u nastavku.

6.3. Sučelje za autentifikaciju

Dio za autentifikaciju je vrlo jednostavan – strukturno se sastoji od jednog ekrana koji nudi prijavu ili registraciju. Početni prikaz nudi prijavu, a klikom na odgovarajući gumb otvara se forma za registraciju. Uspješnom registracijom stvara se učenički račun, a s unesenim podatcima potrebno je prijaviti se u sustav.

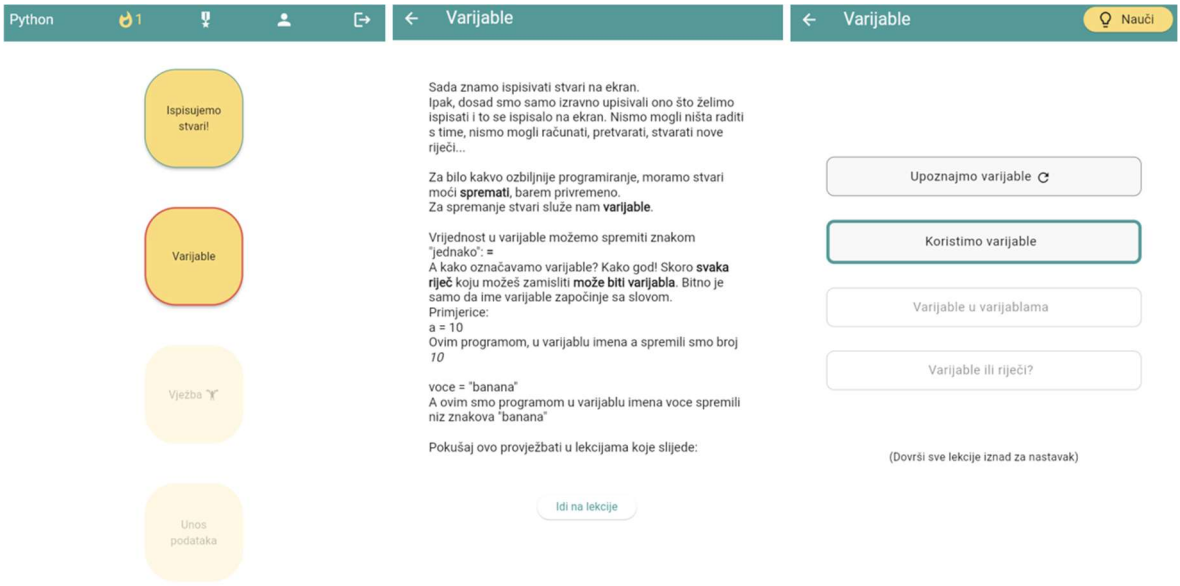
Slika 6.2 - Ekrani za autentifikaciju

6.4. Učeničko sučelje

Glavni fokus aplikacije je na dijelu za učenika jer su upravo učenici primarni korisnici aplikacije.

Nakon prijave u aplikaciju učeničkim računom, učenik dobiva popis svih cjelina u aplikaciji. Cjeline kojima može pristupiti obojene su normalno, dok su one kojima ne može pristupiti poluprozirne. Učenik može pristupiti rješavanju svih cjelina koje je već riješio te prvoj cjelini koju još nije riješio (dakle, cjeline mora rješavati serijski).

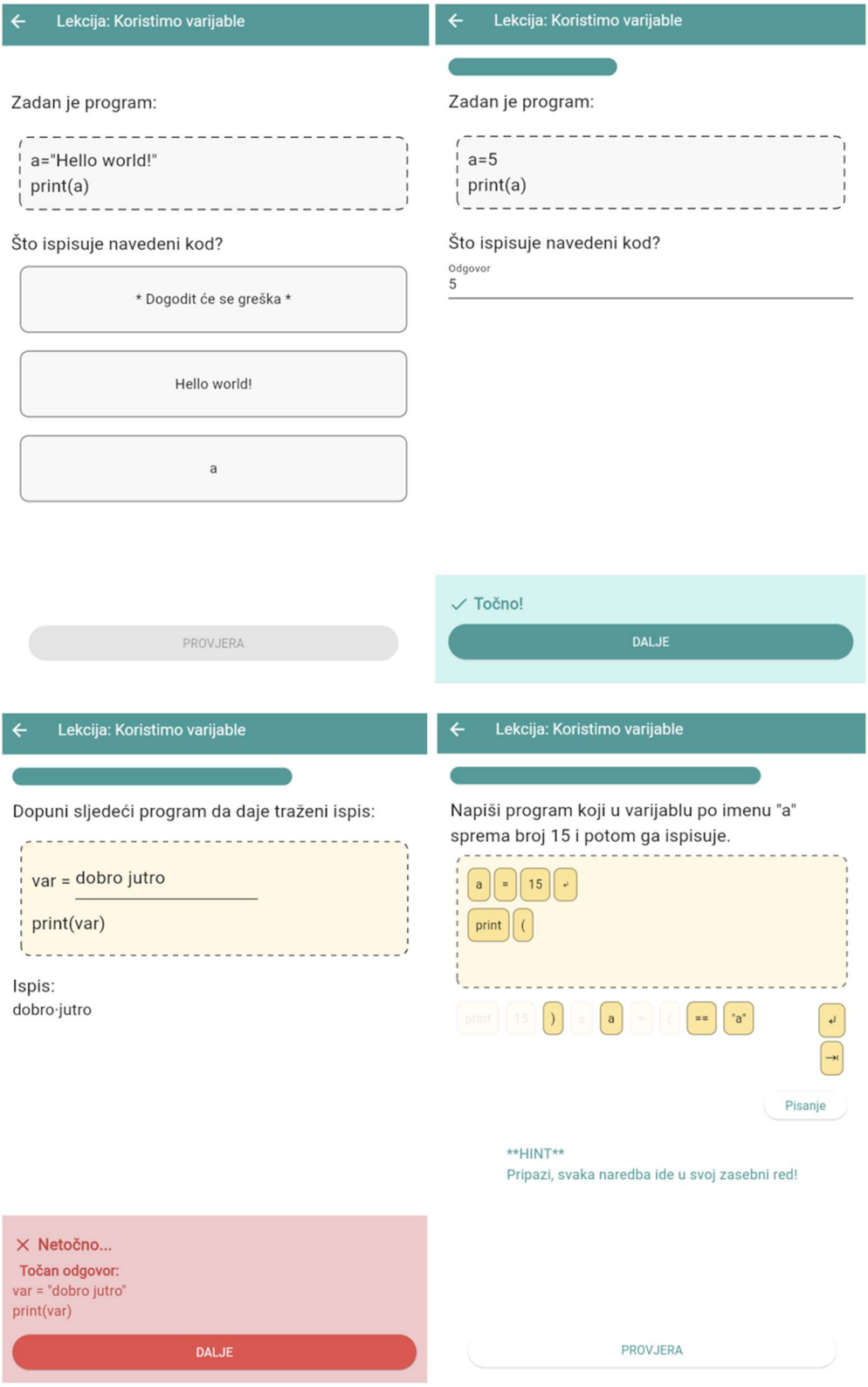
Klikom na cjelinu učenik ulazi u popis lekcija koje se nalaze u toj cjelini. Ponovno, cjeline koje može riješiti prikazane su normalno, dok su one koje ne može riješiti prikazane poluprozirno, a lekcije se također rješavaju serijski. Prije rješavanja lekcija u cjelini, učenik može pročitati savjete za tu cjelinu (prozor *Nauči*), a svaka lekcija može imati i specifične savjete vezane za ciljano gradivo. Savjeti vezani za lekciju prikazat će se ulaskom u lekciju, prije početka rješavanja vježbi.



Slika 6.3 - Proces započinjanja rješavanja lekcije

Ulaskom u lekciju (nakon eventualnih savjeta) učenik redom dobiva vježbe koje treba riješiti. Za četiri tipa vježbi postoje četiri vrste ekrana – ekran za vježbe s višestrukim izborom (MC), ekran za vježbe s kratkim odgovorom (SA), ekran za vježbe s dugim odgovorom (LA) i ekran za vježbe s dopunjavanjem kôda (SCW).

