ŠOLSKI CENTER CELJE

Srednja šola za kemijo, elektrotehniko in računalništvo

E-dentiteta: Odsev digitalnega

Raziskovalna naloga

Avtorji:

Žan Škorja, R-4.b

Dominik Brezovšek, R-4.b

Nik Mejak, R-4. b

Mentor:

Boštjan Lubej, dipl. inž. inf. in tehn. kom.

Mestna občina Celje, Mladi za Celje

Celje, 2024

ŠOLSKI CENTER CELJE

Srednja šola za kemijo, elektrotehniko in računalništvo

E-dentiteta: Odsev digitalnega

Raziskovalna naloga

Avtorji:

Žan Škorja, R-4.b

Dominik Brezovšek, R-4.b

Nik Mejak, R-4. b

Mentor:

Boštjan Lubej, dipl. inž. inf. in tehn. kom.

Mestna občina Celje, Mladi za Celje

Celje, 2024

IZJAVA*

Mentor/-ica Boštjan Lubej v skladu z 20. členom Pravilnika o organizaciji mladinske raziskovalne dejavnosti »Mladi za Celje« Mestne občine Celje, zagotavljam, da je v raziskovalni nalogi z naslovom E-dentiteta: Odsev digitalnega, katere avtor/-ica je Žan Škorja, Dominik Brezovšek, Nik Mejak:

- besedilo v tiskani in elektronski obliki istovetno,
- pri raziskovanju uporabljeno gradivo navedeno v seznamu uporabljene literature,
- da je za objavo fotografij v nalogi pridobljeno avtorjevo dovoljenje in je hranjeno v šolskem arhivu,
- da sme Osrednja knjižnica Celje objaviti raziskovalno nalogo v polnem besedilu na knjižničnih portalih z navedbo, da je raziskovalna naloga nastala v okviru projekta Mladi za Celje,
- da je raziskovalno nalogo dovoljeno uporabiti za izobraževalne in raziskovalne namene s povzemanjem misli, idej, konceptov oziroma besedil iz naloge ob upoštevanju avtorstva in korektnem citiranju,
- da smo seznanjeni z razpisni pogoji projekta Mladi za Celje.

Celje, 4.4.2024

Žig šole

ŠOLSKI CENTER CELJE
SREDNJA ŠOLA ZA KEMIJO, ELEKTROTEHNIKO IN RAČUNALNIŠTVO
CELJE, Pot na Lavo 22
1

Podpis mentorja

Podpis odgovorne osebe

*

POJASNILO

V skladu z 20. členom Pravilnika raziskovalne dejavnosti »Mladi za Celje« Mestne občine Celje je potrebno podpisano izjavo mentorja (-ice) in odgovorne osebe šole vključiti v izvod za knjižnico, dovoljenje za objavo avtorja (-ice) fotografskega gradiva, katerega ni avtor (-ica) raziskovalne naloge, pa hrani šola v svojem arhivu.

ZAHVALA

Zahvaljujemo se vsem, ki ste kakorkoli pomagali pri izdelavi raziskovalne naloge. Brez vaše pomoči naloga ne bi nastala, pa naj je šlo le za spodbudne besede, majhno idejo, nasvete ali kritike pri izdelovanju izdelka.

Najprej bi se zahvalili našemu mentorju profesorju Boštjanu Lubeju za ves trud, vztrajnost, podporo in čas, ki ga vložil, da smo raziskovalno nalogo pripeljali do konca. Zahvalili bi se mu tudi za tehnični pregled naloge.

Profesorici Tjaši Verdev bi se zahvalili za slovnični pregled te raziskovalne naloge, profesorici Rosani Breznik pa za pregled in popravo angleškega povzetka v nalogi.

Vsem zgoraj omenjenim posameznikom se še enkrat zahvaljujemo, saj brez Vas raziskovalna naloga ne bi uspela.

POVZETEK

Raziskovalna naloga predstavlja idejno zasnovo, uporabljena ogrodja in orodja ter končne rezultate naše aplikacije za shranjevanje in varno preverjanje dijaških izkaznic ter izkaznic organizacij, E-dentiteta. Raziskovalna naloga se osredotoča na uporabljene postopke in principe, ki so potrebni za dosego optimalnega delovanja spletne aplikacije ter njeno maksimalno varnost. Na koncu je aplikacija, predstavljena v tej raziskovalni nalogi, primerjana z lansko rešitvijo te problematike, ovrednotena pa je predvsem stabilnost, hitrost in zanesljivost obeh aplikacij.

Ključne besede: E-dentiteta, digitalne izkaznice, odzivnost aplikacije, optimizacija, varnost, primerjalna analiza

ABSTRACT

This research paper represents the general idea, frameworks and tools used to build the final product and final results of our web application for storing and securely authenticating student identification cards and organisation cards, called E-dentity. Research paper focuses on techniques and principles that were used in order to optimise our application as well as its maximum possible security. At the end of the paper, we compared our solution to the solution that we built last year. We rated both solutions based on their stability, speed, performance and reliability.

Key words: E-dentity, digital identification cards, application speed, optimisation, security, comparative analysis

KAZALO

KAZALO SLIK

UPORABLJENE KRATICE

CORS – (angl. Cross-Origin Resource Sharing) – skupna raba virov navzkrižnega izvora

CSS – (angl. Cascading Style Sheets) – programski jezik za pisanje stilskih predlog

HTML – (angl. Hyper Text Markup Language) – označevalni programski jezik za programiranje spletnih strani

IDE – (angl. Integrated Development Environment) – integrirano razvojno okolje

MVC – (angl. Model–View–Controller) – Model-Pogled-Nadzornik

ORM – (angl. Object–Relational Mapping) – objektno-relacijsko preslikavanje

PHP – (angl. PHP (Personal Home Page): Hypertext Preprocessor) – programski jezik za programiranje spletnih aplikacij

QR koda – (angl. quick response code) – črtna koda

RAM – (angl. Random Access Memory) – pomnilnik z naključnim dostopom

SHA256 – (angl. Secure Hash Algorithms 256) – varni zgoščevalni algoritem

SQL – (angl. Structured Query Language) – strukturirani povpraševalni jezik za delo s podatkovnimi bazami

SSL – (angl. Secure Sockets Layer) – tehnologija za zagotavljanje varnosti internetne povezave

