Mrežna aplikacija za učenje i poučavanje putem video lekcija

Janjiš, Marko

SVEUČILIŠTE U SPLITU

PRIRODOSLOVNO-MATEMATIČKI FAKULTET

DIPLOMSKI RAD

MREŽNA APLIKACIJA ZA UČENJE I POUČAVANJE PUTEM VIDEO LEKCIJA

Marko Janjiš

Mentor: prof. dr. sc. Andrina Granić

Neposredni voditelj: dr. sc. Jelena Nakić

Split, veljača 2021.

Temeljna dokumentacijska kartica

Diplomski rad

Sveučilište u Splitu
Prirodoslovno – matematički fakultet
Odjel za informatiku
Ruđera Boškovića 33, 21000 Split Hrvatska

MREŽNA APLIKACIJA ZA UČENJE I POUČAVANJE PUTEM VIDEO LEKCIJA

Marko Janjiš

SAŽETAK

Mrežne aplikacije za učenje i poučavanje na daljinu putem video lekcija su danas često u upotrebi kako bi se proširilo ili steklo novo znanje. U ovom radu opisani su osnovni koncepti koje bi takva aplikacija trebala imati. Obrađene su popularne tehnologije i njihovi principi rada. Osim toga opisani su instrukcijski modeli koji uvelike pomažu prilikom izrade metodički oblikovanih video lekcije. *Crodemy* je aplikacija namijenjena za učenje i poučavanje putem video lekcija kojoj je uloga približiti tehnologiju studentima hrvatskog govornog područja. Cijela aplikacija je napravljena na hrvatskom jeziku kako bi stručna terminologija bila razumljivija korisniku s prostora RH. Motivacija za izradu rada je jezična barijera i pojava COVID-19 virusa 2019. godine. Platforma je rađena po uzoru na poznate mrežne aplikacije ovog tipa kao što su: *Udemy*, *Coursera* i *Khan Academy*. Korisniku omogućuje prijavu i registraciju te pregled dostupnih tečajeva. Svi tečajevi na aplikaciji su besplatni te ih korisnik može upisati samo ako je registriran odnosno prijavljen. Svaki tečaj se sastoji od nekoliko video lekcija na kojima pojedinac može steći osnovne znanja i koncepte programiranja u *ReactJS*-u, *Pythonu* tečajevima za odrasle ili *Micro*:*bit*-u koji je namijenjen svim dobnim skupinama. Crodemy je prototipna aplikacija koja ima prostora za razvoj te bi daljnjim napretkom trebala poslužiti kao dodatni izvor znanja uz klasične edukacijske sadržaje koje nudi obrazovni sustav.

Ključne riječi: *mrežna aplikacija, tečaj, funkcija, učenje, video lekcija, MERN arhitektura, instrukcijski dizajn, React, Node, MongoDB, prijava, registracija, odjava*

Rad sadrži: *42 stranica, 19 grafičkih prikaza, izvornik je na hrvatskom jeziku*

Mentor: dr. sc. Andrina Granić, redoviti *profesor Prirodoslovno – matematičkog fakulteta, Sveučilišta u Splitu*

Neposredni voditelj: dr. sc. Jelena Nakić, polslijedoktorand *Prirodoslovno – matematičkog fakulteta, Sveučilišta u Splitu*

Ocjenjivači: dr. sc. Andrina Granić, redoviti *profesor Prirodoslovno – matematičkog fakulteta Sveučilišta u Splitu*

dr. sc. Jelena Nakić, *poslijedoktorand, Prirodoslovno – matematičkog fakulteta, Sveučilišta u Splitu*

dr. sc. Nikola Marangunić, *Docent, Prirodoslovno – matematički fakultet, Sveučilišta u Splitu*

Basic documentation card

Graduate thesis

University of Split
Faculty of science
Department of informatics
Ruđera Boškovića 33, 21000 Split, Croatia

WEB APPLICATION FOR LEARNING AND TEACHING USING VIDEO LESSONS

Marko Janjiš

ABSTRACT

Web applications for distance learning and teaching using video lessons are often used today to expand or acquire new knowledge. This paper describes the basic concepts that such an application should have. Popular technologies and their working principles are discussed. In addition, instructional models are described that greatly help in creating pedagogically quality video lessons. *Crodemy* is an application intended for learning and teaching through video lessons whose role is to bring technology closer to Croatian-speaking students. The entire application is made in the Croatian language in order to make the professional terminology more understandable to the user from the territory of the Republic of Croatia. The motivation for the work is the language barrier and the appearance of the COVID-19 virus in 2019. The platform is modeled on well-known web applications of this type such as: *Udemy*, *Coursera* and *Khan Academy*. It allows the user to log in and register and view available courses. All courses on the application are free and the user can enroll only if he is registered or logged in. Each course consists of several video lessons in which an individual can acquire basic knowledge and programming concepts in *ReactJS*, *Python* courses for adults or *Micro*:*bit*, which is intended for all age groups. *Crodemy* is a prototype application that has room for development and with further progress should serve as an additional source of knowledge in addition to the classic educational content offered by the education system.

Key words: *web application, course, function, learning, video lesson, MERN architecture, Instructional design, React, Node, MongoDB, login, sign up, logout*

Thesis consists of: *42 pages, 19 figures, original language: Croatian*

Mentor: Andrina Granić, Ph. D. *Full professor of Faculty of Science, University of Split*

Supervisor: Jelena Nakić Ph. D. *Postdoctoral Researcher of Faculty of Science, University of Split*

Reviewers: **Andrina Granić Ph. D.** *Full professor of Faculty of Science, University of Split*

Jelena Nakić Ph. D. *Postdoctoral Researcher of Faculty of Science, University of Split*

Nikola Marangunić Ph. D. *Associate Professor o Faculty of Science, University of Split*

Želim iskazati zahvalu neposrednoj voditeljici dr. sc. Jeleni Nakić koja me vodila kroz izradu ovog rada. Također želim zahvaliti na njenom zalaganju kroz cijelo moje obrazovanje na Prirodoslovno-matematičkom fakultetu te svim novi znanjima koja sam stekao od nje.

Najveće hvala mojim roditeljima, bratu i sestri koji su me podupirali kad je bilo najteže, svaki moj pad nije bio uzaludan, a upravo oni su pružali ruku kako bih ustao i dali mii vjetar u leđa.

Veliko hvala mojim prijateljima koji su me podupirali sve ove godine i svaki moj prolaz proslavili dostojno, a svaki pad proživjeli zajedno sa mnom.

Zahvaljujem se svojim kolegama na fakultetu jer smo zajedno plakali u najtežim trenutcima i zajedno se smijali i veselili svakoj prijeđenoj stepenici tokom studija.

Sadržaj

1. Uvod

Aplikacija Crodemy napravljena je s ciljem popularizacije mrežnih aplikacija za učenje i podučavanje na hrvatskom jeziku. Većina ljudi u svijetu koristi engleski jezik za komunikaciju te se upravo on najviše koristi u IT svijetu kada je riječ o personaliziranom učenju preko online sustava.

Učenje i poučavanje korištenjem stranoga jezika može dovesti do neiskorištavanja potencijala pojedinca koji je željan stjecanja znanja putem platformi ovog tipa. Naravno, pamćenje tijekom učenja i razumijevanje stručne terminologije ovisi o pojedincu, no zasigurno je svakom učeniku jednostavnije pratiti nastavu na materinskom jeziku. Upravo je ta jezična barijera postala glavnom motivacijom izrade ovog diplomskog rada. Većina platformi koje se danas nalaze na tržištu nemaju tečajeve na hrvatskom jeziku.

Aplikacija Crodemy u potpunosti je na hrvatskom jeziku, počevši od korisničkog sučelja do video lekcija koje se nalaze na njoj. Zbog sličnosti jezika može se upotrebljavati i na regionalnoj razini, odnosno, bez poteškoća tečajeve mogu pratiti stanovnici Bosne i Hercegovine, Crne Gore, Srbije i Slovenije.

Osim jezične barijere, dodatna motivacija za izradu ovog diplomskog rada bila je pojava virusa COVID-19. U 2020. godini mogli smo uočiti velike probleme s kojima se susreo obrazovni sustav. Unatoč teškom vremenu, nastavnici, učitelji i studenti diljem zemlje sudjelovali su u izradi video lekcija za učenike osnovnih i srednjih škola. Napravili su veliki pothvat i pokazali koje sve mogućnosti nam pruža današnja tehnologija. Upravo u tim trenutcima poželjno je imati ovakvu platformu koja će poslužiti kao jedan veliki izvor znanja za buduće i sadašnje naraštaje. Cilj je prikupiti što više obrazovnog sadržaja na jednom mjestu kako bi osobe svih dobnih skupina, u bilo kojem trenutku, mogle učiti u svojim domovima.

2. Pregled mrežnih aplikacija namijenjenih učenju i poučavanju

Učenje i poučavanje putem mrežnih aplikacija danas je u porastu [Jansen i Schuwer, 2015; I. E. Allen i J. Seaman, 2014]. Prema istraživanjima Jansena i Schuwera (2015) 47,8% institucija koje pripadaju sustavu Bologne koristi jedan od masovnih otvorenih online tečaja dok 17,9% nudi pet ili više njih. U EU ukupno 71,7% institucija se koristi ovim tipovima platformi. U SAD-u ovaj trend se prema Allenu i Seamanu (2014) počeo pojavljivati već 2002. godine. Naime s vremenom su se dogodila kritična razdoblja, točnije 2006. i 2007. godine u kojima je zabilježen pad upotrebe online tečajeva na visokim učilištima. Dubinskim istraživanjem kako ove platforme utječu na učenje svake godine se brojka povećavala za mali postotak koji je s vremenom postao značajan.

Najveći doprinos tom trendu donio je razvoj tehnologije. Na računalnoj mreži mogu se naći različiti pružatelji tečajeva, video lekcija, prezentacija i interaktivnih materijala koji su namijenjeni stjecanju i širenju znanja pojedinca. U ovom poglavlju govorit ćemo o najpoznatijim online tečajevima, a po uzoru na njih napravljen je praktični dio ovog rada.

Platforme o kojima ćemo govoriti su:

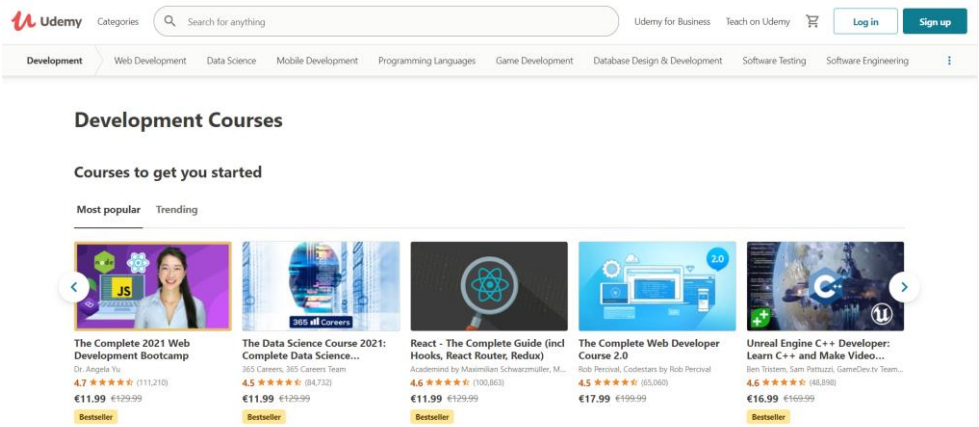
- Udemy
- Coursera
- Khan Academy

2.1. Udemy

Udemy je jedna od najpoznatijih platformi kojoj je uloga online edukacija. Broji preko 130 000 tečajeva (Udemy, 2021.) iz različitih područja. Najzastupljenija grana je računalna znanost, međutim, velik broj lekcija pripada dizajnu, marketingu, glazbi, zdravlju i poslovanju. Osnovan je 2010. godine, a njegovi osnivači su Eren Bali, Oktay Caglar i Gagan Biyani.

