Usvajanje gramatičkih struktura engleskoga jezika uz upotrebu mobilne aplikacije

Župarić, Adrian

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

Sveučilišni studij

USVAJANJE GRAMATIČKIH STRUKTURA ENGLESKOGA JEZIKA UZ UPOTREBU MOBILNE APLIKACIJE

Završni rad

Adrian Župarić

Osijek, 2022.

SADRŽAJ

1. UVOD

Sve je veći trend korištenja modernih tehnologija i mnoga su sveučilišta pokrenula programe za proširenje uporabe (elektroničkog) e-učenja za promicanje aktivnog i neovisnog učenja, razvijanje odgovornosti studenata za njihov studij, povećanje samodiscipline i samomotivacije [1]. Mobilno se učenje pokazalo korisnim, učinkovitim, interaktivnim i praktičnim [2]. Pametni telefoni korisnicima pružaju funkcionalnost sličnu računalu. Uspostavili su veliku nadmoć nad standardnim mobilnim telefonima jer vlasnici imaju mogućnost instaliranja raznih aplikacija. Mobilne aplikacije omogućuju korisnicima jednostavan i funkcionalan pristup informacijama, proizvodima, uslugama i procesima koji su im potrebni u stvarnom vremenu i optimizirane su za praktičnu interakciju. Engleski jezik govori oko 1,5 milijardi ljudi diljem svijeta [3]. Sposobnost razumijevanja engleskog jezika omogućuje komunikaciju u brojnim zemljama. Nove tehnologije i globalizacija učinili su da stalno budemo okruženi engleskim pojmovima. Radi velikog broja govornika engleskog jezika, javila se velika potreba samoučenja jezika uporabom tehnologije i aplikacija.

1.1. Zadatak završnoga rada

Cilj je ovog završnog rada izraditi iOS aplikaciju za usvajanje gramatičkih struktura engleskoga jezika. Aplikacija će biti napisana u programskom jeziku Swift, koji je korišten za funkcionalnost cijele aplikacije, a za bazu će se podataka koristiti Firebase. CocoaPods je korišten za integriranje Firebase-a, Xcode je korišten za IDE, a za arhitekturu koda MVC. U završnom radu Firebase je korišten za autentifikaciju korisnika te kao baza podataka za pohranjivanje cjelina i praćenje korisnikovog napretka.

Korisnici će imati mogućnost registriranja i prijave/odjave. Registriranje i prijava u aplikaciju služi kako bi korisnik mogao nastaviti učiti tamo gdje je stao. Aplikacija će se sastojati od različitih cjelina gramatičkih struktura engleskoga jezika. Svaka će cjelina imati teorijski dio te nakon toga kviz znanja. Teorijski će dio sadržavati slučajeve uporabe, formiranje rečenice te, ovisno o cjelini, objašnjenja. Kviz će znanja sadržavati pitanja temeljena na teoriji. Kako bi korisnik mogao prijeći na iduću razinu postotak će riješenosti trebati biti 100 %. Korisniku će biti prikazano objašnjenje za odgovore u kvizu. Korisnik će trebati prolaziti cjeline

određenim redoslijedom, odnosno neće ih moći preskakati jer će biti zaključane. Aplikacija je namijenjena za učenike osnovne i srednje škole.

2. PREGLED SLIČNIH RJEŠENJA

Trenutno na tržištu postoje brojne aplikacije za učenje engleskog jezika. Većina je aplikacija slična po pitanju sadržaja i funkcionalnosti te se uglavnom sastoje od dijela proučavanja teorije i provjere znanja. Poneke aplikacije koriste zvučno učenje gdje je klikom na gumb moguće čuti izgovor riječi i rečenice. Aplikacije na tržištu pokušavaju obuhvatiti cijelu gramatiku, ali ima i onih koje se orijentiraju na samo jedno područje. U sljedećim su poglavljima izdvojene i objašnjenje tri aplikacije najsličnijih aplikaciji izrađenoj za potrebe ovog završnog rada.

2.1. English Grammar (Tenses Test)

English Grammar (Tenses Test)[1] je aplikacija koja pomaže pri učenju glagolskih vremena engleskog jezika na lagan i učinkovit način. Sadrži prošlo, sadašnje i buduće vrijeme te preko dvije tisuće zadataka vezanih uz gramatiku engleskog jezika. Redoslijed rješavanja gramatičkih cjelina nije određen i cjeline se mogu preskakati. Postotak riješenosti kviza nije bitan za daljnje rješavanje te je on tu samo kako bi korisnik dobio povratnu informaciju o tome je li shvatio gradivo. Ukoliko je odgovor netočan ne dobije se objašnjenje, nego samo točan odgovor, što je nedostatak ove aplikacije. Pitanja su sastavljena od višestrukih odabira. Aplikacija ima teorijski i praktični dio, a sadrži detaljna objašnjenja svake gramatičke cjeline. Namijenjena je početnicima. Može ju se preuzeti s App Store-a i besplatna je. Pojednostavljenog je korisničkog sučelja.

2.2. LearnEnglish Grammar

LearnEnglish Grammar[2] je aplikacija za vježbanje gramatike engleskoga, japanskog i španjolskog jezika. Aplikacija nudi preko tisuću zadataka kako bi korisnik poboljšao svoje znanje gramatike engleskoga jezika. Sadrži četiri razine; od početnika do naprednog korisnika. Sastoji se od dvadeset i pet gramatičkih cjelina koje korisnik može nasumično odabirati. Ne

[1] https://apps.apple.com/us/app/english-grammar-tenses-test/id995714288
[2] https://apps.apple.com/us/app/learnenglish-grammar-uk-ed/id488099900

sadrži dio s teorijom, nego samo praktične zadatke. Moguće je odabrati vježbanje ili ispit, a prilikom vježbanja potrebno je odabrati težinsku razinu zbog čega je aplikacija namijenjena za sve razine znanja. Ispit se sastoji od višestrukog odabira odgovora, nadopunjavanja rečenica te odabira riječi koja ne odgovara u rečenici. Prilikom netočnog odgovora nije pruženo objašnjenje, nego samo prikaz točnog odgovora, što je, kao i poglavlju 2.1., nedostatak navedene aplikacije. Postotak riješenosti nije presudan za daljnje cjeline, nego služi kako bi korisnik dobio povratnu informaciju o usvojenosti pojedine cjeline. Aplikacija je besplatna i sadrži reklame, no moguća je nadoplata i isključivanje prikazivanja reklama.