SSR - (angl. Server-Side Rendering) - generiranje vsebine spletne strani na serverju

SVN – (angl. Subversion) – podverzije

UI – (angl. User interface) – uporabniški vmesnik

1 UVOD

Živimo v svetu, kjer si vsakdana ne moremo predstavljati brez pametnega telefona na dosegu naših rok. Dostopnost kjerkoli in vedno bolj vsestranska uporabnost je omogočila razvoj različnih digitalnih rešitev, ki so nadomestile že skoraj vse vsakodnevne stvari – od opomnikov, nakupovalnih listkov pa vse do plačevanja z digitalno bančno kartico. Zamisel za projekt je prišla iz razvoja naše digitalne e-denarnice, ki smo jo ustvarili prejšnje leto. Le-ta je bila zelo ozko usmerjena in slabo dodelana. Zaradi tega smo se odločili, da bomo letos na novo definirali ciljno skupino uporabnikov in pri izdelavi uporabili modernejše tehnologije za ustvarjanje spletnih aplikacij.

Med razvojem te aplikacije smo se soočali s pomembno problematiko, ki smo se jo odločili temeljito preučiti in raziskati. Naš cilj je bil preoblikovati tradicionalni koncept fizičnih kartic v digitalno obliko in stopiti v korak s časom. S tem smo omogočili večjo praktičnost in varnost uporabnikom pri identifikaciji. Zavedamo se, da se smernice minulih let vedno bolj nagibajo k dolgotrajni družbi, ki bi imela manjši vpliv na okolje, zato smo želeli prispevati k temu cilju z zmanjšanjem proizvodnje fizičnih kartic.

Naša aplikacija je bila zasnovana s ciljem, da olajša proces preverjanja identitete posameznikov. Namesto fizičnih izkaznic bi uporabniki imeli svojo identifikacijsko izkaznico shranjeno v digitalni obliki. Naša ciljna skupina so bile, predvsem zaradi relevantnosti in realizacije ideje, izobraževalne inštitucije in podobne ustanove, ki lahko za interne ali eksterne potrebe svojim članom izdajo izkaznice.

Seveda smo pri tem poizkušali aplikacijo narediti uporabniku čim bolj prijazno in predvsem varno. Le-to smo dosegli z uporabo različnih knjižnic. Več o idejni in aplikacijski zasnovi pa je zapisano v nadaljevanju raziskovalne naloge.

1.1 Hipoteze

Cilj raziskovalne naloge je bil uspešno postavljen in delujoč primerek aplikacije, ki bi različnim nivojem uporabnikov omogočala vse potrebne administrativne funkcije. Pri implementaciji smo želeli zagotoviti čim večjo enotnost in varnost.

V raziskovalni nalogi smo postavili naslednje hipoteze:

1) Aplikacija bo imela ob enakih pogojih za najmanj 40% boljšo odzivnost.
2) Z uporaba Laravel ogrodja v kombinaciji z procesorjem za predloge Laravel Blade bo boljša za odzivnost aplikacije kot pa uporaba Laravel zalednega[1] sistema in uporaba ogrodja Vue.js za čelni[2] del aplikacije.
3) Uporaba knjižnice Auth0, v kombinaciji s SSL certifikatom, bo poskrbela za dovolj dobro varnost naše aplikacije.
4) Uporaba predpomnjena uporabnikov in nekaterih podatkov v Redis podatkovni bazi bo izboljšala odzivni čas aplikacije za najmanj 20% v primerjavi z uporabo Eloquent ORM poizvedb v SQL podatkovno bazo.

1.2 Metode raziskovanja

Za pripravo raziskovalne naloge nismo imeli na razpolago strokovnih virov in literature za to tematiko. Večinoma smo si pa pomagali z aplikacijami, ki ponujajo podobne ali enake produkte. Večino stvari smo morali narediti samostojno, saj na tem področju še ni neke enovite rešitve ali pa se le-ta zelo razlikuje od naše idejne zasnove.

[1] angl. backend
[2] angl. frontend

1.3 Struktura raziskovalnega dela

V raziskovalni nalogi smo sprva predstavili temo in si zadali hipoteze. V drugem poglavju smo predstavili že obstoječe rešitve na trgu za hranjenje kartic. V tretjem poglavju smo predstavili vsa orodja in programe, ki smo jih uporabili za izdelavo aplikacije. Potek praktičnega dela raziskovalne naloge smo predstavili v četrtem poglavju. V predzadnjem poglavju smo predstavili analizo in rezultate naše raziskovalne naloge. V zadnjem, petem poglavju, pa smo ovrednotili hipoteze.

2 Obstoječi trg

Na začetku raziskovalne naloge smo ponovno, tokrat podrobneje, preučili in raziskali trg obstoječih rešitev hranjenja kartic. Ugotovili smo, da na obstoječem trgu ni veliko rešitev za tovrstni problem. Kljub naraščanju digitalizacije in potrebi po varnem shranjevanju in upravljanju identifikacijskih kartic obstaja le omejeno število aplikacij, ki ponuja ustrezne funkcionalnosti. Med trenutno dostopnimi rešitvami so najbolj znane ID123, BizConnect in ISIC (International Student Identity Card). Kljub dejstvu, da te aplikacije nudijo nekatere funkcionalnosti za upravljanje identifikacijskih kartic, še vedno obstajajo določene pomanjkljivosti in omejitve, ki zahtevajo nadaljnje raziskovanje in razvoj celovitejših rešitev.

2.1 ID123

ID123 je e-denarnica, ki je na voljo prek spletnega brskalnika in mobilne aplikacije. Namenjena je podjetjem, šolam in drugim organizacijam, ki želijo digitalizirati svoje identifikacijske kartice. Aplikacija omogoča shranjevanje in upravljanje identifikacijskih kartic na mobilni napravi, ne da bi bila potrebna internetna povezava. Uporabnikom je omogočena funkcija, da lahko dodajo podpis na svojo mobilno identifikacijsko kartico, prav tako pa imajo možnost v aplikaciji shranjevanja več digitalnih identifikacijskih kartic. Te kartice vključujejo tudi črtne kode, ki jih je mogoče skenirati s tretjimi napravami za skeniranje in s tem verifikacijo veljavnosti kartice. (ID123 Inc.)

