MOBILNA APLIKACIJA ZA USAVRŠAVANJE GOVORA DJECE S AUTIZMOM

Pinjuh, Jure

Undergraduate thesis / Završni rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Algebra University College / Visoko učilište Algebra**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:225:129542

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-28**

Repository / Repozitorij:

Algebra Univerity College - Repository of Algebra Univerity College

VISOKO UČILIŠTE ALGEBRA

ZAVRŠNI RAD

MOBILNA APLIKACIJA ZA USAVRŠAVANJE GOVORA DJECE S AUTIZMOM

Jure Pinjuh

Zagreb, veljača 2020.

„Pod punom odgovornošću pismeno potvrđujem da je ovo moj autorski rad čiji niti jedan dio nije nastao kopiranjem ili plagiranjem tuđeg sadržaja. Prilikom izrade rada koristio sam tuđe materijale navedene u popisu literature, ali nisam kopirao niti jedan njihov dio, osim citata za koje sam naveo autora i izvor, te ih jasno označio znakovima navodnika. U slučaju da se u bilo kojem trenutku dokaže suprotno, spreman sam snositi sve posljedice uključivo i poništenje javne isprave stečene dijelom i na temelju ovoga rada".

U Zagrebu, 18.02.2020.

Jure Pinjuh

Predgovor

Želim se zahvaliti profesoru Aleksanderu Radovanu i ostalim profesorima Visokog učilišta Algebra na pruženom znanju kroz ove tri akademske godine.

Posebno se zahvaljujem svojoj obitelji i djevojci na pruženoj potpori tijekom obrazovanja.

Prilikom uvezivanja rada, Umjesto ove stranice ne zaboravite umetnuti original potvrde o prihvaćanju teme završnog rada kojeg ste preuzeli u studentskoj referadi

Sažetak

Tema ovog rada izrada je mobilne i web aplikacije koja bi osobama s autističnim poremećajem omogućila komunikaciju s drugima preko simbola, teksta i pretvaranja teksta u govor. Aplikacija spada u kategoriju AAC (engl. *alternative and augmentative communication*) aplikacija pa kao takva služi kao pomagalo u komunikaciji. U radu je opisan razvoj baze podataka, web servisa te web i mobilne aplikacije. Tijekom izrade rada korištene su moderne tehnologije kao što su ASP.NET Web API, Dapper, Angular i NativeScript.

Ključne riječi: autistični poremećaj, AAC, mobilna aplikacija, web aplikacija, ASP.NET, Dapper, Angular, NativeScript

Summary

The theme of this paper is the development of mobile and Web application that would enable people with autistic disorder to communicate using symbols, text and text-to-speech. Application is an AAC (augmentative and alternative communication) application, and as such it serves as a communication tool. This paper describes the development of database, Web service and Web and mobile application. Modern technologies such as ASP.NET Web API, Dapper, Angular and NativeScript were used during the development.

Key words: autistic disorder, AAC, mobile application, web application, ASP.NET, Dapper, Angular, NativeScript

Sadržaj

1. Uvod 3

2. Definicija autističnog poremećaja 4

3. Augmentativna i alternativna komunikacija (AAC) 5

3.1. Općenito o augmentativnoj i alternativnoj komunikaciji (AAC) 5

3.2. Primjena augmentativne i alternativne komunikacije (AAC) 5

3.3. Učenje i razvijanje vještina komunikacije korištenjem augmentativne i alternativne komunikacije (AAC) 5

4. Korištene tehnologije 7

4.1. Microsoft SQL Server 7

4.2. ASP.NET 8

4.2.1. ASP.NET Web API 8

4.3. Angular 8

4.4. NativeScript 9

5. Baza podataka 10

5.1. Izrada tablica 10

5.2. Izrada procedura za rad s podacima 12

5.3. Er dijagram baze podataka 13

6. Web Servis 14

6.1. Izrada modela 14

6.2. Izrada sloja pristupa podacima 15

6.3. Izrada upravljača i sloja poslovne logke 16

6.4. Implementacija JSON Web Token autentikacije 18

7. Angular web i nativna aplikacija 21
7.1. Priprema okruženja za razvoj Angular nativne aplikacije 21
7.2. Kreiranje Angular projekta 21
7.3. Izrada modela Angular aplikacije 22
7.4. Izrada servisa za dohvaćanje i rad s podacima 23
7.5. Izrada komponenti aplikacije 24
7.5.1. Izrada korisničkog sučelja za web 25
7.5.2. Izrada korisničkog sučelja za mobilne uređaje 29
7.5.3. Izrada poslovne logike komponenti 31
8. Zaključak 35
Popis kratica 36
Popis slika 37
Popis tablica 38
Popis kôdova 39
Literatura 40

1. Uvod

„Poremećaj iz spektra autizma je vrlo složen neurorazvojni poremećaj koji zahvaća sve aspekte dječje ličnosti (komunikacija, motorika, ponašanje i učenje). Naime, djeca s autizmom vide, čuju i osjećaju, ali te utiske teško sklapaju u smislenu cjelinu, pa se stoga povlače u vlastiti svijet u kojem nalaze sigurnost. Stoga osobe s autizmom imaju poteškoća u izražavanju svojih osjećaja, želja, potreba, sposobnosti i problema s kojima se svakodnevno bore, a što se odražava na njihovo ponašanje.“[1] Kod učenja, lakše će upamtiti i naučiti određene informacije ukoliko ih preoblikuju u vizualno, ističu se u rješavanju vizualno – spacijalnih i perceptivnih zadataka, te zadataka koji podrazumijevaju sklapanje. I kod određenih aspekata govora, fonologije, rječnika, jednostavne gramatičke strukture, numeričkog računanja mogu pokazati jaku stranu, no unatoč tim aspektima javljaju se teškoće u provedbi razgovora, te upotrebi govora u socijalnom smislu.

Cilj ovog rada je izrada aplikacije kroz koju bi im se omogućila lakoća u komuniciranju i izražavanju svojih osjećaja, želja i potreba preko simbola, teksta i pretvaranja teksta u govor. Svaka riječ predstavljena je slikom i tekstom kako bi osoba lakše mogla razumjeti značenje riječi pa je moguće tako smislenim dodavanjem slika u red stvoriti rečenicu koju ostali mogu čuti, na način da se tekst pretvara u govor. Stvorene rečenice je također moguće slati kao poruke. Sve riječi su kategorizirane kako bi ih bilo lakše pronaći i povezati s drugim sličnim riječima. Aplikacija ima nekoliko kategorija s osnovnim riječima koje su najčešće korištene u govoru, ali omogućuje i dodavanje vlastitih kategorija s vlastitim riječima i slikama koje će se spremati u bazu podataka. Tako da sve ono što žele izraziti, omogućilo bi im se na taj način, te tako uvelike i olakšalo.

Rad se sastoji od osam poglavlja. Uvod kao prvo poglavlje predstavlja objašnjenje osnovnih pojmova o kojima će se govoriti u radu, cilj i svrha rada te tema. Sljedeća poglavlja opisuju autizam i tehnologije koje će se koristiti za izradu aplikacije te samu izradu aplikacije. Zadnje poglavlje sadrži sveukupni zaključak završnog rada.

2. Definicija autističnog poremećaja

Autizam je neurorazvojni poremećaj koji se javlja u ranom djetinjstvu (do 3. godine života) ili pri samom rođenju djeteta. Oštećuje gotovo sve psihičke funkcije, i traje cijeli život. Označava karakteristično promijenjeno ponašanje u svim područjima središnjeg živčanog sustava: socijalnom, intelektualnom, emotivnom, motoričkom i perceptivnom. Definira se simptomima u ponašanju koji uključuju: kvalitativno oštećenje društvenih interakcija, kvalitativni poremećaj komunikacije te ograničeno, ponavljajuće i stereotipno ponašanje, interese i aktivnosti. Intelektualno funkcioniranje djece s autizmom je različito i kreće se od prosječne inteligencije do inteligencije s intelektualnim teškoćama. Na testovima inteligencije postižu lošije rezultate na dijelovima na kojima se ispituje govor i govorne funkcije nego na neverbalnim testovima. Među njima postoje velike individualne razlike: neka su djeca s izrazitim teškoćama funkcioniranja kod koje prevladava niska intelektualna razina, niska emocionalna zrelost, nerazvijen govor, te veća izraženost autističnog poremećaja, dok su druga djeca „visoko funkcionalni autisti“ kod kojih se javljaju znatno visoka funkcionalnost u određenim područjima (kreativnost, matematika, fizika i sl.). Autistični poremećaj sa sobom nosi i neke negativne posljedice. Bez obzira na dob, kod autistične se djece javljaju agresivnost, autoagresivnost i destruktivnost. Uzroci takvog ponašanja mogu biti različiti; to može biti način nerazumijevanja okoline ili način izražavanja vlastite tuge, boli ili očaja, ili jednostavno način izražavanja. Zbog ovakvih oblika ponašanja, koji su teški i za dijete ali i za njegovu okolinu, potrebno ih je prevenirati, koliko god je moguće i okolinu djece s autizmom upoznati o tom stanju i posljedicama istog. Autizam zaustavlja dijete da razvije normalnu komunikaciju i da ostvari uobičajene društvene odnose. Djeca s autizmom vide, čuju, dodiruju, ali te utiske teško mogu sklopiti u smislenu cjelinu, zbog čega se povlače u svoj svijet, u kojem nalaze sigurnost. Zatvaranjem u vlastiti svijet javlja se izoliranost od druge djece i ljudi, te njihova zaokupljenost interesima i opsesivnim radnjama koji se neprestano ponavljaju. U rehabilitaciji djece s autizmom mogu se koristiti psihoterapija, bihevioralna terapija, terapija sredine, grupna terapija, te drugi tretmani i terapije npr. glazbena i likovna terapija, terapija igrom ili kineziterapija.[2]