2.3. English Grammar in Use & Test

English Grammar in Use & Test[3] je besplatna aplikacija koja sadrži teorijske cjeline i testove za provjeravanje usvojenosti gramatičkih cjelina engleskoga jezika. Namijenjena je za B1 i B2 razinu znanja. Korisnik bira cjelinu koju želi proći, a ista sadrži teorijski dio i primjere. Moguće je polagati kviz iz svake cjeline zasebno, ali je moguće i generirati nasumični ispit. Kviz se sastoji od pitanja u kojima je potrebno odabrati jedan od nekoliko ponuđenih odgovora. Ukoliko korisnik pogrešno odgovori, prikazan mu je točan odgovor te objašnjenje. Korisnik, neovisno o broju točnih odgovora, može ići dalje na druge cjeline. U aplikaciji se ne prati napredak korisnika. Moguće je cjeline staviti u favorite radi lakšeg pristupanja cjelini. Aplikacija je u cijelosti besplatna, ali često prikazuje reklame, što ometa čitanje i pristupanje kvizovima.

2.4. Osvrt na postojeća rješenja

Izdvojene su aplikacije slične te obuhvaćaju većinu gramatike engleskog jezika. Gramatika je podijeljena u manje cjeline nakon koje slijedi testiranje. Prvim ulaskom u aplikacije nema inicijalnog testiranja te korisnik može rješavati cjeline redoslijedom kojim želi. Svaka aplikacija ima testove, ali nemaju sve teorijski dio. Primjerice, *LearnEnglish Grammar* nema teorijski dio jer je namijenjena samo za vježbanje gramatike. Svaka aplikacija obuhvaća većinu gramatičkih cjelina, no nisu sve detaljno objašnjene. Testovi su uglavnom sastavljeni od

[3] https://apps.apple.com/us/app/english-grammar-in-use-test/id1526935199

višestrukih odgovora, no ima i zadataka nadopunjavanja. Postotak riješenosti nekog testa nije ključan prilikom prelaska na rješavanje zadataka neke druge cjeline. Jedino se u *English Grammar in Use & Test* aplikaciji daje objašnjenje ako je odgovor netočan.

Prednosti su analiziranih aplikacija dobro napravljeni testovi te jednostavnost korištenja. Nedostatci su aplikacija pojava reklama i nemogućnost praćenja napretka korisnika, odnosno nastavka učenja na drugom uređaju. Niti jedna aplikacija nema inicijalnog testiranja, što znači da ne usmjerava korisnika na razinu znanja i sadržaja njemu prikladnim. U tom se slučaju oslanjaju na to da korisnik sam zna koji mu sadržaji trebaju, što je vrlo teško za očekivati od početnika.

Aplikacija izrađena za potrebe završnog rada slična je analiziranim aplikacijama po obuhvaćenim cjelinama, no razlikuje se po tome što u analiziranim aplikacijama nema linearnosti i korisnik može odabrati koju god cjelinu želi, što je riješeno u ovom radu. Ovaj rad ima i teorijski dio i praktični dio. Praktični dio sadrži zadatke na nadopunjavanje. Aplikacija ne pruža objašnjenje netočnog odgovora, što je nedostatak te nema inicijalnog testa koji korisnika svrstava u određenu razinu znanja i nudi mu cjeline primjere njoj, čime se otvara mogućnost za nadogradnju aplikacije.

3. KORIŠTENE TEHNOLOGIJE I PROGRAMSKA ARHITEKTURA

3.1. Operacijski sustav iOS

iOS je mobilni operacijski sustav koji je razvio Apple Inc. isključivo za svoje sklopovlje. Predstavljen 2007. za prvu generaciju iPhonea, iOS je od tada proširen za podršku drugim Apple uređajima kao što su iPod Touch (rujan 2007.) i iPad (predstavljen u siječnju 2010., a dostupan od travnja 2010.). Od ožujka 2018., Apple-ov App Store sadrži više od 2,1 milijuna iOS aplikacija, od kojih je 1 milijun izvorno izrađeno za iPad [4]. Ove mobilne aplikacije zajedno su preuzete više od 130 milijardi puta. iOS je trenutno drugi najzastupljeniji operacijski sustav, odmah poslije Androida [5]. Razlog zašto je Android popularniji od iOS-a je različit raspon cijena Android uređaja. Glavna hardverska platforma za iOS je ARM arhitektura (ARMv7, ARMv8-A, ARMv8.2-A, ARMv8.3-A). Izdanja iOS-a prije iOS-a 7 mogu se izvoditi samo na iOS uređajima s 32-bitnim ARM procesorima (ARMv6 i ARMv7-A arhitekture). iPhonei su općenito puno glatkiji, brži i imaju znatno bolji vijek trajanja u usporedbi s Android pametnim telefonima. iOS je najviše korišten u Kini [6].

Slika 3.1. Početni zaslon iOS operacijskog sustava

Korisničko sučelje iOS-a temelji se na izravnoj manipulaciji i korištenju višestrukih dodirnih gesti kao što su povlačenje prstom, dodir, štipanje i obrnuto štipanje. Kontrolni elementi sučelja uključuju klizače, prekidače i gumbe. Siri je osobni pomoćnik integriran u iOS. Siri koristi glasovne upite i korisničko sučelje za odgovaranje na pitanja, davanje preporuka i izvođenje radnji delegiranjem zahtjeva skupu internetskih usluga. Program se prilagođava individualnoj upotrebi jezika, pretraživanjima i preferencijama korisnika uz kontinuiranu upotrebu.