Platforma omogućava korisniku da izabere ulogu nastavnika (instruktora) ili studenta prilikom prijave. Instruktori mogu izrađivati video lekcije na temu koju su odabrali. Osim video lekcija mogu priložiti dokumente kao što su: PowerPoint prezentacije, PDF, komprimirane datoteke (eng. *ZIP*) i audio datoteke. Studenti nemaju prethodno spomenute mogućnosti, ali imaju mogućnost pohađanja tečaja iz područja o kojem žele naučiti ili nadograditi svoje prethodno znanje. Objavljivanje rada kako bi ga instruktor ili drugi sudionici tečaja mogli ispraviti ili postavljanje pitanja

i rješavanje testova samo su neke od mogućnosti ponuđene studentima. Važno je obratiti pažnju na ispravljanje rada jer se na taj način razvija suradnja i zajednica (eng. *community*). Upravo ta komunikacija između većeg broja korisnika može pojedincu pružiti različite oblike rješenja istog problema, a samim time i mogućnost odabira pristupa rješavanja zadataka. Osim funkcionalnosti koje su omogućene, ova platforma ima sučelje (Slika 2.1.1), koje je jednostavno i prilagođeno svakom tko želi stjecati nova znanja. Na početnoj stranici mogu se uočiti preporučeni tečajevi i hiperveze koje vode na prijavu ili registraciju ovisno o tome ima li korisnik otvoren račun ili ne. Prije same prijave korisnik može pogledati sve kategorije tečajeva i detaljan opis istih; na taj način mu je omogućeno da izabere područje koje mu najviše odgovara.



Slika 2.1.1 - Sučelje Udemy platforme

Kako je već spomenuto na početku poglavlja, praktični dio rada najviše se bazirao upravo na *Udemy* platformi, odnosno na veliki dio njenih funkcionalnosti. Važno je naglasiti da su druge mrežne aplikacije ovog tipa slične i da imaju većinu zajedničkih opcija.

2.2. Coursera

Coursera isto tako ubrajamo u popularne platforme koje služe obrazovanju od kuće, odnosno, stjecanju i širenju znanja pojedinaca. Osnovana je nešto kasnije od prethodno spomenutog *Udemy-ja,* točnije 2012. godine. Osnivači su profesori sveučilišta Standford (eng. *Stanford University*) Andrew Ng i Daphne Koller. Na ovoj mrežnoj aplikaciji također su ponuđene razne kategorije koje korisnik može izabrati, ovisno o području njegova zanimanja. Neka su od područja koja je moguće izabrati matematika, logika, poslovanje, jezici, računalna znanost, zdravlje i drugi. Princip na kojem funkcionira Coursera znatno je sličan onom kod Udemy-ja, no postoje neke razlike. Uloga studenta

gotovo je identična dok uloge predavača (instruktora) nema. Uzrok tome način je na koji ova platforma funkcionira.

Kod *Coursere* naglasak je na profesionaliziranom pristupu gdje su predavači većinom zaposlenici poznatih sveučilišta. Već tu možemo uočiti da su neke mogućnosti ove platforme drugačije od prethodno spomenute ili pak *Khan Academy*-ja o kojem ćemo govoriti nešto kasnije. Naime, kada se korisnik prijavi ima priliku upisati tečajeve iz različitih područja i sudjelovati na predavanjima u stvarnom vremenu upravo na tim sveučilištima preko video veze. To nužno ne znači da su svi tečajevi osmišljeni na ovaj način jer postoje i tečajevi kojima korisnik može pristupiti kada želi, takvi se tečajevi ne izvode u stvarnom vremenu. Zahvaljujući brojnim sveučilištima, odnosno, partnerima, *Coursere* ideja ovog tipa nastave je zaživjela. Nekolicina partnera koji su pomogli kako bi se ovaj oblik predavanja ostvario su i:

- Sveučilište Princeton – (eng. *Princeton University*)
- Sveučilište Yale – (eng. *Yale University*)
- Sveučilište London – (eng. *London University*)
- Sveučilište Yonsei – (eng. *Yonsei University*)

Kao i kod *Udemy-ja,* da bi korisnik pohađao neki tečaj mora se na njega pretplatiti. Razlika između njihovih pretplata odnosi se na trajanje pretplate. Na Courseri za određene tečajeve postoji vrijeme trajanja zbog izvođenja nastave u stvarnom vremenu, a kod *Udemy-ja* jednom kada platimo pretplatu na neki tečaj, nema ograničenja.

Važno je osvrnuti se na razlike između ove dvije platforme, tj. procijeniti koji pristup odgovara studentu. Jedan je tradicionalan, odnosno, klasični oblik nastave u stvarnom vremenu preko video poziva ili video lekcije kojima se uvijek može pristupiti te nema nikakve interakcije s predavačem. Naravno, veliki utjecaj na odabir platforme ima omjer cijene i kvalitete, ali također ovisi i o pojedincu, je li korisnik platformw osoba kojoj odgovara da sama istražuje i uči ili joj je potrebna izravna motivacija nastavnika.

Razlika u izgledu sučelja (Slika 2.2.1) ove dvije platforme minimalna je, ali ako ćemo uzeti u obzir da iza *Coursere* stoji podrška većeg broja partnera, onda možemo naslutiti da je sučelje organiziranije i jednostavnije za snalaženje korisnika nego što je to slučaj na platformi *Udemy*. Važno je napomenuti da obje platforme pri završetku tečaja dodjeljuju certifikat i diplomu koja potvrđuje završetak njihovog tečaja.

Slika 2.2.1 – Korisničko sučelje Coursera platforme

2.3. Khan Academy

Khan Academy neprofitabilna je mrežna aplikacija namijenjena personaliziranom učenju pojedinca. Osnovao ju je Sal Khan 2008. godine kako bi objašnjavao matematiku svom rođaku. Kao i kod *Udemy-ja* sastoji se od snimljenih video lekcija koje se izvode na elektroničkoj ploči. Za razliku od prethodno spomenutih aplikacija, *Khan Academy* usredotočuje se na tradicionalne znanosti: matematiku, ekonomiju, fiziku, računalno programiranje i humanističke znanosti. Nadalje, pokriva sve predmete koji se nalaze u školi te je prilagođena svim dobnim skupinama počevši od djece koja još idu u vrtić do odraslih osoba koje žele proširiti svoje znanje.

Lekcije koje se nalaze na ovoj platformi kratke su i usredotočene na jednostavnost. Pružaju velik broj sadržaja koji omogućuje korisniku vježbanje i praćenje svoga napredak tijekom pohađanja tečaja. Upravo se *Khan Academy* ističe u motivaciji – tijekom pohađanja nastave student, ovisno o uspjehu, dobiva značke koje su vidljive na njegovom računu/profilu. Uočljive su i glavne razlike i sličnosti ove mrežne aplikacije u odnosu na već spomenute. *Khan Academy* u potpunosti je besplatan te funkcionira na principu donacija, pruža motivaciju korisniku, ima mogućnost prijave kao nastavnik ili student, lekcije su namijenjene za sve dobne skupine. Sučelje *Khan Academy-ja* poprilično je slično *Udemy-ju* i *Courser-i,* a jedini je nedostatak preglednost tečajeva. Kategorije su, može se reći, neuredno prikazane.

2.4. Ostale popularne mrežne aplikacije namijenjene poučavanju i učenju

Udemy, *Coursera* i *Khan Academy* platforme su koje služe personaliziranom učenju i poučavanju, ali nisu jedine. Online tečajevi danas su postali sastavni dio obrazovanja pojedinca i na računalnoj mreži može se pronaći velik broj mrežnih aplikacija kojima je glavna uloga olakšati učenje. U prethodnim poglavljima, istaknute su neke razlike koje se javljaju na tim platformama i

svaka od njih ima svoj jedinstveni pristup, no istovremeno su i slične jer su im interes i cilj, u konačnici, isti. Osim navedenih, postoji i još nekoliko popularnih mrežnih aplikacija namijenjenih učenju i poučavanju koje nisu opisane:

- Udacity
- Alison
- Lynda
- Futurlearn
- edX
- Bloc
- CodeCademy

Razlika između aplikacije koja će biti opisana u ovom radu i gore navedenih jest ciljana jezična skupina što je glavni motiv, uz pojavu virusa COVID-19, koji je znatno utjecao na obrazovni sustav diljem svijeta [Shahzad. et. al., 2020]. Hrvatski se jezik rijetko pojavljuje na stranicama ovog tipa što studentima i učenicima s prostora Republike Hrvatske otežava učenje i razumijevanje stručne terminologije u različitim znanstvenim područjima.

3. Modeli instrukcijskog dizajna i video lekcije

Video lekcija je jedan od digitalnih materijala za učenje kojim se služe mrežne aplikacije namijenjene online učenju. Osim toga, važnu ulogu imaju i drugi digitalni sadržaji kao što su prezentacije, tekstualni dokumenti i audio datoteke. Za razliku od tradicionalnog oblika nastave, online učenje omogućava korisniku interaktivno učenje vlastitim ritmom, kad on to želi. Prilikom organizacije digitalnih sadržaja, predavači se koriste različitim modelima instrukcijskog dizajnirana

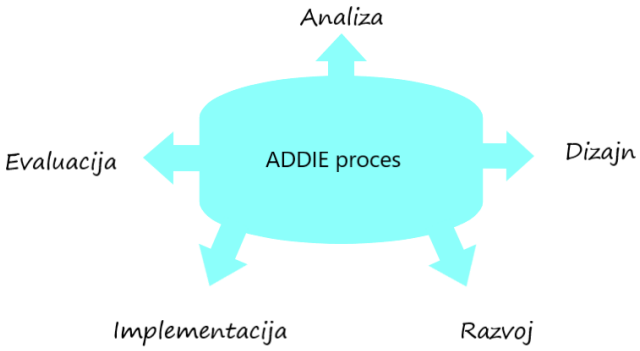
- ADDIE model
- Merrill-ovi instrukcijski principi
- Gagne-ovih devet instrukcijskih događaja
- Bloom-ova taksonomija

3.1. ADDIE model

ADDIE model jedan je od prvih koji se pojavio [Gutierrez 2018.]. Još uvijek se koristi u praksi za izradu online tečajeva unatoč tome što su njegovi koncepti odavno u primjeni. Sastoji se od pet faza (Slika 3.1.1) koje dizajner, odnosno, vlasnik tečaja mora slijediti kako bi napravio kvalitetan tečaj.

1. *analiza* - prva faza u kojoj dizajner odgovara na pitanje zašto je tečaj potreban te kojoj je populaciji/dobnoj skupini namijenjen
2. *dizajn* - u drugoj fazi, odabire se strategija podučavanja, metodičke komponente i određuje se konačni cilj
3. *razvoj* - u trećoj fazi gledaju se zaključci iz prethodnih faza te se izrađuju materijali koji su u skladu s njima
4. i*mplementacija* - u četvrtoj fazi tečaj se isporučuje te se prate i bilježe rezultati korisnika/učenika
5. *evaluacija* - posljednja faza odnosi se na procjenu utjecaja tečaja koji se temelji na povratnim informacijama, najčešće u obliku anketa.

Nakon što je završen cijeli ciklus ADDIE modela, povratne informacije koje su prikupljene prilikom evaluacije se promatraju te se ispravljaju moguće pogreške, a zatim se cjelokupni proces ponavlja.



Slika 3.1.1 - Pet faza ADDIE modela

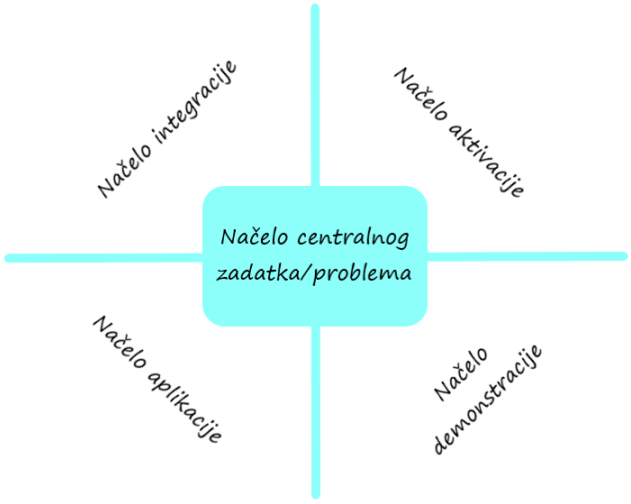
3.2. Merrillovi instrukcijski principi

Ovaj model instrukcijskog dizajna nastao je 2002. godine, a osmislio ga je David Merrill. Njegovi principi su usmjereni da se pojedincu pokuša usaditi maksimalno znanje iz svakog tečaja, radilo se to o video lekcijama, animacijama, prezentacijama ili nekom drugom obliku digitalnog sadržaja za učenje. Merrill i suradnici (2008.) u svom radu opisuju pet osnovnih načela (Slika 3.2.1) kojima se treba voditi prilikom korištenja ovog instrukcijskog dizajna. Njihova glavna prednost je skalabilno izvođenje zadataka koji utječu na napredak studenta.