3. Augmentativna i alternativna komunikacija (AAC)

3.1. Općenito o augmentativnoj i alternativnoj komunikaciji (AAC)

Potpomognuta komunikacija ili augmentativna i alternativna komunikacija integrirana je skupina sastavnica koja uključuje simbole, pomagala, strategije i tehnike koje korisnici rabe s ciljem jačanja komunikacije. Osobe s ozbiljnim govornim ili jezičnim problemima oslanjaju se na augmentativnu i alternativnu komunikaciju (AAC) kako bi dopunili postojeći govor ili zamijenili govor koji nije funkcionalan. Namijenjena je osobama koje se ne mogu služiti govornim jezikom ili imaju teškoće jezičnoga razumijevanja (djeca i osobe s motoričkim teškoćama (cerebralna paraliza, distoni sindrom i sl.), višestrukim teškoćama, intelektualnim teškoćama, komunikacijskim teškoćama i poremećajem iz spektra autizma, oštećenjem vida, oštećenjem sluha, djeca s neurorazvojnim rizikom, djeca s jezičnim i govornim teškoćama.[3]

3.2. Primjena augmentativne i alternativne komunikacije (AAC)

Cilj upotrebe nekog od oblika potpomognute komunikacije je ostvarenje funkcionalne komunikacije, cjelovito sudjelovanje u dobno primjerenim aktivnostima te stjecanje novih znanja i iskustava. Kod nekih osoba ona je samo privremeno sredstvo komunikacije, a kod nekih se koristi kao primarno ili trajno sredstvo komunikacije.[3]

3.3. Učenje i razvijanje vještina komunikacije korištenjem augmentativne i alternativne komunikacije (AAC)

Metode AAC uključuju odabir slikovnih, grafičkih i zvučnih poruka ili znakova iz dostupnog fonda. Prijenos poruka ostvaruje se uporabom navedenih pojedinačnih znakova ili njihovom kombinacijom. Pomagalo AAC-u je svaki uređaj, elektronski ili neelektronski, koji se koristi za odašiljanje ili primanje informacije. Za to se koristi audio-vizuografičkim simbolima. Primjeri: olovka i papir, komunikacijske kartice, komunikacijske ploče, komunikacijske

mape, statički i dinamički komunikatori, elektronička komunikacijska pomagala, uz različite uređaje za prilagođeni pristup istima (prilagođeni miševi, tipkala i ostalo).[4]

Primjer AAC komunikacijskog pomagala prikazan je na slici (Slika 3.1.).

Slika 3.1. Primjer komunikacijske ploče

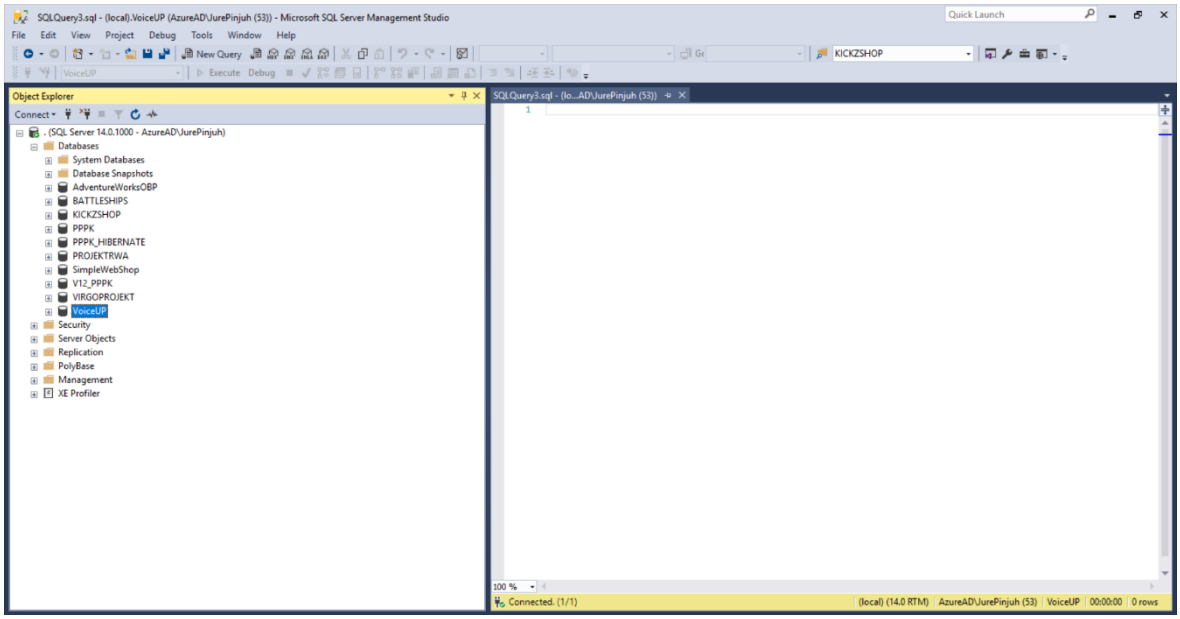
4. Korištene tehnologije

4.1. Microsoft SQL Server

SQL Server je sustav za upravljanje relacijskim bazama podataka razvijen od strane Microsofta koji se prvi put pojavio 1989. godine kao SQL Server 1.0. Zadnja službena verzija je SQL Server 15.0.[5]

Kao sustav za upravljanje relacijskim bazama podataka primarna funkcija mu je spremanje i dohvaćanje podataka. Glavni način za spremanje i dohvaćanje podataka je putem upita koji se pišu određenom vrstom SQL-a zvanom T-SQL koji se na SQL proširuje kako bi obuhvatio proceduralno programiranje te dodaje mogućnost korištenja raznih matematičkih funkcija i datumskih funkcija.

Prilikom razvoja aplikacije korišten je Microsoftov alat Microsoft SQL Management studio, prikazan na slici (Slika 4.1. SQL Server Management Studio), koji kroz grafičko sučelje omogućava konfiguriranje i upravljanje svim komponentama SQL Servera. Korištenjem uređivača skripti SQL Management studia napisane su sve skripte za kreiranje baze podataka, njenih tablica i procedura koje služe za dohvaćanje i obradu podataka.



Slika 4.1. SQL Server Management Studio

4.2. ASP.NET

ASP.NET je programski okvir otvorenog koda koji je dizajniran za web razvoj. Razvojnim inženjerima omogućava razvoj dinamičkih web aplikacija, stranica i servisa.

ASP.NET proširuje .NET programski okvir dodavanjem posebnih alata i biblioteka za web razvoj kao što su:

- Osnovni okvir za obradu web zahtjeva
- Programska sintaksa Razor koja se koristi za izradu dinamičkih web stranica
- Biblioteke za uobičajene web obrasce kao što je MVC
- Sustavi za provjeru autentičnosti.[6]

Web Forms, MVC i Web API su jedni od brojnih programskih modela koje ASP.NET podržava za razvoj web aplikacija.

4.2.1. ASP.NET Web API

ASP.NET Web API programski je okvir koji se koristi za izradu web servisa temeljenih na HTTP protokolu koristeći .NET programski okvir. Za razliku od ostalih web programskih okvira koji koriste .NET programski okvir, Web API kao odgovor na zahtjeve ne vraća poglede, već vraća tekstualne podatke u JSON ili XML formatu. Do sada su objavljene dvije verzije programskog okvira, a najnovija je Web API 2.0 koja je korištena za razvoj web servisa koji konzumiraju Angular klijentske aplikacije.

Putem HTTP protokola klijentska aplikacija šalje zahtjeve koji u tijelu poruke nose podatke koje servis obrađuje i vraća ih nazad klijentskoj aplikaciji u JSON formatu.

4.3. Angular

Angular, poznat kao Angular2+, programski je okvir otvorenog koda koji se koristi za razvijanje jednostraničnih klijentskih web aplikacija. Pisan je u programskom jeziku TypeScript i razvio ga je Google. Prvi put se pojavljuje kao AngularJS 2010. godine kao verzija 1.0 koja je pisana u JavaScript programskom jeziku. Arhitektura AngularJS aplikacija zasnivala se na MVC arhitekturi u kojoj model kao središnja komponenta izražava ponašanje aplikacije, pogled generira izlaz na temelju podataka iz modela, dok kontroler prihvaća ulazne podatke te ih pretvara u naredbe koje šalje modelu i pogledu. U listopadu

2014. godine objavljuje se verzija 2.0 koja se naziva Angular i potpuno mijenja arhitekturu i koncept programskog okvira. Za razliku od AngularJS-a nova verzija ne koristi kontrolere, već se arhitektura aplikacije zasniva na hijerarhiji komponenti koje su direktive s predloškom.[7]

Svaka komponenta sadrži nekoliko datoteka od kojih svaka ima svoju ulogu:

- TypeScript klasa koja sadrži „@Component“ dekorater i obavlja ulogu obrade podataka
- HTML predložak koji određuje što se prikazuje korisniku
- Datoteke za stiliziranje HTML predloška koja mogu biti u CSS, SASS, Stylus ili Mess formatu.

Angular također sadrži servise koji se koriste za obradu podataka koji nisu vezani samo za jednu komponentu. Servisi sadrže dekorater „@Injectable“ koji omogućava da se kod servisa koristi kao *dependency* u klasama i komponentama. Važan dio Angular aplikacija su moduli koji predstavljaju blok koda posvećen jednoj značajki aplikacije. Svaka Angular aplikacija mora sadržavati najmanje jedan modul, a može sadržavati i veći broj modula ovisno o stupnju kompleksnosti aplikacije.

4.4. NativeScript

NativeScript je programski okvir otvorenog koda koji služi za razvoj aplikacija na iOS i Android platformama. Aplikacije se razvijaju JavaScript programskim jezikom ili bilo kojim drugim programskim jezikom koji se prevodi u JavaScript. Jedan od takvih jezika je TypeScript pa tako NativeScript podržava razvoj s programskim okvirima kao što je Angular. Mobilne aplikacije razvijene NativeScript-om rezultiraju nativnim aplikacijama kao što su one razvijene u Xcodeu ili Android studiju. Aplikacije mogu pozivati API-je na iOS i Android platformama koristeći JavaScript virtualne mašine koje procesiraju JavaScript kod na mobilnim uređajima. Takve virtualne mašine su:

- Google V8 za Android
- WebKit JavaScriptCore za iOS.

NativeScript prvi put se pojavljuje 2015. godine s verzijom 1.0, a trenutno dostupna verzija je 6.3.0. koja je korištena u razvoju Angular nativne aplikacije koja se obrađuje u ovom radu.[8]

5. Baza podataka

Baza podataka je zbirka pohranjenih zapisa u računalu na sustavan način tako da joj se računalni program može obratiti prilikom odgovaranja na problem.[9] Poglavlje Baza podataka prikazuje sve tablice, spremljene procedure i ER dijagram baze koja se obrađuje u ovom završnom radu. Baza podataka koja se obrađuje u radu je relacijska baza podataka verzije Microsoft SQL Server 2017.