3.2. Swift programski jezik

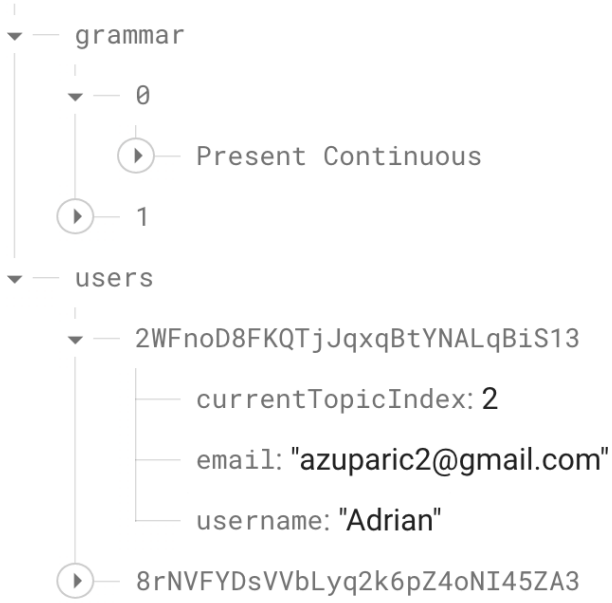
Swift je moćan i intuitivan programski jezik za iOS, iPadOS, macOS, tvOS, watchOS. Moderan je jezik koji ima čistu i izražajnu sintaksu [7]. Razvio ga je Chris Lattner iz tehnološke tvrtke Apple Inc. i prvi je put predstavljen 2014. godine na Worldwide Developers Conference (WWDC). Temelji se na tehnikama naučenim u Objective-C, a prilagođen je današnjim standardima kako bi uključio kraću sintaksu i lakšu čitljivost, lakoću održavanja i sigurnost. Swift kod je sastavljen i optimiziran kako bi se postiglo najviše moguće iz modernog sklopovlja. Swift se temelji na modernim praksama koje se također mogu vidjeti u drugim modernim programskim jezicima kao što su JavaScript, Ruby i Kotlin. Dostupan je u najnovijim verzijama Xcode-a i može se koristiti za izradu aplikacija od iOS 7 ili novijih i macOS uređaja koje datiraju iz Mac OS X 10.9 ili novijih. Swift je otvorenog koda, što je prednost, kao i brži razvoj, laka čitljivost i jednostavno održavanje. Relativno je mlad jezik te to predstavlja nedostatak. Prema zadanim postavkama, Swift ne izlaže pokazivače i druge nesigurne pristupnike, za razliku od Objective-C, koji sveprisutno koristi pokazivače za upućivanje na instance objekta. Zadržava ključne koncepte Objective-C-a, uključujući protokole (engl. *protocols*) i zatvaranja (engl. *closures*), često zamjenjujući bivšu sintaksu čišćim verzijama i dopuštajući da se ti koncepti primjenjuju na druge jezične strukture, kao što su enumeracije („enums“).

```
21      override init(frame: CGRect) {
22          super.init(frame: frame)
23          configureView()
24      }
25
26      required init?(coder aDecoder: NSCoder) {
27          super.init(coder: aDecoder)
28          configureView()
29      }
30
31      func configureView() {
32          xibView = self.loadViewFromNib(nibName: "ExpensesDetailsHeaderView")
33          self.addSubview(xibView)
34      }
35
36      override func layoutSubviews() {
37          super.layoutSubviews()
38          layer.shadowColor = UIColor.shadow.cgColor
39          layer.shadowOffset = CGSize(width: 0.0, height: 2.0)
40          layer.shadowOpacity = 1.0
41          layer.shadowRadius = 4.0
42          layer.masksToBounds = false
43          xibView.frame = self.bounds
44      }
45
46  }
```

Slika 3.2. Primjer Swift koda

3.3. Firebase

Firebase je platforma koju su razvili James Tamplin i Andrew Lee 2011. godine. Google je 2014. kupio Firebase i sada je to njihova vodeća ponuda za razvoj aplikacije. Firebase je platforma smještena u oblaku, integrirana s *Google Cloud Platformom*, koja koristi alat za kreiranje i sinkronizaciju projekata te omogućuje povećanje broja korisnika. Firebase pruža usluge za iOS, Android, Web i Unity. U radu koristimo bazu podataka u stvarnom vremenu (engl. *Realtime database*). Baza podataka u stvarnom vremenu jedan je veliki JSON (JavaScript Object Notation) objekt, a JSON je lagani format za razmjenu podataka koji ljudima olakšava čitanje i pisanje te ga strojevi lako analiziraju i generiraju. Postoje tri glavne kategorije usluga koje Firebase pruža - izrada bolje aplikacije, poboljšanje kvalitete aplikacije i unaprjeđenje aplikacije.



Slika 3.3. Primjer baze podataka u stvarnom vremenu

Firebase baze podataka mogu se skalirati u smislu veličine. Ažuriranje podataka i izvanmrežni pristup čine baze podataka upotrebljivima na primjenu u stvarnom vremenu, kao i sinkronizacija više baza podataka. Firebase ne zahtijeva plaćanje za većinu svojih usluga. Jedan od glavnih problem baze podataka u stvarnom vremenu su ograničenja postavljanja upita. Ne pruža mogućnost filtriranja jer je cijela baza podataka velika JSON datoteka što otežava izradu složenih upita.

3.4. Xcode IDE

Xcode, koji je korišten za razvoj i testiranje ove aplikacije, je Apple-ovo integrirano razvojno okruženje za macOS. Koristi se za razvoj programa za macOS, iOS, iPadOS, watchOS i tvOS. Objavljen je 2003. godine, a može se preuzeti iz trgovine aplikacijama [8]. Xcode nije moguće pokrenuti na ostalim platformama kao što su Windows i Linux. Registrirani programeri mogu preuzeti pretpregledna izdanja i prethodne verzije putem mrežne stranice *Apple Developer*[4]. Xcode podržava izvorni kod za programske jezike C, C++, Objective-C, Objective-C++, Java, AppleScript, Python, Ruby, ResEdit i Swift. Xcode IDE je cjeloviti paket i koristeći