Poleg tega ID123 ponuja večplastno varnost za zaščito uporabniških identitet in preprečevanje goljufij, med drugim z avtentikacijo v dveh korakih. Vse uporabniške podatke aplikacija varno šifrira in shranjuje v oblaku. (ID123 Inc.)

Kljub temu pa ID123 morda ni tako široko uporabljena zaradi visokih cen naročnine, konkurence na trgu, omejene funkcionalnosti ali pomislekov glede varnosti podatkov.

2.2 BizConnect

BizConnect je izjemno učinkovit skener poslovnih kartic, ki omogoča hitro digitalizacijo poslovnih stikov z visoko natančnostjo in hitrostjo. S svojo napredno tehnologijo prepoznavanja besedila uporabnikom omogoča enostavno zajemanje in pretvorbo podatkov s poslovnih kartic v digitalno obliko, kar olajšuje upravljanje stikov ter vzpostavljanje poslovnih povezav. (BizConnect, Inc.)

Kljub temu da aplikacija ponuja zanimivo in hitro rešitev za digitalizacijo identifikacijskih kartic, pa ne vsebuje vseh funkcionalnosti, ki bi si jih uporabniki želeli. Zaradi tega morda ni najbolj optimalna rešitev za vse potrebe na tem področju.

2.3 ISIC - International Student Identity Card

ISIC kartica, kratica za Mednarodno študentsko identifikacijsko kartico, je posebna identifikacijska kartica, namenjena študentom po vsem svetu. Ta kartica ponuja študentom številne ugodnosti in popuste tako doma kot tudi v tujini. ISIC kartica je prepoznavna po svojem modrem dizajnu in vsebuje fotografijo imetnika kartice, ime in priimek, datum rojstva ter datum izteka veljavnosti kartice. (ISIC Association)

ISIC kartica je priznana in sprejeta v več kot 170 državah po vsem svetu, podpira jo tudi Mednarodna organizacija študentskih izkaznic (ISIC Association). Namenjena je vsem polnoletnim študentom, ki se izobražujejo na visokošolskih ali srednješolskih ustanovah, ne glede na njihovo narodnost ali status. (ISIC Association)

ISIC ima tudi mobilno aplikacijo, ki študentom omogoča lažji dostop do svoje ISIC kartice in ugodnosti, ki jih ta prinaša. Le-ta imetnikom omogoča enostaven dostop do digitalne kartice, kjer imetniki pregledujejo podatke o kartici, kopirajo njeno številko in prejmejo obvestila pred iztekom veljavnosti. Poleg tega imajo v aplikaciji možnost zbiranja vavčerjev iz omejenih ponudb in jih shranijo v digitalno denarnico za prihodnjo uporabo. Prav tako omogoča obveščanje o najnovejših novicah iz lokalnega urada ISIC ali šole imetnika ter upravljanje ISIC profila, nalaganje profilne fotografije ali posodabljanje kartice. (ISIC Association)

3 Uporabljene tehnologije

V raziskovalni nalogi smo uporabili kar nekaj sodobnih tehnologij, s katerimi je naša končna izvedba aplikacije zaživela. Zaradi tega smo se odločili, da bomo vse te tehnologije na kratko predstavili.

3.1 Figma

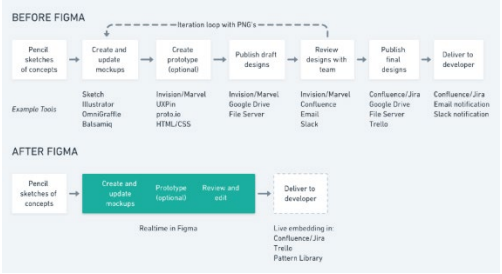
Na začetku smo potrebovali ideje, kaj vse bo naša aplikacija omogočala, zato smo si zaradi lažje predstave zasnovali vse funkcije aplikacije v spletnem orodju Figma. Figma je spletno orodje, ki se uporablja za dizajniranje uporabniških vmesnikov in celovito grafično oblikovanje. Zelo dobra prednost tega orodja je njegova izvrstna implementacija možnosti dizajniranja v skupinah. (Kopf)

Prednost tega orodja je tudi to, da je podprto na vseh platformah. To nam je prišlo prav, saj uporabljamo različne operacijske sisteme, še kako dobrodošlo pa je v večjih podjetjih, kjer uporabljajo računalnike različnih operacijskih sistemov. Zaradi dejstva, da je Figma spletna aplikacija, zagotavlja enako uporabniško izkušnjo na vseh sistemih.

Figma omogoča tudi urejanje dizajna projekta s strani več oseb hkrati. V vsakem trenutku tako takoj vidimo, kaj je nekdo spremenil in kje se le-ta nahaja na projektu. S tem je zelo primerno orodje za uporabo, kjer člani ekipe med sestankom usklajujejo izgled njihove aplikacije.

Še ena prednost aplikacije je ta, da omogoča integracijo v pogovorno aplikacijo Slack, kjer so vsi komentarji in spremembe v trenutku objavljeni v pogovor. Med drugim aplikacija tudi ponuja možnost deljenja datoteke z nastavljanjem pravic, kaj lahko uporabnik na drugi strani počne z zasnovo. Obstaja tudi možnost deljenja samo posameznih zavihkov. (Kopf)

Figma je med drugim ena izmed prvih omogočila oblikovalcem deljenje zasnove v obliki HTML povezave na druge spletne strani.



Slika 1: Prikaz delovanja Figme

(Vir: prevzeto od (Kopf))

3.2 PhpStorm

Za našo aplikacijo smo uporabili PhpStorm, orodje razvito s strani podjetja JetBrains, ki je integrirano razvojno okolje (IDE), namenjeno predvsem razvijalcem v programskem jeziku PHP. To orodje ponuja širok nabor funkcij, ki izboljšujejo produktivnost in olajšujejo proces razvoja PHP aplikacij. PhpStorm je eden izmed razvojnih okolij, ki jih zagotavlja podjetje JetBrains.