5.1. Izrada tablica

Tablica je osnovni objekt relacijske baze podataka i u njoj su pohranjeni podaci. Baza podataka koja se obrađuje u radu sadrži ukupno pet tablica od kojih tri tablice predstavljaju entitete, a dvije tablice predstavljaju relacije između entiteta. Sve tablice kreirane su pokretanjem SQL skripte korištenjem SQL Server Management studia.

```
CREATE TABLE CATEGORY (
IDCATEGORY INT PRIMARY KEY IDENTITY NOT NULL,
CATEGORYNAME NVARCHAR(100) NOT NULL,
CATEGORYPICTURELOCATION NVARCHAR(MAX) NOT NULL,
ISGENERAL BIT NOT NULL,
CATEGORYKEY nvarchar(100) NULL,
OWNERID int FOREIGN KEY REFERENCES [USER](IDUSER) NULL
)
```

Kôd 5.1. Kod za kreiranje tablice kategorija

Pokretanjem skripte za kreiranje baze podataka i tablica kreirane su tablice:

- Korisnik (engl. *user*)

Tablica korisnik početni je entitet baze podataka i sadrži osnovne podatke o korisniku koji su potrebni za prijavu kao što su: korisničko ime i lozinka. Tablica sadrži atribut `IDUser` koji je primarni ključ tablice i služi identifikaciji korisnika. Također tablica sadrži ograničenje jedinstvenog ključa na atributu `NICKNAME` kako bi unos više korisnika s istim korisničkim imenom bio onemogućen.

- Kategorija (engl. *category*)

Tablica kategorija ključni je dio baze podataka jer omogućava strukturirano kategoriziranje riječi. Sastoji se od općih podataka o kategoriji kao što su: naziv kategorije, putanja do fotografije na serveru koja označava kategoriju, ključ kategorije koji omogućava dijeljenje kategorije s drugim korisnicima. Atribut `CATEGORYKEY` sadrži ograničenje jedinstvenog ključa kako bi se svaka kategorija razlikovala po ključu koji se dijeli drugim korisnicima. Tablica također sadrži atribut `ISGENERAL` koji je tipa `BIT` i označava je li kategorija generalna ili ju je kreirao netko od korisnika. `BIT` omogućava unos nule ili jedinice pa tako u ovoj bazi podataka nula (0) označava korisničku kategoriju, dok jedinica (1) označava generalnu kategoriju koju mogu koristiti svi korisnici. Atribut `IDCATEGORY` predstavlja primarni ključ tablice. Posljednji atribut u tablici je `USERID` koji je strani ključ i pokazuje na zapis u tablici Korisnik te govori o podatku tko je kreirao kategoriju. Atribut `USERID` može biti `NULL`, što znači da je određena kategorija generalna. Kreiranje ove tablice prikazano je kodom (Kôd 5.1.).

- Riječ (engl. *word*)

Tablica riječ predstavlja riječ kao simbol prikazan fotografijom i samom riječju. Sastoji se od općih podataka o riječi kao što su: tekst riječi, putanja do fotografije na serveru koja označava riječ. Atribut `IDWORD` predstavlja primarni ključ tablice. Tablica također sadrži atribut `CATEGORYID` koji je strani ključ i pokazuje na zapis u tablici Kategorija te govori kojoj kategoriji pripada određena riječ.

- Korisnik – kategorija (engl. *user - category*)

Ova tablica predstavlja relaciju između korisnika i kategorije, odnosno prikazuje koji korisnik može koristiti kategoriju koju je netko drugi kreirao. Atribut `IDUSERCATEGORY` predstavlja primarni ključ tablice, dok atributi `USERID` i `CATEGORYID` predstavljaju strane ključeve koji pokazuju na zapise u tablicama Korisnik i Kategorija.

- Povezane kategorije (engl. *linked categories*)

Tablica povezanih kategorija prikazuje koje kategorije je određeni korisnik povezao kako bi im lakše pristupao. Takve kategorije većinom sadrže riječi koje su povezane i u rečenicama se često pojavljuju jedna za drugom. U tablici postoji atribut `IDLINKEDCATEGORIES` koji predstavlja primarni ključ tablice. Atributi `FIRSTCATEGORY` i `SECONDCATEGORY` predstavljaju strane ključeve koji pokazuju na zapise u tablici Kategorija, dok atribut

USERID predstavlja strani ključ koji pokazuje na tablicu Korisnik i označava korisnika koji je povezao određene kategorije.

5.2. Izrada procedura za rad s podacima

Spremljena procedura pripremljeni je SQL kod koji se može ponovno upotrebljavati. Ako postoji neki SQL upit koji se često koristi, dobro ga je spremiti kao spremljenu proceduru tako da ga ne bi morali pisati iznova. Mogu biti izvršene pozivanjem EXECUTE naredbe iz T-SQL koda, ali mogu biti izvršene i izvan poslužitelja baze pozivanjem procedure iz programskog koda. Korištenje spremljenih procedura pruža velik broj prednosti kao što su:

- Brzo izvršavanje jer su spremljene na poslužitelju baze podataka
- Laka izmjena

Ako postoji potreba da se neka funkcionalnost mijenja, moguće je samo promijeniti kod spremljene procedure bez promjene koda aplikacije.

- Ponovna upotreba koda

Kada se spremljena procedura jednom napiše i spremi u bazu podataka uvijek se može koristiti bez ponovnog pisanja koda.

U bazi podataka koja se obrađuje u ovom radu postoji dvadeset spremljenih procedura pa su tako sve funkcionalnosti aplikacije pokrivene s odgovarajućom procedurom.

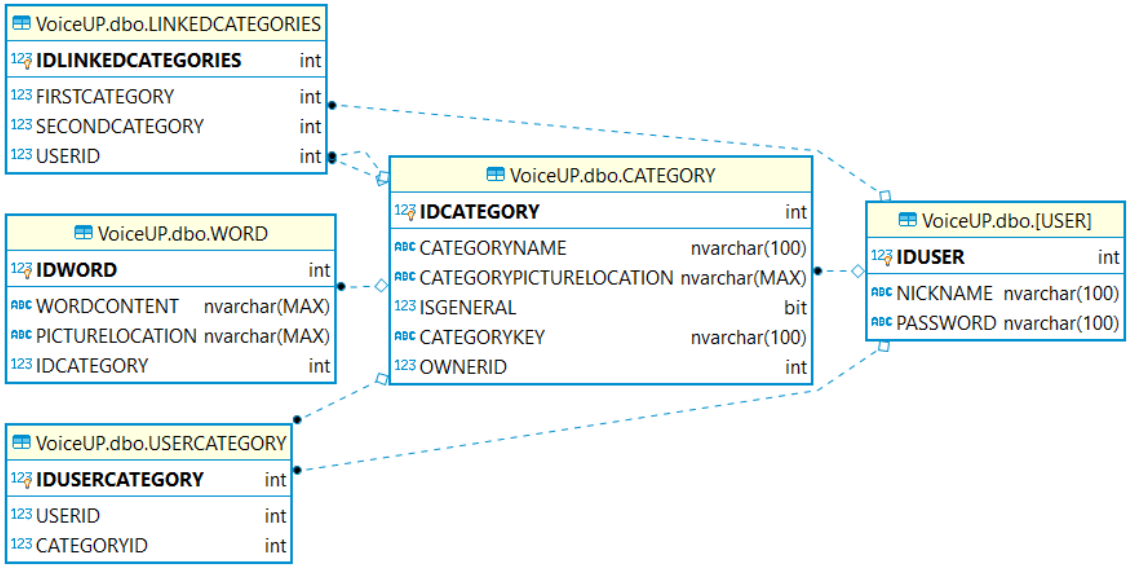
```
CREATE PROCEDURE [dbo].[ADDWORD]
(
@WORDCONTENT NVARCHAR(MAX),
@WORDPICTURE NVARCHAR(MAX),
@CATEGORYID INT,
@IDWORD INT OUTPUT
)
AS
BEGIN
INSERT INTO WORD(WORDCONTENT,PICTURELOCATION,IDCATEGORY)
VALUES(@WORDCONTENT,@WORDPICTURE,@CATEGORYID)
SET @IDWORD=SCOPE_IDENTITY();
END
GO
```

Kôd 5.2. Kod za kreiranje procedure za dodavanje riječi

U kodu (Kôd 5.2.) prikazana je procedura za unos riječi koja kao ulazne parametre prima riječ, putanju do fotografije i jedinstveni ključ kategorije. Nakon uspješnog unosa, procedura vraća novi jedinstveni ključ kao `OUTPUT` parametar.

5.3. ER dijagram baze podataka

ER (engl. *Entity relationship*) dijagram grafički je prikaz osnovni elemenata ER modela. Dijagram grafički prikazuje entitete, njihove atribute i veze, odnosno relacije između entiteta. Veze između entiteta mogu biti unarne, binarne ili n-arne.



Slika 5.1. ER dijagram baze podataka VoiceUP

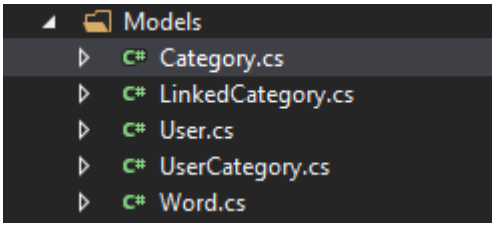
Na slici (Slika 5.1.) se prikazuje ER dijagram baze podataka koja se opisuje u ovom radu sa svim entitetima, njihovim atributima i odnosima između njih.

6. Web Servis

Web servis ili Web API je API kojem se može pristupiti preko HTTP protokola i može biti razvijen u mnoštvu tehnologija kao što su Java ili .NET. Servis koji se obrađuje u ovom završnom radu je .NET Web API 2 servis razvijen po objektno orijentiranim principima. Također, navedeni servis koristi slojevitu arhitekturu u kojoj se mogu razlikovati sloj pristupa podacima i sloj poslovne logike. U ovom poglavlju opisuje se izrada modela web servisa, sloja pristupa podacima, sloja poslovne logike te implementacija JWT autentikacije.