1. *Načelo centriranog zadataka (problema)* – studentima se zadaje problem iz svakodnevnog okruženja, usko povezanim uz sadržaj koji se obrađuje
2. *Načelo aktivacije* – prethodna znanja i iskustva na području koje se obrađuje znatno utječu na stjecanje novih mentalnih modela studenta te omogućuje strukturiranje sadržaja
3. *Načelo demonstracije* – konzistentan demonstracija studentima pruža bolji izvedbu tokom rješavanja kompleksnih problema do kojih se došlo skalabilnim putem
4. *Načelo aplikacije* – aplikacija pruža mogućnost kontinuiranog pružanja povratne informacije za prethodnu demonstracijsku fazu
5. *Načelo integracije* – integracijom prethodnih načela promiču se nova usvojena znanja i vještine u svakodnevnom životu te potiču brže rješavanje sličnih problema

Merrill u svom radu opisuje tri skupine koje su testirane, a pružene su im različite faze odnosno načela njegovog instrukcijskog dizajna prilikom učenja Excel-a. Naime prva grupa je jedina koristila sva načela i njihovi rezultati su imali prolaznost 89% dok u drugoj grupi koja je imala izravan pristup

e-učenju bez početne faze je imala prolaznost 68%, a zadnja testna skupina je imala pristup samo završnim lekcijama te je 34% ispitanika uspješno završilo tečaj.



Slika 3.2.1 - Pet načela Merrill-ovog instrukcijskog dizajna

3.3. Gagneovih devet instrukcijskih događaja

Gagneov model se razlikuje od gore navedenih zbog biheviorističkog pristupa. Naime Gagne je bio obrazovni psiholog te je svojih devet instrukcijskih događaja povezao s procesom učenja na način da privlačenje učenikove pažnje tj. izazivanjem podražaja prilikom se informacije prenose iz kratkoročnog u dugoročno pamćenje [Krull i Oras, 2010]. Upravo kako bi to Gagne postigao primijenio je sljedeće događaje:

1. *Privlačenje pozornosti* – vođenje dijaloga i poticanje studenta na aktivnost verbalnu ili fizičku (primjer ustajanja učenika)
2. *Informiranje studenata* – obavještavanje studenata o gradivu koje će se obrađivati te pojašnjenje postupaka vrednovanja
3. *Poticanje sjećanja vezana za prethodna znanja* – ponavljanje prethodno naučenog gradiva pruža lakšu nadogradnju znanja i povezivanje novog i starog
4. *Predstavljanje sadržaja* – novi sadržaj se prikazuje u malim i jasnim ulomcima kako bi se postigao inkrementirajući učinak
5. *Pružanje smjernica* – pokazivanje primjera, pružanje podrške i davanje instrukcija kako bi se približio sadržaj studentima
6. *Provjera uspjeha učenja* – uključivanje studenata u razne aktivnosti kako bi se provjerilo njihovo znanje te mogla dati povratna informacija

7. *Povratna informacija* – pružanje iste može studenta navesti na ispravan način rješavanja nekog problema, a samim tim poboljšava i zadržava prijenos naučenog sadržaja
8. *Organizacija i upravljanje općim razrednim aktivnostima* – zadavanje različitih aktivnosti druženje i različiti načini rada s učenicima (pojedinačan, grupni rad, projektni zadatak itd.)
9. *Opća strategija poučavanja i atmosfera u učionici* – na osnovu prethodnih događaja stvaraju se nove strategije i atmosfera u učionici

Istraživanja Krulla i Orase (2010) koje je provedeno na estonskom sveučilištu Tartu, istraživači su donijeli zaključak da Gagneov nastavni model potiče psihološke podražaje pa je interpretacija nastavnog sadržaja razumljivija i motivira učenika na učenje u različitim granama znanosti. Kontrolna skupina studenata nastavničkih smjerova pokazala je napredak u percepcijama i razmišljanjima istih.

3.4. Bloomova taksonomija

Bloomova taksonomija je instrukcijski dizajn koji je osmislio obrazovni psiholog Benjamin Bloom. On učenje dijeli u šest razina koje su međusobno zavisne, odnosno imaju hijerarhijsku strukturu, to znači da napredak ovisi o razini na kojoj se učenik nalazi te se gleda da li je uspješno prošao prethodne faze [Sabatura, 2013]. Kako bi se mogao strukturirati i planirati proces usvajanja novog sadržaja potrebno je slijediti sljedeće korake:

1. *Prisjećanje* – na ovoj razini učenik se prisjeća važnih informacija iz dugoročnog pamćenja koje su povezane s trenutnim gradivom
2. *Razumijevanje* – razina u kojoj učenik stvara sliku i poimanje o slikovnim, slušnim i grafičkim informacijama
3. *Primjena* – na ovoj razini učenici koriste procedure za rješavanje problema te ih implementiraju
4. *Analiza* – učenici u na ovoj razini vrše podjelu materijala na manje dijelove te određuju međuzavisnost istih, motiv ovog koraka je dobra organizacija
5. *Evaluacija* – razina na kojoj je potrebno izvršiti ocjenjivanje na temelju jasno prethodno jasno postavljenih kriterija (važnu ulogu ima povratna informacija)
6. *Stvaranje* – strukturiranje i spajanje svih prethodnih elemenata u jednu funkcionalnu cjelinu, reorganizacija se također provodi u ovoj razini ako je potrebna

Slika 3.4.1 - Šest razina učenja Bloom-ove taksonomije [AZZO 2021]

Važno je navesti još jednu podjelu koja se koristi u Bloom-ovoj taksonomiji, a to je na višu i nižu kognitivnu razinu. Studenti koji su još uvijek na dnu gore prikazane piramide tj. u nisu prešli primjenu pripadaju nižoj kognitivnoj razini dok studenti koji se nalaze iznad primjene pripadaju višoj kognitivnoj razini.

3.5. Izrada video lekcije

U ovom dijelu poglavlja govorit će se o načinima izrade video lekcija koji su osnovni digitalni sadržaj mrežnih aplikacija namijenjenih za online učenje. Forma na koji način će se snimiti video lekcija nije standardizirana, no postoji nekoliko oblika snimanja digitalnog sadržaja koji se koriste u online nastavi. Ovisno o tečaju, upotrebljavaju se različiti formati kao što je tradicionalni pristup (snimanje predavača skupa s pločom što učenika podsjeća na nastavu u učionici), Khan format u online učenju u krupni plan stavlja sadržaj (ne vidi se predavačevo lice već se u kadru nalazi samo sredstvo za pisanje i podloga), a danas se također koriste web emisije koje prilikom predavanja koriste prezentacije i animacije dok se u pozadini čuje samo predavačev glas. U praksi se koriste varijacije ovih formata u kojima se oni kombiniraju i rezultiraju dinamikom video lekcije koja utječe na pažnju učenika.

Ou, Joyner i Goel (2019) tvrde da je upotreba video lekcija u porastu u zadnjem desetljeću ponajviše zbog razvoja uređaja koji imaju kameru kao što su pametni telefoni i prijenosna računala te nam ona omogućuju snimanje bilo gdje i u bilo kojem trenutku. Osim toga besplatno dijeljenje medija i internetski prijenos (eng. *streaming*) znatno olakšavaju instruktorima, nastavnicima i predavačima da objave svoje materijale. Oni su u svom istraživanju koristili sustav kojim upravlja umjetna inteligencija te su studentima isporučili 26 video lekcija. Svaka od njih je podijeljena u četiri faze nastave:

1. *Aktivacija* – studentima se u ovoj fazi predstavljaju teme koje će biti obrađene u lekciji te ih pokušavaju motivirati da povežu prethodna gradiva s onim koje slijedi
2. *Demonstracija* – studenti se u ovoj fazi lekcije upoznavaju s metodama i funkcijama aplikacije, gradivo je prezentirano pomoću grafičkih i animiranih ilustracija
3. *Primjena* – studentima su unutar video lekcije ugrađeni interaktivni sadržaji kako bi mogli vježbati i učvrstiti dosadašnje stečeno znanje
4. *Integracija* – pruža sažetak gradiva te nagovještava sljedeće gradivo koje će biti obrađeno, a studentima se daje naputak da zapišu sadržaj i razmisle o njemu

Možemo uočiti da ove četiri faze koje su korištene u istraživanju se podudaraju s Merrillovim instrukcijskim dizajnom. Ou, Joyner i Goel su na kraju istraživanja proveli anketu na koju je odgovorilo 1224 ispitanika. Stopa odgovora iznosila je 65% gdje je 90% studenata odgovorilo da su video lekcije bile informativne i od velike pomoći tokom učenja. Što se tiče vježbi čak se 80% ispitanika složilo da su bile kvalitetno organizirane i zahvaljujući povratnim informacijama njihovo znanje se znatno poboljšalo.

Prema Ginnakosu (2013) video lekcije imaju brojne pedagoške prednosti jer učenici imaju mogućnost pregledavanja sadržaja kako bi usvojili podatke koji su im promaknuli. Osim toga nastavnici mogu lako transformirati i prilagoditi nastavni sadržaj i okruženje kako bi se koristili na prikladan način. Video lekcije su korisne za ponavljanje, mogu pružiti zanimljiv interaktivan sadržaj, uvode novo okruženje na koje se korisnik prilagođava te se lako prilagođavaju u slučaju da učenici nemaju pozitivne povratne informacije u nekim dijelovima gradiva. Predavači trebaju integrirati tehnologije u svoje tečajeve, samo je pitanje njihove implementacije. Ginnakos također tvrdi da je Izrada video lekcija složen postupak te zahtjeva temeljito planiranje, postupak provedbe i poznavanje teorija učenja. Kako bi ovi utjecaji bili mogući nastavnik bi trebao koristiti neki od gore navedenih modela instrukcijskih dizajna.