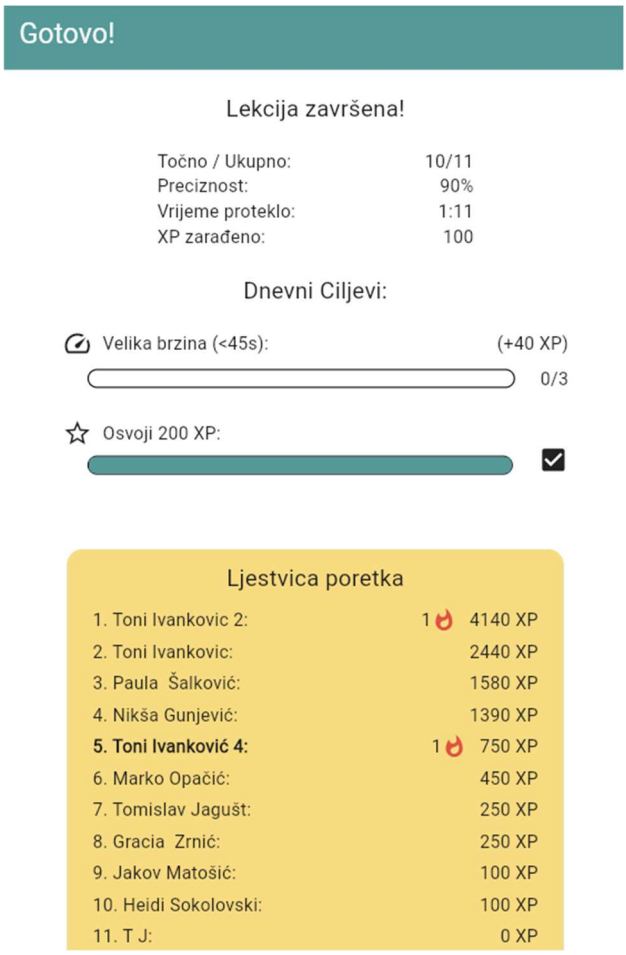
Svi ekrani prikazuju napredak u lekciji (eng. *progress bar*). Učenik, kada je spreman predati odgovor na vježbu, može pritisnuti gumb provjera. Aplikacija odgovor provjerava na poslužitelju, te povratnu informaciju vraća učeniku. U slučaju točnog odgovora, učenik može jednostavno nastaviti dalje, dok u slučaju pogrešnog odgovora učenik dobiva informaciju o točnom odgovoru te također može nastaviti dalje.



Slika 6.4 - Vrste vježbi

Pogrešno odgovorene vježbe ponavljaju se na kraju lekcije, sve dok ih učenik ne riješi točno, kao što je opisano u potpoglavlju 3.5.2.

Nakon što je sve vježbe u lekciji učenik riješio točno, dobiva povratnu informaciju o svom rješavanju lekcije – broj točnih odgovora naspram ukupnog broja odgovora, točnost rješavanja (eng. *accuracy*) i vrijeme koliko je trajalo rješavanje lekcije. Osim podataka o rješavanju lekcije, dobiva i podatke o napretku vezanom za elemente igrifikacije – dobiveni XP bodovi, napredak na dnevnim ciljevima te ljestvicu poretka sa svim učenicima iz razreda ili škole.



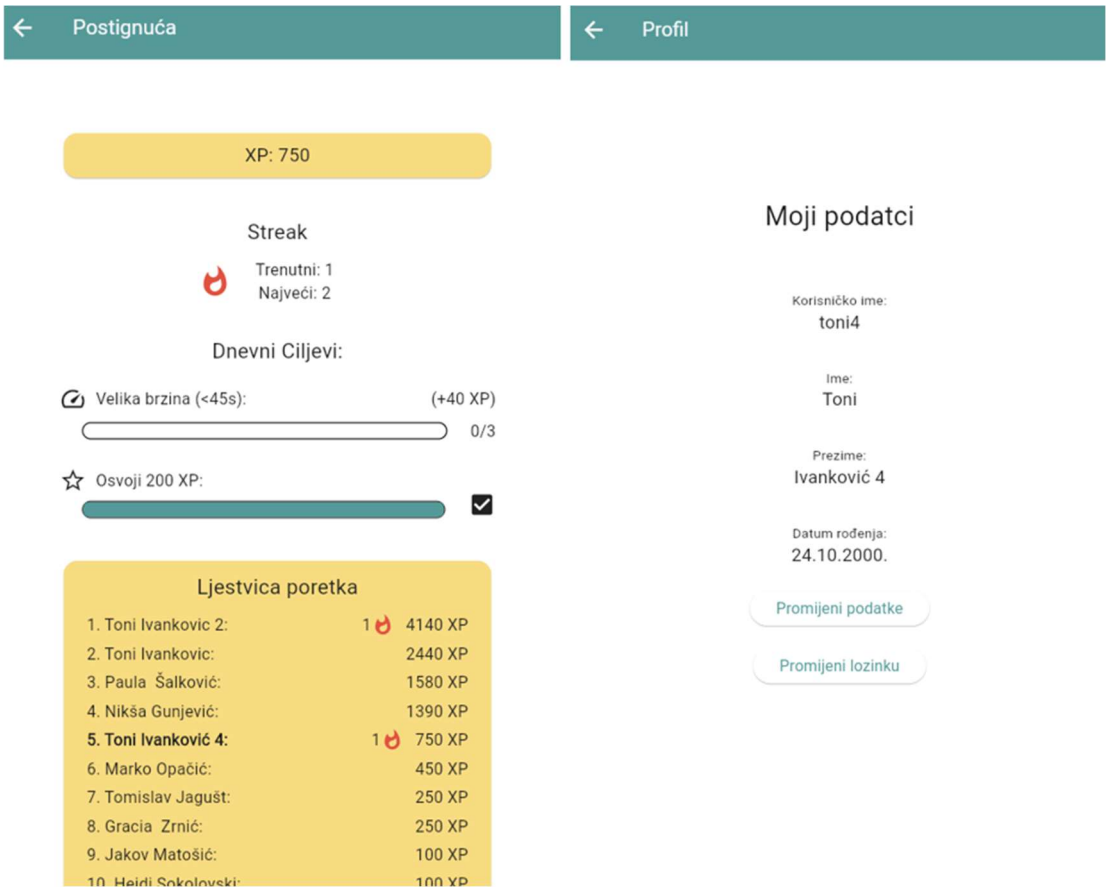
Slika 6.5 - Povratna informacija nakon riješene lekcije

Završetkom lekcije, učeniku se otključava sljedeća lekcija u cjelini, a završetkom zadnje lekcije u cjelini učeniku se otključava sljedeća cjelina i prva lekcija u njoj.

Osim dijela za učenje, učenik ima još dva ekrana koje može vidjeti – ekran s osobnim podatcima i ekran s postignućima.

Na ekranu s osobnim podatcima učenik vidi svoje osobne podatke (ime, prezime, datum rođenja) koje može urediti, a može promijeniti i lozinku.

Na ekranu s postignućima učenik može vidjeti svoj ukupni broj XP bodova, svoj *streak* (niz dana rješavanja), svoje dnevne izazove te ljestvicu poretka svih učenika u njegovom razredu ili u njegovoj školi. Na rang ljestvici učenici su rangirani po ukupnom broju osvojenih XP bodova, a trenutni korisnik podebljano je otisnut (eng. *bold*), radi lakšeg pronalaženja sebe na ljestvici. Na ljestvici se prikazuju samo imena i prezimena učenika, kako bi se smanjio nered tj. pretrpanost informacijama.



Slika 6.6 - Ekrani s postignućima i osobnim podatcima

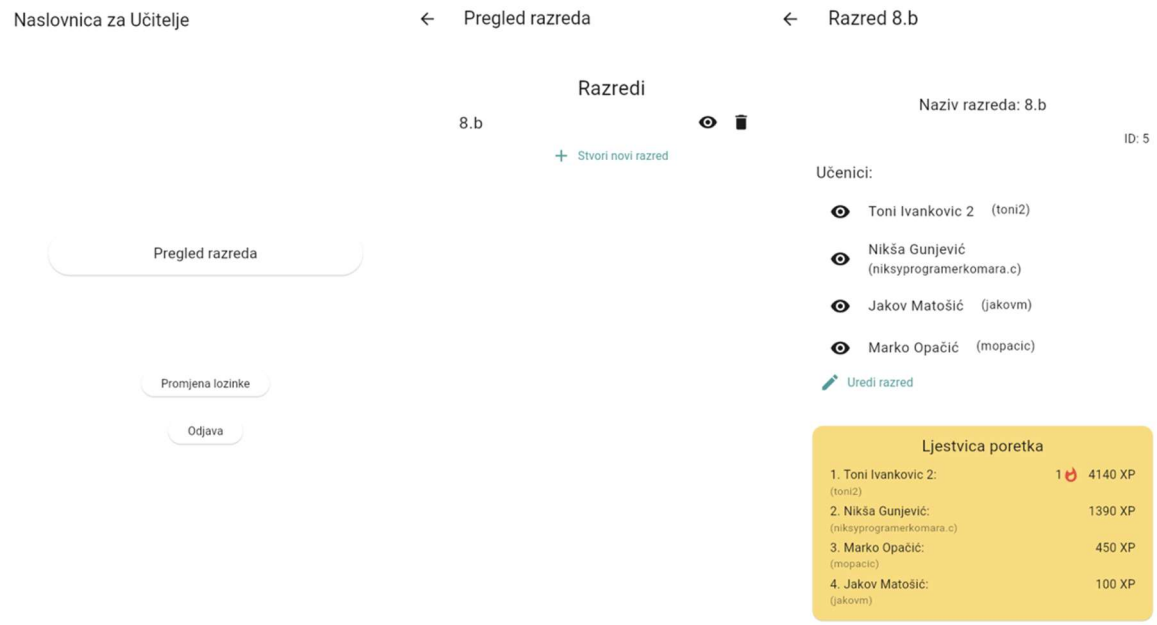
6.5. Učiteljsko sučelje

Prijavom u aplikaciju, učitelj nailazi na vrlo jednostavno sučelje. Ima opciju promjene lozinke, ili, primarnu funkciju, pregleda razreda.