[4] https://developer.apple.com/

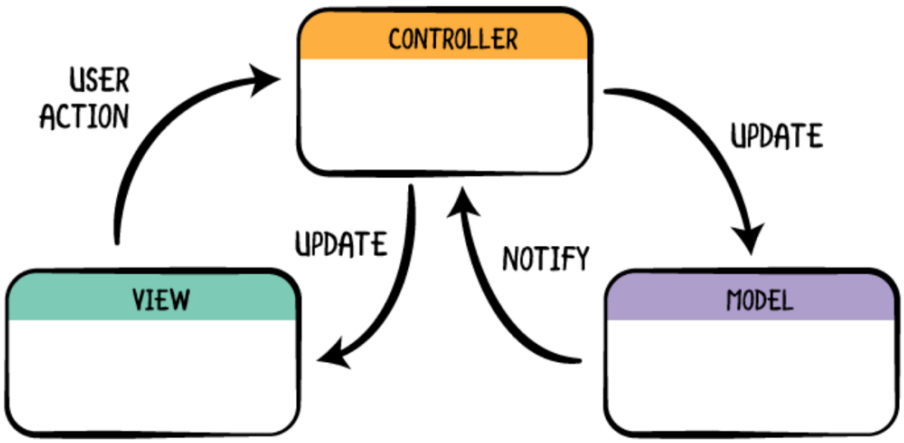
ga, programeri mogu obavljati više zadataka u rasponu od dizajniranja korisničkog sučelja, pisanja koda za aplikacije, kompajliranja i testiranja koda te provjere grešaka u kodu. Svi korisnici macOS-a mogu besplatno koristiti Xcode, ali kako bi distribuirali aplikacije na više platformi Apple-ove trgovine, potrebno je pretplatiti se na Apple Developer program, čija je cijena 99 USD godišnje. Simulator u Xcode-u omogućuje jednostavno testiranje aplikacije. Veliki nedostatak Xcode-a je što je podržan samo na Apple OS-u.

3.5. CocoaPods

Cocoapods je upravitelj ovisnosti za Swift i Objective-C Cocoa projekte. CocoaPods je snažno inspiriran kombinacijom Ruby projekata RubyGems i Bundler. Pokreće se iz naredbenog retka, instalira ovisnosti za aplikaciju specifikacijom ovisnosti umjesto ručnog kopiranja izvornih datoteka. Cocoapods ovisnosti pokreće Molinillo koji također koriste drugi veliki projekti kao što su Bundler, RubyGems i Berkshelf [9]. U završnom radu Cocoapods su korišteni kako bi se integrirala Firebase baza podataka te Firebase autentikacija.

3.6. MVC

Pojam MVC je kratica za model-pogled-kontroler (engl. *Model-View-Controller*). MVC je obrazac dizajn programa koji odvaja logiku aplikacije prema odgovornostima - model upravlja podatkovnom strukturom aplikacije, pogled upravlja načinom na koji su informacije predstavljene u korisničkom sučelju, a kontroler prihvaća unos i šalje naredbe modelu i pogledu. U projektu je korištena MVC arhitektura kako bi projekt bio čitljivi, jasniji i strukturalno lakši za održavanje. Svim se datotekama upravlja u odgovarajućim mapama, tako da svatko može jednostavno pronaći datoteke i održavati ih.



Slika 3.4. Primjer MVC arhitekture[5]

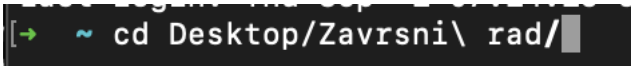
[5] https://www.raywenderlich.com/1000705-model-view-controller-mvc-in-ios-a-modern-approach

4. RAZVOJ APLIKACIJE I PRIKAZ FUNKCIONALNOSTI

4.1. Instalacija ovisnosti putem CocoaPods

Nakon kreiranja projekta potrebno je ubrizgavanje ovisnosti (engl. *Dependency injection*) putem CocoaPods-a. U projektu su korišteni FirebaseAuth i FirebaseDatabase kako bi se mogao koristiti Firebase autentifikacija i Firebase baza podataka.

Za instalaciju CocoaPods upravitelja ovisnosti koristi se terminal unutar macOS operacijskog sustava. Nakon otvaranja terminala potrebno je promijeniti trenutni direktorij u direktorij u kojem je kreiran projekt naredbom *cd*.

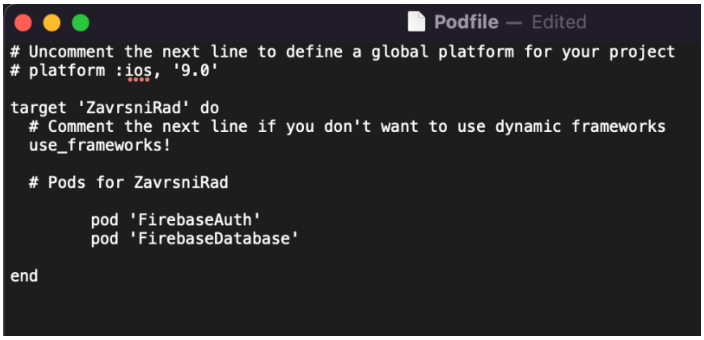


Slika 4.1. Promjena direktorija

Nakon premještanja u direktorij projekta, potrebno je inicializirati *Podfile. Podfile* služi kako bi navele sve ovisnosti koje se koriste u projektu. Naredba za inicializiranje *Podfile*-a je *pod init*.

Slika 4.2 Inicializacija *Podfile* datoteke

U direktoriju bi se projekta trebalo stvoriti *Podfile*. *Podfile* je potrebno otvoriti uređivačem teksta te u njega definirati ovisnosti koje se žele koristiti u projektu.



Slika 4.3 *Podfile* s ovisnostima

Nakon definiranja ovisnosti potrebno je spremiti promjene te pokrenuti instalaciju naredbom *pod install.* Ukoliko je instalacija uspješno završila, moguće je koristiti ovisnosti u projektu.

Slika 4.4. Instalacija ovisnosti

4.2. Registriranje korisnika u bazu podataka

Prilikom prvog pokretanja aplikacije korisnik se treba registrirati u aplikaciju. Za registraciju je potrebno unijeti korisničko ime (engl. *username*), adresu elektroničke pošte (engl. *email*) i zaporku (engl. *password*). Korisničko ime i adresa elektroničke pošte moraju biti jedinstveni. To znači da jedno korisničko ime i jedna adresa elektroničke pošte može imati samo jedan račun. Nakon uspješne registracije korisnik se može prijaviti u aplikaciju.

