Web aplikacija za upravljanje projektima unutar tvrtke

Kiš, Marijo

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

Sveučilišni diplomski studij

WEB APLIKACIJA ZA UPRAVLJANJE PROJEKTIMA UNUTAR TVRTKE

Diplomski rad

Marijo Kiš

Osijek, 2021.

SADRŽAJ

1. UVOD

Zadatak ovog diplomskog rada je izraditi web aplikaciju koja će pomoći i olakšati koordinaciju na projektima između zaposlenika neke tvrtke. Iako je aplikacija primjenjiva za razne tvrtke, u ovome radu primarno će biti prikazano korištenje sustava unutar poduzeća koje se bavi razvojem softverskih rješenja.

Koordinacija na projektima postaje zahtjevnija proporcionalno tome što je projekt kompleksniji i što više ljudi na njemu radi. Projekt ima svoj početak i očekivani rok završetka koji se može znatno premašiti ukoliko osobe koje rade na projektu ne surađuju na ispravan način. Vrlo lako se može dogoditi da se neka sitnica pretvori u veliki problem koji je kasnije teško ispraviti i otkloniti nakon što sustav naraste i izgradi se na krivim temeljima. Također, vođenje zaduženja za pojedine zadatke, informacije o zaposlenicima, količina odrađenih sati po zadatku i ukupno po projektu te ostale informacije relevantne za menadžment i administraciju, mogu postati nepregledni ukoliko se te informacije vode na zasebnim sustavima koji međusobno ne komuniciraju.

U drugom poglavlju predstavljen je pregled područja problematike nakon čega su pojašnjeni sami pojmovi projekta i projektnog menadžmenta te primjena tehnologije u toj sferi. Četvrto poglavlje donosi opis i pojašnjenja korištenih tehnologija. Peto poglavlje sastoji se od objašnjenja procesa izrade baze podataka, poslužiteljske aplikacije pa sve do klijentske, a nakon njega dolazi poglavlje o korištenju aplikacije te na koncu i zaključak.

1.1. Zadatak diplomskog rada

Izraditi web aplikaciju koja bi olakšala upravljanje i koordinaciju zaposlenika neke tvrtke u radu na različitim projektima. Predvidjeti sljedeće uloge: administrator koji može dodavati nove zaposlenike, generirati izvješća o radu na projektima, uvid u radne sate zaposlenika na pojedinim projektima, kreiranje projekta i postavljanje voditelja projekta; voditelj projekta koji može uključiti zaposlenike na rad na projektu, ima uvid u radne sate na projektu i može dodjeljivati poslove suradnicima; suradnik na projektu može vidjeti zadane poslove i evidentirati odrađeni posao.

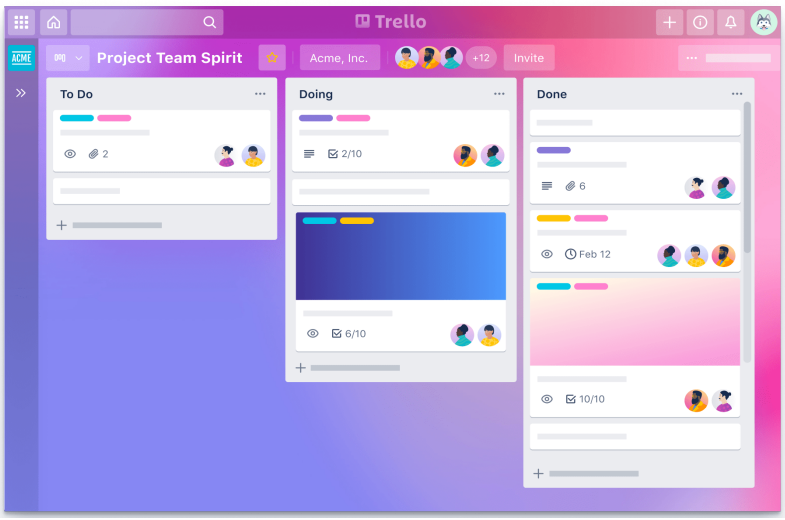
2. PREGLED PODRUČJA

Postoje razna softverska rješenja za probleme i izazove modernog projektnog menadžmenta. Na tvrtki je da odabere ono koje je najbolje za njezin način rada i krene sa sustavnom implementacijom odabranog softvera. Nikada ne postoji idealan softver, svaki ima svoje prednosti i nedostatke u odnosu na konkurenciju, no najbitnije je izabrati onaj koji će zadovoljiti najviše zahtjeva načina i politike samog poslovanja tvrtke.

Kako je za potrebe ovog diplomskog rada izrađena web aplikacija, tako su i postojeća rješenja koja su uzimana u obzir također web aplikacije (ili aplikacije koji imaju i svoju online inačicu).

2.1. Trello

Trello je softver za organizaciju na projektu koji se temelji na *kanban* pločama (slika 2.1.). Dolazi s unaprijed definiranim temama koje se mogu birati prema opisu i sferi projekta. Korake projekta je moguće podijeliti u željene dijelove (npr. dizajn, razvoj ili test), a zadatke je moguće smjestiti u pojedine korake. Za svaki zadatak je moguće definirati dodatne informacije kao što su rok izvršenja ili zadužena osoba. Prebacivanje zadataka je vrlo jednostavno samim povlačenjem iz jednog koraka u drugi. Moguće je dodavati komentare i pratiti listu svih aktivnosti na zadatku/projektu, a spajanjem s vanjskim softverima omogućavaju se funkcionalnosti kao što su automatske obavijesti i mnoge druge.

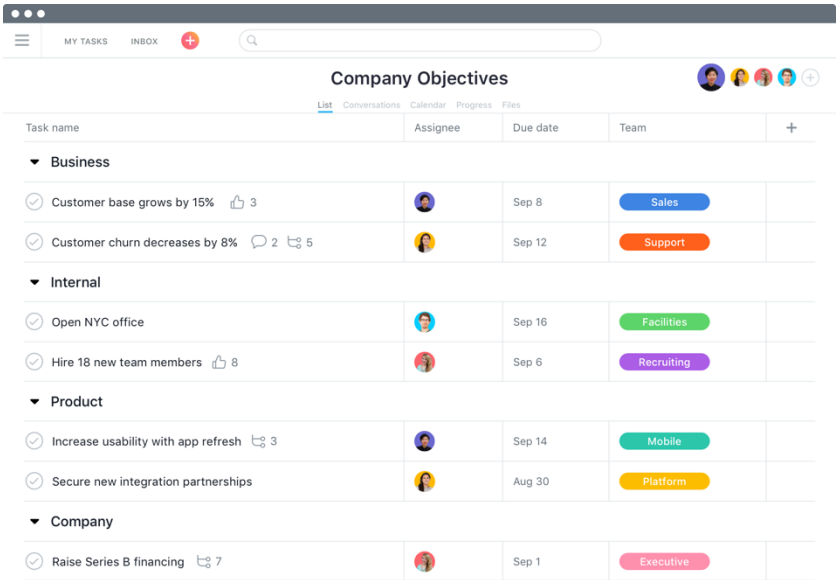


Sl. 2.1. *Trello*[1]

[1] Preuzeto: https://trello.com/ (6.9.2021.)

2.2. Asana

Asana svoje rješenje temelji na principu liste zadataka, no taj aspekt podiže na razinu tima. Omogućava podjelu projekta na sekcije i podliste (slika 2.2.), a dolazi s analitikom za pregled bitnih faktora projekta (npr. pregled postotka riješenih zadataka). Također podržava definiranje timova te komunikaciju kako bi tim bio što efikasniji i brži čak i ako rade na različitim mjestima.

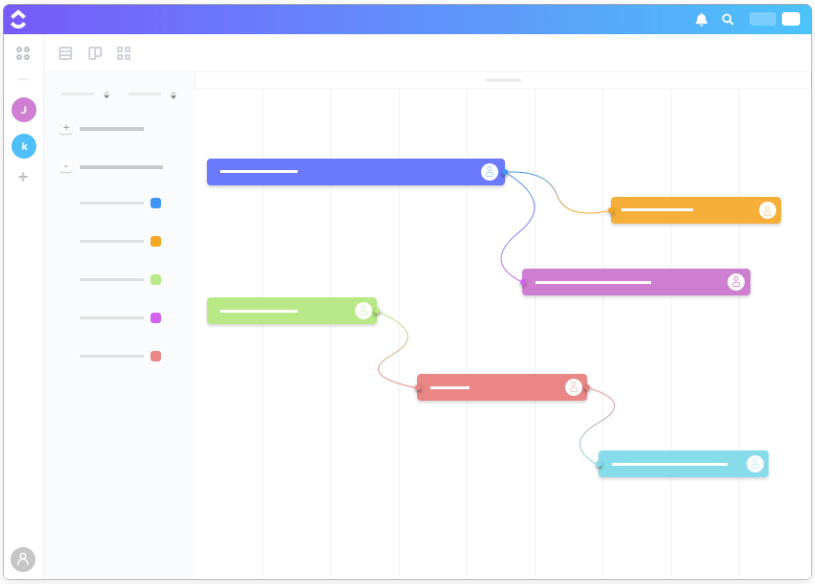


Sl. 2.2. *Asana*[2]

2.3. ClickUp

ClickUp donosi raznolike mogućnosti prikaza tijeka izvođenja zadataka od jednostavne liste pa sve do kompliciranijih prikaza kao što su *Ganttogram*[3] i umne mape. Sam proces postavljanja projekta, otvaranje zadataka i dodjeljivanje istih zaposlenicima, nije jednostavan kao npr. kod *Trella*, ali je olakšan preciznim vodičem. Nudi široki spektar prikaza najvažnijih pokazatelja uspješnosti provođenja projekta, a u isto vrijeme je skoro potpuno besplatan. Na slici 2.3. prikazana je info grafika sučelja aplikacije.

[2] Preuzeto: https://asana.com/product (6.9.2021.)
[3] Grafički prikaz toka projekta gdje se zadatci smještaju na vertikalnu os dok horizontalna os predstavlja vrijeme

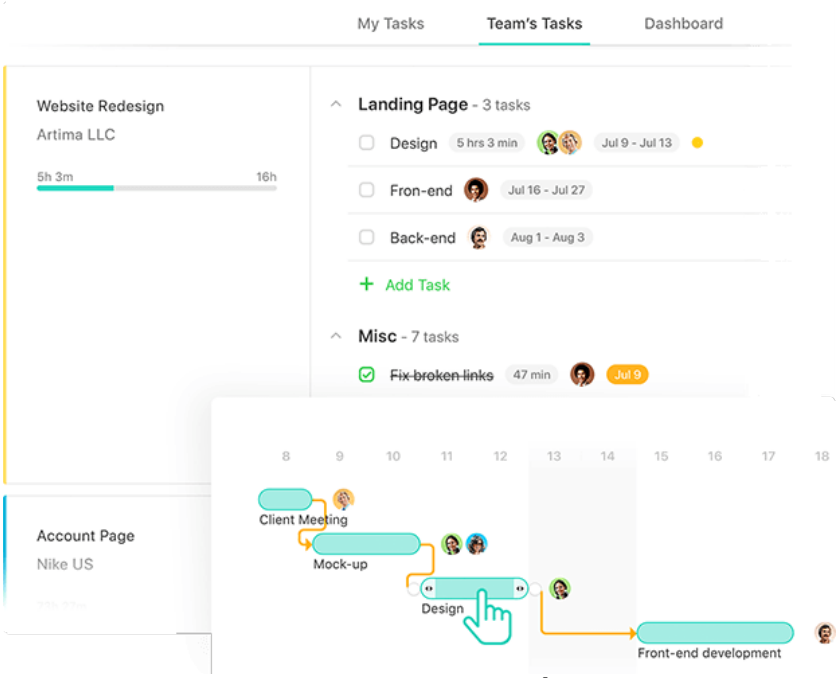


Sl. 2.3. *ClickUp*[4]

2.4. Paymo

Paymo uz mogućnosti prioritiziranja i podjelu zadataka, također nudi i opciju praćenja vremena provedenog na pojedinom zadatku. Sastoji se od tri glavne skupine u koje je moguće organizirati svoj rad: klijenti, projekti i zadatci. Kada dođe vrijeme za ispostavu računa klijentu, *Paymo* nudi opciju kreiranja izvještaja koji nudi sve najbitnije informacije o odrađenom poslu kao što je prikaz provedenog vremena po svakom zadatku ili vizualizacija odrađenog posla kroz grafove. Također, omogućava automatsko postavljanje naplate klijentima putem sustava za plaćanje (npr. uz *PayPal*). Na slici 2.4. prikazan je primjer korištenja aplikacije.