U sučelju pregleda razreda, učitelj može stvoriti novi razred, ili pregledati (i potom urediti) ili obrisati neki od postojećih razreda.

U pregledu razreda učitelj vidi naziv razreda, popis svih učenika i ljestvicu poretka. Za razliku od ljestvice poretka koju vide učenici, učitelju se prikazuju i korisnička imena

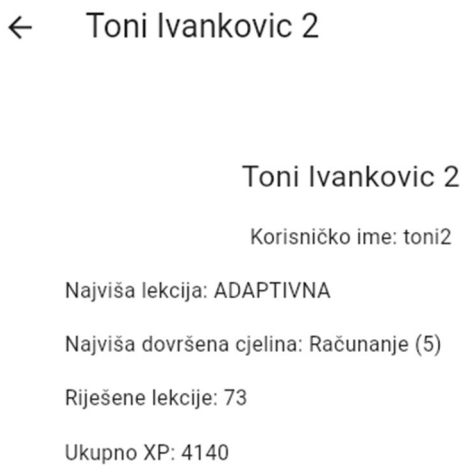
učenika, zbog potencijalnog slučaja jednakih imena i prezimena učenika. Razred je moguće urediti, a moguće je i pregledati svakog učenika u razredu.



Slika 6.7 - Pregled razreda

Uređivanjem razreda može se promijeniti ime razreda i učenici koji su u njemu.

U pregledu pojedinog učenika, dostupne su informacije: najviša lekcija koju je učenik dovršio, najviša cjelina koju je učenik dovršio, ukupan broj riješenih lekcija, te ukupan iznos osvojenih XP bodova.



Slika 6.8 - Pregled pojedinog učenika

I pri stvaranju razreda i pri uređivanju razreda, potrebno je definirati ime razreda te učenike koji su u razredu. Pri dodavanju učenika dobiva se popis svih učenika u toj školi (čak i onih

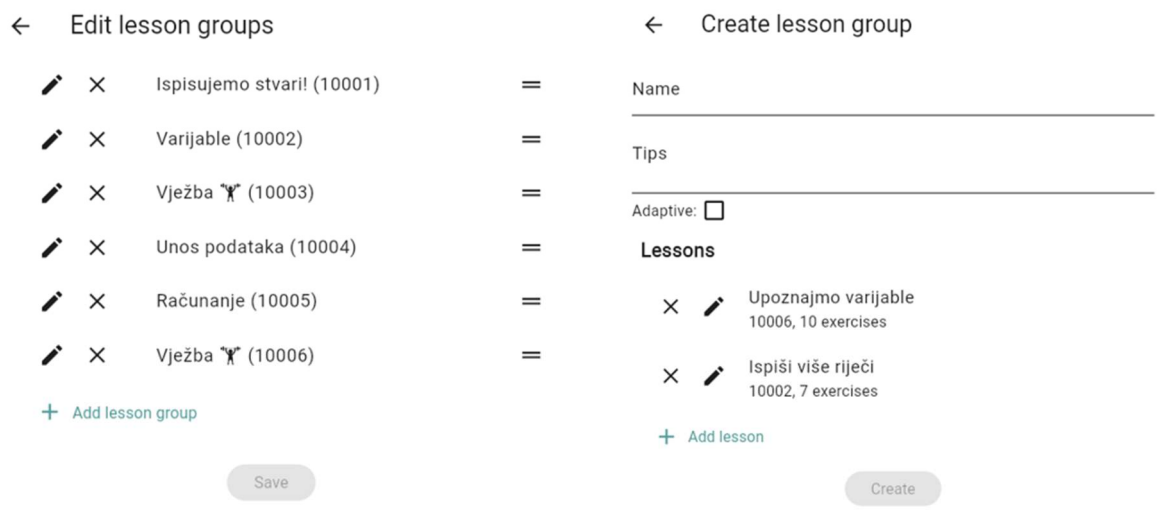
koji već jesu u nekom drugom razredu), a popis se može filtrirati upisom upita u odgovarajuće polje. Označavanjem učenika oni se dodaju u razred. Na popisu nisu prikazani učenici koji su već dodani u trenutni razred.

Slika 6.9 - Dodavanje učenika u razred

6.6. Sučelje za kreatora

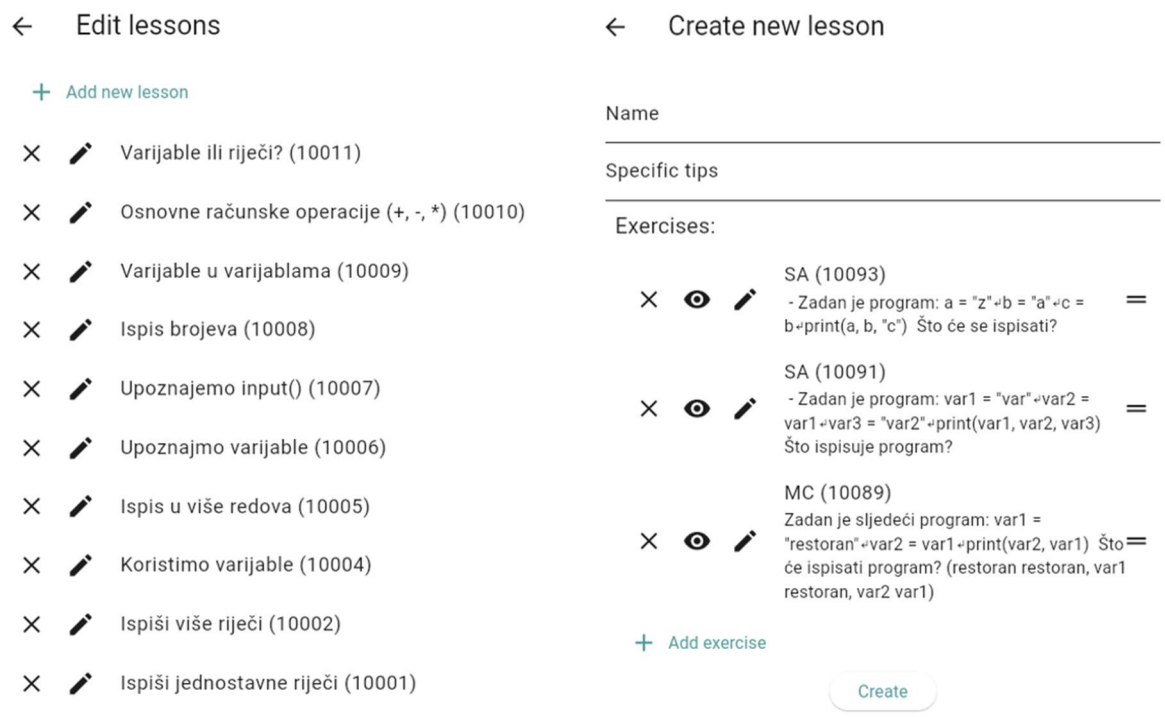
Prijavom u sustav, kreator može odabrati jednu od tri opcije – cjeline, lekcije ili vježbe.

U pregledu cjelina, kreator im može preurediti poredak, može dodati novu lekciju, izbrisati ili urediti neku od postojećih. Stvaranje nove cjeline zahtijeva definiranje imena, savjeta, radi li se o adaptivnoj cjelini te, ako se ne radi o adaptivnoj cjelini, popis lekcija u cjelini. Radi olakšanja cjelovitog stvaranja tečaja, odabrane lekcije mogu se uređivati nakon što su dodane u cjelinu, a moguće je i izravno stvoriti novu lekciju ako ne postoji odgovarajuća stvorena lekcija. Uređivanje cjeline ima gotovo jednako sučelje.