3.6. Alati za uređivanje video lekcija

Prilikom izrade digitalnih sadržaja koriste se razne tehnologije. Da bi se izradile video lekcije, postoje različiti alati koji omogućuju snimanje i uređivanje videa. Većina njih ima slične značajke, neki ipak maju naprednije funkcije, ali su složeniji za korištenje, a postoje i oni koji imaju osnovne funkcije, ali su jednostavni za upotrebu. Rezanje, filteri, dodatni zvuk, bistrenje, spajanje… samo su neke od funkcionalnosti koje pružaju ovi alati. Većina profesionalnih alata za uređivanje videa naplaćuje se, međutim, postoji nekoliko njih koji su besplatni i pružaju niz mogućnosti koje će

poslužiti korisniku koji je početnik u ovom tipu aplikacija. Neki su od alata koji se danas koriste u ove svrhe:

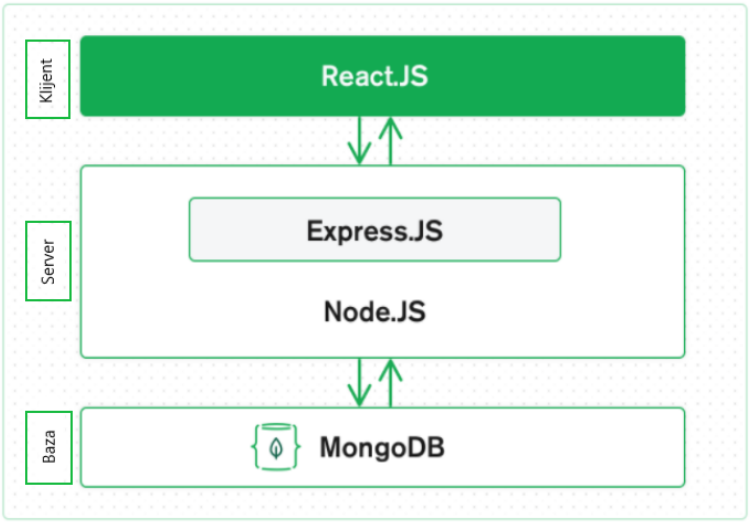
- Lightworks
- Lumen5
- Nero Video
- Corel VideoStudio
- Filmora from Wondershare
- DaVinci Resolve

4. MERN arhitektura

Sve popularnija u području mrežnog razvoja (eng. *Web development*) je i MERN arhitektura (Slika 3.2.1) koja se koristi u praktičnom dijelu ovog rada. Omogućava jednostavan način povezivanja klijentske i serverske strane uz upotrebu ne relacijske baze podataka. MERN je kratica za četiri tehnologije koje se koriste prilikom izrade mrežnih aplikacija [MongoDB, 2021.]:

- MongoDb – baza podataka dokumenata
- Express.js – biblioteka okruženja Node.js
- React.js – JavaScript razvojno okruženje (klijent)
- Node.js – JavaScript razvojno okruženje (server)

Osim prethodno spomenute arhitekture postoji još varijacija koje su slične, ali koriste druga razvojna okruženja koja su orijentirana na korisničko sučelje (klijent). MEVN i MEAN imaju jednake tehnologije kojima je fokus baza podataka i serverski dio aplikacije ,međutim, za izradu klijentskog dijela MEVN koristi *Vue.js*, a MEAN *Angular.js*. Svaka od njih ima svoje prednosti i nedostatke, međutim, prema istraživanju Kaluže i Vukelića (2018.) *React.js* je nakon *Angular.js*-a bio najtraženija tehnologija na *StackOverflow*-u s porastom od 1.7% u odnosu na druge u vremenskom razdoblju od 6 mjeseci u 2017. godini na području razvoja mrežnih aplikacija.



Slika 3.6.1 - Vizualni prikaz MERN arhitekture [MongoDB, 2021]

4.1. React

Na vrhu arhitekture nalazi se *React.js*, odnosno, razvojno okruženje namijenjeno za izradu dinamičkog korisničkog sučelja. Prepoznatljiv je po svojoj modularnosti i brzini zbog čega je došlo do velikog porasta upotrebe ovog okruženja proteklih godina. Glavna značajka koja doprinosi njegovoj brzini jest prikaz cjelokupnog sadržaja na jednoj stranici (eng. *Single-page rendering*), a korištenjem JSX (eng. *JavaScript XML*) sintakse očituje se jednostavnost upotrebe JavaScript-a, HTML-a (eng. *Hyper Text Markup Language*) i CSS-a (eng. *Cascading Style Sheets*). Osim spomenutih tehnologija, *React.js* zadnjih godina koristi funkcionalni pristup programiranja što programerima omogućava lakšu upotrebu u trenutcima kada projekt postaje veći.

4.1.1. Organizacija i modularnost React aplikacije

Pojam modularnost odnosi se na podjelu programa u male dijelove koji se zatim spajaju u jednu cjelinu. Zasniva se na principu *„podijeli pa vladaj“* gdje je svaki dio jedna nezavisna komponenta i moguće ju je iskoristiti na više mjesta istovremeno. Cilj je ovog pristupa smanjenje kompleksnosti, pregledniji kod, kontrola promjenjivih domena aplikacije, ponovna upotreba i lakši rad na kodu u timovima. Upravo *React.js* pruža te mogućnosti korištenjem komponenti koje će biti detaljnije opisane u narednim poglavljima.

Prilikom organizacije korisnik mora prvotno izabrati uređivač koda (eng. *code editor*) u kojem želi izraditi aplikaciju. Danas je sve popularniji *Microsoftov VS Code* u kojem je izrada *React.js* projekta olakšana zahvaljujući *Node.js* paketu (eng. *Node package manager*). Sve što je potrebno da bi se izradio predložak za ovaj tip aplikacije jest upisati sljedeće naredbe u terminal:

```
npm install create-react-app
npm init create-react-app ime_aplikacije
```

Struktura projekta prepuštena je korisniku, ali je važno napomenuti kako se treba pridržavati određenih pravila imenovanja direktorija. Potrebno je razlikovati sljedeće mape i u njima kreirati datoteke koje se odnose na taj dio aplikacije:

- *Pages* – datoteke koje se odnose na primarne stranice aplikacije
- *Components* – komponente koje služe za modifikaciju pojedinih dijelova glavnih stranica
- *Shared* – datoteke koje se odnose na izgled komponenti često korištenih korisničkih sučelja
- *Public* – mapa za pohranu glavnih vizualnih karakteristika aplikacije
- *Node_modules* – datoteke svih instaliranih biblioteka i paketa.

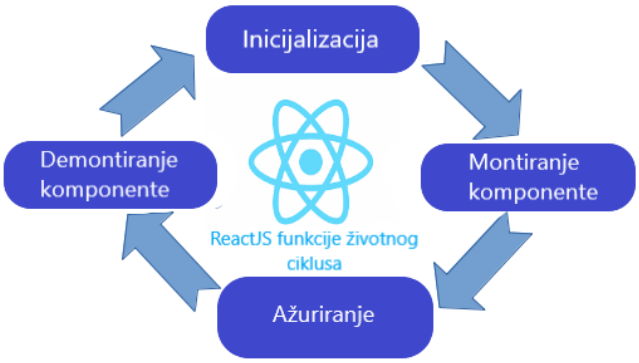
4.1.2. Klasne i funkcionalne komponente

Već je spomenuto kako se razvojno okruženje *React.js* ističe po svojoj modularnosti, a upravo tome doprinose komponente. Važno je razlikovati dva oblika pisanja istih, klasne komponente (eng. *statefull component*) zastarijevaju te se manje koriste u praksi, dok je upotreba funkcionalnih komponenti (eng. *stateless component*) češća.

Glavnu razliku između ova dva pristupa čini način pisanja, pristupanje i korištenje memorije te izvršavanje logike. Klasne komponente imaju stanje (eng. *state*), to je objekt koji posjeduje sva polja, odnosno, podatke koje je potrebno koristiti u komponenti. Na taj se način očituje upotreba memorije u ovom konceptu zasnovanom na objektno-orijentiranom pristupu. Upravo zbog toga se i nazivaju klasne komponente. Ovaj proces odvija se kroz četiri faze (Slika 4.1.1) upotrebom funkcija životnog ciklusa (eng. *lifecycle functions*):

- inicijalizacija (eng. *initialization*)
- montiranje komponente (eng. *mount*)
- ažuriranje (eng. *update*)
- demontiranje komponente (eng. *unmount*).

Funkcionalne komponente imaju drugačiji način obrade podataka koji se dohvaćaju iz memorije. Koristi kuke (eng. *hooks*), tj. ugrađene funkcije koje zamjenjuju funkcije životnog ciklusa, pružaju niz drugih mogućnosti kao što je upotreba lokalnog stanja, manipulacija poviješću (eng. *history*), referenciranje HTML elemenata itd. Neke od najčešće korištenih funkcija jesu `useState()`, `useRef()`, `useEffect()`, `useSelector()`, `useDispatch()`.



Slika 4.1.1 – Četiri faze životnog ciklusa klasne komponente [A. Nigam, 2019]

4.1.3. JSX sintaksa

JSX produžetak je sintakse programskog jezika JavaScript. Njegov izgled podsjeća na klasični HTML predložak, međutim, pruža mogućnost izvršavanja logike i uređivanja prikaza mrežnih stranica. Najčešće se te dvije radnje odvajaju u ostalim razvojnim okruženjima gdje programer mora izrađivati predložak koji će se prikazati korisniku u HTML-u, a logiku mora pisati u odvojenoj datoteci JavaScript-a. JSX upravo ujedinjuje te dvije tehnologije i ubrzava rad na projektu zahvaljujući jednostavnoj sintaksi. Za nekoga tko se prvi put susreće s ovakvim oblikom pisanja koda, programiranje može biti otežano, ali korisnik ovog pristupa navikne se na njegove karakteristike veoma brzo. Primjer koda napisanog JSX sintaksom izgleda kako slijedi

```
import React from 'react';
import './Card.css';
const Card = props => {
  return (
<div className={`card ${props.className}`}
style={props.stele}>
      {props.children}
    </div>
  );
};
export default Card;
```

Kod 4.1 – Primjer JSX sintakse

4.2. Node.js

Node.js razvojno je okruženje namijenjeno za izradu poslužiteljskog dijela aplikacije (eng. *server*) te isto kao i *React.js* koristi JavaScript programski jezik. Njegova je uloga dohvaćanje sadržaja iz baze podataka te prosljeđivanje istog klijentu.

Postupak koji se odvija na poslužiteljskom dijelu može se opisati kroz nekoliko koraka. Svaki put kada klijent pošalje zahtjev na mrežnom pregledniku, aktivira se događaj (eng. *event*) koji na poslužiteljskoj strani poziva funkciju. Funkcija, ovisno o tom zahtjevu, pristupa bazi podataka te odrađuje jednu od CRUD operacija (eng. *Create, Read, Update, Delete*). Nakon što se izvršila jedna od tih operacija, odgovor se vraća klijentu.

4.2.1. Asinkroni i sinkroni pozivi

Operacije slanja i primanja informacija na poslužitelju mogu biti sinkrone i asinkrone. Razlika se očituje u vremenu izvršavanja tih dvaju procesa. Kada dođe do poziva operacije, a proces se blokira dok ne dođe do njenog završetka, onda se govori o sinkronom pozivu, a ako je operacija pokrenuta i proces nije zaustavljen, koristi se pojam asinkroni poziv. Kod *Node.js*-a omogućen je paralelizam, samim tim ne dolazi do zaustavljanja procesa te se određena logika može izvršavati za vrijeme trajanja poziva. Iz navedenog je razvidno da je on u potpunosti asinkrono razvojno okruženje. Korištenjem tog pristupa korisnik se ne treba brinuti oko blokiranja procesa, niti jedna funkcija povratnog poziva (eng. *callback function*) ne izvodi izravan unos ili ispis.

4.2.2. Node moduli

Moduli, odnosno, biblioteke skup su funkcija koje je potrebno unijeti u aplikaciju da bi joj se pružile različite funkcionalnosti. Neki od njih već su ugrađeni u samo razvojno okruženje i ne zahtijevaju dodatnu instalaciju paketa dok postoje oni koje je potrebno instalirati. Korisnik alata također ima mogućnost izrade vlastitih modula kako bi unaprijedio poslužiteljsku stranu aplikacije. Najčešće korišteni ugrađeni moduli su *http*, *File System*, *url* i *email* (w3schools, 2021.).

- *HTTP* modul omogućuje prijenos informacija koristeći se istoimenim protokolom (eng. *Hyper Text Transfer Protocol*), a uloga mu je pregledavanje zahtjeva na ulazima i vraćanje odgovora klijentu
- *File System* modul služi za čitanje datoteka koje se nalaze na korisnikovom računalu
- *URL* modul prevodi adresu mrežne stranice na način da objekt adrese vraća kao svojstva kojima se naknadno može pristupiti i pročitati ih
- *Email (nodemailer)* modul služi kako bi se slala elektronička pošta putem *Node.js* poslužitelja.

Kako bi se korisnik mogao koristiti modulima koji nisu ugrađeni unutar okruženja, potrebno je koristiti upravitelj node paketima (eng. *Node package manager*). Njegova upotreba je jednostavna - u terminal se upiše naredba `npm install` i ime paketa. Primjerice, ako korisnik želi instalirati *express.js* koji je namijenjen za jednostavniju, bržu i pregledniju izradu poslužiteljskog dijela aplikacije, u konzolu upisuje sljedeću naredbu:

```
npm install express
```

Instalacija se može razlikovati ovisno o operativnom sustavu kojeg korisnik upotrebljava. Gore navedeni primjer odnosi se na OS Windows.