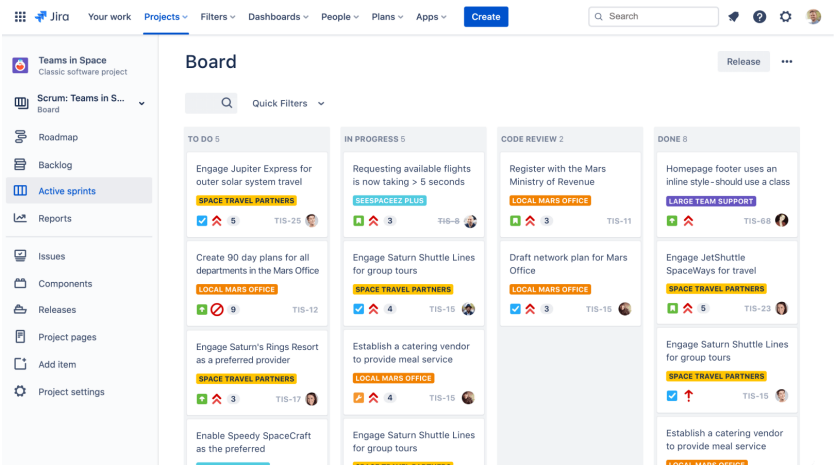
[4] Preuzeto: https://clickup.com/ (6.9.2021.)



Sl. 2.4. *Paymo*[5]

2.5. Jira

Jira (slika 2.5.) je izrađena za *Scrum* i *Agile* načinu rada. Pri samoj prijavi, korisniku se postavlja nekoliko pitanja prema kojima aplikacija predlaže najbolju metodologiju rada. Izvrstan je i popularan alat među programerima i inženjerima jer omogućava jednostavnu integraciju s ostalim alatima za razvoj kao što su *GitHub*, *GitLab*, *Sentry* i *Jenkins*.



Sl. 2.5. *Jira*[6]

[5] Preuzeto: https://www.paymoapp.com/tour (6.9.2021.)
[6] Preuzeto: https://www.atlassian.com/software/jira (6.9.2021.)

3. PROJEKTNI MENADŽMENT

Projekt je definiran kao privremena težnja za stvaranje i nastanak jedinstvenog proizvoda ili usluge. Projekt mora imati jasno definiran početak i kraj, a ono što ga dijeli od svakodnevnog poslovanja jesu jedinstvenost i privremenost. Putem projekta se stvara, dok se menadžmentom organizira, planira, rukovodi i nadgleda, odnosno osigurava se ispravan proces izvođenja, a putem spone između projekta i menadžmenta nastao je projektni menadžment.

Projektni menadžment označava planiranje, upravljanje i organiziranje aktivnosti i resursa na projektu, a sve u svrhu točne provedbe definiranih zahtjeva i osiguravanja izvršavanja projekta u definiranom vremenu, odnosno postizanje projektnih ciljeva. Glavne funkcije su mu: odlučivanje, planiranje, organiziranje, upravljanje ljudskim resursima, vođenje, kontroliranje i upravljanje promjenama.

Razvojem računalnih tehnologija, računala i softverska rješenja ušla su u sva područja života pa tako i u projektni menadžment. Softverska rješenja postala su primjenjiva za sve vrste projekata, a objedinjuju alate koji omogućavaju jednostavnije planiranje, organiziranje te kontinuirano kontroliranje izvedbe projekta. Prve računalne primjene i rješenja za ovu problematiku javljaju se sredinom 70-ih godina prošloga stoljeća, no minus su im bili sporost, kompleksnost i ograničenost. Napretkom računarstva softveri za projektni menadžment su postali sve moćniji i sve jednostavniji za korištenje. Današnja najpopularnija i najzastupljenija rješenja iz ove domene su: *Microsoft Project*, *PeopleSoft*, *Planview* i drugi.

4. KORIŠTENE TEHNOLOGIJE

U ovom će poglavlju biti objašnjene korištene tehnologije za izradu aplikacije na kojoj se ovaj diplomski rad temelji. Pojasnit će se tri glavne sastavnice razvoja web korisničkog sučelja, Node.js, Adonis.js, MySQL, Nuxt.js, Vuetify.js te Git.

4.1. Tri glavne sastavnice razvoja web korisničkog sučelja

Govoreći o razvoju web korisničkog sučelja, nameću se tri sastavnice koje omogućavaju izradu virtualnog i logičkog dijela korisničkog sučelja (slika 4.1.), a to su: HTML (engl. *Hyper Text Markup Language*), CSS (engl. *Cascading Style Sheets*) i JavaScript.

HTML je opisni jezik za kreiranje HTML dokumenata. Prvi se put spominje 1991. godine i definirao ga je Tim Barners-Lee. HTML se temelji na posebnim oznakama pomoću kojih se gradi struktura sadržaja dokumenta. Na temelju tih oznaka, preglednik prikazuje sadržaj. Elementi su najčešće sačinjeni od početne i završne oznake dok se sadržaj nalazi između njih. Tako tvoreni elementi mogu sadržavati i atribute koji definiraju dodatnu informaciju o elementu i navode se u početnoj oznaci. Atributi su većinom par imena i vrijednosti. Trenutno aktualni standard jezika je HTML5 usvojen 2014. godine.

Način na koji će se HTML elementi prikazati definira CSS opisni jezik. Predložio ga je kreator Håkon Wium Lie u vremenu kada je radio s Timom Bernersom-Leejem u CERN-u. U to vrijeme se pojavilo još nekoliko opisnih jezika za stil, no CSS je prihvaćen 1996. godine. Preglednici čitaju pravila prikaza definirana u CSS-u i na taj način mijenjaju prikaz sadržaja HTML elementa (npr. promjena boje). Pravilo se sastoji od selektora kojim se dohvaća željeni element koji se želi izmijeniti te od deklaracijskog bloka sa skupom svojstava i vrijednosti koja će utjecati na prikaz. Aktualni standard je CSS3.

JavaScript je dinamični skriptni programski jezik visoke razine. Izvršava se direktno bez pretvorbe u strojni jezik (interpretativnost). Iako je prvotno kreiran za web preglednike, unutar zadnjih godina je popularan i na poslužiteljskoj strani, a također pronalazi primjenu i u izradi računalnih te mobilnih aplikacija. Podržava više programskih paradigmi (objektno orijentiranu, imperativnu i funkcionalnu), a temelji se na prototipovima putem kojih se ostvaruje nasljeđivanje. JavaScript je razvio Brendan Eich 1995. godine. Standardizira se prema Ecma International[7] radi isporuke programskog jezika standardiziranog na

[7] Europska udruga za standardizaciju informacijskih i komunikacijskih sustava

međunarodnoj razini. Najnovija verzija je ECMAScript 2021, no kako web preglednici još ne koriste funkcionalnosti i mogućnosti ovoga standarda, a dosta njih ni neke ranije, najaktualniji standard prema zastupljenosti na tržištu je ECMAScript 2015 ili skraćeno ES6.

Sl. 4.1. *HTML, CSS i JavaScript*[8]

4.2. Node.js

Prvotno, JavaScript je korišten samo u web preglednicima koji mu pomoću svojih pokretača omogućavaju izvođenje, no s vremenom se javila želja i potreba izvođenja JavaScripta i izvan preglednika, što zbog samih mogućnosti jezika, a što zbog njegove široke primjene u web razvoju. Jedna od platformi koja je doskočila rješavanju ove potrebe je Node.js. Kako bi se omogućilo izvođenje izvan preglednika, Node koristi Googleov V8 pokretač, isti onaj koji koristi Google Chrome preglednik. Node predstavlja paradigmu „*JavaScript svuda*“, odnosno objedinjuje razvoj web aplikacije oko jednog programskog jezika. Razvio ga je Ryan Dahl 2009. godine. Godinu kasnije, predstavljen je i rukovatelj paketima pod nazivom npm (engl. *Node Package Manager*). Sama jezgra Nodea sadrži ključne funkcionalnosti, dok se one naknadno mogu nadograđivati i proširivati putem raznih paketa. Temelji se na programiranju pokretanom događajima i asinkronosti (ne dolazi do blokiranja među dijelovima koda). Na slici 4.2. prikazan je logo Nodea.

[8] Preuzeto: https://p92.com/binaries/content/gallery/p92website/technologies/htmlcssjs-overview.png (9.7.2021.)

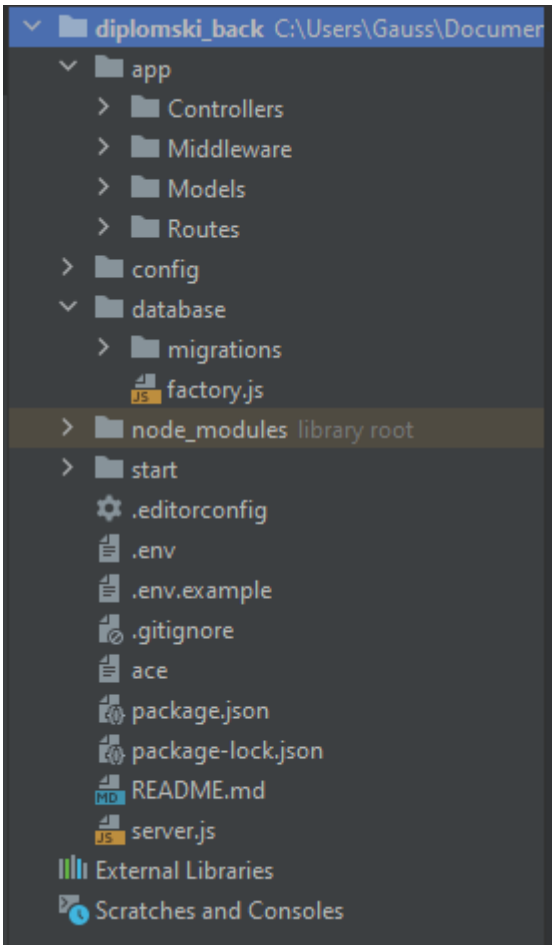
Sl. 4.2. *Node.js*[9]

4.3. Adonis.js

Adonis.js je Node.js MVC[10] razvojno okruženje. Omogućava rukovanje HTTP (engl. *Hypertext Transfer Protocol*) zahtjevima, radnje na bazama podataka, autentikaciju korisnika i još mnogo toga. Razvio ga je Aman Virk.

Princip rada i komunikacije između klijenta i poslužitelja je sljedeći: HTTP zahtjevom koji se šalje s klijenta rukuje Adonisov *Server modul* koji izvršava sve među-programe (engl. *middleware*) na poslužiteljskoj razini. Ukoliko je sve u redu, idući na red dolazi *Router* koji pokušava naći rutu koja odgovara zatraženom URL-u (engl. *Uniform Resource Locator*). Ukoliko se ruta ne pronađe, izbacuje se iznimka da ruta nije pronađena, a u slučaju povoljnog ishoda, pozivaju se svi globalni među-programi i među-programi definirani na zatraženoj ruti. Ako nitko od njih ne prekine proces zahtjeva, sljedeći na redu je rukovatelj tom rutom unutar kojega je potrebno prekinuti zahtjev. Adonis tada poziva među-programe koji su definirani da se pozivaju na prekid zahtjeva i podatci se šalju klijentu.