Pritiskom na gumb *Register* poziva se metoda *didTapRegister* (Slika 4.5.) koja koristi provjerava jesu li svi podaci upisani te ako jesu, Firebase metoda *createUser* upisuje podatke korisnika u bazu podataka.

```
@objc func didTapRegister(){
    guard let email = registerView.emailTextField.text, email != "" else { fieldIsEmpty(fieldName: "Email"); return }
    guard let password = registerView.passwordTextField.text, password != "" else { fieldIsEmpty(fieldName: "Password"); return}
    guard let username = registerView.usernameTextField.text, username != "" else { fieldIsEmpty(fieldName: "Username"); return}
    Auth.auth().createUser(withEmail: email, password: password) { authResult, error in
        if error == nil && authResult != nil {
            let changeRequest = Auth.auth().currentUser?.createProfileChangeRequest()
            changeRequest?.displayName = username
            changeRequest?.commitChanges(completion: { error in
                if error == nil {
                    self.ref.child("users").child((authResult?.user.uid)!).setValue(["email": email, "username": username, "currentTopicIndex": 0])
                    self.loginSuccess()
                }
            })
        } else {
            let alert = UIAlertController(title: "Failure", message: "Something went wrong, please try again", preferredStyle: .alert)
            alert.addAction(UIAlertAction(title: "OK", style: .default))
            self.present(alert, animated: true)
        }

    }
}
```

Slika 4.5. didTapRegister metoda

4.3. Prijava

Ukoliko korisnik ima korisnički račun, moguća je prijava u aplikaciju. Potrebno je upisati elektroničku poštu i zaporku te pritisnuti gumb *Login*. Metoda *didTapLogin* (Slika 4.6.) provjerava jesu li polje za upisivanje elektroničke pošte i zaporka prazni. Ukoliko su prazni, pojavljuje se prozor s obavijesti da je jedno od tih dvaju polja prazno. Ukoliko polja nisu prazna, poziva se Firebase metoda koja kao parametre prima elektroničku poštu i zaporku te vraća odgovor postoji li korisnik s tom zaporkom i elektroničkom poštom i je li to dvoje ispravno uneseno. Ako je odgovor ispravan, spremaju se podaci trenutnog korisnika i prelazi se na idući prozor.

```
@objc func didTapLogin(){
    guard let email = loginView.emailTextField.text, email != "" else { fieldIsEmpty(fieldName: "Email"); return }
    guard let password = loginView.passwordTextField.text, password != "" else { fieldIsEmpty(fieldName: "Password"); return}
    Auth.auth().signIn(withEmail: email, password: password) { authResult, error in
        if let user = authResult?.user, error == nil{
            UserInfo.shared.email = user.email
            UserInfo.shared.username = user.displayName
            UserInfo.shared.uID = user.uid
            self.ref.child("users").getData { error, snapshot in
                let users = snapshot?.value as? [String: [String: Any]]
                guard let uid = UserInfo.shared.uID else { return }
                guard let user = users?[uid] else {
                    return
                }
                if let currentTopicIndex = user["currentTopicIndex"] as? Int {
                    UserInfo.shared.currentTopicIndex = currentTopicIndex
                } else {
                    UserInfo.shared.currentTopicIndex = 0
                }
                let menuVC = MenuViewController()
                self.view.window?.rootViewController = UINavigationController(rootViewController: menuVC)
            }
        } else {
            let alert = UIAlertController(title: "Error", message: error?.localizedDescription, preferredStyle: .alert)
            alert.addAction(UIAlertAction(title: "OK", style: .default))
            self.present(alert, animated: true)
        }
    }
}
```

Slika 4.6. didTapLogin metoda

4.4. Glavni izbornik

Nakon uspješne prijave prikazuje se zaslon pregleda svih tema. Prilikom prikazivanja glavnog izbornika pozvana je metoda *getData* (Slika 4.7.) koja Firebase metodom *.observe(.value)* dohvaća sve gramatičke teme iz baze podataka.

```
func getData() {
    ref.child("grammar").observe(.value) { snapshot in
        DispatchQueue.main.async { [weak self] in
            if let dict = snapshot.value as? [[String: [Any]]]{
                for topic in dict {
                    self?.topics.append(topic.first!.key)
                }
                if let currentTopic = UserInfo.shared.currentTopicIndex{
                    self?.headerView.progressBar.progress = Float((currentTopic))/Float((self?.topics.count ?? 0))
                }
                self?.tableView.reloadData()
            }
        }
    }
}
```

Slika 4.7. getData metoda

Nakon što se uspješno dohvate sve teme, iste se prikazuju pomoću *UITableView* (Slika 4.8.). *UITableView* upravlja osnovnim izgledom tablice koristeći ćelije (UITableViewCell) koje prikazuju sadržaj. Svaka ćelija sadrži ime teme te je pozadinskom bojom određeno je li tema otključana ili ne; crvena boja prikazuje zaključane teme, a bijela otključane.

```
func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
    topics.count
}

func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
    let cell = tableView.dequeueReusableCell(withIdentifier: "cell", for: indexPath)
    cell.textLabel?.text = topics[indexPath.row]
    guard let topicIndex = UserInfo.shared.currentTopicIndex else { return UITableViewCell()}
    if indexPath.row > topicIndex {
        cell.backgroundColor = .red
    } else {
        cell.backgroundColor = .white
    }
    return cell
}
```

Slika 4.8. Postavljanje *UITableView-a*

Na glavnom izborniku moguće je vidjeti napredak (postotak riješenosti) s prilagođenim *UIView* elementom. *UIView* se može definirati kao objekt pomoću kojeg je moguće kreirati i upravljati pravokutnim područjem ekrana. Moguće je imati bilo koji broj pogleda unutar pogleda kako bismo stvorili hijerarhijsku strukturu *UIView-a*. *UIViewom* se upravlja pomoću metoda i svojstava definiranih u klasi *UIView* koja nasljeđuje *UIKit*.