4.2.3. Express.js

Express.js razvojno je okruženje *Node.js*-a koje je potrebno instalirati prije upotrebe. Pruža niz značajki koje olakšavaju korisniku pisanje koda te izradu poslužiteljskog dijela mrežne aplikacije. Ima robustan pristup, odnosno, dovoljno je nekoliko linija koda kako bi se izradio jednostavan poslužitelj. Nakon instalacije *express.js*-a korisnik sučelja ga uključuje u aplikaciju te definira port na kojem će se moći pristupiti serveru. Da bi mogao raščlaniti tijelo zahtjeva koje dolazi prilikom poziva, potrebno je koristiti modul za raščlambu (eng. *body-parser*). Nakon što su ta dva paketa uključena u aplikaciju, moguće je definirati rute koje izvršavaju funkcije povratnog poziva te rade zahtjevnu operaciju na serveru. Na samom kraju definira se funkcija koja osluškuje port na kojem je pokrenut poslužitelj. Primjer kreiranja osnovnog *express.js* servera prikazan je u sljedećem kodu:

```
Const PORT = 3001

Const express = require('express')

Const bodyParser = require('body-parser')

Const App = express()

App.use(bodyParser.json())

//Here we write routes

App.listen(PORT,()=>console.log('Server started at
port 3001 successfully))
```

Kod 4.2 – Primjer express.js poslužitelja

4.3. MongoDb baza podataka

MongoDb bazirana je na dokumentacijskom modelu, odnosno, spada u NoSQL baze podataka. Dokument ima JSON format što znači da se svi podatci mapiraju u objekte što olakšava njihovu upotrebu i restrukturiranje. Arhitektura *MongoDb*-a zasniva se na dizajniranom sustavu distribucije koji omogućava inteligentnu raspodjelu podataka, dokumentacijskom modelu, globalnoj bazi podataka u više oblaka kako bi se mogla koristiti bilo gdje i bilo kada.

SQL (eng. *Structured Query Language*) relacijska je baza podataka, a kako i samo ime govori, zasniva se na vezama između tablica. Često zahtijevaju planiranje strukture jer svi podatci moraju slijediti niz pravila kako bi se postigla konzistentnost i izbjegla redundancija. Da bi se njima manipuliralo, koristi se jezik strukturiranog upita (eng. *Structured Query Language - SQL*). NoSql baza podataka orijentirana je na dinamički pristup podatcima koji se ne spremaju u tablice već se

koriste u obliku dokumenata, stupaca, grafova ili para ključ-vrijednost. Jednostavnija je za upotrebu te ne zahtijeva pisanje upita da bi se manipuliralo strukturom podataka. Izrada se ne definira unaprijed te svaki dokument može imati svoju jedinstvenu strukturu što rezultira skraćivanjem vremena planiranja izrade baze podataka. Prije izrade mrežnih aplikacija korisnik mora proučiti koji su njeni zahtjevi kako bi se odlučio koju od dvije vrste baza koristiti, SQL ili NoSQL, te uzeti u obzir njihove razlike:

1. SQL je relacijska, a NoSQL je nerelacijska baza
2. SQL ima unaprijed definiranu strukturu, NoSQL ima dinamičku strukturu podataka
3. SQL baziran je na tablicama, NoSQL na dokumentima
4. SQL koristi se za strukturirane, a NoSQL za nestrukturirane podatke.

5. Aplikacija Crodemy

Mrežna aplikacija *Crodemy* platforma je koja služi za personalizirano online učenje. Tehnologije koje su doprinijele razvoju aplikacije pripadaju MERN arhitekturi gdje se veći dio aplikacije odnosi na razvojno okruženje *React.js* u kojem je napravljeno korisničko sučelje. U sklopu tog okvira, koristi se *JavaScript* programski jezik kojemu je glavna uloga izvršavanje logike, HTML i CSS namijenjeni su za vizualni dio, a zajedno su integrirani u JSX sintaksu. *Bootstrap* je implementiran unutar aplikacije kako bi bila prilagodljiva različitim uređajima poput tableta, laptopa, pametnog telefon i sličnih. Za izradu servera, odnosno, poslužitelja korišten je *Node.js* kojem je temelj *JavaScript*. Zaslužan je za komunikaciju korisničkog sučelja i baze. Svi podatci pohranjeni su u MongoDB bazu koja omogućava spremanje istih u JSON objekt te su na taj način prilagođeni jednostavnoj upotrebi.

Cilj je aplikacije učiniti online učenje jednostavnije studentima koji se nalaze na prostoru Republike Hrvatske na način da su svi video materijali napravljeni na hrvatskom jeziku. Upotrebom prethodno spomenute tehnologije ostvaren je zanimljiv dizajn, funkcionalnost i jednostavnost korištenja aplikacije.

5.1. Kreiranje projekta

Dva međusobno povezana programa osmišljena su odvojeno i čine aplikaciju. Jedan se odnosi na korisničko sučelje, a drugi je program poslužitelj zadužen za komunikaciju s bazom podataka. Prilikom izrade projekta korišten je *Visual Studio Code* u kojem se *React.js* aplikacija postavlja upisivanjem sljedećih naredbi u terminal:

```
npm install create-react-app
npm create-react-app crodemy
```

Prva je linija zadužena za instalaciju paketa što omogućava izradu *React* projekta dok druga stvara direktorij na računalu pod nazivom *crodemy*. U njemu se nalaze instalirani *node* moduli koji su potrebni za osnovni rad. Po instaliranju projekta za izradu korisničkog sučelja aplikacije, u novom terminalu kreira se program za rad u *Node.js* okruženju, a ono je orijentirano na poslužiteljsku stranu aplikacije, odnosno, server. Kako bi se stvorio ovaj program u terminalu potrebno je utipkati:

```
npm init
```

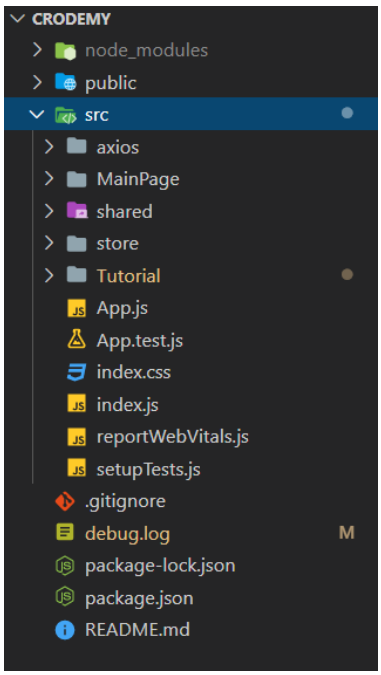
Daljnji razvoj aplikacije ovisi o planiranju, strukturiranju i implementaciji direktorija i funkcionalnosti aplikacije.

5.2. Struktura aplikacije

5.2.1. Korisničko sučelje

Aplikacija je podijeljena u skup direktorija (Slika 5.2.1), oni sadržavaju komponente koje se koriste za prikaz sučelja korisniku. Glavni dio sadržava rute, one služe za preusmjeravanje s jedne stranice na drugu i nalaze se u datoteci *App.js* koja je smještena u glavni direktorij *src*. Prije izrade ruta potrebno je uključiti module u projekt kojima je uloga usmjeravanje:

1. BrowserRouter – komponenta za korištenje URL putanja kako bi se pristupilo stranicama
2. Switch – komponenta zaslužna za pretragu tražene rute unutar svog tijela od strane korisnika
3. Route – provjerava je li zatražena ruta odgovara njenom svojstvu *path*, a kada pronađe odgovarajuću rutu izvršava prikaz stranice koja se nalazi na toj putanji

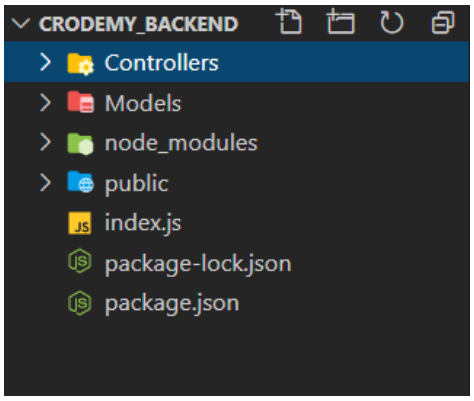


Slika 5.2.1 - Struktura direktorija Crodemy aplikacije

Nadalje, u strukturi direktorija važna je datoteka *index.js*, ona je ulazna točka u mrežnu aplikaciju (eng. *entry point*), a *index.css* zadužen je za osnovne stilove. Prilikom pokretanja, indeks.js poziva komponentu *App*, ona pregledava putanju oblika '/' i preusmjerava korisnika na početnu stranicu aplikacije.

5.2.2. Poslužitelj

U ovom dijelu aplikacije koristi se *Node.js* jer omogućava jednostavnu izradu modela i kontrolera koji su glavni akteri serverske strane. Struktura je nešto jednostavnija (Slika 5.2.2) nego kod korisničkog sučelja. U samom korijenu aplikacije nalazi se *indeks.js* datoteka koja je ulazna točka, ona je odgovorna za pokretanje poslužitelja na određenom portu te osluškuje pozive koji dolaze i poziva funkcije za izvršavanje operacije ovisno o dolazećim zahtjevima.



Slika 5.2.2 - Struktura direktorija poslužitelja

Mapa *Models* u sebi sadržava tri datoteke, dvije od njih imaju predložak objekata koji će se čitati ili spremati u bazu, a treća je zadužena za ostvarivanje veze između poslužitelja i baze podataka. Jedan od njih naziva se *users.js*, a u njemu je definirana klasa koja opisuje kako izgleda objekt korisnika. Njena definicija prikazana je sljedećim kodom:

```
class User {
    constructor(user){
        this.user_id = user.user_id
        this.first_name = user.first_name;
        this.last_name = user.last_name;
        this.email = user.email;
        this.password = user.password;
        this.courses = user.courses
    }
}
```

Kod 5.1 – Primjer modela korisnika

Drugi model naziva se *courses.js;* u njemu je definirana klasa, odnosno, predložak za stvaranje objekta jednog tečaja. Svojstva koja sadrži jedan tečaj vidljive su na sljedećem primjeru koda:

```
class Course {

    constructor(course){

        this.course_id = course.course_id

        this.course_name = course.course_name

        this.course_image = course.course_image

        this.course_description = course.course_description

        this.lessons = []

        this.users = []

    }

}
```

Kod 5.2 – Primjer modela tečaja

Mapa *Controllers* sadrži dva kontrolera u kojima se nalaze funkcije povratnog poziva te izvršavaju poslovnu logiku na poslužitelju. Kontroleri pristupaju bazi te dohvaćaju, spremaju ili brišu podatke unutar nje. Definirani kontroleri su *CourseController.js* i *UserController.js*, a kako im samo ime govori, svaki od njih izvršava promjene u bazi nad podatcima koji pripadaju tečajevima ili korisnicima.