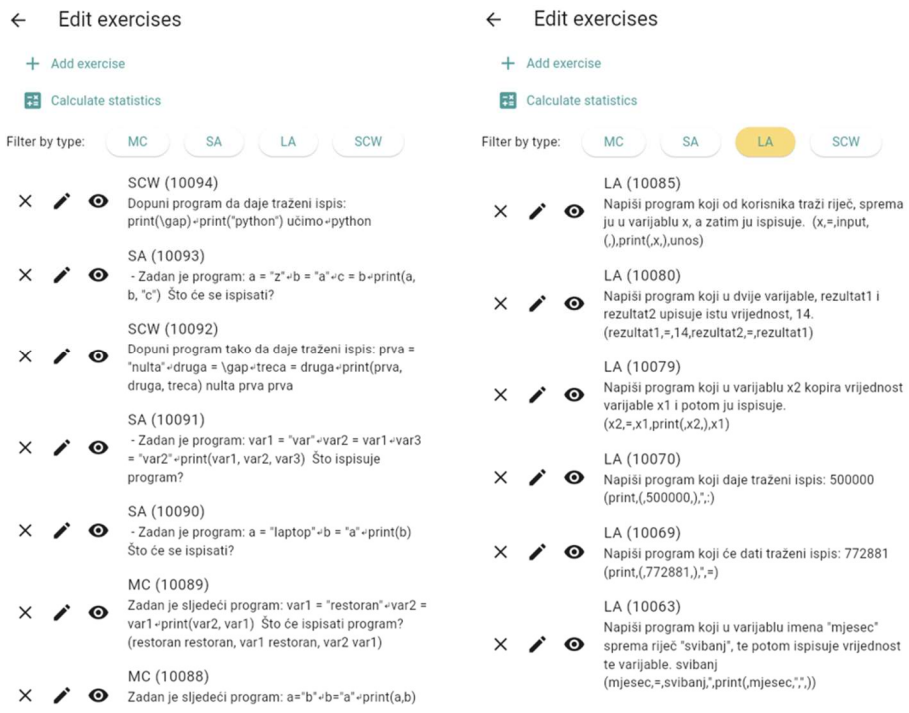
Slika 6.10 - Uređivanje i stvaranje cjelina

Odabirom opcije pregleda lekcija na početnom ekranu, kreator dobiva popis postojećih lekcija, od kojih svaku može urediti ili obrisati, te opciju stvaranja nove lekcije. Stvaranje lekcije zahtijeva definiranje njenog imena, opcionalnih savjeta vezanih uz lekciju te popisa vježbi koje lekcija sadrži. Odabranim vježbama može se preuređivati redoslijed. Ponovno, radi olakšanja cjelovitog stvaranja tečaja, pri odabiru vježbe moguće je i stvoriti novu vježbu, ako već ne postoji odgovarajuća vježba, a postojeće vježbe moguće je filtrirati po tipu (MC, SA, LA i SCW). Iako su u popisu vježbi navedene sve bitne informacije za vježbu, kreator u popisu vježbi uvijek ima opciju pregleda pojedine vježbe, čime dobiva pogled jednak učeničkom pri rješavanju te vježbe, a vježbu može i pokušati riješiti, kako bi osigurao da vježba prihvaća ili odbija neki odgovor. Uređivanje lekcije, ponovno, nudi gotovo jednako sučelje kao ono za stvaranje lekcije.



Slika 6.11 – Uređivanje i stvaranje lekcija

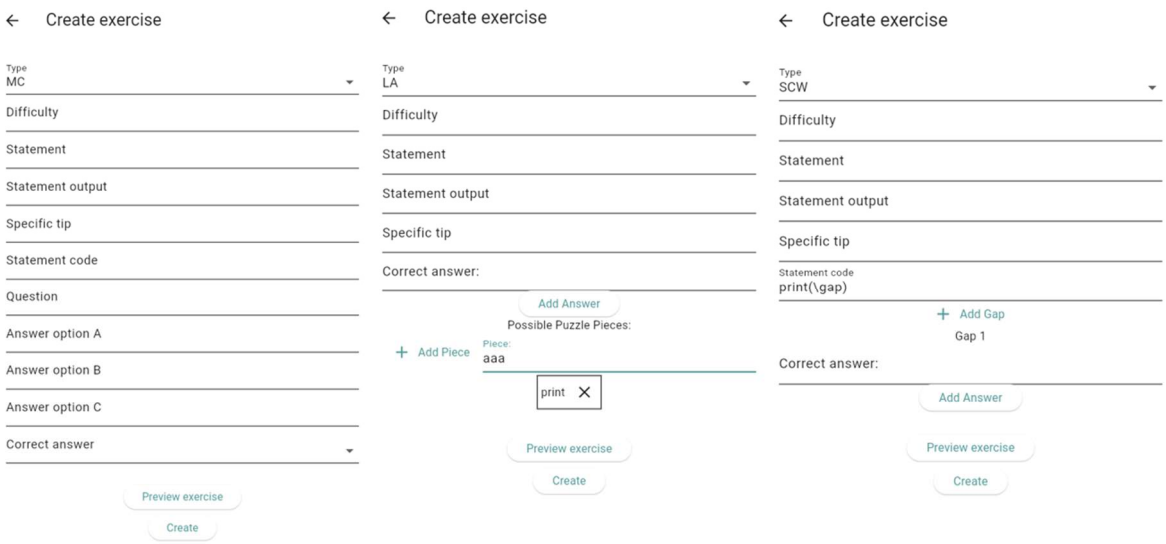
Odabirom opcije pregleda vježbi na početnom ekranu, kreator dobiva popis svih postojećih vježbi. Svaku od vježbi na popisu može izbrisati, urediti ili pregledati iz perspektive učenika. Vježbe na popisu može filtrirati po tipu, a može i dodati novu vježbu. Forma za stvaranje vježbe ovisi o tipu vježbe jer različiti tipovi vježbi imaju različite atribute. U svakom trenutku stvaranja vježbe (nakon unosa nužnih atributa), kreator može pregledati vježbu iz perspektive učenika.



Slika 6.12 - Pregled vježbi

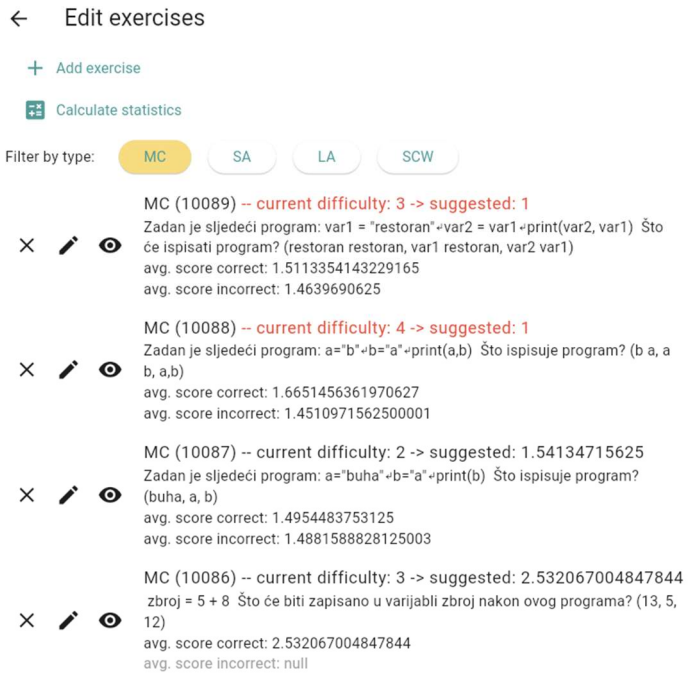
Specifični dio kreiranja MC vježbi su tri ponuđena odgovora te odabir točnoga. Za ostale tipove vježbe, moguće je definirati više točnih odgovora radi omogućavanja otpornosti na pogreške. Za LA vježbe moguće je definirati *dijelove slagalice*, tj. ponuđene dijelove kôda koje učenik može koristiti ako odabere način slagalice pri rješavanju vježbe.

Za SCW vježbe potrebno je pisati kôd koji će učenik nadopunjavati, gdje svaku prazninu označava s izrazom „\gap“, koji može ručno upisati, ili ga automatski dodati pritiskom na odgovarajući gumb. Potom, za svaku od praznina moguće je definirati točne ponuđene odgovore. Provjera odgovora vrši se ocjenjivanjem točnosti odgovora na svakoj od praznina zasebno. Iako bi postojale opcije gdje bi neki odgovori mogli biti točni samo u kombinaciji, ovaj način odabran je radi skalabilnosti – za n odgovora na jednu prazninu i m odgovora na drugu prazninu, broj opcija koje je potrebno unijeti na ovaj način je $O(n + m)$, dok bi za unos kombinacija trebalo $O(n * m)$ kombinacija. Na ovaj način potreban je unos manjeg broja opcija, ali se žrtvuje ekspresivnost, tj. neke vježbe nije moguće ostvariti ovim tipom pitanja.



Slika 6.13 - Forme za stvaranje vježbi

Pri pregledu vježbi također je moguće izračunati statistike za vježbe. S obzirom na to da je izračun statistike relativno skup, on se pokreće ručno umjesto da se dohvaća svaki put kad se zatraži pregled vježbi. Izračunom statistike uz svaku vježbu prikazuju se dodatni podatci – trenutna i predložena težina vježbe, prosječna razina učenika koji su odgovorili točno i prosječna razina učenika koji su odgovorili netočno. Ako se trenutna i predložena razina razlikuju za više ili jednako 1, promjena je dodatno istaknuta crvenom bojom, kako bi kreatoru skrenula pažnju da je vjerojatno potrebno korigirati tu vježbu.



Slika 6.14 - Izračun statistika za vježbe i predložene težine

6.7. Sučelje za administratora

Dio za administratora vrlo je jednostavan – administrator može stvoriti novog učitelja ili novog kreatora. Za oboje mora definirati korisničko ime i lozinku, a za učitelja još dodatno mora definirati i školu kojoj pripada te ime i prezime.