[9]Preuzeto:https://upload.wikimedia.org/wikipedia/commons/thumb/d/d9/Node.js_logo.svg/330px-Node.js_logo.svg.png (9.7.2021.)
[10] Model-View-Controller; uzorak za dizajn softvera koji aplikaciju dijeli na više dijelova, odnosno na model, pogled i kontroler



Sl. 4.3. *Struktura projekta*

Na slici 4.3. je prikazana struktura poslužiteljske strane aplikacije diplomskog rada. Unutar *app* direktorija smještena je logika aplikacije. Direktorij *app/Controllers* sadrži sve HTTP i WS (engl. *WebSocket*) upravljače (engl. *controller*). Upravljači se vežu na jednu ili više ruta te objedinjuju srodne logike upravljanja zahtjevom (npr. logika za dohvaćanje, uređivanje i brisanje korisnika). *App/Midleware* sadrži sve definirane među-programe. Omogućuju transformacije na zahtjevu klijenta ili odgovoru poslužitelja. *App/Models* objedinjuje sve modele, dok *app/Routes* sadrži datoteke unutar kojih se povezuju metode HTTP zahtjeva s pripadajućom metodom upravljača. Unutar *config* direktorija nalaze se postavke aplikacije kao što su postavke CORS-a (engl. *Cross-Origin Resource Sharing*), baze podataka i drugih. Migracije su dokumentirane promjene na bazi podataka, a mogu se pronaći unutar *database* direktorija gdje se nalazi migracija za svaku pojedinu tablicu. Kako bi se koristile migracije, prvotno se mora uključiti pružatelj migracija (engl. *Migrations Provider). Start* sadrži sve datoteke koje se pokreću pri samome podizanju aplikacije. Na slici 4.4. prikazan je logo Adonis razvojnog okruženja.

Sl. 4.4. *Adonis.js*[11]

4.4. MySQL

MySQL (slika 4.5.) je jedna od najpopularnijih baza podataka otvorenog koda. To je relacijska baza koju je razvila kompanija MySQL AB (današnji Oracle Corporation) 1995. godine. Pisana je u C i C++ programskom jeziku. Temelji se na strukturnom upitnom jeziku, SQL-u (engl. *Structured Query Language*), pomoću kojega se vrši dodavanje, pretraživanje, izmjena i brisanje podataka. SQL je neproceduralni jezik koji kao naredbe koristi riječi iz engleskog jezika (npr. *select*, *update*, *delete*) zbog čega je vrlo intuitivan za korištenje.

Sl. 4.5. *MySQL*[12]

[11]Preuzeto:https://res.cloudinary.com/practicaldev/image/fetch/s--N1ri0IYn--/c_imagga_scale,f_auto,fl_progressive,h_720,q_auto,w_1280/https://dev-to-uploads.s3.amazonaws.com/uploads/articles/2r7pwf06szmde50ryv93.png (9.7.2021.)
[12] Preuzeto: https://www.mysql.com/common/logos/includes-mysql-167x86.png (9.7.2021.)

4.5. Nuxt.js

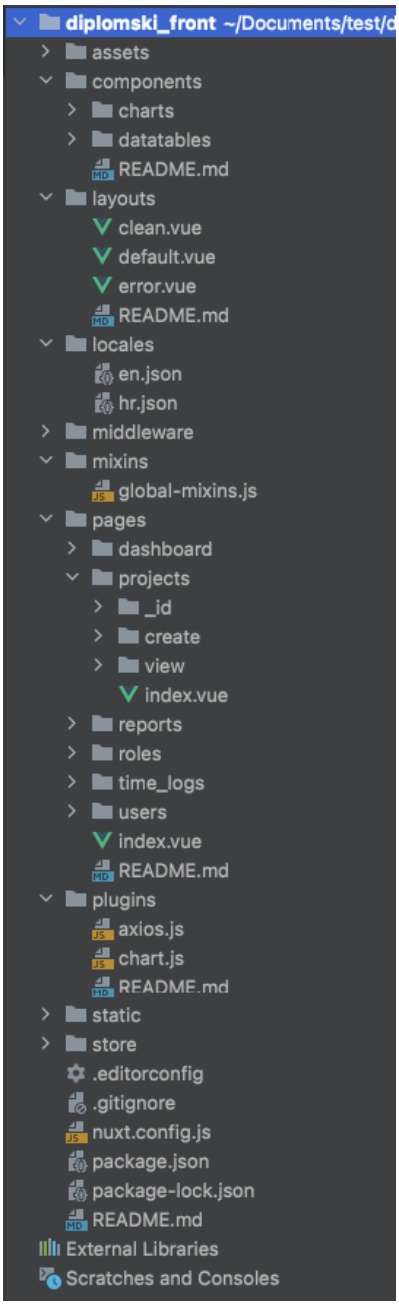
Nuxt.js je web okvir otvorenog koda u čijim su temeljima Node.js, Vue.js, Babel.js i Webpack. Prva verzija puštena je 2016. godine, a razvili su ju Alexandre Chopin, Sebastien Chopin i Pooya Parsa. Na slici 4.6. nalazi se logo Nuxt.js web okvira.

Sl. 4.6. *Nuxt.js*[13]

Krasi ga izrazito modularna arhitektura što uvelike olakšava poslove kao što su npr. dodavanje Google analitike ili PWA (engl. *Progressive Web App*) mogućnosti kroz službene pakete koji se jednostavno instaliraju putem npm-a. Samom instalacijom dobiva se maksimalno optimizirana aplikacija. Njegova jednostavnost, intuitivnost i jasna dokumentacija čine ga lako upotrebljivim čak i onima koji su tek zakoračili u svijet razvoja klijentskih aplikacija.

U usporedbi sa Vue.js okvirom na kojemu se temelji, beneficije koje Nuxt.js donosi su automatsko generiranje usmjerivača (engl. *router*) na temelju strukture datoteka i direktorija, lakše upravljanje meta oznakama, SEO (engl. *Search Engine Optimization*) poboljšanja te SSR (engl. *Server-Side Rendering*) način izvođenja. U SSR načinu, web preglednik šalje zahtjev koji dolazi do Node.js poslužitelja. Zatim se generira HTML koji će se potom poslati nazad pregledniku. Nakon što preglednik dobije sadržaj, nastupa Vue.js hidratizacija, odnosno proces u kojemu sadržaj postaje vidljiv i interaktivan. Tada navigaciju među stranicama preuzimaju Nuxt poveznice na klijentskoj strani, stoga se više ne kontaktira poslužitelj sve dok se grubo ne osvježi stranica.

[13] Preuzeto: https://nuxtjs.org/logos/nuxt.svg (9.7.2021.)



Sl. 4.7. *Struktura projekta*

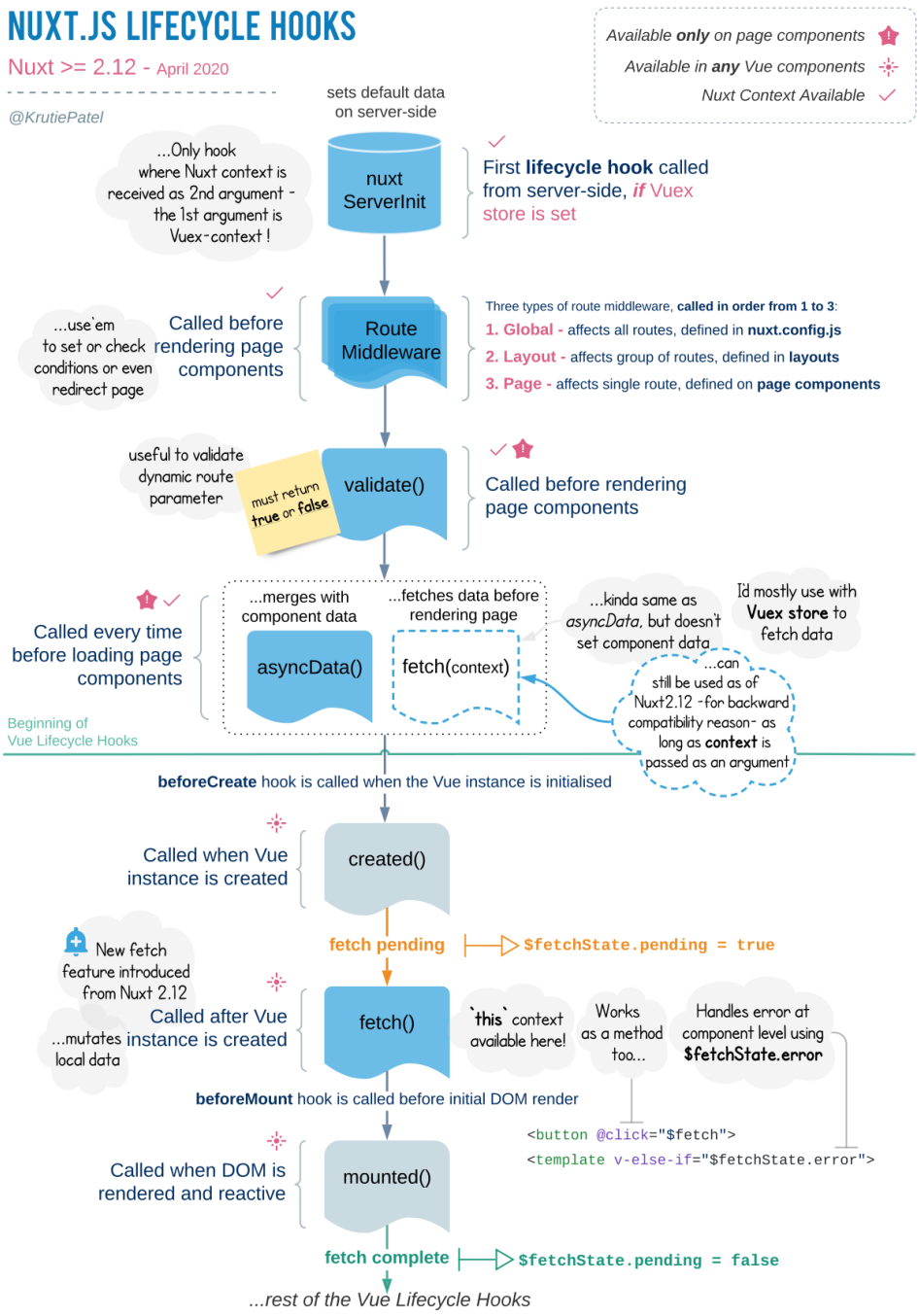
Govoreći o strukturi projekta (slika 4.7.), prvi na red dolazi *.nuxt* direktorij, odnosno direktorij gradnje. Automatski je generiran i skriven. Kreira se pokretanjem naredbe *nuxt dev* ili *nuxt build*. *Assets* direktorij sadrži datoteke kao što su *Sass* (engl. *Syntactically Awesome Style Sheets*) datoteke i fontovi. Nuxt pruža mogućnost pisanja vlastitih Vue.js komponenti koje se zatim mogu uvesti na neku stranicu, raspored (engl. *layout*) ili u neku drugu komponentu. Komponenta sadrži strukturu unutar *template*, logiku unutar *script* i stilove unutar *style* oznaka, a smještaju se unutar *components* direktorija. *Layout* direktorij omogućava

kreiranje rasporeda koji će se koristiti na više stranica. *Default.vue* raspored bit će automatski primijenjen na sve stranice na kojima ne bude specificirano drugačije. Npr. u *default.vue* može se definirati ladica za navigaciju i traka aplikacije koje će se zatim primjenjivati kroz cijelu aplikaciju nakon prijave, dok će se za stranicu prijave definirati poseban *clean.vue* raspored koji neće sadržavati nikakve elemente te će se on eksplicitno navesti kao aktivan raspored na toj stranici. *Middleware* direktorij sadrži sve među-programe koji će se pozivati prije iscrtavanja same stranice ukoliko se na njoj specificira primjena tog među-programa. Upotreba u praksi ovakvog jednog među-programa bila bi zaštita pristupa određenim stranicama ukoliko korisnik nije prijavljen u sustav. Osim što bi ga poslužitelj odbio HTTP status kodom 401[14] ili 403[15], na klijentskoj strani među-program vrši definirano preusmjeravanje na stranicu za prijavu sve dok se korisnik ne prijavi, nakon čega mu je omogućen pristup sustavu i među-program ga više neće preusmjeravati. Svi pogledi i rute smješteni su unutar *pages* direktorija kojega Nuxt čita i automatski prema njima konfigurira usmjerivač. Dodatna mogućnost je kreiranje dinamičkih stranica na način da im se ispred imena doda donja crtica (znak _). Na taj način usmjerivač će tako kreiranu stranicu promatrati kao nešto promjenjivo, odnosno nešto što neće imati fiksno ime. Primjer ovakve prakse bilo bi kreiranje singlice[16] za uređivanje korisnika koja bi tada imala naziv *_id.vue* i bila bi smještena unutar *users* direktorija. Usmjerivač bi tada mogao navigirati na (npr.) *users/1* ili *users/2* stranicu koja bi imala jednu strukturu, stil i logiku, a ovisno o identifikatoru koji je postavljen umjesto *_id*, korisnik bi pomoću tog identifikatora slao zahtjev poslužitelju za dohvaćanje drugog korisnika, odnosno drugi podatci bi bili prikazani na stranici nakon dohvaćanja podataka. *Plugins* direktorij sadrži JavaScript proširenja koja se dodaju aplikaciji i ona se pokreću prije instanciranja korijena aplikacije. *Static* direktorij sadrži sve datoteke koji se neće mijenjati, Nuxt ih automatski poslužuje i može im se pristupiti putem korijenskog URL-a projekta. Nuxt.js dolazi konfiguriran kako bi pokrio većinu standardnih zahtjeva korisnika, no ukoliko je potrebno dodatno konfiguriranje, to se može učiniti unutar *nuxt.config.js* datoteke.