6.1. Izrada modela

Modeli predstavljaju podatke aplikacije. Kreirani su tako da u potpunosti predstavljaju tablice iz baze podataka.



Slika 6.1. Prikaz svih modela web servis

Slika 6.1. prikazuje pet klasa koje predstavljaju modele aplikacije. Prikazane klase služe spremanju stanja modela u memoriju prilikom obrade podataka.

```
public class Category
    {
        public Category()
        {
        }
        public int IDCategory { get; set; }
        public string CategoryName { get; set; }
        public string PictureData { get; set; }
        public string CategoryPictureLocation { get; set; }
        public string CategoryKey { get; set; }
        public bool IsGeneral { get; set; }
        public int OwnerID { get; set; }

    }
```

Kôd 6.1. Prikaz klase kategorija (engl. *category*)

Kôd 6.1. prikazuje klasu kategorija koja je ujedno model web servisa. Dio klase `public Category(){}` predstavlja konstruktor klase koji omogućava inicijalizaciju objekta tipa klase `Category`. Ostatak klase definira svojstva (engl. *properties*) klase koja odgovaraju stupcima tablice kategorija u bazi podataka.

6.2. Izrada sloja pristupa podacima

Sloj pristupa podacima (engl. *Data access layer*) je sloj u arhitekturi aplikacija koji služi pristupu podacima koji su pohranjeni u nekoj vrsti trajne pohrane. Trajna pohrana korištena u ovom radu je baza podataka opisana u poglavlju Baza podataka pa tako sloj pristupa podacima služi komunikaciji s bazom podataka.

Sloj pristupa podacima opisanog web servisa sastoji se od jednog sučelja (engl. *interface*) i dvije klase:

- Sučelje `IRepository`

Sučelje definira sve metode koje će se moći pozivati iz klasa koje će ga implementirati. Sve metode prikazane su u kodu (Kôd 6.2.).

```
public interface IRepository
  {
      int AddCategory(Category category);
      int DeleteCategory(Category category);
      int AddUser(User user);
      int AddWord(Word word);
      int DeleteWord(Word word);
      IEnumerable<Category> GetCategories();
      Category GetCategoryByID(int idCategory);
      Category GetCategoryByKey(string categoryKey);
      IEnumerable<Category> GetGeneralCategories();
      IEnumerable<Word> GetWords(Category category);
      User GetUser(string nickname);
      User GetUser(int idUser);
      IEnumerable<LinkedCategory>
      GetLinkedCategories(LinkedCategory link);
      IEnumerable<UserCategory>GetUserCategories(User
      user);
      int LinkCategories(LinkedCategory linked);
      int LinkUserCategory(UserCategory uc);
```

```
        int UpdateWord(Word word);
        int UpdateCategory(Category category);
        Word GetWord(int idWord);
        int UnlinkUserCategory(UserCategory uc);
    }
```

Kôd 6.2. Prikaz sučelja `IRepository`

- Klasa `SQLRepo`

Ova klasa služi pristupu podacima te implementira sučelje `IRepository`, što znači da implementira sve metode definirane u sučelju. Sadrži svojstvo `connectionString` tipa `string` koje dohvaća iz konfiguracijske datoteke web servisa gdje je definirano. Navedeno svojstvo se prosljeđuje kao parametar konstruktora klase `SqlConnection` koja predstavlja vezu prema bazi podataka. Za pristup podacima koristi jednostavno objektno-relacijsko mapiranje Dapper koje automatski pretvara tablice u C# objekte kada se nazivi svojstva klase podudaraju s nazivima stupaca tablice.

- Klasa `RepoFactory`

Ova klasa sadrži statičku metodu `GetRepo` koja vraća instancu klase koja implementira sučelje `IRepository`.

6.3. Izrada upravljača i sloja poslovne logke

Upravljač (engl. *controller*) je klasa koji obrađuje HTTP zahtjeve. Svaka klasa koja predstavlja upravljač mora završavati s ključnom riječi „Controller“ i mora nasljeđivati `ApiController` klasu. Metode definirane u upravljaču nazivaju se akcijske metode kojima je pridružen URI i vrsta HTTP metode pa tako na temelju URL-a i HTTP metode web servis odlučuje koju će akcijsku metodu izvršiti.

Web servis opisan u ovom završnom radu ima tri upravljača:

- `CategoryController` - upravljač koji obrađuje HTTP zahtjeve koji se odnose na kategorije
- `WordController` - upravljač koji obrađuje HTTP zahtjeve koji se odnose na riječi
- `UserController` - upravljač koji obrađuje HTTP zahtjeve koji se odnose na korisnika

```
[Route("login")]
    [HttpPost]
    public IHttpActionResult LoginUser(User sentUser)
    {
        User user = BLL.GetUser(sentUser.Nickname);
        if (user != null)
        {
           if (PasswordHelper.Verify(sentUser.Password,
                user.Password))
            {
                    string token =
                    JWTHelper.CreateToken(sentUser.Nickna
                    me,Request);
                    return Ok(token);
            }
            else {
                  return Unauthorized();
            }
        }
        else {
              return Unauthorized();
        }
    }
```

Kôd 6.3. Akcijska metoda za prijavu korisnika na `UserController` upravljaču

U prikazanom kodu (Kôd 6.3.) prikazana je akcijska metoda za prijavu korisnika kojoj je definirana putanja atributom `[Route]` i HTTP *post* metoda atributom `[HttpPost]`. Metoda kao parametar prima instancu klase `User` i provjerava postoji li korisnik u bazi podataka. Ako korisnik postoji, provjerava točnost lozinke i ako je lozinka točna, vraća JWT. Ako korisnik ne postoji ili lozinka nije točna, vraća statusni kod 401 - *Unauthorized*.

Sloj poslovne logike je sloj koji služi kao posrednik u razmjeni podataka između upravljača i sloja pristupa podacima. U web servisu koji se opisuje u ovom radu sloj poslovne logike predstavlja statična klasa `BLL` koja na sebi ima instancu klase koja implementira sučelje `IRepository`. Metode klase `BLL` pozivaju metode koje dohvaćaju podatke iz baze podataka i te podatke pripremaju za slanje preko HTTP protokola. Uz pripremu podataka definira i implementira poslovna pravila aplikacije.

```
public static int DeleteCategory(Category category) {
       Category cat=
```

```
            GetCategoryByID(category.IDCategory);
            if (cat != null)
            {
                if (cat.OwnerID == category.OwnerID)
                {
                   If
                  (File.Exists(cat.CategoryPictureLocation))
                    {
                   File.Delete(cat.CategoryPictureLocation);
                    }
                    return DeleteCategory(category);
                }
                else
                {
                    return UnLinkUserCategory(new
                    UserCategory {Category=new
                    Category{IDCategory= cat.IDCategory},
                    User = new User { IDUser=category.OwnerID
                  } });
                }
            }
            return 0;
        }
```

Kôd 6.4. Metoda `DeleteCategory` na `BLL` klasi

U kodu (Kôd 6.4.) prikazana je metoda koja služi brisanju kategorije. Metoda provjerava je li korisnik koji želi izbrisati kategoriju onaj korisnik koji ju je dodao. Ako je to taj korisnik kategorija se briše, a u suprotnom se briše redak iz tablice `USERCATEGORY` koji omogućava korisniku da koristi tu kategoriju.

6.4. Implementacija JSON Web Token autentikacije

JWT (JSON Web Token) sigurnosni je token koji se koristi za autentikaciju kod RESTful web servisa. Služi kao spremnik za tvrdnje o korisniku i sastoji se od tri dijela koji su odvojeni točkom. Prvi dio tokena je zaglavlje koje sadrži informacije o tipu tokena i informacije o algoritmu koji se koristi za potpisivanje. Drugi dio tokena sadrži tvrdnje o korisniku, dok treći dio sadrži potpis tokena koji služi provjeri pouzdanosti tokena.[10]

Web servis koji se obrađuje u ovom radu implementira JWT autentikaciju kroz dvije klase.

```
public static class JWTHelper
    {
        public static string CreateToken(string
        nickname,HttpRequestMessage request)
        {
            DateTime starts = DateTime.UtcNow;
            DateTime expires = DateTime.UtcNow.AddMonths(1);
            var Handler = new JwtSecurityTokenHandler();
            var claim = new ClaimsIdentity(new[]
            {
                new Claim(ClaimTypes.Email, nickname)
            });
            const string secrectKey=
          "9f3ba9d41618d246665d5bb00e6e20cd208e5077eab0c039";
            var securityKey = new
            SymmetricSecurityKey(System.Text.Encoding.Default
            .GetBytes(secrectKey));
            var credentials = new
            SigningCredentials(securityKey,
            SecurityAlgorithms.HmacSha256Signature);
            var token =
                (JwtSecurityToken)
                Handler.CreateJwtSecurityToken(
                    issuer: request.RequestUri
                    .GetLeftPart(UriPartial.Authority),
                    audience: request.RequestUri
                    .GetLeftPart(UriPartial.Authority),
                    subject: claim,
                    notBefore: starts,
                    expires: expires,
                    signingCredentials: credentials);

            var tokenString = Handler.WriteToken(token);
            return tokenString;
        }
    }
```

Kôd 6.5. Statična klasa `JWTHelper`

Kôd 6.5 prikazuje statičnu klasu `JWTHelper` koja služi za kreiranje web tokena nakon provjere autentičnosti korisničkog imena i lozinke. Metoda kao parametre prima korisničko

ime i HTTP zahtjev te radi novu instancu klase `JwtSecurityToken` koju u stringu vraća kao povratni parametar. Nakon toga se token sprema na klijentskoj aplikaciji i šalje svakim novim zahtjevom na server. Druga klasa je `TokenValidationHandler` koja pri svakom zahtjevu na serveru provjerava zaglavlje zahtjeva i ako token ne postoji vraća odgovor s HTTP statusnim kodom 401. Ako token postoji, provjerava mu autentičnost istim mehanizmom kako se token kreira uz privatni ključ kojim je stvoren. Kako bi se kod klase `TokenValidationHandler` izvršio nakon svakog zahtjeva, potrebno je instancu klase dodati kao `MessageHandler` u metodu `Register` klase `WebApiConfig`.

```
config.MessageHandlers.Add(new TokenValidationHandler());
```

7. Angular web i nativna aplikacija

U ovom poglavlju opisane su Angular web i nativna aplikacija koje su izrađene kao dio ovoga završnog rada. Kod obje aplikacije razvijan je u istom projektu metodom dijeljenja koda koja omogućava dijeljenje koda poslovne logike između web, Android i iOS platformi, ali omogućava i dodavanje koda koji je specifičan za određenu platformu.