5.2.3. Baza podataka

U bazi podataka nalaze se dvije kolekcije, odnosno, dva dokumenta. Važno je naglasiti da se ne radi o tablicama već o dokumentacijskim modelima jer je korišten MongoDB. Ovaj način manipuliranja podatcima spada u nerelacijske baze što olakšava njihovu modifikaciju, ali i upotrebu. Kada se dohvate putem poslužitelja, njihov je oblik JSON dokument i samim tim na korisničkom sučelju je jednostavnije prikazati podatke jer su proslijeđeni kao objekt. Prva kolekcija naziva se *courses* (Slika 5.2.3), u njoj se nalaze sva svojstva pojedinog tečaja, a druga kolekcija odnosi se na korisnika (Slika 5.2.4).

```
_id: ObjectId("5fd251c7752d4c7d2271b646")
course_id: "1"
course_name: "React (početnici)"
course_image: "/images/react.png"
course_description: "U ovom tečaju imate priliku naučiti osnove React frameworka za izradu ..."
lessons: Array
  0: Object
  1: Object
users: Array
  0: Object
```

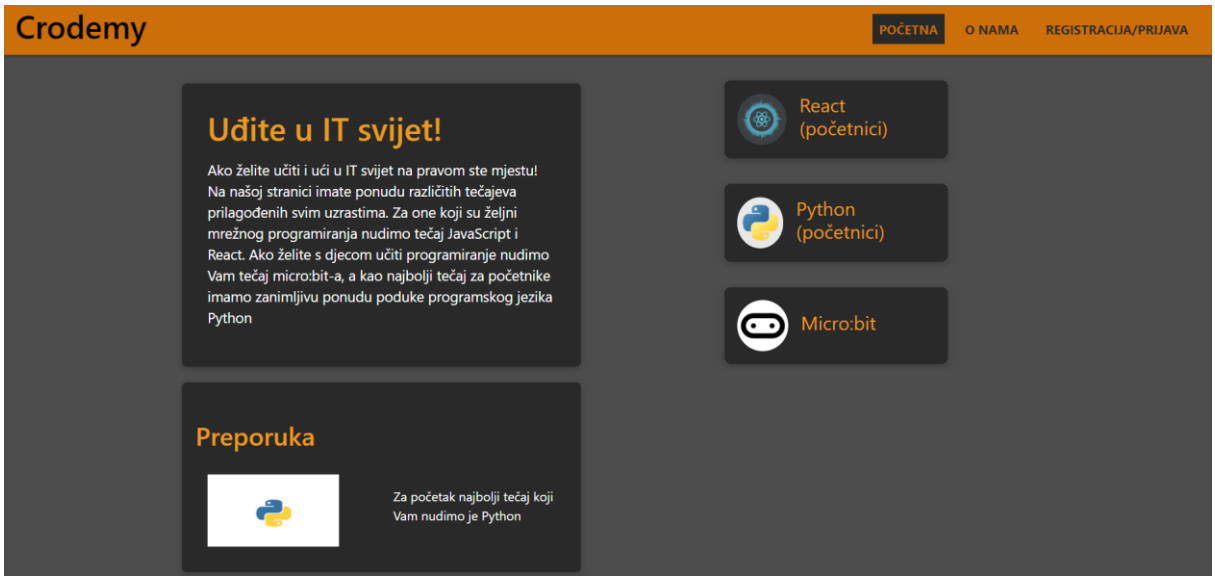

Slika 5.2.3 - Kolekcija tečaja


```
_id: ObjectId("5fd24d325a313417d4a4a33c")
user_id: "1593b79d-7dec-4b9c-8a7d-065878117d54"
first_name: "Marko"
last_name: "Janjiš"
email: "marko@markan.com"
password: "333mj903hrv"
courses: Array
  0: "1"
  1: "2"
  2: "3"
```


Slika 5.2.4 - Kolekcija korisnika

5.3. Početna stranica

Na početnoj stranici korisniku je prikazan opis i ciljevi mrežne aplikacije. Osim toga, izlistani su dostupni tečajevi uz preporuku najboljih lekcija za početnike. Sučelje (Slika 5.3.1) je jednostavno za upotrebu te je prilagođeno korisnicima koji se prvi put susreću s mrežnim aplikacijama namijenjenim za online učenje.



Slika 5.3.1 - Početna stranica Crodemy aplikacije

Na početnoj stranici nalaze se tečajevi za pohađanje, isti se dohvaćaju iz baze podataka gdje se inicijalno nalaze. Logika dohvaćanja tečaja pokreće se onog trenutka kada se dođe do ulazne točke aplikacije. Na korisničkoj strani aplikacije poziva se ruta na poslužitelju pomoću sljedeće funkcije povratnog poziva:

```
useEffect(() => {
        axios.get('/courses').then(response => {
            setCourses(response.data)
            setLoading(false)
        }).catch((error) => {
            alert("Courses not found")
        })
    },[])
```

Kod 5.3 – Poziv rute za dohvaćanje tečajeva

Biblioteka *axios* upravlja http zahtjevima koji se šalju na server. Ona sadržava funkciju `get()` koja prima putanju za poziv povratne funkcije na poslužitelju, zatim dohvaća podatke u bazi i vraća ih korisniku. Primljeni su podatci spremljeni u niz objekata, a zatim se prikazuju na korisničkom sučelju. Sljedeća linija koda prikazuje rutu za dohvaćanje svih tečajeva:

```
App.get('/courses', CourseController.getAllCourses)
```

getAllCourses pripada kontroleru *CourseController.js*, pristupa kolekciji koja se nalazi u bazi podataka. Njena je uloga dohvaćanje svih tečajeva iz baze, a potom ih vraća u obliku niza. Prilikom tog postupka koriste se ugrađene funkcije `find()` za pronalazak podataka i `toArray()`, a njena uloga je mapiranje strukture niza.

```
getAllCourses: (req,res) => {
    db.get().collection('courses').find({}).toArray((err,
result) => {
        if(err){
            res.status(500).send("Courses not found");
        }
        else{
```

```
                res.status(200).send(result);
            }
        })
    }
```

Kod 5.4 – Funkcija povratnog poziva za dohvaćanje tečajeva

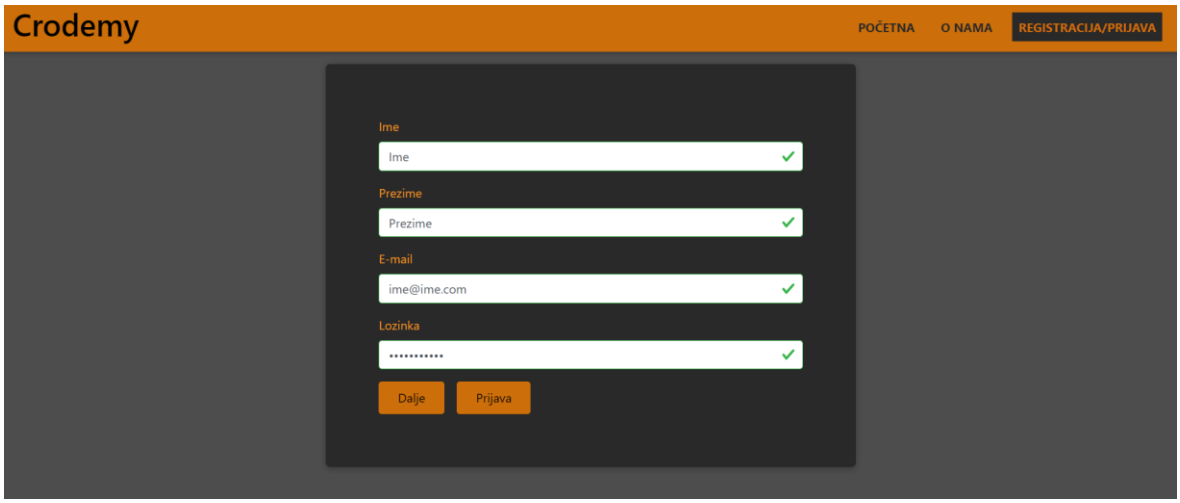
Onog trenutka kada su podatci dohvaćeni, poslužitelj ih šalje korisničkom sučelju gdje se filtriraju i prikazuju. U ovom slučaju koristi se samo naslov tečaja kao osnovni element prikaza. Za njihov prikaz korištena je modularna komponenta te svim elementima proslijeđenog niza dodjeljuje karticu s uređenim stilom.

5.4. Registracija i prijava

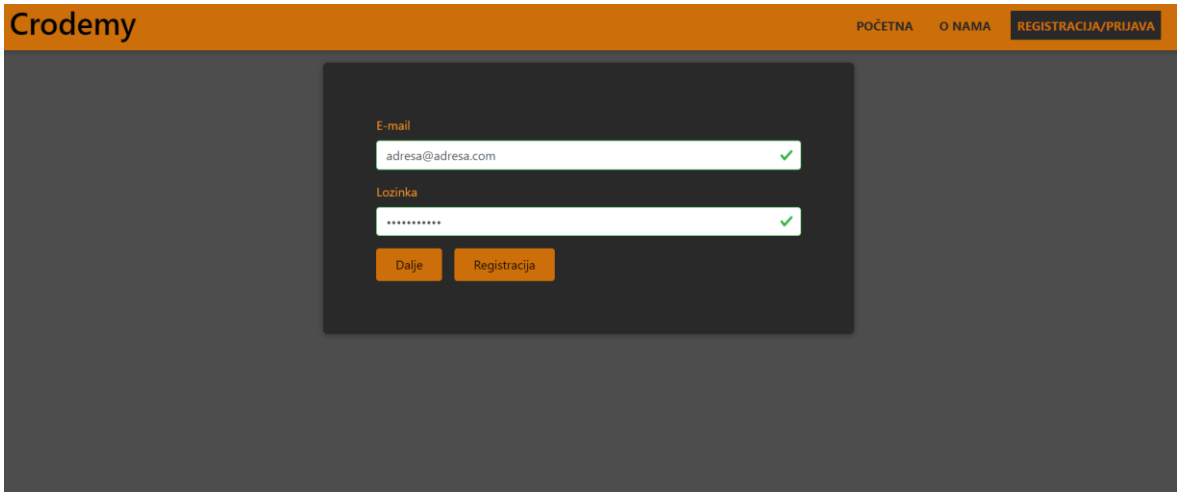
Registracija i prijava su dvije značajke koje gotovo svaka mrežna aplikacija za online učenje i podučavanje posjeduje. Prilikom izrade ove dvije komponente korišteni su elementi korisničkog sučelja (eng. *User Interface*) koji su modularni. Oni se odnose na polja za unos (eng. *input fields*) u kojima će se vršiti validacija, tj. ispravnost unosa. Njihov broj ovisi o tome je li korisnik posjetio stranicu za registraciju ili prijavu. Cijeli postupak sadrži četiri različita unosa:

- Elektronička adresa - Email
- Lozinka
- Ime
- Prezime.

Kod registracije korisnika potrebni su svi podatci - elektronička adresa, lozinka, ime i prezime (Slika 5.4.1), dok je kod prijave potrebno unijeti samo elektroničku adresu i lozinku (Slika 5.4.2).



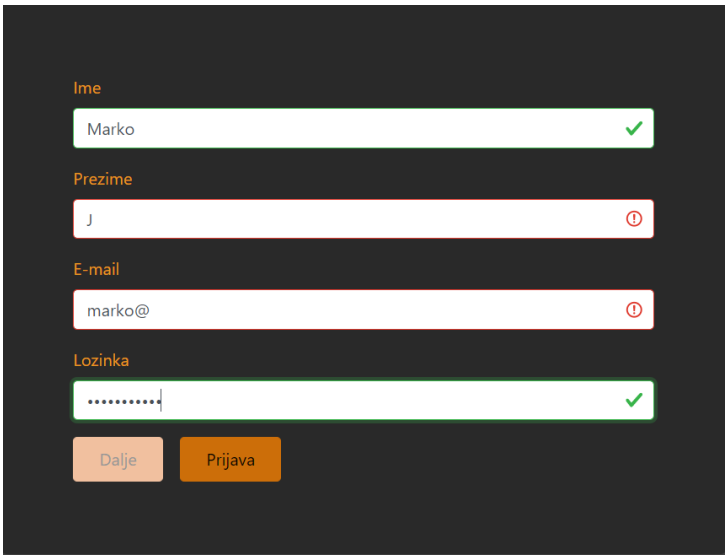
Slika 5.4.1 - Polja za unos (registracija)



Slika 5.4.2 - Polja za unos (prijava)

5.4.1. Registracija korisnika

Prilikom registracije na korisničkom sučelju odvija se logika validacije prije nego se omogući slanje zahtjeva na poslužitelj. Korisnik mora ispravno unijeti ime koje ne smije biti kraće od dva znaka, potom slijedi unos prezimena za koje vrijedi isto pravilo. Nadalje, elektronička adresa mora biti ispravnog formata, a lozinka ne smije biti kraća od sedam znakova. Ispravnost unosa signalizira se kvačicama, ako je unos ispravan, ili uskličnikom u slučaju pogreške (Slika 5.4.3) za što su zaslužni *Bootstrap* stilovi unosa. Za validaciju elektroničke adrese korišteni su regularni izrazi, a za ostale uvjet koji provjerava je li svojstvo `length` polja manje od dozvoljene granice. Da bi se validacija ispravno izvršila potrebne su kontrolne varijable koje su spremljene u lokalno stanje (eng. *local state*) *React.js* komponente *Auth.* Pet je takvih varijabli, četiri se odnose na sva polja, a peta kontrolira je li cijela forma validna.