Kako bi se razvojno okruženje što kvalitetnije moglo koristiti, važno je razumjeti što se događa u pozadini. Izvođenje Nuxta podijeljeno je u tzv. faze života prikazane na slici 4.8.. Važno ih je ispravno koristiti jer se neke faze odvijaju na poslužiteljskoj strani dok se druge

[14] Neautoriziran korisnik
[15] Zabranjeno
[16] Stranica koja predstavlja jedan resurs iz skupa resursa; npr. stranica za prikaz jednog korisnika

odvijaju na klijentskoj (kod SSR načina izvođenja) što može prouzročiti rušenje ili neispravan rad aplikacije ako se fazama ne rukuje na ispravan način.



Sl. 4.8. *Prikaz Nuxtovih faza života*[17]

[17] Preuzeto: https://nuxtjs.org/docs/2.x/concepts/nuxt-lifecycle (9.7.2021.)

4.6. Vuetify.js

Vuetify.js (slika 4.9.) je Vue knjižnica *Material Design* komponenti za bržu izradu korisničkog sučelja. Stvorio ga je John Leider, a prva verzija svijetlo dana je ugledala samim krajem 2016. godine. Pruža unaprijed definirane komponente koje već sadrže strukturu, stil i logiku. Dokumentacija je detaljna s mnogo primjera, a putem raznih atributa i pratitelja događaja, komponentama se može vrlo lako izmijeniti izgled, način rada ili čak i proširiti dostupne funkcionalnosti. Vuetify samom instalacijom bez dodatnog postavljanja omogućava ispravan rad na svim uređajima, od mobitela pa sve do stolnih računala. Također pruža i dodatne funkcionalnosti kao što su definiranje proizvoljnih točaka lomljena za različite veličine zaslona, pristupačnost (a11y), dvosmjernost (LTR/RTL) za čitanje i pisanje s lijeva na desno i obrnuto, internacionalizaciju (i18n), definiranje tema i druge.

Sl. 4.9. *Vuetify.js*[18]

4.7. Git

Git je alat otvorenog koda koji je stvorio Linus Torvalds 2005. godine u svrhu nadziranja distribuiranog razvoja jezgre operacijskog sustava Linux. Alat omogućava verzioniziranje koda i praćenje rada različitih sudionika na projektu te jednostavno spajanje različitih cjelina iz različitih izvora (autora) u jedinstvenu cjelinu. Najvažnija osobina Gita je distribuiranost što označava pojam prema kojemu svaki sudionik ima svoju lokalnu instancu koda koju podiže na udaljeni repozitorij nakon odrađenih promjena.

Neki od popularnijih Git davatelja usluge su GitHub (slika 4.10., lijevo) i GitLab (slika 4.10., desno).

[18] Preuzeto: https://pbs.twimg.com/media/Ei5n6vBWoAEy5gp.png (9.7.2021.)

Sl. 4.10. *GitHub*[19] *(lijevo) i GitLab*[20] *(desno)*

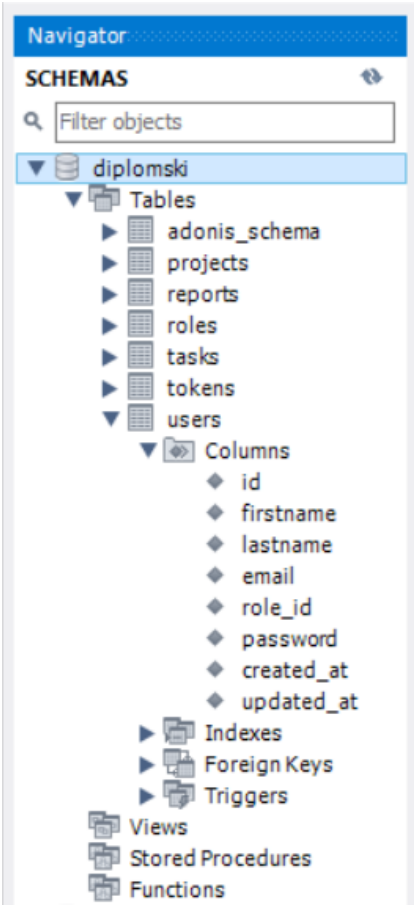
[19] Preuzeto: https://github.com/logos (10.7.2021.)
[20] Preuzeto: https://about.gitlab.com/press/press-kit/ (10.7.2021.)

5. PROCES IZRADE

Ovo poglavlje opisuje proces izrade aplikacije te dodatno pojašnjava njezine najvažnije dijelove. Prvo je ukratko objašnjeno modeliranje i kreiranje baze podataka, zatim poslužiteljska strana te naposljetku i klijentska strana izrađenog sustava.

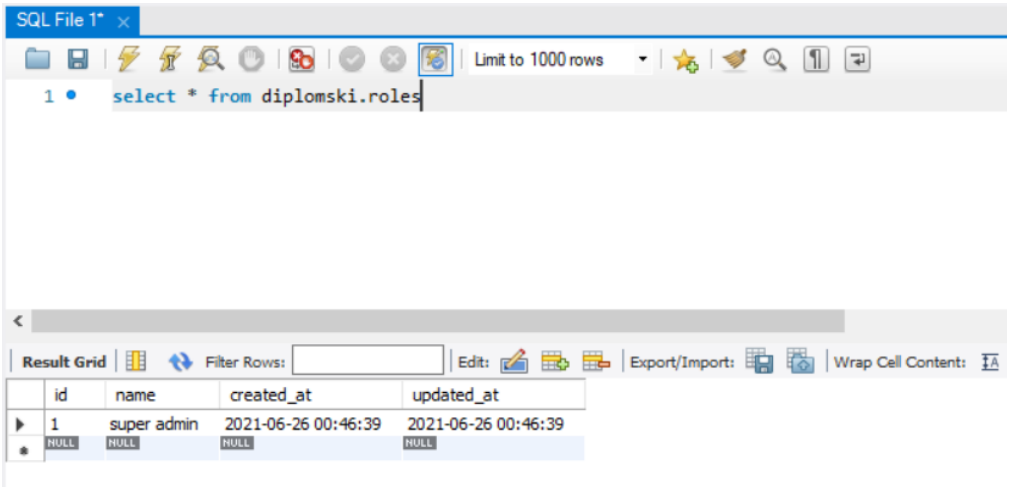
5.1. Baza podataka

Korištena baza podataka za potrebe ovog rada je MySQL relacijska baza podataka i MySQL Workbench alat koji omogućava akcije na bazi putem SQL upita. Pri instalaciji MySQL Workbencha, odrađuje se također i konfiguriranje postavki veze kao što su korisničko ime i lozinka putem kojih se pristupa instanci te priključak (engl. *port*) na kojemu će se nalaziti baza. Uobičajeni priključak za MySQL protokol je 3306, ali moguće je i to izmijeniti prilikom instalacije ili naknadnim otvaranjem instance na nekom drugom slobodnom priključku.



Sl. 5.1. *Prikaz sheme baze podataka*

Na slici 5.1. prikazana je shema izrađene baze podataka. Baza nema kreiranih pogleda, procedura niti funkcija. Kreirane tablice mogu se pronaći unutar *tables* direktorija. Tablica *adonis_schema* je tablica koju automatski kreira Adonis i u nju zapisuje informacije o provedenim migracijama kao što su naziv migracije i vrijeme zadnjeg pokretanja svake od njih. Ostale tablice (osim *tokens*) kreirane su migracijama kroz Adonis okvir. Na slici 5.2. prikazan je primjer dohvaćanja svih upisa SQL upitom iz tablice *roles*. Atributi koji se nalaze u toj tablici su *id* koji se kreira sam i automatski se povećava, *name* kojega klijent šalje poslužitelju te *created_at* i *updated_at* koji označavaju vrijeme upisa i vrijeme zadnje izmjene.

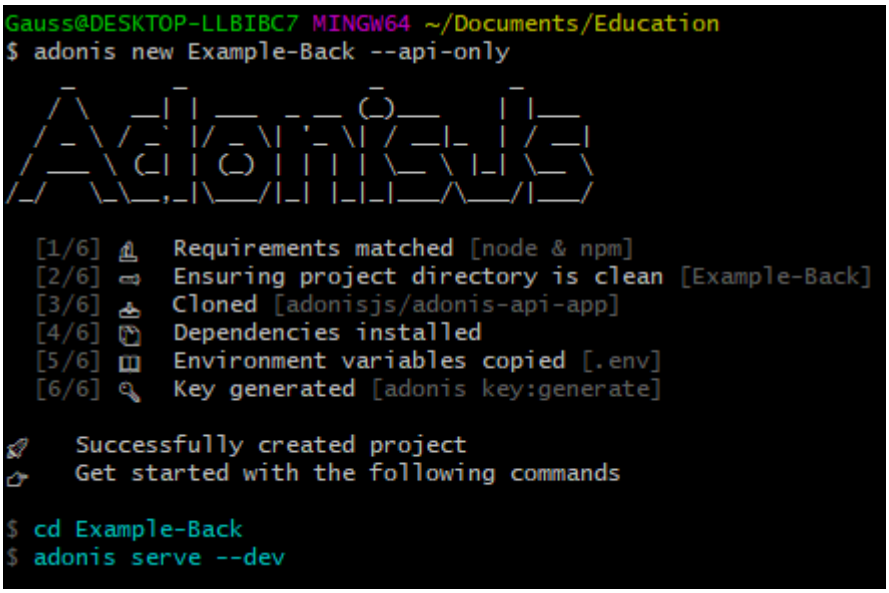


Sl. 5.2. *Dohvaćanje podataka iz tablice*

Spajanje na bazu podataka vrši se na poslužiteljskoj strani konfiguracijom i pružanjem pristupnih podataka te priključka na kojemu je baza postavljena, a proces će biti detaljnije objašnjen u idućem potpoglavlju.

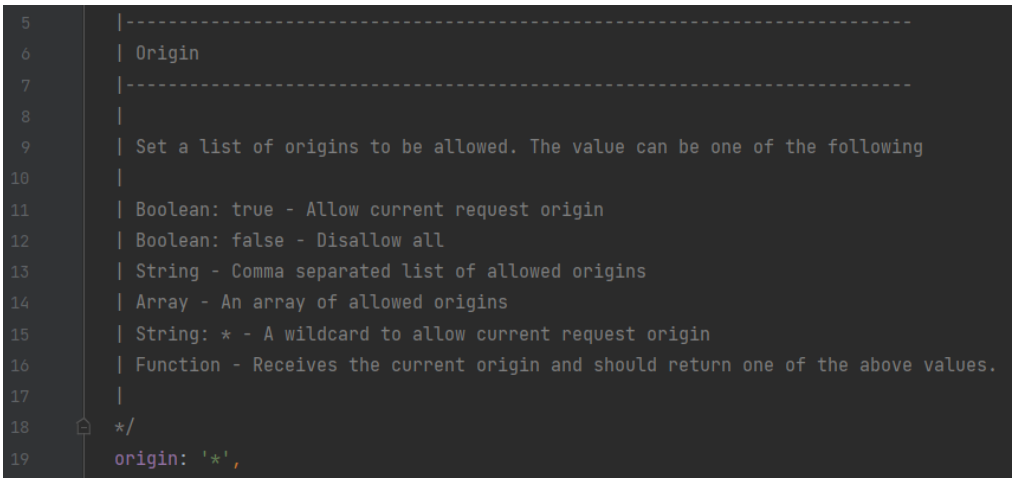
5.2. Poslužiteljska aplikacija

Poslužiteljska aplikacija izrađena je u Adonis.js razvojnom okviru. Kreiranje projekta vrši se putem Adonis CLI alata naredbene linije. Upisivanjem naredbe *adonis new Example-Back* (slika 5.3.) kreira se projekt pod nazivom *Example-Back*, a dodavanjem opcije *–api-only*, projekt će doći unaprijed postavljen za kreiranje API-ja (engl. *Application Programming Interface*) (ostale dvije mogućnosti su bez dodavanja opcije koja će stvoriti *fullstack* projekt ili *slim* koja će pružiti samo ključne mogućnosti okvira). Nakon kreiranja, projekt je moguće pokrenuti naredbom *adonis serve --dev*.