6.8. Postavljanje aplikacije

Razvijena aplikacija objavljena je u web obliku. Kao i za poslužiteljsku aplikaciju, uspostavljen je *CI/CD pipeline* (cjevovod) s pomoću *GitHub Actions Workflowa*. Umjesto na Azure, klijentska web aplikacija objavljena je na AWS. Kompajlirana aplikacija sprema se na servisu *AWS S3*, a pristup njoj omogućen je preko servisa *AWS CloudFront*. CloudFront omogućuje globalnu distribuciju sadržaja, dostavu statičkog i dinamičkog sadržaja, privremeno pohranjivanje (eng. *caching*) te zaštitu od napada.

7. Struktura tečaja

Tečaj se trenutno sastoji od četiri cjeline. One su navedene u nastavku, zajedno s kratkim opisom:

1. ***Ispisujemo stvari!*** – funkcija *print*, *stringovi* u kontekstu *printa*, brojevi u kontekstu *printa*, ispis u više redova, ispis više argumenata
2. ***Varijable*** – spremanje vrijednosti u varijable, korištenje varijabli za ispis, kopiranje vrijednosti iz jedne varijable u drugu, razlika varijabli i *stringova*
3. ***Unos podataka*** – funkcija *input*
4. ***Računanje*** – osnovne aritmetičke operacije osim dijeljenja (zbrajanje, oduzimanje, množenje), cjelobrojno dijeljenje, decimalno dijeljenje, računanje s pomoću varijabli

Prostor za daljnji razvoj je stvaranje daljnjih lekcija. Zamišljen je popis cjelina koje bi bile dio tog tečaja:

5. Modulo operator
6. Problemski zadatci (primjena naučenog)
7. Uvjetno grananje – *if* naredba
8. Uvjetno grananje – *else* naredba
9. Uvjetno grananje – *elif* naredba
10. Problemski zadatci
11. Petlja *while*
12. Petlja *for*
13. Problemski zadatci
14. Liste – eksplicitno stvaranje, iteracija indeksom, dinamičko dodavanje i brisanje elemenata, iteracija po elementima (*for each*)
15. *Stringovi* – konkatenacija, pristup po indeksu, usporedba *stringova*, iteracija po elementima (*for each*)
16. Problemski zadatci
17. 2D liste
18. Rječnici (*dictionary*)
19. Funkcije
20. Rekurzija

8. Ispitivanje funkcionalnosti

Ispitivanje funkcionalnosti aplikacije na stvarnim korisnicima provedeno je u dvije faze – alfa testiranje radi prilagodbe intuitivnosti aplikacije te beta testiranje s učenicima radi ispitivanja efektivnosti igrifikacije.

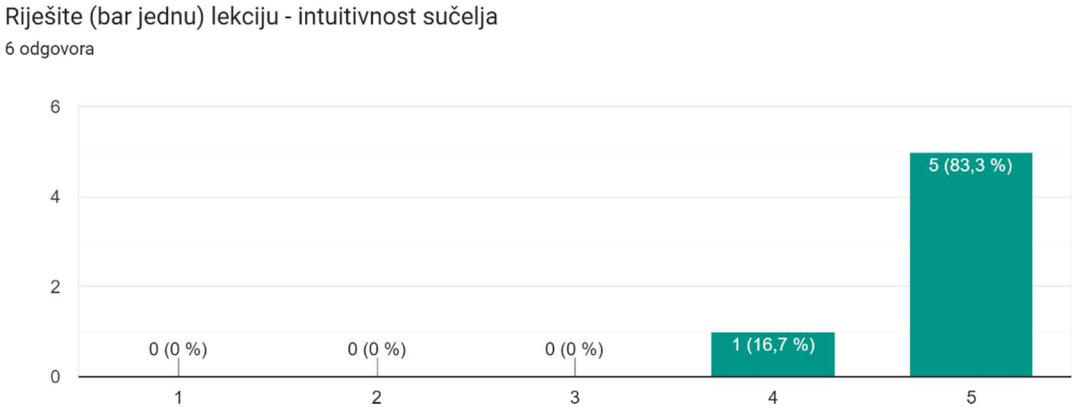
8.1. Alfa testiranje

Alfa testiranje aplikacije provedeno je s korisnicima u dobi 22-25 godina, nakon što su u aplikaciju dodana svi zamišljeni aspekti.. Dio testera ima znatno iskustvo u računarstvu, a dio je nepovezan s područjem računarstva. Alfa testeri imali su zadatak osvrnuti se na intuitivnost aplikacije, općenitu ispravnost funkcionalnosti te osobni dojam. Većina skala za ocjenjivanje bila su 1-5, osim ako je drukčije naznačeno.

Vezano za proces prijave, intuitivnost je na skali 1-5 najviše ocijenjena ocjenama 5 i 4. Prijedlozi vezani za proces prijave bili su omogućiti automatsko popunjavanje podataka za prijavu i spremanje podataka pri registraciji, jasnije istaknut gumb za prebacivanje između prijave i registracije te bolje oblikovanu poruku o pogrešci pri unosu pogrešnog datuma rođenja. Svi prijedlozi potom su implementirani u aplikaciji.

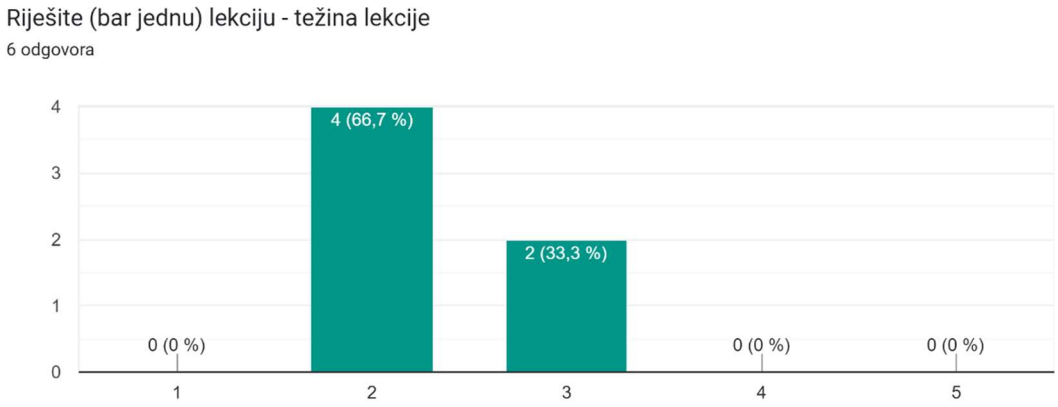
Intuitivnost početka učenja svi su testeri označili ocjenom 5. Povratna informacija bila je jasnije naznačiti prostor na koji je moguće kliknuti (inicijalno je to bio samo trokutić oblika *play*) te da je potrebno objasniti što radi ikonica žaruljice prije ulaska u lekciju u cjelini (dodano je objašnjenje „Nauči“).

Intuitivnost sučelja za učenje 83 % testera ocijenilo je s 5, a 17 % s 4.



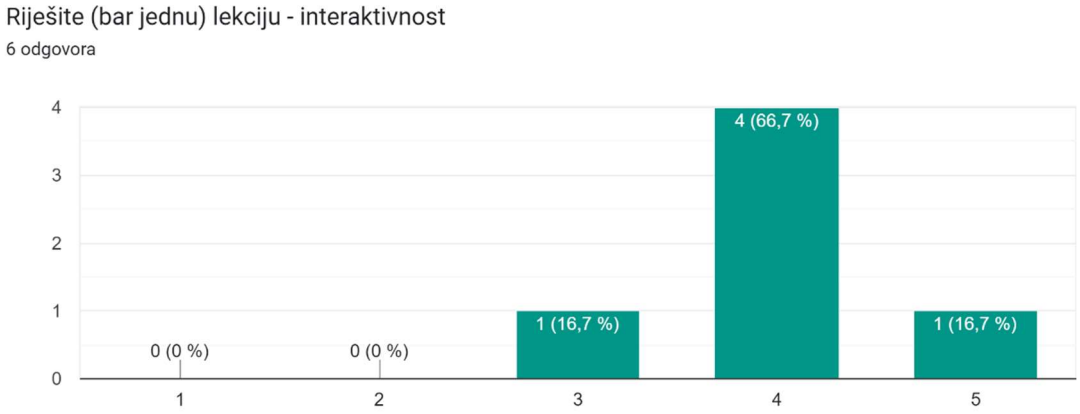
Slika 8.1 – procjene intuitivnosti sučelja za učenje

Težinu prve lekcije 67 % ocijenilo je s 2, a 33 % s 3, gdje 1 označava prelagano, a 5 preteško. Kao odgovor na ovu povratnu informaciju, u početne je lekcije ubačeno nešto više *trik pitanja*.



Slika 8.2 – procjena težine prve lekcije

Interaktivnost u procesu učenja 17 % ocijenilo je s 3 i s 5, a 66 % ocijenilo je s 4.