Odabirom na otključanu temu poziva se metoda prikazana na Slici 4.9. Poziva se Firebase metoda *.observe(.value)* kojoj se predaje odabrana tema te u odgovor sadrži rječnik (engl. *Dictionary*) koji za ključ (engl. *key*) dobiva redni broj pitanja, a za vrijednost (engl. *value*) pitanje u obliku niza (engl. *String*). Ukoliko se dobije odgovor, poziva se novi prozor.

```
func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
    tableView.deselectRow(at: indexPath, animated: true)
    guard let topicIndex = UserInfo.shared.currentTopicIndex else { return }
    if indexPath.row <= topicIndex {
        ref.child("grammar").child(String(indexPath.row)).child(topics[indexPath.row]).observe(.value){ snapshot in
            let topic = snapshot.value as? [[String: [Any]]]
            guard let topic = topic else { return }
            let topicVC = TopicViewController(topic: topic)
            topicVC.delegate = self
            self.navigationController?.pushViewController(topicVC, animated: true)
        }
    }
}
```

Slika 4.9. Odabiranje teme

4.5. Tema gramatike

Prikaz teme gramatike sastoji se od elementa *UICollectionView*[6] (Slika 4.10.). *UICollectionView* je objekt koji upravlja uređenom zbirkom podatkovnih stavki i predstavlja ih pomoću prilagodljivih izgleda. Pojedinačni pogled naziva se ćelija.

[6] https://developer.apple.com/documentation/uikit/uicollectionview

```
func collectionView(_ collectionView: UICollectionView, numberOfItemsInSection section: Int) -> Int {
    titles.count
}

func collectionView(_ collectionView: UICollectionView, cellForItemAt indexPath: IndexPath) -> UICollectionViewCell {
    switch indexPath.row%2{
    case 0:
        let cell = collectionView.dequeueReusableCell(withReuseIdentifier: TheoryCollectionViewCell.identifier, for: indexPath) as! TheoryCollectionViewCell
        guard let model = topic[indexPath.row].first?.value as? [String] else { return UICollectionViewCell() }
        cell.configure(with: model)
        return cell

    case 1:
        let cell = collectionView.dequeueReusableCell(withReuseIdentifier: QuizCollectionViewCell.identifier, for: indexPath) as! QuizCollectionViewCell
        guard let model = topic[indexPath.row].first?.value as? [[String: String]] else { return UICollectionViewCell() }
        cell.configure(with: model)
        cell.delegate = self
        return cell
    default:
        let cell = collectionView.dequeueReusableCell(withReuseIdentifier: "cell", for: indexPath)
        return  cell
    }
}

func collectionView(_ collectionView: UICollectionView, layout collectionViewLayout: UICollectionViewLayout, sizeForItemAt indexPath: IndexPath) -> CGSize {
    return CGSize(width: view.frame.width, height: view.frame.height)
}

func configureCollectionView() {
    configureLayout()
    collectionView = UICollectionView(frame: view.frame, collectionViewLayout: layout)
    collectionView?.register(TheoryCollectionViewCell.self, forCellWithReuseIdentifier: TheoryCollectionViewCell.identifier)
    collectionView?.register(QuizCollectionViewCell.self, forCellWithReuseIdentifier: QuizCollectionViewCell.identifier)
    collectionView?.register(UICollectionViewCell.self, forCellWithReuseIdentifier: "cell")
    collectionView?.isPagingEnabled = true
    collectionView?.showsHorizontalScrollIndicator = false

    collectionView?.delegate = self
    collectionView?.dataSource = self

    collectionView?.bounces = false
    collectionView?.backgroundColor = .white
}
```

Slika 4.10. Konfiguracija *UICollectionView-a*

Prikaz teme gramatike sastoji se od naizmjenično prikazanih *UICollectionViewCell-a*[7] teorije i kviza. *UICollectionViewCell* je jedna podatkovna stavka unutar vidljivih granica prikaza zbirke. *UICollectionViewCell* je moguće koristiti predefinirano ili napraviti podklasu za dodavanje dodatnih svojstava i metoda. Prilikom ulaska u temu gramatike moguće je pristupiti prvoj stranici teorije i prvom kvizu. Ukoliko korisnik točno odgovori na sva pitanja, otključavaju mu se iduća dva prikaza.

4.6. Teorija

Teorija je sastavljena od nekoliko rečenica prikazanih u *UIStackView* elementu. *UIStackView* omogućava automatski raspored, stvarajući korisnička sučelja koja se mogu dinamički prilagoditi orijentaciji uređaja, veličini zaslona i svim promjenama u dostupnom prostoru. Elementi se dodavaju metodom *addArrangedSubview*. Metoda *configure()* (Slika

[7] https://developer.apple.com/documentation/uikit/uicollectionviewcell

4.11.) kao parametre dobiva niz (engl. *array*) rečenica koje se for petljom prolaze te se svaka rečenica dodaje *UILabel* elementu.

```
func configure(with values: [String]){
    for value in values{
        let label = UILabel()
        label.text = value
        label.numberOfLines = 0
        label.backgroundColor = .white
        stackView.addArrangedSubview(label)
    }
}
```

Slika 4.11. Konfiguracija zaslona teorije

4.7. Kviz

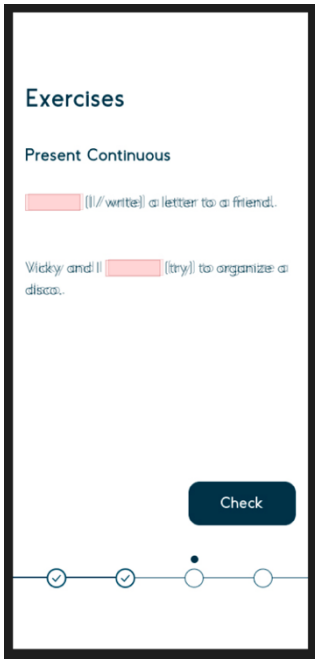
Kviz je tipa *UICollectionViewCell-a* koji sadrži *UIStackView* kako bi svi elementi bili jednako raspoređeni. Svaki dio *UIStackView-a* sadrži *UILabel* koji za tekst sadrži pitanje i *UITextField* za unos odgovora na pitanje. Nakon unosa odgovora korisnik treba pritisnuti gumb za dalje kako bi provjerio odgovore te ukoliko su odgovori točni, otključava se novi dio teme gramatike. Pritiskom na gumb *Check Answers* poziva se metoda na Slici 4.12. For petljom se iterira kroz sve elemente tipa *UITextField* kako bi se provjerili upisani odgovori s točnim odgovorima. Na kraju metode provjerava se broj točnih odgovora s brojem pitanja te ako je broj isti, delegiranjem se obavještava da se otključa idući zaslon teorije i kviza.

```
@objc func didTapCheckAnswers(){
    var correct = 0
    for i in 0..<numberOfQuestions{
        let textField = viewWithTag(i+1) as? UITextField
        print(answers[i])
        if let text = textField?.text{
            if text == answers[i]{
                correct+=1
                textField?.backgroundColor = .green
            } else {
                textField?.text = ""
                textField?.backgroundColor = .red
            }
        }
    }
    if correct == numberOfQuestions{
        delegate?.setCurrentUnlockedScreens()
    } else {
        animateShake()
    }
}
```