Sl. 5.3. *Kreiranje Adonis projekta*

Prvi korak je instalacija *mysql npm* paketa kako bi se omogućilo korištenje i spajanje s priređenom MySQL bazom podataka te dodavanje postavki unutar *.env*[21] datoteke s varijablama za definiranje adrese, priključka, korisničkog imena, lozinke i naziva baze podataka. Također, ukoliko se projekt pokreće lokalno, kako bi se riješili problemi s CORS-om (slika 5.4.), potrebno je putem oznake * dopustiti sve izvornike (engl. *origins*).



Sl. 5.4. *Konfiguracija postavki CORS-a*

[21] Tekstualna datoteka koja sadrži varijable koje se žele predati okruženju aplikacije (engl. application's environment)

Adonis koristi *Lucid* modele koji su implementacija *Activ Record* obrasca u JavaScriptu. Razlozi korištenja su jednostavno dohvaćanje modela, API koji omogućava implementaciju relacija, definiranje *hookova* kako bi kod poštivao DRY (engl. *Don't Repeat Yourself*) načelo, mijenjanje podataka u hodu, serijaliziranje i mnogi drugi.

Na slici 5.5. prikazan je model za korisnika. Unutar *boot* metode nalaze se funkcije koje će se izvesti samo jednom pri *bootanju* modela. *Trait* omogućava dodavanje funkcionalnosti „izvana prema unutra“, odnosno dodavanje metoda klasi modela. Kreiranje *traita* vrši se unutar *Models/Traits* direktorija. Svaki *trait* mora imati *register* metodu koja kao parametre prima klasu modela i opcije. Također, moguće je i proširiti Adonisov *QueryBuilder* pa se kasnije na modelu može pozivati definirana funkcija.

Hook je akcija koja se izvodi prije ili nakon definirane operacije na bazi podataka. Jedan od primjera (koji se može vidjeti i na slici 5.5.) je *heširanje* korisnikove lozinke prije samog upisa u bazu podataka. Adonisov *hash provider* koristi *bcrypt* algoritam koji je izrazito spor što ga čini vrlo sigurnim jer gotovo je nemoguće da napadač iz njega izvuče stvarnu vrijednost. Sporost algoritma i kompleksnost probijanja i dolaska do inicijalne vrijednosti temelji se na broju iteracija koje će algoritam napraviti prije no što vrati *heširanu* vrijednost. Razlika između *heširanja* i enkripcije je ta što se od *heširane* vrijednosti ne može dobiti originalna (osim putem izrazito kompleksnih procesa koje većinom koriste napadači) dok se kod enkriptiranih vrijednosti izvršava dekripcija putem tajnih ključeva kako bi se dobila originalna poruka.

Na modelu je moguće definirati i relacije s drugim modelima. Prilikom definiranja nije potrebno relacije dodati na oba modela. Relacija *hasMany* definira jedan na više vezu što znači da jedan korisnik može imati više tokena (prema primjeru sa slike 5.5.) dok je *belongsTo* inverzna relacija, odnosno definira jedan na jedan vezu, ali na način da uloga u sustavu pripada korisniku pa se zato relacija na ovaj način definira na modelu korisnika.

Lucid modeli donose razne statične metode koje mogu zamijeniti korištenje *Query Builder* sučelja. Tako je uz primjenu *hidden* metode moguće sakriti neki od atributa s modela (u ovom slučaju skriva se korisnikova lozinka, odnosno uklanja se s odgovora poslužitelja).

```
'use strict'

const Model = use('Model')

const Hash = use('Hash')

class User extends Model {
  static boot() {
    super.boot()

    this.addTrait( trait: 'Paginable')

    this.addHook( forEvent: 'beforeSave', handlers: async (userInstance) => {
      if (userInstance.dirty.password) {
        userInstance.password = await Hash.make(userInstance.password)
      }
    })
  }

  tokens() {
    return this.hasMany( relatedModel: 'App/Models/Token')
  }

  role(){
    return this.belongsTo( relatedModel: 'App/Models/Role')
  }

  static get hidden() {
    return ['password']
  }
}

module.exports = User

```

Sl. 5.5. *User model*

Migracije su dokumentirane mutacije na bazi podataka i zahtijevaju dvije metode: *up* i *down*. *Up* metoda koristi se za akcije na tablici. Može biti korištena za kreiranje nove tablice ili za izmjenu postojeće. *Down* metoda može povratiti promjene iz *up* metode.

Nazivlje migracija sastoji se od trenutne vremenske oznake kreiranja migracije i imena same tablice na koju se migracija odnosi.

Na slici 5.6. nalazi se migracija za *users* tablicu. Moguće je definirati tip i naziv podatka pojedinoga stupca (npr. *string* ili *integer* metode koje primaju naziv stupca kao argument, a

metoda *string* još prima i maksimalan broj znakova koji podatak može sadržavati), zatim obveznost pojedinog podatka metodom *notNullable*, jedinstvenost putem metode *unique* ili vezu s drugim tablicama pomoću metode *references* koja postavlja strani ključ.

```
'use strict'

/** @type {import('@adonisjs/lucid/src/Schema')} */
const Schema = use('Schema')

class UserSchema extends Schema {
  up () {
    this.create('users', (table) => {
      table.increments()
      table.string('firstname', 30).notNullable()
      table.string('lastname', 30).notNullable()
      table.string('email', 80).notNullable().unique()
      table.integer('role_id').unsigned().references('roles.id').notNullable()
      table.string('password',128).notNullable()
      table.timestamps()
    })
  }

  down () {
    this.drop( args: 'users')
  }
}

module.exports = UserSchema

```

Sl. 5.6. *Migracija za users tablicu*

Uloga upravljača (engl. *controller*) je odgovoriti na HTTP zahtjev. Upravljači objedinjuju srodnu logiku u jednu datoteku, a vežu se na jednu ili više ruta. Moguće im je pristupiti samo preko rute na kojoj su registrirani, a način registracije je prikazan na slici 5.7.. Definiraju se HTTP metoda, putanja rute te naziv metode iz upravljača. Dodatno je moguće grupirati srodne rute i definirati im prefiks putanje.

Sl. 5.7. *Definiranje ruta i pripadajućih metoda iz upravljača*

Slika 5.8. prikazuje *UserController* upravljač i metodu za kreiranje korisnika (nije prikazan cijeli upravljač iz projekta već samo jedna metoda na kojoj će se detaljnije razjasniti kreiranje upravljača). Metoda *createUser* kao argumente prima *request* i *response* klase s *HTTP Context* objekta. Metodom *post* dohvaćaju se svi podatci iz tijela zahtjeva te se primjenom *sanitize*[22] funkcije uz *normalize_email* iz dobivenog maila uklanjaju svi nepotrebni znakovi. Sljedeći korak je validacija podataka uz definirana pravila kao što su npr. *required* (obveznost) ili *confirmed* (zahtjeva slanje istovjetno ponovljenog podatka na kojemu je pravilo definirano). Ukoliko validacija padne na nekom od podataka, kreiranje korisnika se prekida, a u suprotnom se vrši dodatna provjera postoji li već dobivena email adresa zapisana u sustavu. Ukoliko sve validacije prođu, pozivom metode *create* na modelu i predajom primljenih podataka, vrši se upisivanje u tablicu nakon čega se šalje odgovor kako je zahtjev uspješno završen.

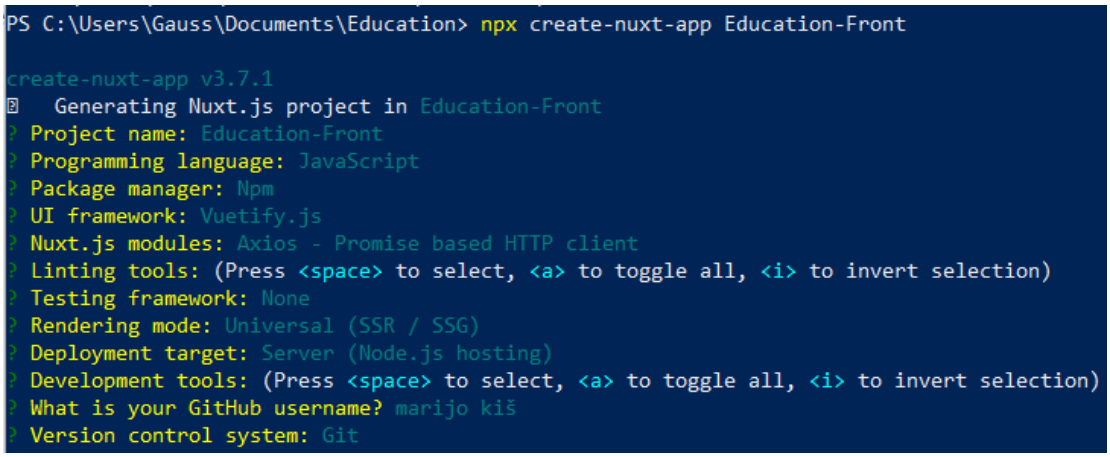
[22] Metoda *sanitize* omogućuje funkcionalnosti kao što su zaštita od *SQL injectiona* ili limitiranje veličine datoteke za upload

```
'use strict'

const User = use("App/Models/User")
const {validate, sanitize} = use("Validator")

class UserController {
  async createUser({request, response}) {

    const allParams = sanitize(request.post(), {
      email: "normalize_email"
    })

    const validation = await validate(allParams, {
      firstname: "required|max:30",
      lastname: "required|max:30",
      email: "required|email|max:80",
      role_id: "required",
      password: "min:6|required|confirmed|max:30"
    })

    if (validation.fails()) {
      return response.badRequest({
        _message: "Incorrect parameters"
      })
    }

    const check = await User
      .query()
      .where("email", allParams.email).first()

    if (check) {
      return response.badRequest({
        _message: "User with provided email already exists"
      })
    }

    await User.create({
      firstname: allParams.firstname,
      lastname: allParams.lastname,
      email: allParams.email,
      role_id: allParams.role_id,
      password: allParams.password
    })

    return response.ok()
  }
}

```

Sl. 5.8. *User upravljač*

5.3. Klijentska aplikacija

Klijentska aplikacija izrađena je uz Nuxt.js i Vuetify.js razvojni okvir. Kreiranje projekta vrši se upisivanjem naredbe *npx create-nuxt-app Example-Front* prilikom čega se kreira projekt pod nazivom Example-Front. Prilikom kreiranja, postoje razne opcije za prilagodbu projekta kao što su preferirani programski jezik (JavaScript ili TypeScript), upravitelj paketima, Nuxt.js moduli, testni okvir te ostale prikazane opcije na slici 5.9.. Pokretanje projekta putem naredbene linije obavlja se upisivanjem i pokretanjem *npm run dev* naredbe.



Sl. 5.9. *Kreiranje Nuxt projekta*

Kako kreiranjem projekta i instalacijom Vuetify.js okvira projekt dolazi uz unaprijed generirane testne stranice i pripadajuće komponente, potrebno ih je obrisati iz *pages* direktorija kako bi se započeo razvoj prema vlastiti željama i potrebama.