Slika 8.3 – procjena interaktivnosti u procesu učenja

Prijedlozi vezani za sadržaj i sučelje u lekcijama bili su da je točkica koja označava razmak bila zbunjujuća. Kao odgovor na ove povratne informacije, u početne je lekcije ubačeno više *trik pitanja*, primjerice hoće li se ispisati *string* s navodnicima ili bez navodnika te je u cjelinu o varijablama ubačena lekcija s *trik pitanjima* u kojima treba prepoznati radi li se o varijablama ili o *stringovima*. Također, u više je vježbi dodana pomoć koja se prikazuje ispod vježbe, a koja se odnosi na objašnjenje značenja znaka · kao oznaku za razmak.

Jasnoća povratne informacije na kraju lekcije ocijenjena je samo opisno, ali svi su komentari bili pozitivni.

Identificirani su elementi aplikacije koji nisu radili kako je bilo zamišljeno:

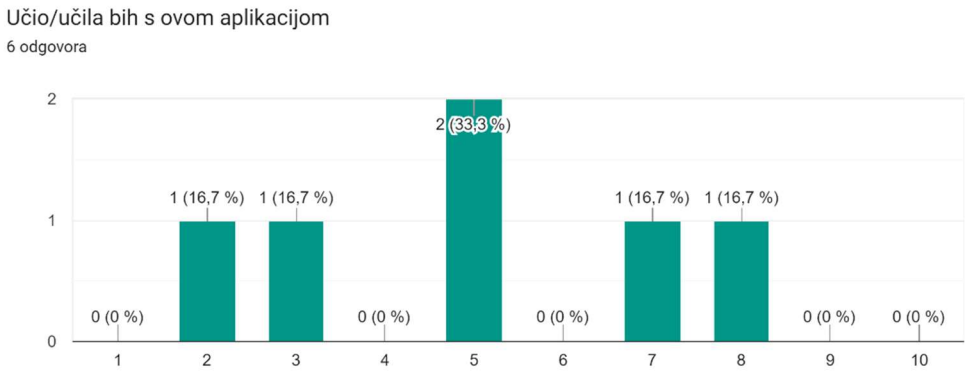
- U vježbama kratkog odgovora, web verzija aplikacije na mobilnim uređajima ponašala bi se neuobičajeno pri unosu teksta u prostor za odgovor, dok se isti problem nije pojavljivao na računalima i na lokalnoj Windows verziji aplikacije. Istraživanje je dalo naslutiti da se radi o internoj pogrešci radnog okvira *Flutter*. Problem je zaobiđen ograničavanjem kratkog odgovora na jedan redak, a vježbe koje su očekivale odgovor u više redaka preoblikovane su.
- Boje u tamnom načinu nisu bile dovoljno usklađene, pa je obraćeno više pažnje na slaganje palete boja i usklađena je u cijeloj aplikaciji.
- Specifična pomoć na pojedinoj vježbi bila je napisana crvenom bojom, što je neke testere zbunilo jer su mislili da se radi o poruci o pogrešci, a ne o pomoći. Boja specifične pomoći promijenjena je u primarnu boju aplikacije, tirkiznu.

- Povratna informacija o pogrešnom odgovoru testerima se činila neskladnom, pa je dizajn promijenjen dodavanjem ikonica, poravnavanjem, razmacima među dijelovima komponente i podebljanjem dijelova teksta.

Radi identifikacije intuitivnosti pronalaska pojedinih komponenata, testeri su trebali označiti na koje su aspekte aplikacije naišli. Kako bi se ispitalo na koje aspekte su naišli normalnim korištenjem aplikacije, bilo je potrebno po sjećanju označiti aspekte koje su susreli, bez povratka u aplikaciju kako bi ih pronašli:

- Svi su testeri susreli objašnjenje gradiva cjeline (pomoć), vježbu pisanja kôda u načinu slagalice i XP bodove
- Većina testera (> 66 %) sjeća se da je susrelo ostale tipove vježbi (s tim da je manje testera susrelo vježbu pisanja kôda u načinu slagalice u usporedbi s drugim tipovima vježbi), streak, povratnu informaciju o točnosti rješavanja te mogućnost izmjene osobnih podataka.
- Oko polovice testera sjeća se da je susrelo i dnevne ciljeve, informacije o svojem profilu te mogućnost izmjene lozinke.

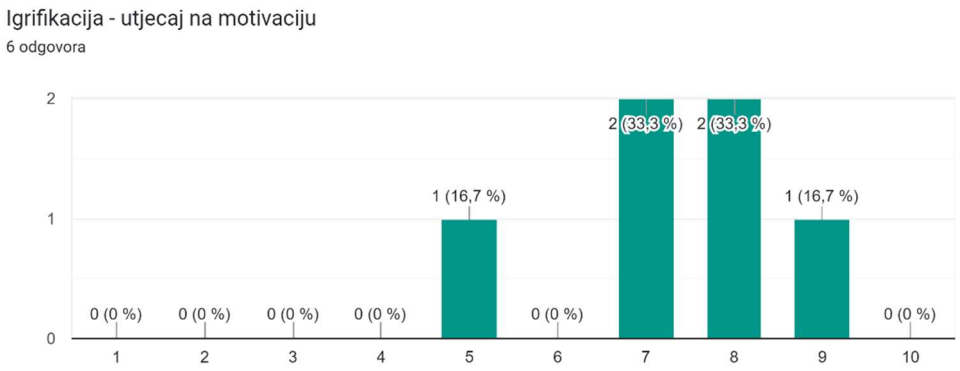
Na pitanje da na skali 1-10 procijene koliko vjerojatno bi učili programiranje s ovom aplikacijom u usporedbi s drugim opcijama poput YouTube *tutoriala*, tečaja, ili drugih online platformi, u situaciji kada bi htjeli naučiti programirati, 33 % dalo je ocjenu 2-3, 33 % ocjenu 5, a 33 % ocjenu 7-8.



Slika 8.4 – procjena vjerojatnosti učenja s pomoću aplikacije

Kada su upitani kako (subjektivno) procjenjuju da je igrifikacija (XP bodovi, ljestvica poretka, dnevni izazovi, *streak*, slagalica kôda) utjecala na njihovu motivaciju na skali 1-10,

gdje ocjena 1 ima značenje izrazitog smanjenja motivacije, a 10 značenje izrazitog povećanja motivacije, čak 83 % smjestilo je utjecaj između 7 i 9. Ovo je pitanje među glavnim pitanjima koje je bilo od interesa za objektivno ispitivanje u sljedećoj fazi.



Slika 8.5 – procjena utjecaja igrifikacije na motivaciju

Također, identificirano je da je ocjenjivanje odgovora preblago – bilo je neosjetljivo na mala/velika slova. S obzirom na to da je u kôdu ta strogoća lako podesiva, neosjetljivost na velika/mala slova je isključena, pa učenik mora ponuditi odgovor čija velika/mala slova odgovaraju traženom odgovoru. Popustljivost u ocjenjivanju vezana za bjeline na početku i na kraju odgovora je zadržana.

8.2. Testiranje s učenicima – učinkovitost igrifikacije

Testiranje s učenicima petog razreda provedeno je u Osnovnoj školi Tina Ujevića u Zagrebu. Učenici petog razreda nisu se susreli s programiranjem u Pythonu jer je ono u kurikulumu informatike smješteno tek u šesti razred. Na taj način, uzeti su učenici bez predznanja, koji bi s ovom aplikacijom mogli naučiti najviše. Učenici viših razreda mogli bi također imati koristi od ove aplikacije, ali bi im vrlo vjerojatno dobro došla funkcionalnost inicijalnog testiranja i pozicioniranja na prvu cjelinu koju dosad nisu usvojili.

Jedna od ideja bila je ispitati učinkovitost aplikacije za svrhu učenja programiranja i programskog razmišljanja. Ipak, za bilo kakvo statistički značajno ispitivanje o usvajanju znanja, učenici bi aplikaciju trebali koristiti duže vrijeme i dovoljno često kako bi iz nje zapravo naučili dovoljno da bi postojala mjerljiva razlika u usporedbi sa znanjem prije

korištenja aplikacije. S obzirom na vremenske okvire rada, ova je opcija izbačena i ostaje kao prostor za daljnje ispitivanje.