7.1. Priprema okruženja za razvoj Angular nativne aplikacije

Angular nativna aplikacija koja se obrađuje u ovom radu razvijana je u Windows 10 okruženju pa tako se u daljnjem tekstu opisuje priprema okruženja Windows 10 za razvoj Angular nativne aplikacije.

Prva stavka potrebna za razvoj je Node.js koji predstavlja JavaScript runtime okruženje. Node.js uključuje lagani web server na kojem se lokalno mogu pokrenuti Angular aplikacije te uključuje npm package manager koji omogućava instalaciju Angular CLI i NativeScript CLI programskih paketa preko konzole izvršavanjem sljedećih naredbi:

```
npm install -g @angular/cli
npm install -g nativescript
```

Nakon instalacije Angular i NativeScript CLI programskih paketa potrebno je instalirati NativeScript Schematics koji omogućava kreiranje Angular projekta koji dijeli kod između mobilne i web aplikacije.

```
npm i -g @nativescript/schematics
```

7.2. Kreiranje Angular projekta

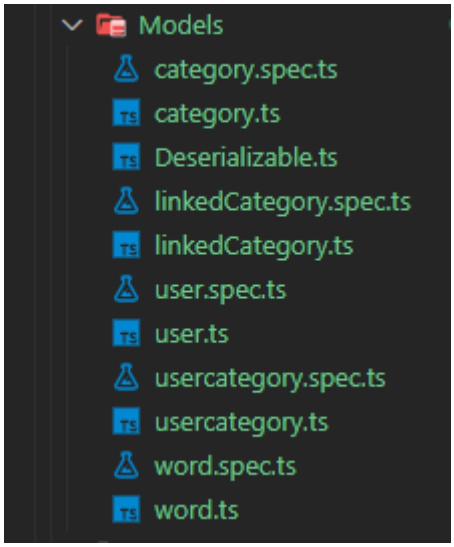
Ispunjavanjem zahtjeva navedenih u prethodnom podnaslovu moguće je kreirati Angular aplikaciju s dijeljenjem koda u dva jednostavna koraka korištenjem konzole. Prvi korak kreiranja projekta pozicioniranje je u željeni direktorij u kojem će se nalaziti projekt. Nakon pozicioniranja u željeni direktorij potrebno je izvršiti sljedeću naredbu:

```
ng new --collection=@nativescript/schematics project-name --
shared
```

U navedenoj naredbi potrebno je `project-name` zamijeniti sa željenim imenom projekta. Nakon izvršavanja naredbe u željenom direktorij biti će stvoren novi direktorij sa svim potrebnim datotekama bitnim za razvoj i inicijalno pokretanje projekta. Kako se radi o projektu koji dijeli kod između mobilne aplikacije za iOS i Android i web aplikacije, moguće ga je pokrenuti na tri načina koja su specifična za navedene platforme. Prvi način je pokretanje web aplikacije izvršavanjem naredbe `ng serve`, dok je drugi način pokretanje aplikacije za Android platforme izvršavanjem naredbe `tns run android`. Treći način je pokretanje aplikacije za iOS izvršavanjem naredbe `tns run ios` koju nije moguće izvršiti na Windows operacijskom sustavu.

7.3. Izrada modela Angular aplikacije

Kao što je navedeno u podnaslovu Izrada modela, modeli predstavljaju podatke aplikacije i kreirani su tako da prikazuju tablice iz baze podataka.



Slika 7.1. Prikaz modela Angular aplikacije

Slika 7.1. prikazuje pet modela Angular aplikacije: category.ts, linkedCategory.ts, user.ts, usercategory.ts i word.ts koje su klase uz pet datoteka koje imaju sufiks spec.ts. Datoteke sa sufiksom spec.ts su datoteke koje generira Angular CLI prilikom izrade nove klase i služe testiranju aplikacije. Datoteka Deserializable.ts sadrži sučelje `Deserializable`, prikazano kodom (Kôd 7.1.), kojeg implementira svaka od klasa kako bi JSON mogle deserijalizirati u objekt.

```
export interface Deserializable {
    deserialize(input: any): this;
```

```
  }
```

Kôd 7.1. Prikaz sučelja `Deserializable`

```
import {Deserializable} from './Deserializable';
export class Word implements Deserializable {

    public IDWord: number;
    public WordContent: string;
    public PictureLocation: string;
    public PictureData: string;
    public IDCategory: number;

   deserialize(input: any): this {
    Object.assign(this, input);
    return this;
    }
}
```

Kôd 7.2. Prikaz klase riječ (engl. *word*)

Kôd 7.2. prikazuje klasu `Word` sa svojim svojstvima i implementacijom sučelja `Deserializable`. Klasa služi čuvanju podataka o riječima koje se dohvaćaju s web servisa.

7.4. Izrada servisa za dohvaćanje i rad s podacima

Angular servisi su klase koje služe dohvaćanju podataka s web servisa, odnosno služe komunikaciji klijentske aplikacije sa servisom. Ukrašene su dekoraterom „@Injectable“ koji omogućava korištenje koda servisa u komponentama bez stvaranja novog objekta.

Angular aplikacija obrađena u ovom radu ima četiri servisne klase od kojih tri služe komunikaciji s web servisom, dok posljednja servisna klasa služi zaštiti ruta aplikacije. Broj servisnih klasa za komunikaciju s web servisom odgovara broju kontrolera na web servisu pa tako i broj metoda u servisima odgovara broju metoda na određenom kontroleru. Na taj način svaka metoda u servisnoj klasi poziva određenu metodu na upravljaču web servisa.

Servisne klase koje služe komunikaciji s web servisom su:

- `CategoryService`: šalje zahtjeve na `CategoryController` upravljač web servisa
- `WordService`: šalje zahtjeve na `WordController` upravljač web servisa
- `UserService`: šalje zahtjeve na `UserController` upravljač web servisa.

```
public getGeneralCategories(): Observable<Category[]> {
  const token = localStorage.getItem('jwt');
  return this.httpService.get<Category[]>(this.root +
  `/category/getgeneral`, {
    headers: new HttpHeaders({
      'Authorization': 'Bearer ' + token,
      'Content-Type': 'application/json'
    })
  }).pipe(
    map(data => data.map(item => new
    Category().deserialize(item)))
  );
}
```

Kôd 7.3. Metoda dohvaćanja generalnih kategorija

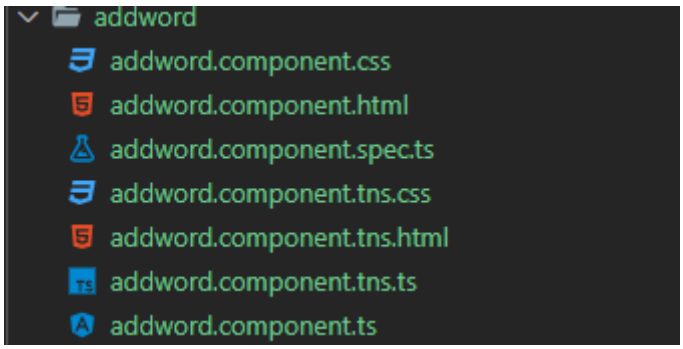
Kôd 7.3. prikazuje metodu za dohvaćanje generalnih kategorija u aplikaciji. Metoda koristi `httpService` kako bi poslala GET zahtjev na definirani URL s autorizacijskim tokenom u zaglavlju. Map dio zahtjeva služi deserijalizaciji svake stavke u JSON nizu objekata.

Servisna klasa koja služi zaštiti ruta aplikacije je `AuthGuard` servisna klasa. `AuthGuard` implementira sučelje `CanActivate` koje definira metodu `canActivate` i omogućava da se servisna klasa koristi kao zaštitnik ruta. Prilikom svakog pokušaja aktivacije određene rute servisna klasa pokušava dohvatiti JSON web token. Ako JWT ne postoji metoda vraća `false`, što znači da ruta ne može biti aktivirana, a ako JWT postoji, provjerava mu se ispravnost i dužina trajanja. U slučaju ispravnosti JWT-a metoda vraća `true` i ruta se može aktivirati, dok u suprotnom metoda vraća `false` i ruta se ne može aktivirati.

7.5. Izrada komponenti aplikacije

Komponenta je glavni gradivni oblik Angular aplikacije.[11] Sastoji se od skriptne datoteke pisane u TypeScript programskom jeziku, HTML predloška koji definira što se korisniku prikazuje i datoteke za stiliziranje HTML predloška. Kako su obje aplikacije razvijene kroz jedan projekt, komponente sadrže set navedenih datoteka. Datoteke koje u svom nazivu

sadrže ključnu riječ „tns“ su datoteke koje se odnose na dijelove komponente za mobilne uređaje, dok ostale predstavljaju dijelove komponente za web.



Slika 7.2. Prikaz datoteka komponente za dodavanje riječi

Komponenta prikazana na slici (Slika 7.2.) sadrži dvije TypeScript datoteke, što znači da kod poslovne logike nije dijeljen kod ove komponente, već odvojen. Kod je odvojen zbog potrebe pozivanja API-a za učitavanje fotografija na iOS i Android platformama što nije moguće kompilirati za web. Ako ne postoji potreba za korištenjem koda specifičnog za određenu platformu dovoljna je jedna TypeScript datoteka koja predstavlja upravljač za iOS, Android i web.

U tablici (Tablica 7.1.) prikazuje se popis svih komponenti web i mobilne aplikacije.