JetBrains je vodilno podjetje na področju razvoja integriranih razvojnih okolij (IDE) in drugih produktov za razvijalce programske opreme. Seveda pa njihovi produkti niso brezplačni, ampak ker smo dijaki, imamo možnost dostopa do JetBrainsovih produktov brezplačno zaradi pobude podjetja za spodbujanje izobraževanja in podpore pri razvoju novih generacij razvijalcev. Ta program, imenovan "JetBrains Students", omogoča dijakom in študentom, da brezplačno uporabljajo široko paleto orodij ter integriranih razvojnih okolij. (JetBrains s.r.o.)

Ena izmed ključnih lastnosti PhpStorma je napredno urejanje kode. S funkcijami, kot so barvanje kode, samodejno dopolnjevanje, preverjanje napak v realnem času, refraktoriranje kode, in podporo za upravljanje z različnimi jeziki, vključno s HTML, CSS, JavaScript in SQL. Poleg tega ima integracijo s sistemom za nadzor različic, kot so Git, SVN in Mercurial, kar olajša upravljanje s projektnimi repozitoriji. Sami smo zaradi lažje organizacije in sprotnega spremljanja sprememb kode naše aplikacije uporabili Git. (JetBrains s.r.o.)

PhpStorm ponuja tudi podporo za številna ogrodja in platforme, kot so Laravel, Symfony, Zend Framework, WordPress, Joomla in drugi. Vsako od teh ogrodij ima svoje specifične značilnosti in prednosti, ki jih PhpStorm vključuje ter omogoča njihovo uporabo. V našem primeru smo uporabili Laravel, ki je odprtokodno ogrodje za razvoj spletnih aplikacij v PHP-ju. (JetBrains s.r.o.)

3.3 GitHub

Za naš projekt smo vključili GitHub repozitorij v PhpStorm, ker želimo izkoristiti prednosti sistema za nadzor različic, kot je Git, in sodelovalno razvojno okolje, ki ga ponuja GitHub.

GitHub je spletna platforma za gostovanje in sodelovanje pri razvoju programske opreme, ki temelji na sistemu za nadzor različic Git. Omogoča razvijalcem, da shranijo, upravljajo in delijo svoje kode z drugimi razvijalci prek spleta. Prav tako ponuja številne funkcije, kot so sledenje spremembam v kodi, upravljanje zavez (angl. issues), razprave (angl. discussions), pregled kode (angl. code reviews), sodelovanje na projektih in še več. (Kinsta Inc., 2023)

Vključitev našega GitHub repozitorija v PhpStorm nam je omogočala enostavno upravljanje s projektom, sledenje spremembam v kodi, izvajanje verzioniranja kode, izmenjavo sprememb z drugimi člani ekipe ter upravljanje s projektom v enem integriranem okolju. To je povečevalo našo učinkovitost, preglednost in sledljivost pri razvoju projekta ter omogočalo boljšo organizacijo in sodelovanje med člani ekipe.

3.4 Laravel

Našo aplikacijo smo se odločili postaviti z ogrodjem Laravel zaradi njegovih naprednih funkcionalnosti, ki olajšajo razvoj spletnih aplikacij v PHP-ju. Laravel je odprtokodno ogrodje, ki ponuja širok nabor orodij in funkcij za učinkovito izdelavo zmogljivih aplikacij. Zasnovan je tako, da olajša vsakdanje naloge razvijalcev in nam omogoča, da se osredotočijo na bistvo razvoja aplikacij, ne pa na ponavljajoče se operacije, kar nam je prihranilo ogromno časa pri razvoju aplikacije.

Laravel ponuja številne napredne funkcionalnosti, med katerimi so elegantna sintaksa in struktura projekta, migracije ter sejanje podatkov, ORM (Object-Relational Mapping), avtentifikacija in avtorizacija, šablonski pogledi ter vgrajena podpora za testiranje. Tako smo imeli možnost učinkovito razviti zmogljivo spletno aplikacijo s čisto in vzdržljivo kodo. (Laravel Holdings Inc.)

Seveda pa smo morali podporo za Laravel sintakso namestiti v PhpStom, kar je bilo nadvse enostavno, saj obstaja vtičnik za Laravel podporo v PhpStorm IDE, razvit s strani podjetja JetBrains. Inicializacija novega projekta je dokaj enostavna, ko poznaš vse male skrivnosti ogrodja. Najprej je treba namestiti Composer, ki ga Laravel uporablja za upravljanje paketov. Ko je Composer nameščen, preprosto ustvarimo novo Laravel aplikacijo tako, da z uporabo terminala izvedemo ukaz za ustvarjanje novega projekta z uporabo Composerja (composer create-project laravel/laravel ime_projeka). Ko je projekt ustvarjen, se premaknemo v korensko mapo projekta in jo odpremo v poljubnem razvojnem okolju, v našem primeru PhpStorm, in začnemo z razvijanjem.

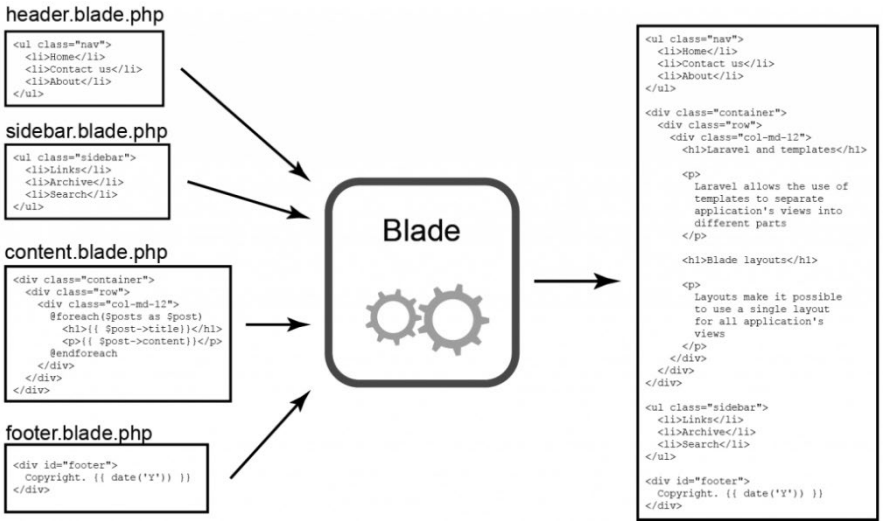
3.4.1 Laravel Blade

Ker smo želeli zgraditi čim bolj enovito aplikacijo, ki bi uporabljala čim bolj podobne elemente, smo se odločili čelni del sistema razviti z Laravel Blade. Pri tem smo predvidevali, da bomo, zaradi deljenja podatkov in pogledov na eni domeni oz. na enakem sistemu, zelo izboljšali odzivnost naše aplikacije, kar se je izkazalo kot resnično. Rezultate primerjave odzivnosti aplikacije bomo podrobneje prikazali pri analizi hipotez, kjer bomo primerjali odzivnost Laravel Blade aplikacije napram aplikacije, ki uporablja Laravel in Vue.js.