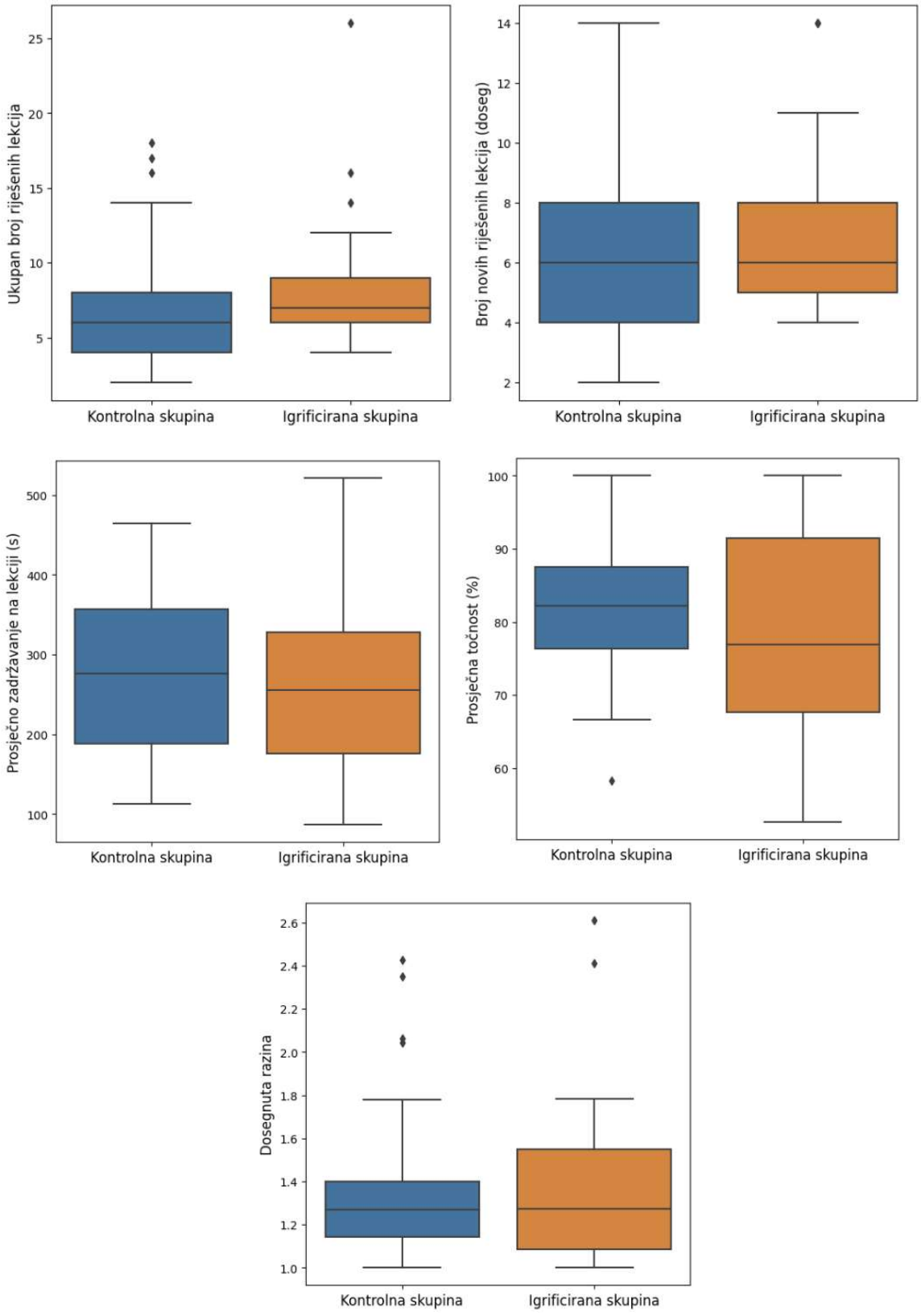
Ideja koja je ispitana bila je upravo povezana s igrifikacijom. Ideja je igrifikacije, kao što je opisano u poglavlju 1.3, da učenici nemaju dojam da uče, već da se igraju.

Učenici su podijeljeni u dvije skupine – *igrificirana skupina*, tj. oni koji su dobili potpunu aplikaciju i kontrolna skupina, tj. oni koji su dobili aplikaciju bez većine svojstava igrifikacije (ljestvice poretka, dnevnih ciljeva, XP bodova, *streaka*). Podjela u skupine potpuno je nasumična, a događa se naizmjenično po redoslijedu registracije.

Nulta hipoteza je da elementi igrifikacije ne utječu na motivaciju učenika. Alternativna hipoteza je da igrifikacija povećava motivaciju učenika. S obzirom na to da motivacija sama po sebi nije mjerljiva veličina, ispitivani su ukupan broj riješenih lekcija, broj novih riješenih lekcija (*doseg* na tečaju), vrijeme provedeno u lekcijama, dosegnuta razina i prosječna točnost rješavanja. Kako ne bi bilo utjecaja na vrijeme provedeno u lekcijama, nijedno igrifikacijsko svojstvo koje je prisutno tijekom rješavanja lekcije nije mijenjano (npr. slagalica kôda ili duljina lekcije).

Ispitivanje je provedeno s N=48 učenika, od kojih je N1=25 bilo u igrificiranoj skupini, a N2=23 bilo u kontrolnoj skupini. Broj učenika po skupini nije jednak jer su iz istraživanja isključeni učenici koji nisu niti pokrenuli aplikaciju (a nasumična skupina im je već dodijeljena unaprijed). Učenici su tijekom jednog tjedna na nastavi informatike i van nastave bili potaknuti rješavati aplikaciju.

S obzirom na veličinu uzorka, ne može se koristiti pretpostavka normalnosti, što je vidljivo i iz sljedećih *box-plotova* – distribucija podataka vrlo je često teško nagnuta u jednu stranu, dok su negdje podatci distribuirani gotovo uniformno.



Slika 8.6 - Box-plotovi izmjerenih rezultata

S obzirom na odsutnost normalnosti podataka, u statističkoj analizi korišteni su medijani umjesto srednjih vrijednosti. Konkretne numeričke vrijednosti prikazane su u tablici, kao i P vrijednosti dobivene koristeći *Mann-Whitney* U-test (tj. *Wilcoxon rank-sum* test).

Tablica 8.1 - Statistička analiza izmjerenih rezultata

Veličina	Kontrolna skupina	Igrificirana skupina	Alternativa	P-vrijednost
Ukupan broj riješenih lekcija	6	7	$N_{kont.} < N_{igrif.}$	0,064
Broj novih riješenih lekcija (doseg na tečaju)	6	6	$N_{kont.} < N_{igrif.}$	0,134
Medijan vremena provedenog u lekciji (s)	276,27	255,31	$T_{kont.} > T_{igrif.}$	0,241
Medijan točnosti	82,21 %	76,92 %	$Acc_{kont.} > Acc_{igrif.}$	0,163
Dosegnuta razina	1,32	1,45	$R_{kont.} < R_{igrif.}$	0,179

Dio rezultata sukladan je očekivanjima, dok je dio nešto drukčiji. I ukupan broj riješenih lekcija i broj jedinstvenih riješenih lekcija (doseg na tečaju) veći su kod igrificirane skupine nego kod kontrolne skupine s relativno malenom P-vrijednosti. Učenici s igrifikacijom dosegli su višu razinu od kontrolne skupine (doduše s P-vrijednosti 0,179). Iznenađujuće, učenici s igrifikacijom imali su manju točnost rješavanja. Učenici s igrifikacijom lekcije su rješavali brže, ali ovaj zaključak ima najmanju statističku značajnost, s P-vrijednosti od čak 0,241.

Moguć razlog bržeg rješavanja lekcija kod učenika s igrifikacijom je što su neki učenici dobili dnevni cilj brzog rješavanja lekcija, ali i što brže žele skupiti XP bodove. S obzirom na to, *brzopletost* bi mogla biti objašnjenje manje točnosti rješavanja u igrificiranoj skupini.

Nije jednostavno utvrditi razlog relativno velikim P-vrijednostima, ali najvjerojatniji uzročnici su relativno malen uzorak i relativno maleno vrijeme korištenja. Uzorak od 48

učenika mogao bi i biti dovoljan kada bi podatci bili normalno distribuirani, ali zbog odstupanja od normalnosti bilo je potrebno koristiti neparametarske metode, koje imaju slabiju statističku snagu i zahtijevaju veći uzorak.

Osim toga, većina aspekata igrifikacije implementiranih u aplikaciji i u ideji postižu najveću snagu kada se koriste kontinuirano – ljestvica poretka, XP bodovi, dnevni izazovi. Niz dana koje učenik rješava vježbe (*streak*) nema gotovo nikakav značaj ako se aplikacija koristi jednokratno. Stoga, očito je i sama postavka eksperimenta ograničila djelotvornost igrifikacije.

Također, treba uzeti u obzir i da je većina svojstava igrifikacije namijenjena natjecanju (XP bodovi, skupljanje bodova dnevnim izazovima, ljestvica poretka). Različiti učenici imaju različite preferencije pa učenici koji, primjerice, preferiraju istraživanje umjesto kompetitivnosti neće ostvariti maksimalnu korist od trenutno implementiranih svojstava.

Za daljnje istraživanje bilo bi potrebno eksperiment provesti na većem uzorku učenika, kroz duži period te učenike potaknuti da rješavaju aplikaciju i izvan nastave. Osim toga, implementacija više igrifikacijskih svojstava mogla bi dati značajnije razlike u rezultatima.

Zaključak

U sklopu ovog rada razvijen je višeplatformski sustav za učenje programiranja u programskom jeziku Python. Odabrana arhitektura je *klijent-poslužitelj*, gdje je klijentska aplikacija implementirana u radnom okviru *Flutter*, a poslužiteljska aplikacija u radnom okviru *.NET*. Za pohranu je odabrana nerelacijska baza podataka *MongoDB*.