Tablica 7.1. Popis komponenti aplikacije

addcategoryComponent	homeComponent
addwordComponent	wordsComponent
editcategoryComponent	loginComponent
editwordComponent	registerComponent

7.5.1. Izrada korisničkog sučelja za web

Korisničko sučelje Angular aplikacija predstavlja HTML datoteka koja se nalazi u komponenti pa tako za svaki ekran aplikacije postoji jedna komponenta. Tijekom izrade korisničkog sučelja korišten je Bootstrap kako bi aplikacija bila responzivna, što znači da se aplikacija skalira ovisno o veličini uređaja na kojemu se prikazuje. Također je korištena Angular Material biblioteka komponenata kako bi web aplikacija i mobilna aplikacija bile

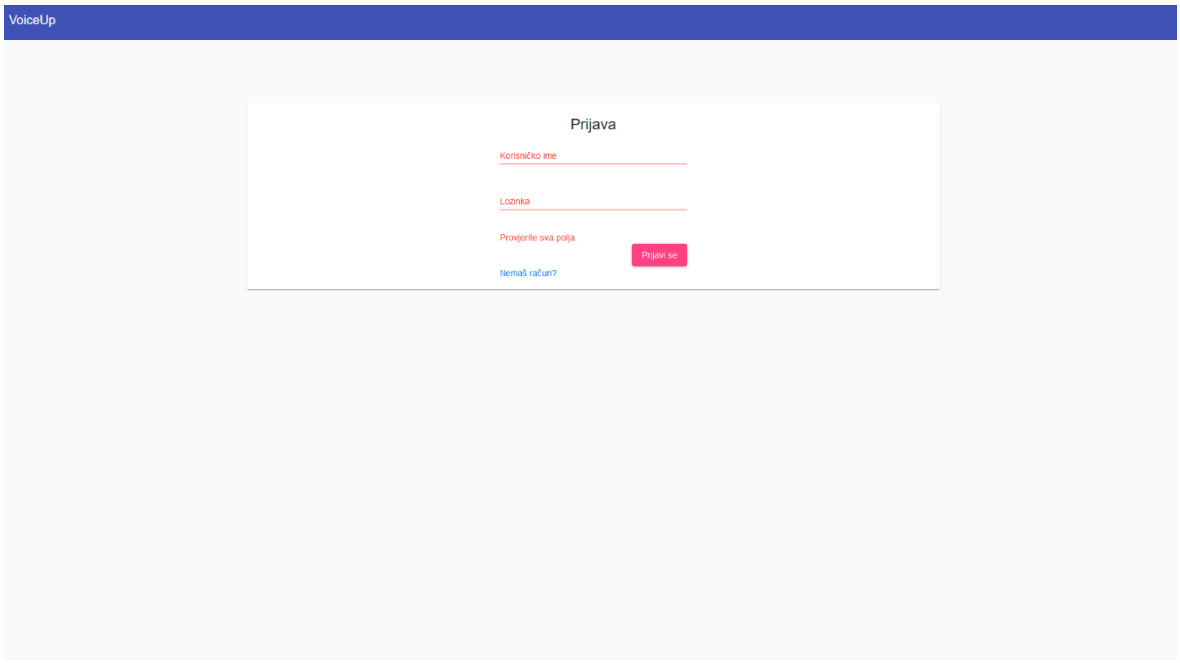
što sličnije. U daljnjem tekstu opisuje se kako Angular proširuje HTML sintaksu i kroz slike se prikazuje nekoliko glavnih komponenta aplikacije.

Angular proširuje sintaksu HTML-a dodavanjem komponenti i direktiva koje se pojavljuju kao novi elementi i atributi.

```
<mat-card class="scrolling-wrapper col-11">
  <mat-card class="card" *ngFor="let word of words"
  style="width: 9rem;height: 9rem;">
      <img mat-card-image style="width:9rem;height:9rem;"
       src="{{word.PictureData}}"
                 class="mat-card-image" />
       <mat-card-subtitle class="mat-card-subtitle text
       center"><b>{{word.WordContent}}</b></mat-card
       subtitle>
   </mat-card>
 </mat-card>
```

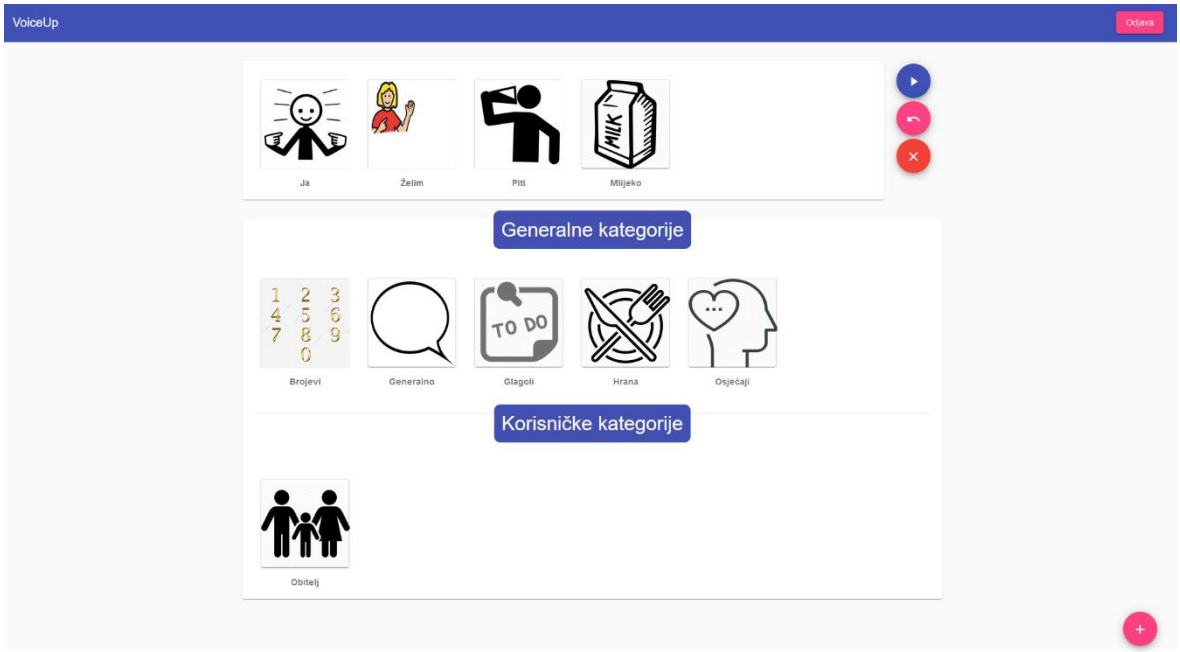
Kôd 7.4. HTML kod koji predstavlja riječ na početnoj komponenti (`homeComponent`)

U kodu (Kôd 7.4.) mogu se vidjeti neka od proširenja koje Angular dodaje na HTML. Prvo proširenje je direktiva `*ngFor` koja omogućava iteraciju po nizu podataka. Primjer direktive `*ngFor` u kodu predstavlja `*ngFor="let word of words"` koja omogućava iteraciju po nizu riječi koji je definiran u upravljaču komponente. Za svaku riječ koja se nalazi u nizu riječi dodat će definirani kod u HTML. Iduće proširenje je `<mat-card>` koje predstavlja komponentu iz biblioteke Angular Material. Posljednje proširenje vidljivo u kodu naziva se interpolacija. Interpolacija omogućuje dodavanje izraza u označeni tekst pa se tako vrijednosti varijabli definiranih u upravljaču komponente ili u određenim direktivama mogu lagano prikazivati. Primjer interpolacije u kodu predstavlja `{{word.WordContent}}` pri čemu se na ekranu prikazuje vrijednost `WordContent` varijable `word` definirane u direktivi `*ngFor` koja iterira kroz niz riječi. Direktiva koja nije prikazana u kodu je `*ngIf` koja omogućava prikaz HTML koda ovisno istinitosti izraza koji prima.



Slika 7.3. Komponenta prijave u sustav

Slika 7.3. prikazuje korisničko sučelje komponente preko koje se korisnici mogu prijaviti u sustav unosom korisničkog imena i lozinke. Također se prikazuje i validacija forme koja je sastavni dio ove komponente. Polja za unos su crvena jer nisu unesene vrijednosti u obavezna polja forme te je prikazana obavijest o grešci.



Slika 7.4. Početna komponenta aplikacije (`homeComponent`)

Na slici (Slika 7.4.) prikazano je korisničko sučelje početne komponente na koju aplikacija preusmjerava nakon uspješne prijave u sustav. Komponenta se sastoji od 3 glavna dijela:

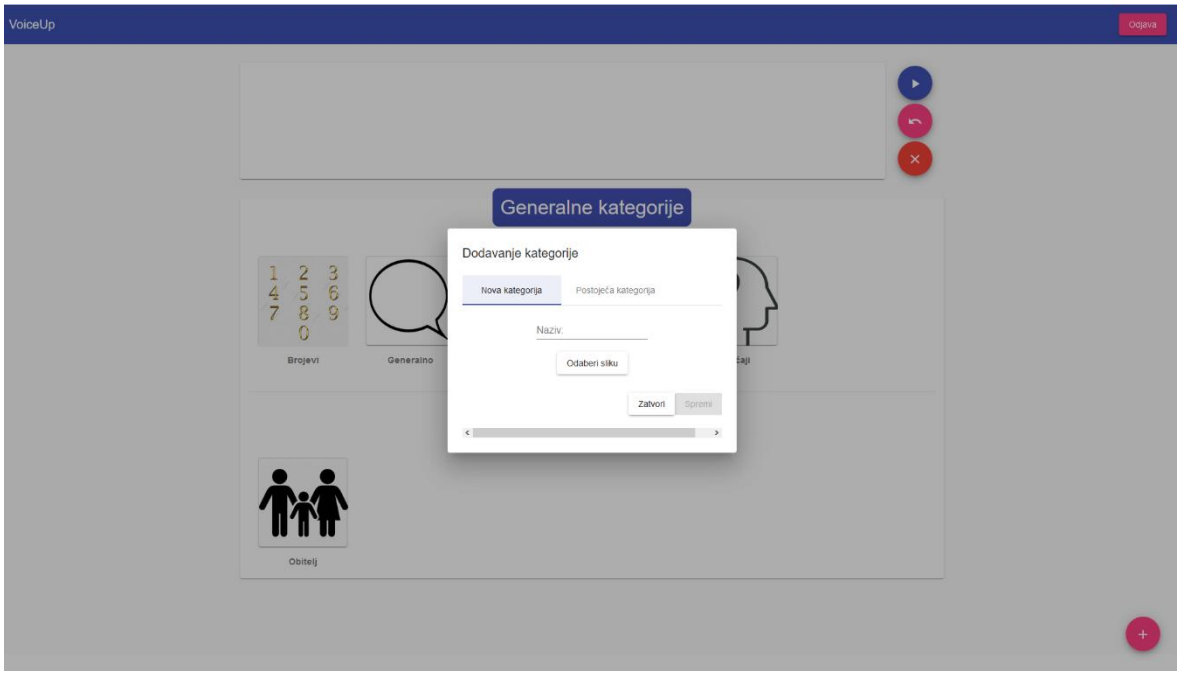
- Početni dio u kojem se prikazuju riječi od kojih se sastavlja rečenica

Ovaj dio sadrži riječi koje je korisnik dodao u red kako bi stvorio rečenicu. Pored ovog dijela nalaze se tri gumba od kojih svaki ima svoju funkciju na komponenti. Nakon pritiska plavog gumba sastavljena rečenica pretvara se u govor, a preostali gumbi služe za brisanje jedne riječi ili čitave rečenice.