Laravel Blade je privzet način zapisovanja kode za čelni del pri uporabi Laravel ogrodja. Pri razvijalcih je zelo priljubljen zaradi zmožnosti ustvarjanja robustnih in hitrih pogledov, ki so uporabni tudi večkrat. Omogoča uporabo zank, spremenljivk, logičnih operacij in ostalih PHP funkcij kar preko razširjene sintakse HTML. (Ramos, 2023)

Mnogo različnih testov je pokazalo, da je Blade mehanizem za poglede eden najhitrejših PHP ogrodij na trgu. To predvsem doseže s tem, da se najprej vsa koda prevede v navadno PHP kodo, kjer se potrebni dinamični podatki in funkcije izvedejo ter »hidrirajo« potrebne elemente, nato pa se pogled v brskalniku izpiše kot HTML dokument. Le-ta se shrani v predpomnilnik strežnika, dokler ne pride do sprememb v kodi. Gre za SSR (Server-Side-Rendering oz. genereiranje na serverju) pristop, ki je v zadnjem času postal zelo popularen pri arhitekturi spletnih aplikacij, saj se njihova vsebina ne spreminja dovolj hitro, da bi upravičila generacijo na strani klienta, saj je ta generacija (strokovni izraz je hidracija, ang. “hydration”) zelo neučinkovita in ponavadi pomeni daljši čas do interaktivnosti aplikacij. Zaradi predpomnjenja je mogoče hitrejše pošiljanje prikaza spletne strani. (Ramos, 2023)

Še ena prednost Laravel Blada je uporaba MVC modela.



Slika 2: Prikaz delovanja Blade

(Vir: Prevzeto od (Huang, 2018))

3.5 Redis

Med ustvarjanjem raziskovalne naloge smo naleteli na problem neučinkovitih, ponavljajočih se poizvedb v podatkovno bazo, ki so upočasnjevale strežnik in delovanje aplikacije. Pri tem smo naleteli na zelo zanimivo rešitev, kako bi lahko odzivni čas aplikacije drastično zmanjšali. Problem ponavljajočih se kompleksnih poizvedb je, da porabijo veliko sistemskih virov zaradi prepošiljanja podatkov iz diska na delovnih pomnilnik in nato preko HTTP zahteve do klienta. Z uporabo predpomnjenja ponavljajočih se poizvedb, bi se odzivni čas aplikacije izboljšali. Ena od možnih rešitev je tako postala uporaba Redis podatkovne baze.

Redis oziroma remote dictionary server je zelo dodelana solucija za shranjevanje podatkov v parih po principu »key:value«, torej »ključ_podatka:vrednost«, vendar pa ponuja še ogromno zelo naprednih funkcij, kot so na primer predpomnjenje zahtev, predpomnjenje podatkovnih poizvedb, abstrakcija SQL podatkovnih tabel. Rešitev je bila predstavljena leta 2009 in jo uporablja zelo veliko tehnološko naprednih aplikacij. Deluje na osnovi tega, da razvijalec v naprej določi različne poizvedbe, ki jih bo potreboval tekom končne aplikacije. Te so po navadi bolj kompleksne za iskanje v podatkovni bazi in bi vzele preveliko časa, da bi jih pridobili iz nje. Zato le-te definiramo v naprej s pomočjo Redisa in le-ta jih bo že izvedel in si jih bo shranil v RAM pomnilnik, kar zelo drastično vpliva čas odziva spletnega strežnika. Seveda zaradi tega strežnik potrebuje več RAM pomnilnika, a je ta količina skoraj zanemarljiva, saj gre tu mogoče za nekaj MB več porabljenega pomnilnika. Ta rešitev pa ob vsaki spremembi tabele v podatkovni bazi, ki je seveda tudi uporabljena v Redis poizvedbah, avtomatsko požene poizvedbo ponovno in si jo spet shrani na RAM pomnilnik. (Redis Ltd.)

Omogoča tudi veliko različnih modulov, katere lahko dodamo Redis knjižnici, ti pa omogočajo še razširitev funkcionalnosti tej soluciji. Na žalost le-teh nismo mogli preizkusiti, saj bi morali kupiti naročnino.

4 Praktični del raziskovalne naloge

4.1 Idejna zasnova

Problem, ki smo ga želeli odpraviti, se je izkazal kot zelo kompleksna ideja. Zasnovna ideja naše aplikacije je bila varno shranjevanje in uporabljanje kartic, v namen izkazovanja resničnosti kartice. Zaradi tega smo morali uvesti kar nekaj varnostnih ukrepov, med drugim šifriranje občutljivih podatkov v podatkovni bazi, shranjevanje ključa seje v bazi in primerjanje le-tega s ključem seje, poslanim v zahtevi ter zaščito pred nepooblaščenimi zahtevami preko namenskih funkcij za prečiščevanje zahtev.

Sprva smo imeli samo tri tipe uporabnikov, to so bili: uporabnik, administrator in člani organizacij.

Naša ideja ja na začetku predvidela samo to, da bi uporabnik prikazoval svoje kartice in dokazal njihovo resničnost ter zaprosil za kartico pri različnih organizacijah. Le-to pa bi preverjale organizacije, ki imajo za to dovoljenje. Te bi izdajale nove kartice za svoje člane, ki so lahko ali odprtega tipa ali tipa, kjer mora uporabnik najprej zaprositi za kartico, prošnjo pa mu more odobriti organizacija. Administrator je služil samo kot upravljalec organizacij, uporabnikov in kartic.