Slika 4.12. *didTapCheckAnswers* metoda

Ako je točno odgovoreno na zadnji dio teme gramatike, prikazuje se *UIAlertController* koji obavještava da je tema gramatike uspješno odrađena te se prikazuje glavni izbornik. Znak da je korisnik netočno odgovorio na pitanje (Slika 4.13.) je ekran koji se zatrese (Slika 4.14.).

Slika 4.13. Netočno riješen zadatak

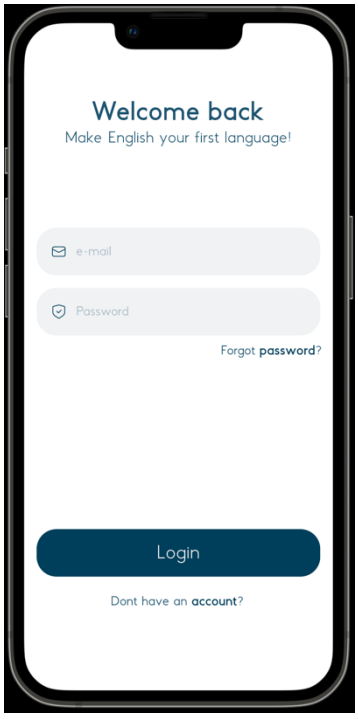


Slika 4.14. Zatreseni ekran

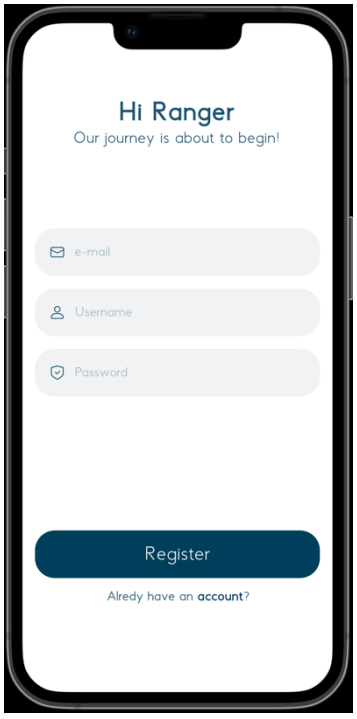
5. KORIŠTENJE I IZGLED APLIKACIJE

5.1. Početni zaslon aplikacija

Ovo poglavlje opisuje korištenje i izgled same aplikacije. Pokretanjem aplikacije prikazan je zaslon za prijavu u aplikaciju (Slika 5.1.) u kojem je moguće u polja upisati korisničko ime i zaporku te je nakon unosa je potrebno pritisnuti gumb *Login*. Ako korisnik nema račun, pritiskom na *Don't have an account* odvodi ga se na zaslon za registraciju (Slika 5.2.) koja sadrži registracijsku formu. Nakon uspješne registracije korisnik se treba prijaviti u aplikaciju.



Slika 5.1. Zaslon za prijavu



Slika 5.2. Zaslon za registraciju

5.2. Glavni izbornik

Nakon registracije i prijave korisniku se prikazuje glavni izbornik (Slika 5.3.) koji se sastoji od napretka korisnika na sredini ekrana koji je prikazan animacijom i postotkom riješenost tema gramatike. Ispod napretka nalaze se ćelije izlistanih tema gramatike. Svaka

ćelija sadrži ime teme i prikaz lokotom je li tema otključana ili ne. U kutu gore desno nalazi se gumb za postavke.

Slika 5.3. Glavni izbornik

5.3. Zaslon teorije i kviza

Prilikom odabira teme gramatike otvara se zaslon u kojem je prikazana teorija gramatike (Slika 5.4.). Teorija sadržava primjere tvorbe, uporabe i primjere. Nakon što korisnik pročita teoriju, povlačenjem prstom u lijevo, prelazi se na zaslon gdje se pojavljuje kviz (Slika 5.5.) temeljen na pročitanoj teoriji.

Slika 5.4. Teorijski dio teme gramatike

Slika 5.5. Kviz prijašnje prikazane teorije

Nakon što korisnik ispuni, pritiskom na gumb *Next* provjerava se je li točno riješio. Ukoliko je korisnik točno riješio (Slika 5.6.), otključavaju mu se iduća dva zaslona, a ako nije točno riješio, polje za unos se zacrveni (Slika 5.7.).

Slika 5.7. Netočno uneseni odgovori

Slika 5.6. Točno uneseni odgovori

6. ZAKLJUČAK

Kako se nalazimo u vremenu u kojemu je rast globalizma i korištenje raznih internetskih tehnologija gotovo pa nezaustavljiv fenomen, tako postoji i sve veća potreba prosječnog pojedinca da nauči i poboljša svoje znanje engleskog jezika, jezika interneta. Svrha je ovog rada pojednostaviti vježbanje i učenje engleskog jezika učenicima osnovne i srednje škole koristeći moderne tehnologije u kombinaciji s tradicionalnim metodama učenja.