- Dio koji sadrži generalne kategorije koje se prikazuju svim korisnicima
- Dio koji sadrži kategorije koje je dodao prijavljeni korisnik

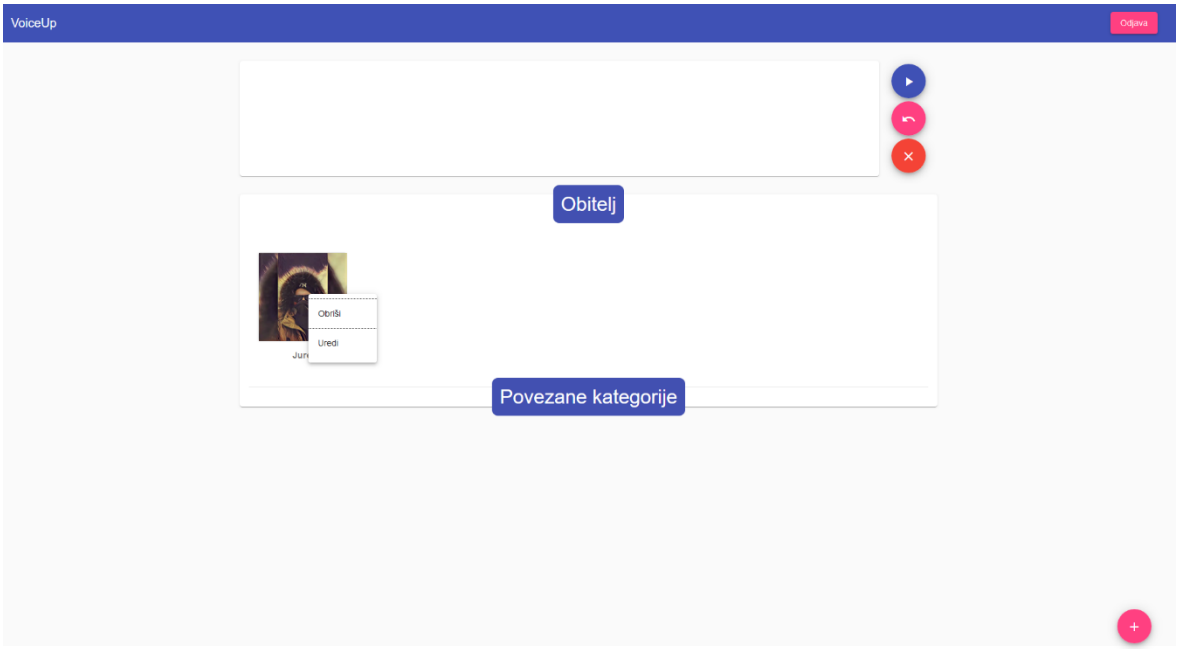
Pritiskom desne tipke miša na fotografiju kategorije otvara se izbornik preko kojeg se kategorije mogu urediti ili izbrisati te se može dijeliti jedinstveni ključ kategorije.

Pritiskom gumba u donjem desnom kutu početne komponente otvara se komponenta za dodavanje kategorije.



Slika 7.5. Komponenta dodavanja kategorije

Preko komponente prikazane na slici (Slika 7.5.) kategorija se može dodati kao nova ili kao postojeća unosom jedinstvenog ključa kategorije koji se može dobiti od korisnika koji je dodao kategoriju.



Slika 7.6. Komponenta za prikaz riječi određene kategorije

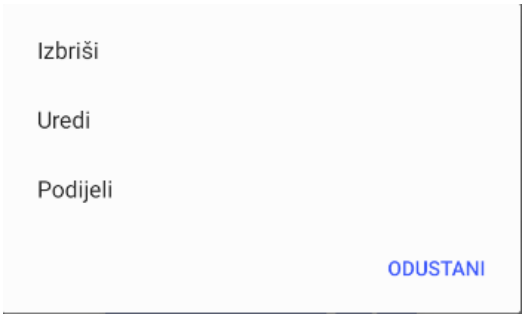
Pritiskom lijeve tipke miša na fotografiju određene kategorije otvara se komponenta prikazana na slici (Slika 7.6.). Na ovoj komponenti nalaze se sve riječi kategorije i ostale kategorije koje je korisnik povezao s odabranom kategorijom. Željena riječ se dodaje u rečenicu pritiskom lijeve tipke miša, dok se pritiskom desne tipke miša otvara izbornik preko kojega je moguće obrisati riječ ili je urediti. Pritiskom na gumb u donjem desnom kutu komponente otvara se komponenta za dodavanje riječi. Uređivanje, brisanje i dodavanje riječi moguće je jedino ako je korisnik kreirao kategoriju u kojoj se riječ nalazi.

7.5.2. Izrada korisničkog sučelja za mobilne uređaje

Korisničko sučelje Angular mobilne aplikacije predstavlja HTML datoteka u komponenti s ključnom riječi „tns“. Iako je datoteka „html“ ekstenzije, sastoji se samo od XML oznaka koje su specifične NativeScript programskom okviru. Nastavno tomu, slaganjem različitih layouta i komponenti koje se definiraju XML oznakama razvija se korisničko sučelje. Uz XML oznake specifične za NativeScript, mogu se koristiti direktive iz Angulara kao što su `*ngIf` i `*ngFor` te interpolacija. U nastavku ovog poglavlja prikazuje se korisničko sučelje nekoliko glavnih komponenti mobilne aplikacije.

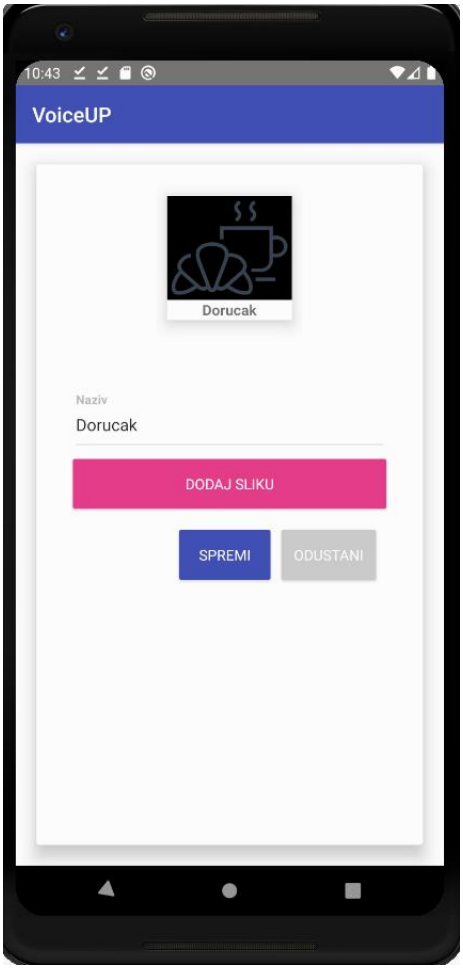
Slika 7.7. Prikaz početne komponente mobilne aplikacije

Slika 7.7. prikazuje korisničko sučelje `homeComponent` komponente na mobilnoj aplikaciji pa tako sadrži iste funkcionalnosti kao korisničko sučelje web aplikacije prikazano na slici (Slika 7.4.). Mobilna aplikacija dodaje novu funkcionalnost dijeljenja sastavljene riječi na društvenim mrežama ili porukom pritiskom gumba u gornjem desnom kutu. Pritiskom na fotografiju kategorije otvara se komponenta za prikaz riječi kategorije, dok se dvostrukim pritiskom prikazuje izbornik preko kojega se kategorija može izbrisati ili urediti te se može dijeliti jedinstveni ključ kategorije.



Slika 7.8. Izbornik za rad s kategorijama

Pritiskom opcije „Uredi“ na izborniku prikazanom na slici (Slika 7.8.) otvara se komponenta, prikazana na slici (Slika 7.9.), koja omogućava uređivanje kategorije s pregledom kako će kategorija izgledati na ostalim komponentama.



Slika 7.9. Prikaz komponente za uređivanje kategorije mobilne aplikacije

Slika 3.1

7.5.3. Izrada poslovne logike komponenti

Poslovna logika komponenti nalazi se u datotekama s ekstenzijom „ts“, što znači da su pisane TypeScript programskim jezikom. Svaka komponenta u projektu s dijeljenjem koda mora sadržavati barem jednu TypeScript datoteku. Moguće je dodavanje TypeScript datoteke s ključnom riječi „tns“ koja predstavlja upravljač komponente za iOS i Android platforme, ali je moguće i dodavanje dviju datoteka od kojih svaka predstavlja upravljač za određenu platformu. Projekt kojim se bavi ovaj radi sadrži dvije komponente koje imaju samo jednu datoteku kao upravljač, dok ostale komponente sadrže dvije datoteke kao upravljač. To su

komponente `loginComponent` i `registerComponent` na kojima nije bilo potrebe pozivanja API-ja na iOS i Android platformama.

Svaka TypeScript klasa uređena je dekoraterom „@Component“ te sadrži metapodatke koji govore Angularu kako procesirati određenu klasu. Također, svaka klasa mora sadržavati konstruktor koji se poziva svaki put kad je klasa instancirana. Dobra praksa kod izrade Angular aplikacija je implementacija sučelja `OnInit` koje predstavlja jedan od dijelova životnog ciklusa komponente. `OnInit` se poziva nakon što se klasa inicijalizira i najčešće se u njoj dohvaćaju podaci koji su potrebni za prikaz komponente kao što je prikazano u kodu (Kôd 7.5.).

```
ngOnInit() {
  this.words = this.wordService.GetWordsFromPlaylist();

    this.categoryService.getGeneralCategories().subscribe(d
    ata => {
    this.generalCategories = data;
        });

    this.userService.getUser(this.nickname).subscribe(data
    => {
    this.id = data.IDUser;
    if (this.id > 0) {

          this.categoryService.getCategoriesForUser(this.id
          ).subscribe(response => {
           this.userCategories = response;
           this.loaded = true;
            });
          }
        });
      }
```

Kôd 7.5. Implementacija metode `ngOnInit` na početnoj komponenti

Glavna funkcija aplikacije je omogućavanje komunikacije osobama s autizmom pa tako se u daljnjem tekstu prikazuje kod koji im to omogućava.