Aplikacija će imat prijavu korisnika i tek nakon uspješne prijave, pojavljuje se sadržaj i kompletni raspored koji omogućava navigaciju kroz sustav. Kako bi se to postiglo, potrebno je kreirati dva rasporeda unutar *layouts* direktorija. Raspored koji će sadržavati aplikacijsku traku, podnožje i navigacijsku ladicu je *default.vue* datoteka, a *celan.vue* će se koristiti na stranici za prijavu. Nuxt automatski kreira *error.vue* raspored koji se pokreće ukoliko dođe do greške u sustavu, odnosno kada klijentska aplikacija „pukne“ zbog neke greške. Ukoliko se na stranici ne navede eksplicitno ime rasporeda koji se koristi, Nuxt će automatski primijeniti *default.vue.* Kreiranjem rasporeda ne smijemo zaboraviti uključiti sadržaj stranice u njega unutar *main* oznaka putem *Nuxt* oznake.

```vue
<template>
  <v-app style="background-color: #fbfaf9">
    <v-navigation-drawer
      v-model="drawer"
      :mini-variant="miniVariant"
      :clipped="clipped"
      fixed
      app
      color="primary"
    >
      <v-list flat dark>
        <v-list-item
          v-for="(item, i) in items"
          :key="i"
          :to="item.to"
          router
          color="secondary"
          exact-active-class="font-weight-bold title"
        >
          <v-list-item-action>
            <v-icon>{{ item.icon }}</v-icon>
          </v-list-item-action>
          <v-list-item-content>
            <v-list-item-title v-text="item.title"/>
          </v-list-item-content>
        </v-list-item>
      </v-list>
    </v-navigation-drawer>
    <v-app-bar :clipped-left="clipped" fixed app>
      <v-btn icon @click.stop="miniVariant = !miniVariant">
        <v-icon>mdi-{{ `chevron-${miniVariant ? "right" : "left"}` }}</v-icon>
      </v-btn>
      <v-toolbar-title v-text="title"/>
      <v-spacer/>
      <v-btn color="error" outlined depressed @click="logout">Log out</v-btn>
    </v-app-bar>
    <v-main>
      <v-container>
        <nuxt/>
      </v-container>
    </v-main>
    <v-footer :absolute="fixed">
      <v-spacer/>
      <span class="caption grey--text"
      >Marijo Kiš © {{ new Date().getFullYear() }}</span
      >
      <v-spacer/>
    </v-footer>
  </v-app>
</template>

```

Sl. 5.10. *Default.vue raspored*

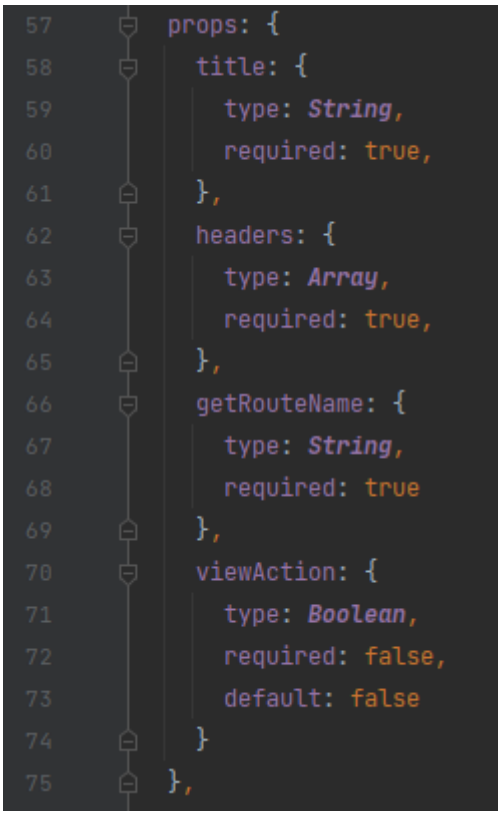
Na slici 5.10. prikazan je *Default.vue* raspored koji će se koristiti kroz cijelu aplikaciju osim na stranici za prijavu. Sastoji se od navigacijske ladice unutar koje je lista svih navigacijskih objekata s ikonom i nazivom. Aplikacijska traka sadrži gumb za sažimanje i proširivanje navigacijske ladice, naziv sustava i gumb za odjavu koji pritiskom poziva metodu *logout.*

HTTP klijent koji se koristi je Axios. Kako bi korištenje bilo što jednostavnije i brže, putem proširenja moguće je definirati globalne postavke. To se vrši kreiranjem datoteke *axios.js* unutar *plugins* direktorija. Na slici 5.11. vidi se postavljanje baznog API URL-a kako bi se prilikom korištenja izbjeglo repetitivno pisanje cijele putanje API-ja. U *onRequest* metodi moguće je definirati postavke pri slanju zahtjeva kao što je npr. dodavanje autorizacijskog zaglavlja i JWT tokena u zaglavlje zahtjeva. Također, dostupno je i npr. definiranje ponašanja prilikom neke greške i to se vrši unutar *onError* metode.

```
1  export default function ({$axios, app}) {
2    const {API_BASE_URL, SSR_API_BASE_URL = API_BASE_URL} = app.$config
3    const baseUrl = process.client ? API_BASE_URL : SSR_API_BASE_URL
4    $axios.setBaseURL(baseUrl)
5    $axios.onRequest( callback: config => {
6    });
7  }
8
```

Sl. 5.11. *Konfiguracija postavki Axiosa*

Komponente olakšavaju proces razvoja na način da poboljšavaju čitljivost koda i omogućavaju izdvajanje logike i strukture koja se ponavlja unutar jedne komponente koja se uključuje na mjestima gdje je potrebna. Cilj je komponentu napraviti što dinamičnijom, odnosno da sve što je promjenjivo bude definirano unutar svojstava, a zatim će se pri uvozu komponente na stranicu predati parametri svojstava koji će biti specifični za tu stranicu ili resurs. Na slici 5.12. prikazano je definiranje svojstava. Svako svojstvo mora imati svoj naziv i tip, a obveznost i inicijalna vrijednost su opcionalne za definiranje. Upotreba će biti objašnjena na temelju *headers* svojstva. Dio koda sa slike 5.12. je dio komponente za tablični prikaz podataka. Kako se podatci dohvaćaju s različitih ruta i kako na svakoj ruti podatci imaju različite atribute, definirano je svojstvo *headers* koje prilikom uvoza komponente na neku stranicu, definira niz objekata koji se sastoje od naslova stupca te ključeva koji su zapravo nazivi atributa. Na taj način moguće je koristiti istu tablicu za prikaz različitih podataka.

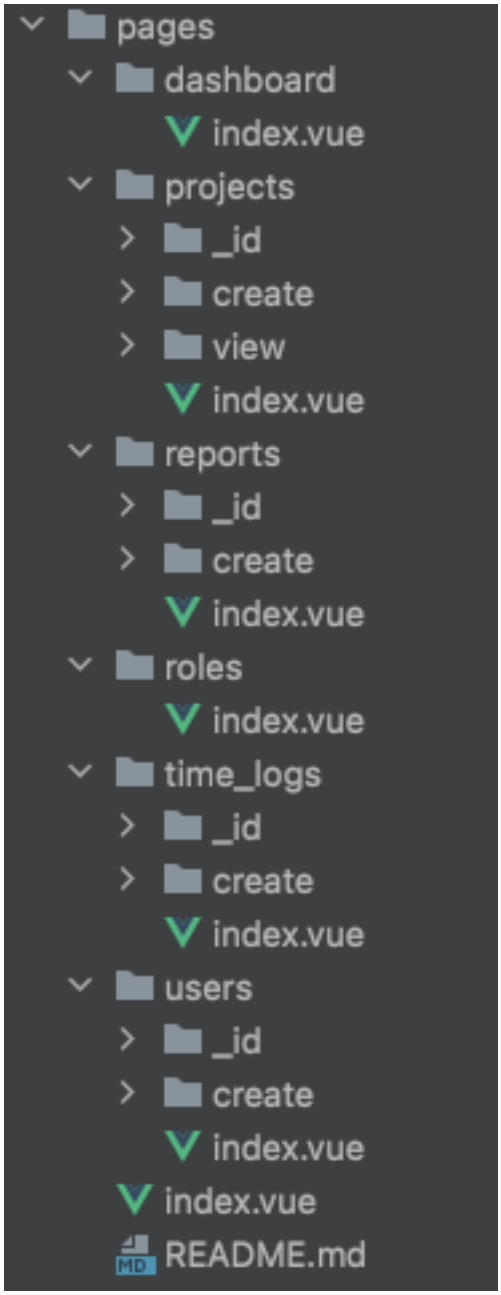


Sl. 5.12. *Definiranje svojstava*

Slika 5.13. prikazuje metodu za dohvaćanje podataka koja je ovisna o vrijednosti svojstva *getRouteName*. Ovime se također osigurava dinamičko recikliranje jedne metode umjesto da se ona definira na svakoj stranici posebno. Kako su podatci *paginirani* (podijeljeni su u skupove po 10 objekata), potrebno je poslužitelju poslati podatak o broju stranice koja se želi dohvatiti, odnosno koji je redni broj grupe od 10 objekata potrebno dohvatiti.

Sl. 5.13. *Metoda za dohvaćanje podataka*

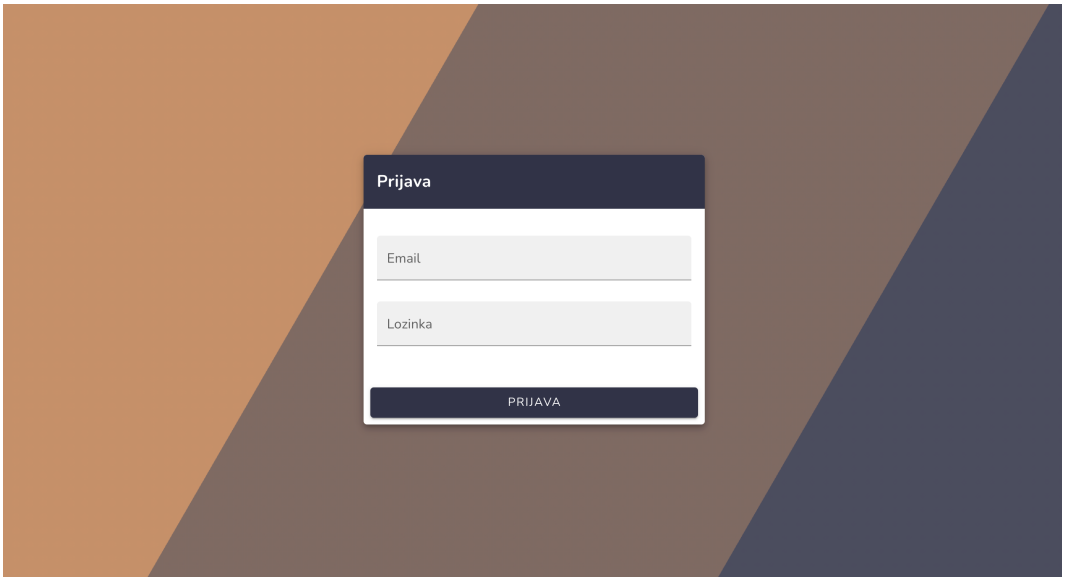
Gradnja CRUD (engl. *Create-Read-Update-Delete*) strukture navigatora koji rute kreira prema strukturi datoteka unutar *pages* direktorija (slika 5.14.) napravljena je na način da direktorij nosi naziv modela iz baze i na *index* stranici te putanje vrši se dohvaćanje liste upisa iz pripadajuće tablice. Unutar tog direktorija nalaze se još dva, a to su: *create* koji će generirati putanju na kojoj će biti smještena forma za kreiranje te *_id* direktorij koji će dinamički mijenjati svoju vrijednost i putanju kako bi se osiguralo dohvaćanje singlice i uređivanje pojedinog upisa.



Sl. 5.14. *Prikaz strukture prema kojoj navigator kreira putanje*

6. KORIŠTENJE APLIKACIJE

Pri ulasku u web aplikaciju, prikazuje se forma za prijavu (slika 6.1.). Kako bi se korisnik prijavio, potrebno je upisati odgovarajuću email adresu i pripadajuću lozinku. Nakon toga se kontaktira poslužiteljska aplikacija koja u bazi provjerava ispravnost danih podataka i ukoliko je sve uredu, klijentskoj aplikaciji se vraća JWT token putem kojega će se autorizirati svi naredni zahtjevi. Token se zapisuje u kolačiće preglednika odakle je modulima dostupan za korištenje. Pozadina iza forme za prijavu je dinamična i animirana što je postignuto CSS animacijama.