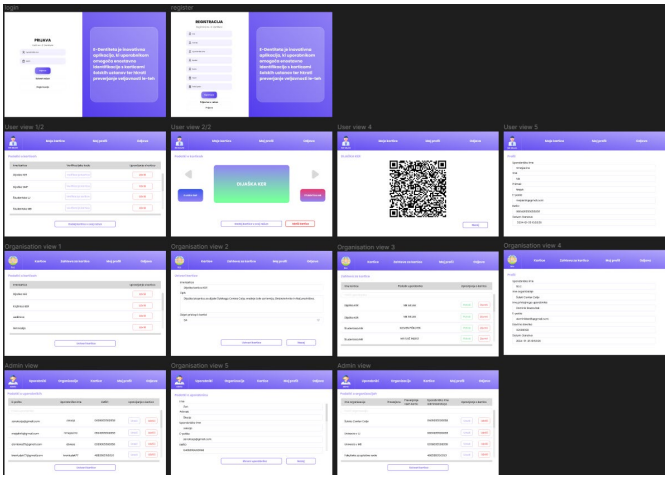
4.2 Varnost podatkov

Vse aplikacije in spletene storitve, ki zbirajo in obdelujejo podatke uporabnikov za namene funkcionalnosti aplikacije, morajo poskrbeti za varno shranjevanje, šifriranje in obdelavo podatkov, poskrbeti za pravočasno obveščanje o morebitnih vdorih v sisteme, ki shranjujejo podatke uporabnikov in omogočiti, da uporabniki izbrišejo svoje podatke iz strežnikov podjetja. V skladu s to uredbo mora naša aplikacija tako ob registraciji jasno navesti zakaj in kako so podatki hranjeni, kako dolgo, uporabnik pa se mora s tem strinjati, sicer ne more uporabljati aplikacije. Prav tako mora imeti uporabnik možnost izbrisa računa in s tem vseh svojih podatkov iz naših strežnikov. To smo dosegli z vnosnim poljem, ki se uporabniku prikaže in ga obvesti o zbiranju podatkov. Vnosno polje se nahaja na strani za registracijo. Prav tako ima uporabnik možnost izbrisa profila pod zavihkom za urejanje profila. Podatki, ki se shranjujejo v podatkovno bazo, so varovani z enkripcijo povezave med podatkovno bazo in aplikacijo (uporaba SSL in CORS politike, ki zahteva avtorizacijksi žeton ob vsaki zahtevi). Prav tako pa je za gesla uporabljen algoritem zgoščevanja SHA256, ki je enosmerni algoritem za pretvorbo poljubno dolgega niza znakov v 256 bitov dolgo kombinacijo znakov, ki se jih ne da pretvoriti nazaj v prvotno obliko, zato je ta način več kot primeren za shranjevanje občutljivih podatkov, kot so na primer gesla. Baza pa je varovana tudi z naključno generiranim geslom, ki ustreza varnostnim standardom, baza pa

na strežniku živi znotraj Docker virtualne instance, kar pomeni da ni dostopna zunaj Docker Engine okolja, tako da so zunanji vdori zelo oteženi.

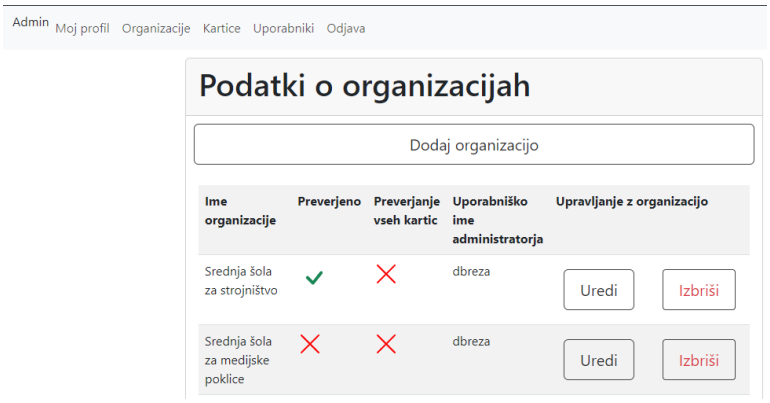
4.3 Prvi koncepti aplikacije

Preden smo začeli z razvojem aplikacije, smo si najprej na hitro ustvarili predlogo , kako bi naš projekt izgledal z uporabo aplikacije Figma.

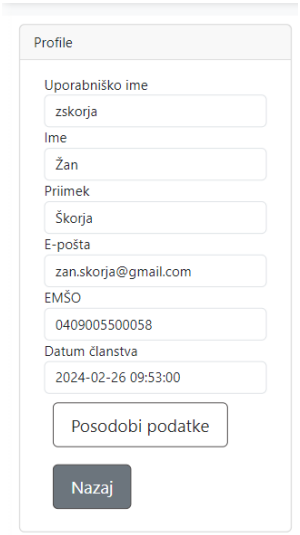


Slika 3: Prikaz vseh konceptov aplikacije

Aplikacijo smo poizkušali razviti sistematsko. Najprej smo začeli z oblikovanjem podatkovne baze in ustvarjanjem zalednega sistema aplikacije. Na začetku smo začeli pisanje kode v Laravel in Vue.js. Kasneje v raziskovalni nalogi pa smo se odločili, da bomo raje uporabili kar Laravel Blade za celotno aplikacijo. Pisanje v Laravel Blade se je izkazala kot dobra odločitev napram Laravelu in Vue.js, saj smo tako lahko enovito zgradili aplikacijo. Aplikacija se je pri tem tudi pohitrila. S pomočjo Laravela smo ustvarili tako imenovane migracije, ki služijo kot PHP stavki, s katerimi se ustvari podatkovna baza. Tako smo se izognili dolgim in zapletenim SQL stavkom. Zatem je sledilo dolgočasno in težaško pisanje modelov. Vsaka entiteta v podatkovni bazi potrebuje tudi model, ker gre za ORM pristop, torej en nivo abstrakcije nad dejansko strukturo podatkovne baze, kjer definiramo vse podatke, ki jih vpišemo in tudi vse povezave do drugih entitet. Ko smo le-to imeli že kar dodelano, smo se lotili še čelnega dela. Najprej smo uporabili kar stilske predloge ogrodja CSS Bootstrap. To smo naredili samo za to, da smo s tem enostavneje preverili, ali vse implementirane funkcije delujejo, kot bi morale. Kmalu je tako nastala prva zasnova naše aplikacije.