Aplikacija poziva API za pretvaranje teksta u govor kako bi stvorena rečenica bila pretvorena u govor. Web aplikacija koristi ResponsiveVoice.js, dok mobilna aplikacija koristi *text-to-speech* API ugrađen na mobilnim uređajima. Kako bi bilo moguće koristiti

ResponsiveVoice.js biblioteku potrebno ju je dodati u projekt kao skriptu u index.html datoteci. Nakon dodavanja u index.html potrebno je deklarirati varijablu `responsiveVoice` u komponenti u kojoj se želi koristiti biblioteka.

```
declare let responsiveVoice: any;
```

Kada se omogući korištenje ResponsiveVoice.js biblioteke potrebno je samo pozvati metodu `speak` na varijabli `responsiveVoice` i kao parametre joj proslijediti tekst koji se želi pretvoriti u govor i jedan od jezika koji su dostupni. Moguće je dodati opcionalne parametre kao što su brzina govora i visina tona kao što je prikazano u kodu (Kôd 7.6.).

```
play() {
    let sentence = '';
    this.words.forEach(element => {
      sentence = sentence + ' ' + element.WordContent;
    });
    if (sentence !== '') {
        responsiveVoice.speak(sentence, 'Croatian Male',
        { pitch: 1.5, rate: 0.1 });
    }
  }
```

Kôd 7.6. Metoda `play` na početnoj komponenti web aplikacije

Za pretvaranja teksta u govor na mobilnoj aplikaciji korišten je NativeScript-TextToSpeech dodatak koji koristi nativne kontrole na mobilnim uređajima. Kako bi bilo moguće koristiti NativeScript-TextToSpeech potrebno ga je dodati u projekt preko konzole pokretanjem komande:

```
tns plugin add nativescript-texttospeech
```

Nakon toga potrebno je importati klase `TNSTextToSpeech` i `SpeakOptions` u komponentu u kojoj se koristi te instancirati objekte klase `TNSTextToSpeech`. `SpeakOptions` je klasa koja služi postavljanju konfiguracije TTS-a pa tako sadrži tekst koji će se pretvarati u govor te ostale opcionalne parametre kao što su brzina govora, glasnoća i jezik. Pretvaranje teksta u govor ostvaruje se pozivanjem metode `speak` na instanci klase `TNSTextToSpeech` koja kao parametar prima instancu klase `SpeakOptions` kao što je prikazanu u kodu (Kôd 7.7.).

```
play() {
  let sentence = '';
```

```
    this.words.forEach(element => {
      sentence = sentence + ' ' + element.WordContent;
    });
    let TTS = new TNSTextToSpeech();
    let speakOptions: SpeakOptions = {
      text: sentence,
      speakRate: 0.1,
      pitch: 1.0,
      volume: 1.0,
      locale: 'hr-HR',
    };
    TTS.speak(speakOptions).then(
      () => {
      },
      err => {
      }
    );
  }
```

Kôd 7.7. Metoda `play` početne komponente mobilne aplikacije

8. Zaključak

Većina današnjih aplikacija počela je sadržajem namijenjenim za zabavu, a tijekom godina, razvoj mobilnih aplikacija ovisio je o potrebama, tehnologiji, brzini prijenosa, sadržaju i tematici te se tako sve više uz zabavu uključuje i koristan sadržaj. Tako se može zaključiti da su mobilne aplikacije trend koji će se konstantno razvijati, a korisnicima, kao što je uvijek i cilj, nudi se nešto novo te im se to omogućuje na pristupačan i zabavan način. U ovom radu se na takav način nešto novo također omogućuje i onima koji imaju poteškoća u izražavanju svojih osjećaja, želja, potreba i problema.

Autizam je cjeloživotna invalidnost koja utječe na način na koji osoba komunicira i odnosi se na druge ljude i svijet oko njih. Aplikacija je zamišljena kao alat koji će roditelj ili rehabilitator koristiti da bi dobio veću pozornost djeteta i samim time povećao učinkovitost komuniciranja i izražavanja osjećaja preko simbola, teksta i pretvaranja teksta u govor. Učinkovito se komunicira kroz izgradnju rečenica uz relevantne slike u obliku rečenice te se osigurava interakcija i međusobno razumijevanje korisnika sa slušateljem.

S obzirom da je poznato da danas djeca imaju veliko zanimanje za pametne uređaje, trebalo bi se više inzistirati na razvoju i implementaciji ICT tehnologija u odgojno-obrazovne ustanove te tako roditeljima i rehabilitatorima olakšati bolje komuniciranje s djecom s autizmom. Ovim radom prikazuje se kako se korištenjem modernih tehnologija može razviti sustav koji ima potencijala da rezultira kvalitetnijom interakcijom i razumijevanjem između djeteta i roditelja ili rehabilitatora.

Popis kratica

AAC	*Alternative and augmentative communication*	alternativna i augmentativna komunikacija
API	*Application Programming Interface*	aplikacijsko programsko sučelje
BLL	*Business Logic Layer*	sloj poslovne logike
ER	*Entity Relationship*	odnos entiteta
HTML	*Hypertext Markup Language*	prezentacijski jezik na webu
HTTP	*Hypertext Transfer Protocol*	protokol prijenosa na webu
ICT	*Information and Communication Technology*	informacijska i komunikacijska tehnologija
JSON	*Javascript Object Notation*	JavaScript objektna notacija
JWT	*JSON Web Token*	web token u JSON obliku
MVC	*Model View Controller*	obrazac softverske arhitekture
SQL	*Structured Query Language*	strukturirani upitni jezik
TTS	*Text-To-Speech*	tekst u govor
URI	*Uniform Resource Identifier*	ujednačeni identifikator sadržaja
URL	*Uniform Resource Location*	ujednačeni lokator sadržaja
XML	*Extensible Markup Language*	jezik za označavanje podataka

Popis slika

Slika 3.1. Primjer komunikacijske ploče 6

Slika 4.1. SQL Server Management Studio 7

Slika 5.1. ER dijagram baze podataka VoiceUP 13

Slika 6.1. Prikaz svih modela web servis 14

Slika 7.1. Prikaz modela Angular aplikacije 22

Slika 7.2. Prikaz datoteka komponente za dodavanje riječi 25

Slika 7.3. Komponenta prijave u sustav 27

Slika 7.4. Početna komponenta aplikacije (`homeComponent`) 27

Slika 7.5. Komponenta dodavanja kategorije 28

Slika 7.6. Komponenta za prikaz riječi određene kategorije 29

Slika 7.7. Prikaz početne komponente mobilne aplikacije 30

Slika 7.8. Izbornik za rad s kategorijama 30

Slika 7.9. Prikaz komponente za uređivanje kategorije mobilne aplikacije 31

Popis tablica

Tablica 7.1. Popis komponenti aplikacije..... 25

Popis kôdova

Kôd 5.1. Kod za kreiranje tablice kategorija 10

Kôd 5.2. Kod za kreiranje procedure za dodavanje riječi 12

Kôd 6.1. Prikaz klase kategorija (engl. *category*) 14

Kôd 6.2. Prikaz sučelja `IRepository` 16

Kôd 6.3. Akcijska metoda za prijavu korisnika na `UserController` upravljaču 17

Kôd 6.4. Metoda `DeleteCategory` na `BLL` klasi 18

Kôd 6.5. Statična klasa `JWTHelper` 19

Kôd 7.1. Prikaz sučelja `Deserializable` 23

Kôd 7.2. Prikaz klase riječ (engl. *word*) 23

Kôd 7.3. Metoda dohvaćanja generalnih kategorija 24

Kôd 7.4. HTML kod koji predstavlja riječ na početnoj komponenti (`homeComponent`) 26

Kôd 7.5. Implementacija metode `ngOnInit` na početnoj komponenti 32

Kôd 7.6. Metoda `play` na početnoj komponenti web aplikacije 33

Kôd 7.7. Metoda `play` početne komponente mobilne aplikacije 34

Literatura

[1] SAVEZ UDRUGA ZA AUTIZAM HRVATSKE, AUTIZAM, https://www.autizam-suzah.hr/autizam/, siječanj 2020.

[2] JELENA ŠAFARIĆ, TERAPIJSKI PRISTUP AUTISTIČNOM DJETETU, https://repozitorij.ufzg.unizg.hr/islandora/object/ufzg%3A1123, siječanj 2020.

[3] ICT-AAC, POTPOMOGNUTA KOMUNIKACIJA, http://rain.ict-aac.hr/potpomognuta-komunikacija, siječanj 2020.

[4] AAC-RU&KA, IMPLEMENTACIJA KOMUNIKACIJSKIH POMAGALA U ODGOJNO – OBRAZOVNE PROCESE, http://www.aac-ruika.com/2018/03/02/implementacija-komunikacijskih-pomagala-u-odgojno-obrazovne-procese , siječanj 2020.

[5] WIKIPEDIA, MICROSOFT SQL SERVER, https://hr.wikipedia.org/wiki/Microsoft_SQL_Server, siječanj 2020.

[6] MICROSOFT, WHAT IS ASP.NET?, https://dotnet.microsoft.com/learn/aspnet/what-is-aspnet, siječanj 2020.

[7] SIMPLILEARN, ANGULARJS VS ANGULAR2 VS ANGULAR4, https://www.simplilearn.com/angularjs-vs-angular-2-vs-angular-4-differences-article, siječanj 2020.

[8] WIKIPEDIA, NATIVESCRIPT, https://en.wikipedia.org/wiki/NativeScript, siječanj 2020.

[9] CSI, BAZA PODATAKA, http://csi.hr/wiki/B, siječanj 2020.

[10] JWT, INTRODUCTION, https://jwt.io/introduction/, siječanj 2020.

[11] ANGULAR, COMPONENT, https://angular.io/api/core/Component, siječanj 2020.