Slika 5.4.3 - Prikaz Bootstrap stilova za validaciju

```
const [formIsValid, setFormIsValid] = useState(false)

const [emailIsValid, setEmailIsValid] = useState(false)

const [passIsValid, setPassIsValid] = useState(false)

const [firstNameIsValid, setFirstNameIsValid] = useState(false)

const [lastNameIsValid, setLastNameIsValid] = useState(false)
```

Kod 5.5 – Kontrolne varijable za validaciju (registracija)

Varijabla `formIsValid` ima važnu ulogu, ovisno o njenom stanju korisnik može poslati zahtjev za registraciju na poslužitelj. Kada je `false` botun za nastavak isključen, odnosno, ne može se poslati zahtjev, tj. onog trenutka kad se njeno stanje promjeni na `true`, korisniku je omogućeno slanje istog. Njena vrijednost zavisi o ostalim kontrolnim varijablama, tj. kada je svima stanje `true` jedino će tada i `formIsValid` biti istina.

Prilikom slanja zahtjeva na poslužitelj poziva se ruta koja pristupa *UserController.js* kontroleru. Izvršava se funkcija povratnog poziva `register`. Ona je kao parametar kroz svojstvo *body* primila podatke koji su poslani s korisničkog sučelja, odnosno, elektroničku adresu, lozinku, ime i prezime. Korištenjem modela stvara se objekt u koji se spremaju podatci te započinje proces provjere postojanja korisnika. U slučaju da baza sadrži te podatke, poslužitelj će vratiti poruku da isti već postoje pa se proces registracije neće moći izvršiti. U suprotnome, korisniku se dodjeljuje novi *id* koji se kreira zahvaljujući biblioteci *uuidv4* koja služi za generiranje jedinstvenih identifikacijskih

brojeva u *Node.js*-u. Sljedeći je korak pristupanje kolekciji *users* koja se nalazi u bazi podataka te se korištenjem ugrađene funkcije `insertOne()` sprema novi korisnik u obliku objekta.

```
user.user_id=uuidv4()
db.get().collection('users').insertOne(user,(err, result) => {
    if(err){
        res.status(500).send("Server error")
    }
    else{
        res.status(200).send("User was saved!")
    }
 } )
```

Kod 5.6 – Funkcija za stvaranje novok korisnika u bazi

Nakon završetka operacije spremanja novog korisnika, poslužitelj šalje status 200 što znači da je registracija završena. U slučaju pogreške, poslao bi se status 500 koji označava internu serversku pogrešku (eng. *Internal server error*). Kako bi se omogućila daljnja upotreba aplikacije, onog trenutka kad se izvrši registracija, korisnik je preusmjeren na stranicu za prijavu.

5.4.2. Prijava korisnika

Kao i kod registracije prije slanja zahtjeva na poslužitelj događa se validacija polja za unos. Međutim, u ovom je slučaju potrebno ispuniti dva podatka, a to su elektronička adresa i lozinka. U ovom slučaju koriste se tri kontrolne varijable zbog manjeg broja polja unosa.

```
const [formIsValid, setFormIsValid] = useState(false)
const [emailIsValid, setEmailIsValid] = useState(false)
const [passIsValid, setPassIsValid] = useState(false)
```

Kod 5.7 – Kontrolne varijable za validaciju (prijava)

Da bi se mogao pozvati zahtjev `formIsValid` mora poprimiti vrijednost `true` te ona ovisi o druge dvije kontrolne varijable. Onog trenutka kada je njeno stanje istina, korisnik klikom na botun „Dalje“ okida događaj koji izvršava poziv na poslužitelj. U tom se trenutku na server proslijeđuju

podatci koji su uneseni funkciji `login()` koja započinje proces pretrage baze podataka. U slučaju pronalaska korisnika, poslužitelj vraća status 200 i preusmjerava ga na početnu stranicu aplikacije, dok u suprotnom dolazi do pogreške statusa 404 koja označava da podatak nije pronađen. Za pretragu se koristi ugrađena funkcija `findOne()` koja je kao parametar primila elektroničku adresu i lozinku korisnika, traženje se izvršava po email-u.

Kako bi korisnik ostao prijavljen, dok god se ne odjavi, koristi se svojstvo `session`. Ono se upotrbljava zbog većeg broja zahtjeva istog korisnika, tj. svaki put kada dođe do novog upita, preko ovog se svojstva uvijek može pristupiti podatcima. Najčešće se upotrebljava kada je potrebno sačuvati sesiju korisnika ili koristiti kolačiće (eng. cookies).

5.5. Upis tečaja

Do upisa tečaja može doći samo onda kada je korisnik prijavljen. Svaki put kada dođe do prijave u lokalno stanje koje je integrirano u *React*, a zove se *Redux*, spremaju se podatci o korisniku unutar objekta `user`. Prije upisa potrebno je odabrati tečaj na početnoj stranici koji se želi upisati; njegovim odabirom korisnik je preusmjeren na stranicu (Slika 5.5.1) gdje se nalazi detaljan opis tečaja i botun „Upiši tečaj“.

Slika 5.5.1 - Prikaz detalja tečaja

Objekt `user` ujedno je i kontrolna varijabla o kojoj ovisi hoće li doći do upisa ili ne. U slučaju da je njegova vrijednost `null`, aplikacija će, u trenutku kada se stisne botun „Upiši tečaj“, preusmjeriti korisnika na stranicu za prijavu. U suprotnom će se poslati zahtjev na poslužitelj te će doći do ažuriranja podataka korisnika u bazi. Upis će se spremiti u objekt `user` te se na taj način, bez odjavljivanja, u istoj sesiji može pristupiti tečaju koji je upravo upisan.

Prilikom slanja zahtjeva, na poslužitelju se izvršava funkcija povratnog poziva `updateUserCourses()`. Kolekcija *users* koja se nalazi u bazi podataka unutar svakog objekta koji predstavlja korisnika sadrži niz u koji se spremaju identifikacijski brojevi upisanih tečajeva. Prethodno spomenuta funkcija prima elektroničku adresu i id koji sprema u niz. Da bi se izvršila operacija ažuriranja koristi se integrirana funkcija *Node.js*-a `updateOne()`.

```
updateUserCourses: (req, res) => {
        console.log(req.body)
        db.get().collection('users').updateOne(
            {email: req.body.email},
            {$addToSet: {courses: req.body.course}},
            (err, result) => {
            if(err){
                res.status(500).send("Server error")
            }
            else{
                res.send(result)
            }
        })
    }
```

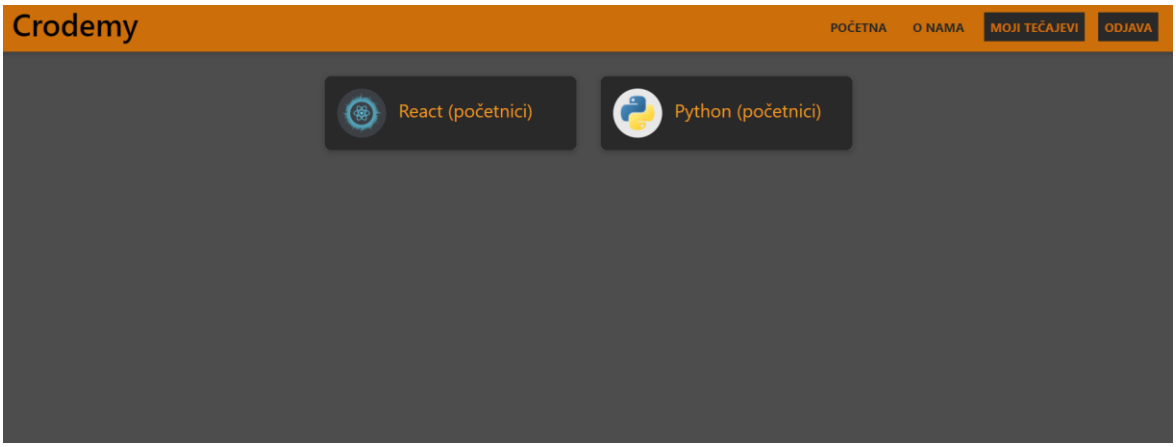
Kod 5.8 – Funkcija povratnog poziva za ažuriranje korisnikovih tečajeva

Prilikom izvršavanja prethodne funkcije server može vratiti dva rezultata, status 500 što znači da je došlo do interne serverske pogreške ili status 200, tj. potvrdu da je tečaj uspješno upisan u niz.

5.6. Korisnikovi tečajevi

Nakon prijave na platformu korisniku se u zaglavlju aplikacije, gdje se nalazi navigacijska traka, pojavljuje još jedna poveznica (eng. *link*) „Moji tečajevi“. Onog trenutka kada se poveznica potvrdi klikom, dolazi do preusmjeravanja na stranicu (Slika 5.6.1) gdje se nalaze svi tečajevi koji su upisani. Sakrivanjem poveznice omogućen je ograničeni pristup između osoba s izrađenim računom i osoba

koje ga nemaju. Da bi to bilo izvedivo, objekt `user`, ranije spomenut u prethodnim poglavljima, ima ulogu kontrolne varijable - kada je njegova vrijednost `null`, poveznica neće biti prikazana.



Slika 5.6.1 - Stranica korisnikovih tečajeva

Da bi se dohvatili tečajevi, na klijentskom dijelu aplikacije nalazi se komponenta koja prilikom prikazivanja izvršava funkciju za pristupanje ruti na poslužitelju. Podatci koje server vrati u obliku su niza te se spremaju u lokalno stanje kako bi se proslijedili sljedećoj komponenti koja će ih prikazati.

```
const [courses, setCourses] = useState([])
```

Na poslužitelju se nalazi ruta za dohvaćanje korisnikovih tečajeva, a njena je glavna uloga pozivanje `getUserCourses()` funkcije. U tom trenutku server pretražuje kolekciju u bazi podataka, a kao parametar traženja uzima identifikacijski broj korisnika kojem traje sesija. Korištenjem `findOne()` funkcije vraća se rezultat koji u sebi sadrži tečajeve korisnika. Pri završetku tog poziva, započinje novi koji, uspoređivanjem korisnikovih s onima u bazi, vraća niz objekata upisanih tečajeva.

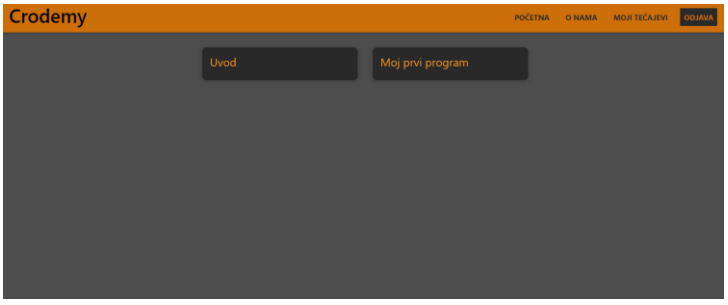
```
db.get().collection('users').findOne({user_id: req.session.user_id
}, (err, result) => {
            if(err){
                res.status(500).send("Internal server error")
            }
            else{
                db.get().collection('courses').find({
                    course_id: {$in: result.courses}
```

```
            }).toArray((err, result) => {
                if(err){
                    res.status(404).send("Course not found")
                }
                else{
                    if(result===null){
                        res.status(404).send("Not found")
                    }
                    else{
                        res.status(200).send(result)
                    }
                }
            })
        }
    })
```

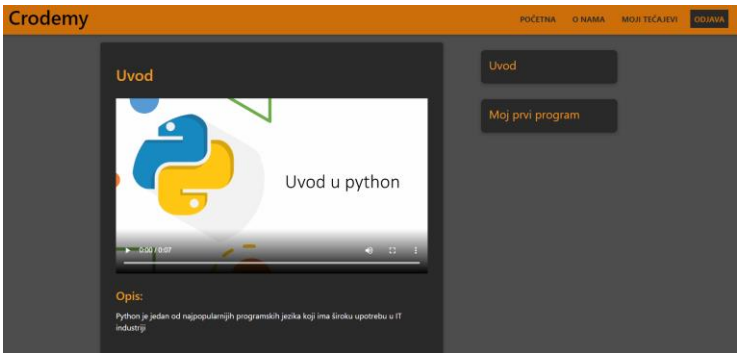
Kod 5.10 – Funkcija za dohvaćanje korisnikovih tečajeva

5.7. Lekcije

Lekcije u aplikaciji mogu se podijeliti na dvije mrežne stranice, jedna prikazuje popis svih lekcija tečaja (Slika 5.7.1) dok se drugoj pristupa pritiskom na jednu od njih i prikazuje se njen sadržaj, tj. naslov, opis i digitalni sadržaj (Slika 5.7.2). Logika za prikaz svih lekcija slična je onoj za dohvaćanje korisnikovih tečajeva, ali se razlikuje u primanju parametara o kojima će biti riječi u ovom poglavlju nešto kasnije.