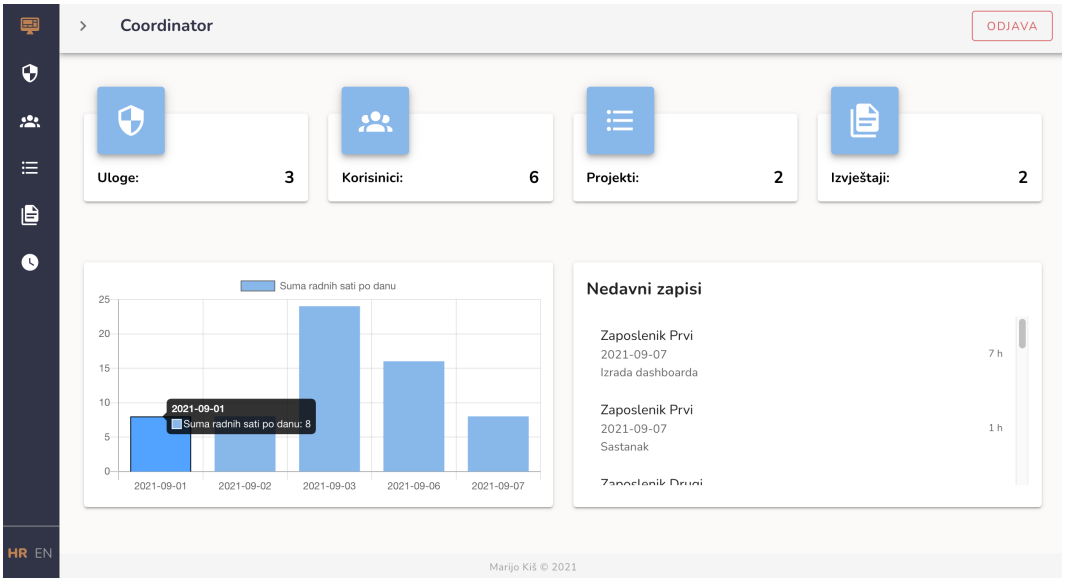


Sl. 6.1. *Forma za prijavu*

Daljnje korištenje aplikacije bit će prokomentirano kroz administratorska prava unutar aplikacije pošto ta uloga u sustavu može vršiti sve akcije. Što se tiče ostalih uloga, svaka od njih ima određena ograničenja rada unutar samog sustava.

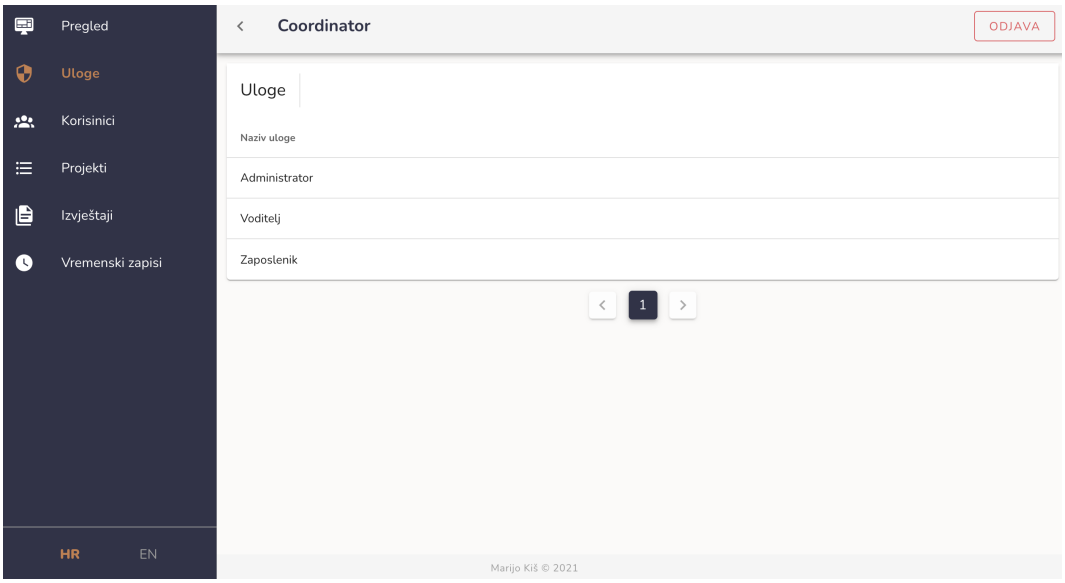
Uspješnom prijavom, aplikacija pogled direktno prebacuje na pregled i statistiku cijeloga sustava (slika 6.2.). S lijeve strane nalazi se navigacijska traka s linkovima na ostale sastavnice sustava, a to su uloge, korisnici, projekti, izvještaji i zapisi radnog vremena. Na samome dnu navigacijske trake, smještena su dva gumba koja omogućavaju prebacivanje jezika aplikacije, a dostupni su hrvatski i engleski jezik. Na vrhu, na aplikacijskoj traci, redom pronalazimo gumb za širenje i sužavanje navigacijske trake, naslov, odnosno naziv aplikacije i gumb za odjavu iz sustava. Glavni sadržaj sastoji se od pobrojanih upisa u bazu za entitete uloge, korisnici, projekti i izvještaji. Graf prikazuje (u slučaju kada je ulogirani korisnik

administrator, u protivnom je prikazana analitika za trenutno prijavljenog korisnika) sumu radnih sati svih zaposlenika grupiranih prema datumu. Zbog veće preglednosti i jednostavnijeg korištenja, prelaskom preko pojedinog stupca, prikazuje se dodatan oblačić u kojemu se točno iščitava vrijednost toga stupca. Desno od grafa smjestio se prikaz koji prikazuje nedavne upise radnih sati (ukoliko je prijavljeni korisnik administrator, tada će vidjeti upise sati svih zaposlenika, dok ostali korisnici vide samo svoje upise).

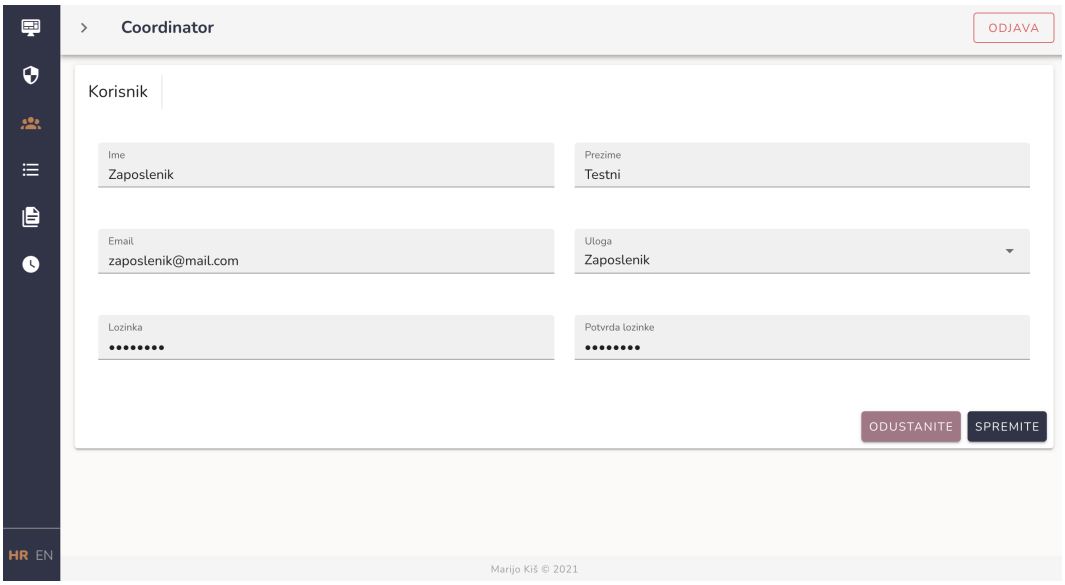


Sl. 6.2. *Pregled i statistika sustava*

Na slici 6.3. prikazana je lista dostupnih uloga i dozvola unutar sustava. Ovome može pristupiti samo administrator, isto kao i modulu za korisnike. Administrator dodaje nove korisnike u sustav, briše ih ili uređuje ukoliko dođe do promjene nekog od podatka. Na slici 6.4. prikazana je forma za kreiranje novih korisnika (zaposlenika). Vidljivo je koje je podatke potrebno ispuniti kako bi spremanje bilo moguće pri čemu je važno pripaziti da email adresa bude u ispravnom obliku te da se lozinka i potvrda lozinke poklapaju. Ukoliko se ne ispune svi validacijski zahtjevi, pojavit će se validacijske poruke koje ukazuju na kršenje određenog validacijskog pravila, a kreiranje će biti onemogućeno dok se sva kršenja ne otklone. Ukoliko je sve uredu i ukoliko se na poslužiteljskoj strani također ispuni validacija da je email adresa unikatna unutar sustava i ostala definirana validacijska pravila, tada će se izvršiti kreiranje novog zaposlenika što će na klijentskoj strani biti popraćeno *toast* porukom i automatskim prebacivanjem pogleda na listu korisnika sustava.



Sl. 6.3. *Uloge u sustavu*

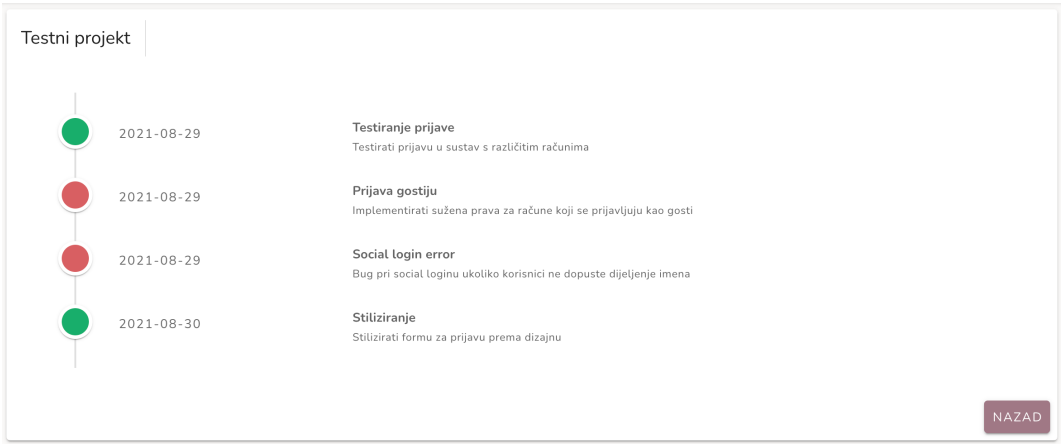


Sl. 6.4. *Kreiranje novog zaposlenika*

Pri kreiranju novog projekta, administrator treba postaviti voditelja koji će biti zadužen za sferu projektnog menadžmenta na razini projekta, odnosno rukovodit će projektom, resursima i zaposlenicima koji rade na njemu. Voditelj projekta može biti osoba koja ima ulogu voditelja unutar sustava. Nakon što je projekt uspješno kreiran, voditelj može započeti s dodavanjem zadataka za zaposlenike, dok zaposlenici kasnije pristupom projektu vide listu zadataka i oznaku statusa zadatka prebacuju na izvršeno nakon što odrade pojedini zadatak.

Govoreći o dopuštenjima na modulu projekta, administrator ima apsolutnu moć, odnosno vidi sve projekte i zadatke na njemu, može uređivati podatke ili izmijeniti voditelja, mijenjati status zadataka ili u krajnjem slučaju i obrisati projekt. Voditelj ne može dodavati nove projekte niti uređivati ili brisati postojeće, već ima kontrolu nad zadatcima na projektu, a zaposlenici mogu vidjeti zadatke i mijenjati im status.