Slika 4: Prvi koncept aplikacije



Slika 5: Prvi koncept aplikacije

4.4 Ponovno postavljanje aplikacije

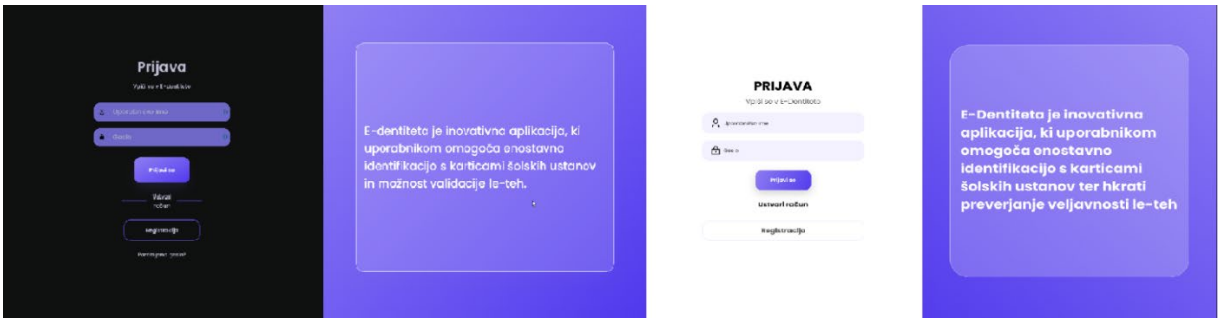
Kmalu pa smo naleteli na probleme, katerih nismo predvideli na začetku. Funkcionalnost aplikacije smo želeli razširiti. To je bila velika težava, saj tega nismo predvideli pri zasnovi podatkovne baze in je ni bilo mogoče prilagoditi. Le-to je bilo treba zato drastično spremeniti in refraktorirati. Pri zasnovi nove podatkovne baze smo upoštevali, da je ne naredimo čisto »zaprto« in si pustimo prostor, kjer bi z lahkoto dodali še kakšno novo funkcionalnost. Dodaten problem je nastal, ko smo začeli pisati svoje stilske predloge, saj smo ugotovili, da je bilo večino HTML elementov odveč.

Ideja naše nove aplikacije je temeljila na šestih različnih tipih uporabnikov: uporabnik, dijak, profesor, administrator oziroma upravljalec organizacije in sistemski administrator ter partner.

Uporabnik je najbolj primitiven del naše aplikacije. Ta ima samo možnost urejanja svojega profila in zahtevanja pridružitve k organizaciji. Le-to odobri administrator organizacije. Ta ima poleg tega še funkcijo, da odstranjuje uporabnike iz organizacije, ureja in izdaja nove kartice organizacije ter ureja zahteve za kartico, če je katera izmed kartic zaprtega tipa. Poleg tega imamo še uporabnika, ki ga mi imenujemo profesor. Ta lahko tako kot upravljalec organizacije ureja zahteve po kartici, za organizacijo, kateri pripada. Predzadnji tip uporabnika se imenuje dijak. Tega si uporabnik prisluži, ko pripada neki organizaciji, v katero je bil sprejet. Zadnja vrsta uporabnikov pa je sistemski administrator. Ta ureja in dodaja organizacije ter zunanje partnerje. Zunanji partnerji oziroma angleško vendorji pa preverjajo izkaznice organizacij, s katerimi sodelujejo in na podlagi preverjanj omogočajo določene ugodnosti uporabnikom.

Sprva je bila ideja, da bi naš sistem bil pol-odprte narave. To je bilo mišljeno tako, da bi moral administrator vsakega uporabnika ročno vpisati v aplikacijo. A to bi bilo zelo zamudno delo za upravljalce raznih organizacij, zato smo se odločili, da si uporabnik sam ustvari račun, a je ta praktično neuporaben, dokler ga vsaj ena organizacija ne potrdi.

Po uspešni prenovi podatkovne baze in HTML kode našega projekta smo se spet lotili razvoja funkcionalnosti naše aplikacije. Najprej smo uvedli šest tipov uporabnikov, ki smo jih našteli na začetku. Zaradi tega je prišlo do drastične prenovitve vseh povezav s podatkovno bazo in logike zalednega sistema. Kmalu je nastal drugi koncept naše aplikacije, s katerim smo bili zelo zadovoljni, saj se je zelo približal Figmini idejni zasnovi.



Slika 6: Primerjava naše zasnove(leva) in zasnove v Figmi(desna)

4.5 Optimizacija naše aplikacije

Ker nismo želeli, da bi naša aplikacija samo uspešno zaživela, temveč da bi prenesla ogromne obremenitve uporabnikov in bi zagotavljala hitro odzivnost, smo se po uspešno postavljeni aplikaciji osredotočili na optimizacijo z vidika hitrosti. Najprej smo odkrili, da je naša podatkovna baza zelo počasna z odzivi po neki poizvedbi. Kompleksne in ponavljajoče se poizvedbe smo si želeli pohitriti. Iskali smo različne rešitve, od tega da bi migrirali podatkovno bazo v NOSQL ali Firebase, a smo po tehtni raziskavi odkrili, da s tem ne bi veliko spremenili. Med brskanjem po vsevednem spletu smo naleteli na ogrodje Redis. Le-ta je našim potrebam zelo ustrezal, saj ni bilo potrebno narediti nobene migracije sistema, enostavno smo ga lahko implementirali v kodo, od koder smo ga uporabili za predpomnjenje ponavljajočih se poizvedb. Pisanje Redis poizvedb poleg tega sploh ni težko. Pri tem gre za to, da nekemu ključu pripišemo rezultat poizvedbe in pripišemo zraven podatek, kako dolgo je le-ta veljaven. Vse to Redis shrani v RAM pomnilnik, od koder je dostop bistveno hitrejši, kot pa postopek normalne obdelave zahteve po podatkih.

Z implementacijo ogrodja Redis smo aplikacijo bistveno pohitrili. Pred implementacijo je povprečna poizvedba po strežniku trajala 30 ms, po implementaciji pa samo 2 ms.