Slika 5.7.1 - Stranica s popisom svih lekcija tečaja



Slika 5.7.2 - Stranica s detaljnim prikazom lekcije

Na klijentskoj strani za ispis naslova svih lekcija koristi se niz koji je dohvaćen s poslužitelja. U trenutku kada korisnik odabere jedan od svojih tečajeva, uzima se njegov jedinstveni identifikator preko kojeg se vrši pretraga na poslužitelju te se dobiju detalji istog.

Na serveru se podatci dohvaćaju iz baze podataka u obliku objekta koji sadrži niz pod nazivom *lessons* nakon čega se na klijentu spremaju u lokalno stanje preko kojeg se prikazuju na stranici. Funkcija povratnog poziva namijenjena za pretragu naziva se `getLessonData()` i kao parametar prima identifikator tečaja. U slučaju da se podatci ne mogu dohvatiti, poslužitelj će ispisati poruku statusa 404 koja upućuje na to da tečaj ne postoji, a u protivnom će biti status 200 te će se klijentu vratiti rezultat pretrage.

```
getLessonData: (req,res) => {
        db.get().collection('courses').findOne({
            course_id: req.params.id
        }, (err, result) => {
            if(err){
                res.status(404).send("Course not found")
            }
            else{
                if(result===null){
                    res.status(404).send("Not found")
                }
                else{
                    res.status(200).send(result)
```

```
                    }
                }
            })
        }
```

Kod 5.11 – Funkcija povratnog poziva za pretragu tečaja

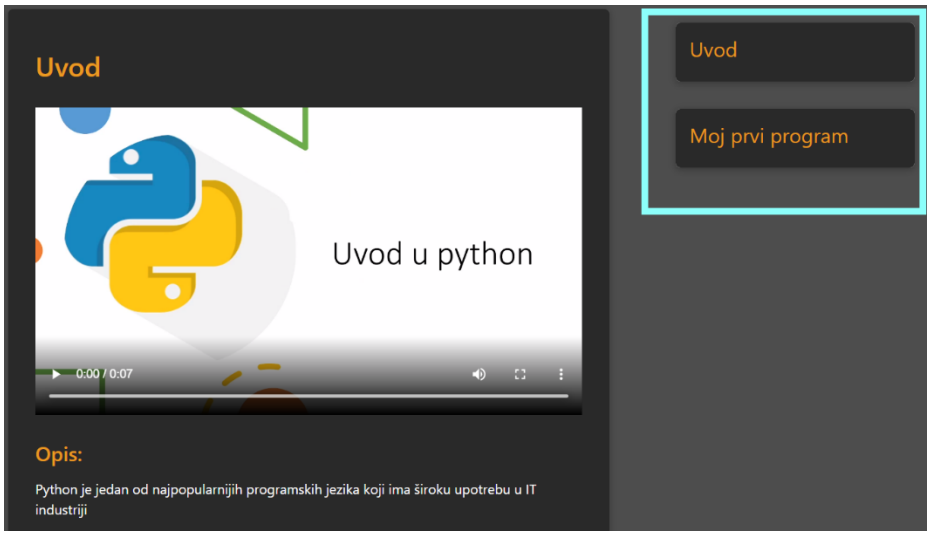
Druga stranica koja se odnosi na prikaz lekcije u sebi sadrži sve njene detalje. Podatci nisu dohvaćani s poslužitelja jer su spremljeni u prethodnoj komponenti koja ih proslijeđuje ovoj stranici onog trena kada se odabere specifična lekcija. U nizu svaka od njih posjeduje svoj jedinstveni identifikator, a on služi za pristup elementu niza, tj. lekciji čije je detalje potrebno prikazati. Kod ovog dijela aplikacije važno je napomenuti da se prikazuje digitalni sadržaj za učenje, odnosno, video. U programiranju mrežnih aplikacija ključno je vrijeme izvođenja programa, zbog čega je važno kako će se takav medij prikazati. Video se ne smije dohvatiti u potpunosti jer bi se na taj način gubila efikasnost, već je potrebno postepeno dohvatiti paketić videa koji će se prikazivati. Kako je React.js okruženje koje pripada otvorenim izvorima biblioteka (eng. *open source*) i pruža mogućnost upotrebe različitih modula, prikaz videa ne stvara problem. Oni su spremljeni na poslužitelju dok se u bazi podataka nalazi putanja do njih. Na klijentskoj strani aplikacije potrebno je instalirati paket pod nazivom `react-player`, a njegova instalacija vrši se upisivanjem sljedeće naredbe u terminal:

```
npm install react-player
```

Ova biblioteka pruža mogućnost postepenog učitavanja videa s poslužitelja, samo je potrebno njenoj komponenti proslijediti putanju. Ima i ugrađena svojstva koja programeru olakšavaju prikaz kontrola te podešavanje okvira za prikaz videa.

```
<ReactPlayer url={video} controls={true}/>
```

Na ovoj stranici nalazi se još jedna funkcionalnost koja olakšava korisniku upotrebu mrežne aplikacije. Na desnoj strani zaslona klijent može dinamički promijeniti lekciju (Slika 5.7.3) zahvaljujući lokalnom stanju prethodne komponente koja cijelo vrijeme pruža pristup cjelokupnom tečaju.



Slika 5.7.3 - Dinamički prikaz lekcija

5.8. Odjava korisnika

Pri završetku korištenja aplikacije korisnik ima mogućnost odjave. U zaglavlju Crodemy-ja nalazi se poveznica „Odjava“ na koju je potrebno kliknuti. Tim postupkom okida se događaj koji na poslužitelju poziva rutu `logout`. U tom trenutku izvršava se istoimena funkcija povratnog poziva u čijem tijelu dolazi do prekidanja sesije trenutnog korisnika. Da bi se prekinula, koristi se funkcija `destroy()` koja pripada objektu `session` o kojem se govorilo u prethodnim poglavljima.

```
logout: (req, res) => {
        if(req.session.user){
            req.session.destroy((err)=>{
                if(err){
                    console.log(err);
                    res.status(500).send('Server error')
                }else{
                    res.status(200).send("Successfuly logout");
                }
            });
        }
        else{
```

```
            res.status(400).send('Session not found');
        }
    }
```

Kod 5.12 – Funkcija povratnog poziva za prekidanje sesije korisnika

Zaključak

Cilj ovog rada bio je stvaranje mrežne aplikacije koja će pojedincu omogućiti personalizirano online učenje na hrvatskom jeziku. Prilikom izrade aplikacije, korištene su razne tehnologije koje služe za razvoj mrežnih aplikacija. Ova činjenica govori o tome da izrada edukacijskog sustava zahtjeva znanja u različitim područjima IT industrije, a samim tim iziskuje puno vremena kako bi se izradila. Poznavanje React-a, Node.js-a i MongoDB baze podataka uvelike olakšava stvaranje sustava ovog tipa, no kada bi došlo do daljnjeg razvoja, poželjnije bi bilo koristiti relacijsku bazu podataka.

Današnji sustavi, koji služe za provođenje nastave na daljinu, znatno su razvijeniji od Crodemy aplikacije te pružaju niz funkcionalnosti koje nisu opisane u ovom radu. Neki od nedostataka ove aplikacije odnose se na povratne informacije učenika i nedostatak uloga (instruktor, učenik i administrator). Ovo je jedan početak gdje je definiran smjer u kojem bi se Crodemy trebao kretati i važno je da se njegov razvoj nastavi kako bi došlo do osvježenja obrazovnog sustava.

U prethodnim poglavljima moguće je uočiti napredak tehnologije te važnost tehnologije u procesu učenja. Više nije upitno trebamo li težiti korištenju tehnologija za učenje i poučavanje već moramo planirati na koje sve načine tehnologiju možemo upotrijebiti za ostvarenje cilja.

Literatura

[1] D. Jansen i R. Schuwer., (2015.), *Institutional MOOC strategies in Europe*, Maastricht, NED, EADUT, p. 7-9

[2] I. Elaine Allen i J. Seaman, (2014.), *Grade Change - Tracking Online Education in the United States*, nepoznato, USA, Babson Survey Reserch Group and Quahog Research Group, Ch. 1

[3] Udemy, (2021.), https://www.udemy.com/, (Zadnji pristup 06.01.2021.)

[4] A. Shahzad, et. al., (2020.), *Effects of COVID-19 in E-learning on higher education institution students: the group comparison between male and female*, *Quality & Quantity journal,* 54

[5] K. Gutierrez, (18.05.2018.), *SHIFT blog, https://www.shiftelearning.com/blog/top-instructional-design-models-explained,* (Zadnji pristup: 12.02.2021.)

[5] D. Merrill et al., (2008.), *Handbook of research on educational comunications and technology*, 3rd ed., New York, USA, Lawrence Erlbaum Associates, pp. 176-177

[6] E. Krull i K. Oras, (2010.), *Promoting student teacher's lesson analysis and observation skills by using Gagne's model of an instructional unit, Journal of Education for Teaching*, 36 (2), pp. 197-210, https://www.researchgate.net/publication/233464221_Promoting_student_teachers'_lesson_analysis_and_observation_skills_by_using_Gagne's_model_of_an_instructional_unit, (Zadnji pristup 10.02.2021)

[8] J. Sabatura (27.08.2013.), *Using Bloom's Taxonomy to Write Effective Learning Objectives*, University of Arkansas, USA, https://tips.uark.edu/using-blooms-taxonomy/, (Zadnji pristup 12.02.2021.)

[7] C. Ou, D. A. Joyner i A. K. Goel, (2019.), *Designing and developing video lessons for online learning: A seven-principle model*, Online Learning, 23 (2), pp. 89-90

https://link.springer.com/article/10.1007/s11135-020-01028-z, (Zadnji pristup 06.01.2021.)

[8] Giannakos M., (2013), Usability Design for Video Lectures, *EurolTv13*, pp. 163-164, https://www.researchgate.net/publication/262235488_Usability_design_for_video_lectures, (Zadnji pristup 31.01.2021.)

[9] M. Kaluža i B. Vukelić, (2020.), Comparison of front-end frameworks for web applications development, *Journal of the Polytechnic of Rijeka,* 6 (1), pp. 261-282 https://www.bib.irb.hr/945431, (Zadnji pristup 10.01.2021.)

[10] MongoDB, (2021.), https://www.mongodb.com/mern-stack, (Zadnji pristup 15.01.2021.)

[11] Coursera, (2021.), https://www.coursera.org/. (Zadnji pristup 06.01.2021.)

[12] Khan Academy, (2021.), https://www.khanacademy.org/, (Zadnji pristup 07.012021.)

[13] w3schools, (2021.), https://www.w3schools.com/nodejs/nodejs_http.asp, (Zadnji pristup 20.01.2021.)

[14] Nigam A., (18.03.2019.), *How to understand component's lifecycle methods in ReactJS*, https://www.freecodecamp.org/news/how-to-understand-a-components-lifecycle-methods-in-reactjs-e1a609840630/, (Zadnji pristup 10.02.2021.)

Slike