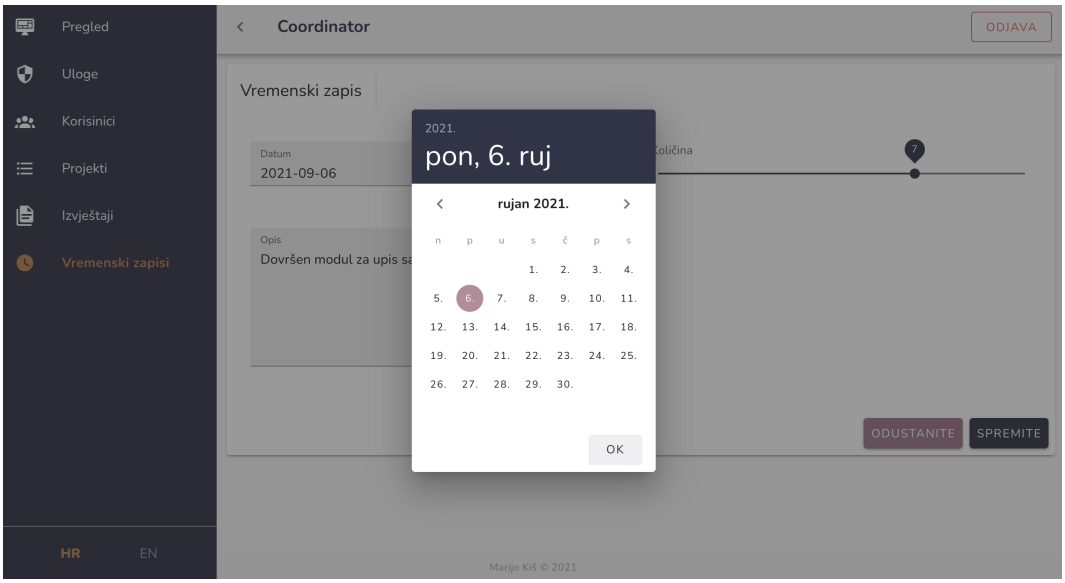
Na slici 6.5. prikazana je kartica s listom zadataka na projektu. Zadatci koji još nisu dovršeni označeni su crvenom bojom. Od ostalih informacija tu su datum kreiranja, naslov zadatka i opis. Prebacivanje statusa vrši se klikom na pojedini zadatak.



Sl. 6.5. *Lista zadataka na projektu*

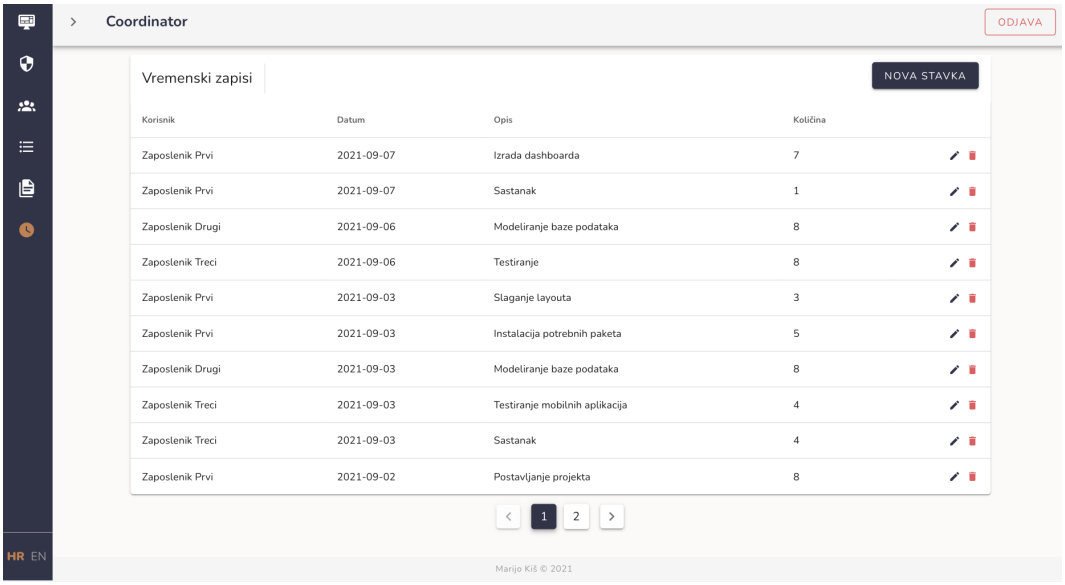
Izvještaji se kreiraju za pojedini projekt, a sastoje se od naslova te od samog sadržaja izvještaja. Kreiraju ih voditelji projekta kako bi dokumentirali napredak u radu ili npr. kako bi dokumentirali odrađeni posao i zapažanja tokom pojedinog sprinta. Ovom modulu mogu pristupiti samo administratori i voditelji.

Lista vremenskih zapisa (zapisi radnog vremena) dostupna je svim ulogama unutar sustava. Svaki korisnik za sebe upisuje broj odrađenih sati uz odabir datuma te kratki opis odrađenog posla unutar tog radnog perioda. Broj radnih sati se odabire povlačenjem klizača do željene vrijednosti, a datum se odabire s kalendara (slika 6.6.).



Sl. 6.6. *Upisivanje radnih sati*

Slika 6.7. prikazuje tablicu vremenskih zapisa radnog vremena. U gornjem desnom uglu nalazi se gumb koji otvara formu za stvaranje novog upisa dok je ispod tablice smještena paginacijska traka koja pokazuje dostupni broj stranica upisa i omogućava prebacivanje između stranica, odnosno upisa. Uređivanje pojedine stavke vrši se klikom na ikonicu olovke što otvara formu za uređivanje, a brisanje se obavlja na način da se klikne ikonica kantice nakon čega se otvara potvrdni prozor putem kojega se traži dodatna potvrda korisnika da je siguran da želi obrisati odabranu stavku.



Sl. 6.7. *Prikaz vremenskih zapisa*

7. ZAKLJUČAK

Računala i softverska rješenja za razne probleme postali su neizostavni dio svakodnevice društva. Cilj ovog diplomskog rada bio je izraditi web aplikaciju koja će modernizirati i olakšati proces projektnog menadžmenta unutar neke tvrtke.

Korisnik sa svim dopuštenjima unutar sustava je administrator koji može kreirati sve ostale korisnike, odnosno zaposlenike i voditelje. Sustav djeluje prema ulogama u poduzeću i prema tome su krojena ovlaštenja za akcije kao što su kreiranje, uređivanje i brisanje, a i sami pregled podataka. Vode se informacije o zaposlenicima. Projekte kreiraju administratori, a zadatke voditelji projekta. Zadatci se dodjeljuju zaposlenicima koji ih mogu zatvoriti nakon što obave definirani posao. Voditelji projekta mogu kreirati izvješća za pojedine projekte koja se koriste za administraciju i arhivu. Svi zaposlenici moraju upisivati odrađene sate što je još jedna od funkcionalnosti koja pridonosi korisnosti sustava jer se na taj način mogu odrediti zalaganja pojedinih članova projektnog tima te eventualno uočiti usko grlo u kojemu dolazi do problema i zbog kojeg se mora napraviti preraspodjela resursa i trajanja određenih zadataka.

U svakom rješenju puno je mjesta za poboljšanja i napredak. Neka od nadogradnji koja bi se mogla dodati u sustav su vođenje digitalnog kadrovskog kartona o zaposleniku u kojemu bi se nalazile sve informacije o njemu kao što su osobni podatci, iskustvo, kompetencije, plaća i drugo. Nadalje, upisivanje sati rada moglo bi se pretvoriti u kalkulator prema kojemu će se računati plaća i bonusi za brzo i kvalitetno odrađen posao. Na koncu, ovaj je projekt dobar temelj da uz nadogradnje i proširenja u potpunosti digitalizira poslovanje neke tvrtke.

LITERATURA

[1] Projektni menadžment - Prof.dr.sc. Vlado Majstorović, Projektni menadžment, Sveučilište u Mostaru, Mostar, 2010.

[2] HTML - https://developer.mozilla.org/en-US/docs/Web/HTML (datum posjete: 7.7.2021.)

[3] CSS - https://developer.mozilla.org/en-US/docs/Web/CSS (datum posjete: 7.7.2021.)

[4] JavaScript - https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript (datum posjete: 7.7.2021.)

[5] Node.js - https://nodejs.org/en/about/ (datum posjete: 8.7.2021.)

[6] Adonis.js - https://legacy.adonisjs.com/ (datum posjete: 8.7.2021.)

[7] MySQL - https://www.mysql.com/ (datum posjete: 8.7.2021.)

[8] Nuxt.js - https://nuxtjs.org (datum posjete: 9.7.2021.)

[9] Vuetify.js - https://vuetifyjs.com/en/ (datum posjete: 9.7.2021.)

[10] Git - https://git-scm.com/ (datum posjete: 9.7.2021.)

[11] Zapier - https://zapier.com/blog/free-project-management-software/ (datum posjete: 6.9.2021.)

[12] Trello - https://trello.com/ (datum posjete: 6.9.2021.)

[13] Asana - https://asana.com/ (datum posjete: 6.9.2021.)

[14] ClickUp - https://clickup.com/ (datum posjete: 6.9.2021.)

[15] Paymo - https://www.paymoapp.com/ (datum posjete: 6.9.2021.)

[16] Jira - https://www.atlassian.com/software/jira (datum posjete: 6.9.2021.)

SAŽETAK

Cilj ovog diplomskog rada je izraditi web aplikaciju koja će olakšati upravljanje i koordinaciju zaposlenika neke tvrtke u radu na različitim projektima. Početak ovog rada uvodi u problematiku upravljanja resursima na razvoju projekta. Nadalje, objašnjene su korištene tehnologije pri izradi projekta kao što su HTML, CSS, JavaScript, Node.js, MySQL, Adonis.js, Nuxt.js, Vuetify.js i Git. Glavni dio donosi pojašnjenje procesa izrade uz pregled važnijih dijelova arhitekture i objašnjava korištenje sustava. Na kraju seminara nalaze se prijedlozi za poboljšanje i nadogradnju sustava.

Ključne riječi: Adonis.js, MySQL, Nuxt.js, Upravljanje projektima, Web

ABSTRACT

Web application for project management

The main goal of this master thesis is to create a web application which helps in project management process. Firstly, there is an introduction to the problems of project management. Furthermore, HTML, CSS, JavaScript, Node.js, MySQL, Adonis.js, Nuxt.js, Vuetify.js and Git are being explained. The main part provides clarification of implementation with the overview of the most significant parts of the software architecture and shows usage flow example of an application. In the end, there are suggestions for enhancements and upgrades of this software.

Keywords: Adonis.js, MySQL, Nuxt.js, Project management, Web

ŽIVOTOPIS

Marijo Kiš rođen je 2. svibnja 1997. godine u Virovitici. Pohađao je Osnovnu školu Vladimira Nazora Virovitica gdje biva nagrađen kao izuzetno uspješan učenik (završio je svih 8 razreda s prosjekom 5,00). 2012. godine upisuje Gimnaziju Petra Preradovića u Virovitici, opći smjer. Tijekom osnovnoškolskog i srednjoškolskog obrazovanja bio je sudionik raznih natjecanja od kojih je značajnije rezultate postizao na županijskoj razini iz predmeta biologija i geografija. 2016. godine polaganjem državne mature završava srednju školu te iste godine upisuje preddiplomski sveučilišni studij na Fakultetu za elektrotehniku, računarstvo i informacijske tehnologije Osijek, smjer Računarstvo. Postaje stipendist općine Lukač na temelju uspješnosti tokom srednjoškolskog obrazovanja. 2019. godine završava preddiplomski sveučilišni studij i stječe akademski naziv sveučilišnog prvostupnika inženjera računarstva. Iste godine upisuje diplomski sveučilišni studij Računarstvo, izborni blok Informacijske i podatkovne znanosti. U vrijeme pisanja ovog diplomskog rada, zaposlen je u Gauss Developmentu gdje radi kao frontend developer i voditelj frontend odjela, a također i kao mentor novim kolegama. Vrlo se dobro služi engleskim jezikom, a posjeduje osnovno znanje njemačkog jezika te vozačku dozvolu B kategorije.

Potpis: _____

Mojoj obitelji, mojoj najvećoj motivaciji, hvala Vam!

PRILOZI

CD

- izvorni kod
- .docx i .pdf verzija diplomskog rada