Razvijena aplikacija korisnicima omogućuje rješavanje pojedinih vježbi, grupiranih u kratke lekcije, koje su dalje grupirane u smislene cjeline. Implementirana su četiri tipa vježbi – vježbe višestrukog izbora, vježbe kratkog odgovora, vježbe dugog odgovora (pisanja kôda) te vježbe dopunjavanja kôda.

Radi povećanja motivacije učenika i poboljšanja iskustva učenja, u aplikaciju su dodani elementi igrifikacije te elementi adaptivnosti. Implementirani elementi igrifikacije su XP bodovi, dnevni ciljevi, ljestvica poretka, niz dana korištenja (*streak*), kratke lekcije te slagalica kôda. Implementirani elementi adaptivnosti su povratna informacija nakon vježbi/lekcija, ponavljanje pogrešno odgovorenih vježbi, prilagodba razine učenika, adaptivne lekcije te prilagodba težine zadataka.

Implementiran je početni dio tečaja, koji je namijenjen učenju ispisa podataka, rada s varijablama, unosa podataka te aritmetičkih operacija. Predstavljena je i ideja nastavka tečaja, obrađujući razna područja uvoda u programiranje, poput grananja, petlji, listi, znakovnih nizova i funkcija.

Aplikaciju su testirali studenti radi identifikacije mogućih poboljšanja u dizajnu i intuitivnosti korisničkog sučelja i korisničkog iskustva. Nakon njihove identifikacije, većina je poboljšanja implementirana u aplikaciji, rezultirajući pozitivnim povratnim informacijama.

Istraživanje provedeno na uzorku od 48 učenika pokazalo je rezultate koji navode na učinkovitost igrifikacije u svrhu motivacije učenika. Ipak, rezultati nisu konkluzivni jer imaju relativno velike P-vrijednosti, najvjerojatnije zbog nedovoljno velikog uzorka i postavke eksperimenta u kojoj igrifikacija nije mogla ostvariti puni učinak. Daljnje istraživanje s dužim trajanjem bilo bi potrebno za sigurniji zaključak o učinkovitosti implementiranih elemenata igrifikacije.

Literatura

[1] E. E. a. C. E. Agency, »Informatics education at school in Europe,« [Mrežno]. Available: https://www.minedu.gov.gr/publications/docs2022/Informatics_education_Report.pdf. [Pokušaj pristupa 12 05 2024].

[2] E. L. Deci i R. M. Ryan, »The “What” and “Why” of Goal Pursuits: Human Needs and the Self-Determination of Behavior,« *Psychological Inquiry,* svez. 11, br. 4, p. 42, 2000.

[3] N. D. Saidin, F. Khalid, R. Martin, Y. Kuppusamy i N. A. P. Munusamy, »Benefits and Challenges of Applying Computational,« International Journal of Information and Education Technology, 2021.

[4] »personalized-learning-vs-adaptive-learning,« 2021. [Mrežno]. Available: https://elmlearning.com/blog/personalized-learning-vs-adaptive-learning/. [Pokušaj pristupa 13 05 2024].

[5] »adaptive-learning-what-is-it-what-are-its-benefits,« 2021. [Mrežno]. Available: https://educationaltechnology.net/adaptive-learning-what-is-it-what-are-its-benefits-and-how-does-it-work/. [Pokušaj pristupa 13 05 2024].

[6] S. Deterding, D. Dixon, R. Khaled i L. Nacke, »From Game Design Elements to Gamefulness: Defining "Gamification",« 2011.

[7] J. Hamari, J. Koivisto i H. Sarsa, »Does Gamification Work? — A Literature Review of Empirical Studies on Gamification,« 2014.

[8] Y. Xiao i K. F. Hew, »Personalized gamification versus one-size-fits-all gamification in fully online learning: Effects on student motivational, behavioral and cognitive outcomes,« *Learning and Individual Differences,* 2024.

[9] B. Reeves i J. Leighton Read, »Ten Ingredients of Great Games«.

[10] »Best Language Learning Apps for 2024,« 2024. [Mrežno]. Available: https://www.cnet.com/tech/services-and-software/best-language-learning-apps/. [Pokušaj pristupa 14 05 2024].

[11] »Duolingo Really Works - Methods,« [Mrežno]. Available: https://www.duolingo.com/efficacy/method. [Pokušaj pristupa 14 05 2024].

[12] »Hour of Code,« [Mrežno]. Available: https://hourofcode.com/hr. [Pokušaj pristupa 15 05 2024].

[13] »Codecademy,« [Mrežno]. Available: https://www.codecademy.com/learn. [Pokušaj pristupa 15 05 2024].

[14] »Codecademy - Wikipedia,« [Mrežno]. Available: https://en.wikipedia.org/wiki/Codecademy. [Pokušaj pristupa 15 05 2024].

[15] »Codecademy Revolutionizing Online Education in Coding,« [Mrežno]. Available: https://medium.com/@shaluisha29/codecademy-revolutionizing-online-education-in-coding-7c4fea718def. [Pokušaj pristupa 15 05 2024].

[16] »Lightbot - Wikipedia,« [Mrežno]. Available: https://en.wikipedia.org/wiki/Lightbot. [Pokušaj pristupa 15 05 2024].

[17] »Lightbot,« [Mrežno]. Available: https://lightbot.com/. [Pokušaj pristupa 15 05 2024].

[18] [Mrežno]. Available: https://res.cloudinary.com/lwgatsby/f_auto/www/uploads/2023/05/client-server-network.jpg. [Pokušaj pristupa 08 06 2024].

[19] P. Pale, J. Petrović i B. Jeren, »Learning Theories - Problem-Based Learning,« FER, [Mrežno]. Available: https://www.learning-theories.org/doku.php?id=instructional_design:problem-based_learning. [Pokušaj pristupa 11 06 2024].

[20] J. Sweller, »Cognitive load theory,« *Psychology of Learning and Motivation,* 2011.

[21] ».NET Framework - Wikipedia,« [Mrežno]. Available: https://en.wikipedia.org/wiki/.NET_Framework. [Pokušaj pristupa 09 06 2024].

[22] T. Collings, »Controller-Service-Repository architecture,« 10 08 2021. [Mrežno]. Available: https://tom-collings.medium.com/controller-service-repository-16e29a4684e5. [Pokušaj pristupa 03 06 2024].

[23] »Flutter,« [Mrežno]. Available: https://flutter.dev/. [Pokušaj pristupa 09 06 2024].

[24] »Flutter architectural overview,« [Mrežno]. Available: https://docs.flutter.dev/resources/architectural-overview#composition. [Pokušaj pristupa 09 06 2024].

Sažetak

Integracija adaptivnosti i igrifikacije u sustav za učenje programiranja

Povećanjem potrebe za digitalnim vještinama u svakodnevnom životu te istovremenim smanjenjem koncentracije učenika, razvoj interaktivnih sustava za podučavanje digitalnih kompetencija koji se prilagođavaju učeniku postaje neophodan. U sklopu ovog diplomskog rada razvijen je sustav za učenje programiranja u jeziku Python. Klijentska aplikacija razvijena je u radnom okviru *Flutter*, poslužiteljska u radnom okviru *.NET*, a za bazu podataka odabran je *MongoDB*. Kako bi se povećala motivacija učenika, u aplikaciji su implementirani elementi igrifikacije poput XP bodova, dnevnih ciljeva i ljestvice poretka. Radi povećanja kvalitete učenja, implementirani su i elementi adaptivnog učenja poput prilagodljivih lekcija, algoritma za prilagodbu razine učenika i algoritma za prilagodbu težine vježbi. Sustav je testiran radi otklanjanja pogrešaka i povećanja intuitivnosti korisničkog sučelja. Sustav je iskorišten i za ispitivanje učinkovitosti igrifikacije. Dobiveni rezultati pokazuju naznake učinkovitosti korištenja igrifikacije radi povećanja motivacije učenika, ali potrebno je dodatno istraživanje, s većim uzorkom korisnika.

Ključne riječi: edukacija, digitalne kompetencije, igrifikacija, adaptivnost, Flutter, .NET, MongoDB

Summary

Integration of Adaptivity and Gamification in a Programming Learning System

With the increasing need for digital skills in day-to-day life, and the simultaneous decline in students' concentration, developing interactive systems for teaching digital competencies that adapt to the student becomes essential. As part of this master thesis, a system for learning programming in Python language was developed. The client application was developed using the *Flutter* framework, the server-side app using the *.NET* framework, and *MongoDB* was chosen for the database. To enhance student motivation, gamification elements, such as XP points, daily quests, and a leaderboard were implemented in the app. To improve the quality of learning, adaptive learning elements were also implemented, such as adaptive lessons, a student score adjustment algorithm, and an exercise difficulty adjustment algorithm. The system was tested to eliminate bugs and to increase the intuitiveness of the user interface. Additionally, the system was used to evaluate the effectiveness of gamification. The results indicate that gamification may be effective in increasing student motivation, but further research with larger sample size is needed.

Keywords: education, digital competencies, gamification, adaptivity, Flutter, .NET, MongoDB