U okviru ovoga je rada izrađena aplikacija za samoučenje gramatičkih cjelina engleskog jezika koja sadrži teorijski dio i kviz za provjeru znanja, a korisnik može pratiti svoj napredak. Aplikacija sadrži prijavu i registraciju tako da ju je moguće koristiti na više uređaja. Uz strelovit rast tržišnog udjela pametnih telefona, Apple-ova platforma iOS, jedna od dviju najzastupljenijih platformi za pametne telefone, predstavlja kvalitetan ekosustav i platformu za aplikaciju koju ovaj rad opisuje. Korištene su tehnologije UIKit za korisničko sučelje, ali je i prije izrade rada, razmotren SwiftUI. Iako bi podatci vjerojatno bili lakše organizirani i bolje strukturirani s relacijskim bazama podataka, brzina, efikasnost i lakoća implementacije Firebase dokumentnih baza podatka ipak je bolji izbor za ovakav tip aplikacije. Naspram analiziranih postojećih rješenja, aplikacija iz ovog rada donosi koncizno, pregledno i jednostavno rješenje. Aplikacija omogućuje korisniku praćenje dosadašnjeg napretka, linearan prolazak kroz gradivo gramatike engleskog jezika i interaktivno testiranje naučenog gradiva. Sve ti elementi zajedno tvore jednu kohezivnu cjelinu koja značajno olakšava učenje stranog jezika uz nastojanje da korištenje aplikacije bude zanimljivo, ugodno te, možda najvažnije od svega, učinkovito.

Kako je tržište pametnih uređaja veliko, a engleski jezik vrlo širok pojam, u ovome radu postoji nekolicina točaka po kojima bi se rad mogao proširiti i poboljšati. Prvenstveno, izrada Android verzije značajno bi proširila potencijalnu korisničku bazu. Internetska verzija aplikacije također bi se mogla razmotriti kako bismo u konačnici dosegnuli veći broj korisnika. Samo gradivo engleskog jezika koje se obrađuje kroz aplikaciju moglo bi se proširiti s vježbama vokabulara i govora uz korištenje raznih oblika multimedije poput video i zvučnih zapisa.

LITERATURA

[1] M. Kuimova, D. Burleigh, H. Uzunboylu, R. Bazhenov, Positive Effects of Mobile Learning on Foreign Language Learning, TEM Journal. 7(4): 837-841, ISSN 2217-8309, DOI: 10.18421/TEM74-22, studeni 2018.

[2] R. Metrku, The Use of Smartphones English Language Learning Apps in the Process of Learning English: Slovak EFL Students Perspectives, srpanj 2021.

[3] The most spoken languages worldwide in 2022, dostupno na: https://www.statista.com/statistics/266808/the-most-spoken-languages-worldwide/ [pristupljeno: 17. 8. 2022.]

[4] iOS, dostupno na: https://en.wikipedia.org/wiki/IOS [pristupljeno: 19. 8. 2022.]

[5] T. Hamed, S. C. Kremer, Computer and Information Security Handbook (Third Edition), 2017

[6] Number of Apple iPhone devices in use in the U.S., China and the rest of the world in 2017, dostupno na: https://www.statista.com/statistics/755625/iphones-in-use-in-us-china-and-rest-of-the-world/ [pristupljeno: 3.9.2022.]

[7] Swift, dostupno na: https://developer.apple.com/swift/ [pristupljeno: 23. 8. 2022.]

[8] Xcode, dostupno na: https://en.wikipedia.org/wiki/Xcode [pristupljeno: 24. 8. 2022.]

[9] CocoaPods, dostupno na: https://en.wikipedia.org/wiki/CocoaPods [pristupljeno: 29. 8. 2022.]

SAŽETAK

Engleski jezik je najraširenij jezik te ga uče milijuni ljudi od kojih se neki za to koriste tehnologijom. Pametni telefoni omogućuju korisnicima da gdje god se nalazili imaju pristup svakakvim aplikacijama. Neke od tih aplikacija su aplikacije za učenje stranog jezika. Trenutno postoje brojne aplikacije za učenje engleskog jezika za različita predznanja i ciljeve korisnika. Cilj je ovog rada bio izraditi aplikaciju za učenje gramatike engleskog jezika koja će se rješavati određenim redoslijedom, te mogućnost praćenja korisnikovog napretka. Prednosti su aplikacije izrađen redoslijed gramatike kojim korisnik uči; počinje se od jednostavnijih primjera i teorije do težih. Smjer učenja sastavljen je na temelju puta učenja u osnovnim i srednjim školama. Nedostatci su ista vrsta ispitnih pitanja. Za izradu korisničkog sučelja korišten je *UIKit*, a za bazu podataka, u kojem je spremljena sva teorija, kvizovi i korisnički podaci, korišten je Firebase. Aplikacija je namijenjena za učenike osnovnih i srednjih škola. Mogla bi se nadograditi tako da profesori mogu pratiti napredak svojih učenika. Potpuno je besplatna te ne sadrži nikakve reklame.

Ključne riječi: aplikacija za samoučenje, Firebase, gramatika engleskog jezika, iOS aplikacija, Swift

ABSTRACT

English is the most widely spoken language learned by millions of people some of whom use technology for self-learning. Smartphones allow users to have access to any application wherever they are. Some of these apps are apps for learning a foreign language. Currently, there are numerous English language learning apps for different users' knowledge and goals. The goal of this work was to create an application for learning English grammar that will be solved in a certain order, and the possibility of monitoring the user's progress. The advantages of the application are the order in which the user learns grammar starting from simpler examples and theory to more difficult ones. The course of study is composed on the basis of the course of study in primary and secondary schools. Disadvantages are the same type of exam questions. UIKit is used to create the user interface, and Firebase is used for the database, where all theory, quizzes and user data are stored. The application is intended for primary and secondary school students. It could be upgraded so that teachers can track their students' progress. It is completely free and does not contain any advertisements.

Keywords: iOS app, self-learning app, English grammar, Firebase, Swift

ŽIVOTOPIS

Adrian Župarić rođen je u Vinkovcima 11. 2. 2000., a živi u Vinkovcima i Osijeku. Nakon završene osnovne škole Ivana Gorana Kovačića, upisuje Tehničku školu Ruđera Boškovića u Vinkovcima. Po završetku srednje škole upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija u Osijeku, preddiplomski sveučilišni studij Računarstva 2018. godine na kojem i trenutno studira završavajući treću godinu preddiplomskog studija.

Adrian Župarić

_____