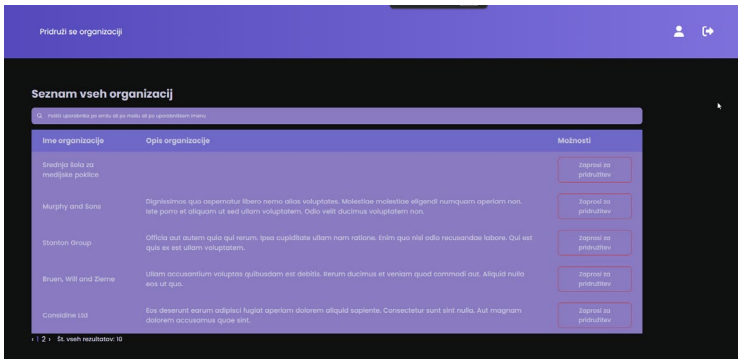
Druga stvar, s katero smo našo aplikacijo optimizirali, je bila ta, da smo v tabelah, kjer je več kot 5 vrstic podatkov, razdelili podatke na več strani z uporabo tehnike imenovane paginacija. S tem se na enkrat od strežnika zahteva bistveno manj podatkov, kot pa če bi moral na enkrat prikazati na primer 1000 uporabnikov in še več. Seveda pa bi bilo iskanje zelo zamudno, zato smo implementirali tudi iskalnik, kjer iščemo po uporabniškem imenu, EMŠU ali e-naslovu.

4.6 Predstavitev aplikacije

Na kratko bomo predstavili vsak uporabniški pogled in njegove funkcije.

4.6.1 Uporabnik

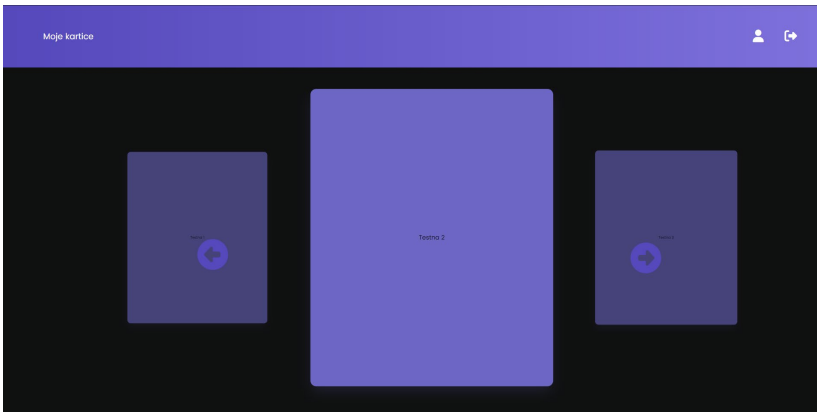
Uporabnik je najenostavnejši tip uporabnikov našega sistema. Ta ima le funkcijo, da spremeni svoje osebne podatke in zaprosi za pridružitev k organizaciji.



Slika 7: Pogled uporabnika

4.6.2 Dijak

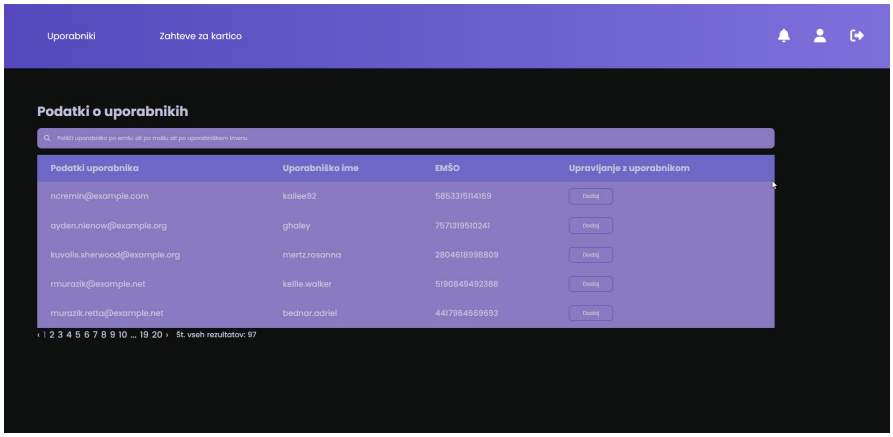
Dijak je druga vrsta uporabnika. Ta postane dijak takrat, ko ga neka organizacija ali profesor potrdi kot člana organizacije. S tem se njegova funkcionalnost zveča na to, da ima možnost zaprositi za kartico organizacije.



Slika 8: Pogled dijaka

4.6.3 Profesor

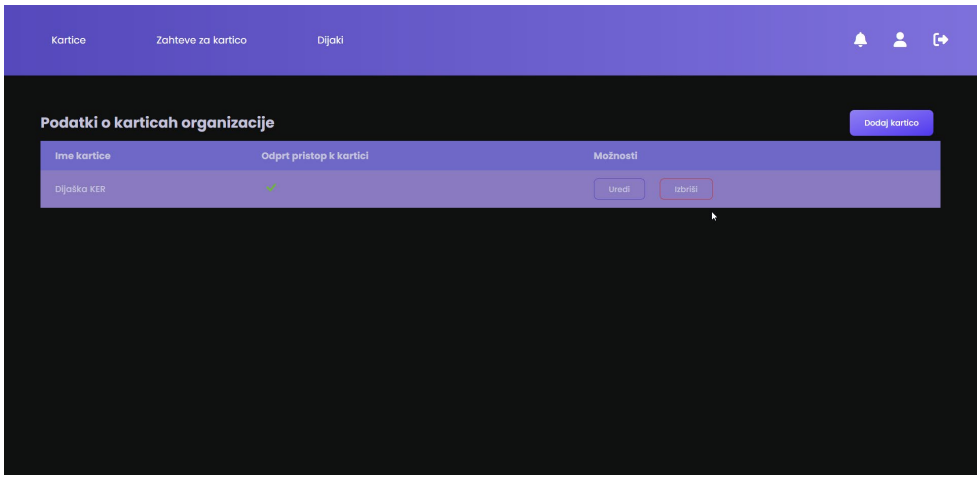
Profesor je entiteta naše aplikacije, ki ima zmožnost dodajati uporabnika svoji organizaciji in pregledovati zahteve za pridobitev kartice s strani dijakov. Pregleduje tudi ali je kartica resnična, če ta dijak pripada njegovemu razredu.



Slika 9: Pogled profesorja

4.6.4 Administrator oziroma upravljalec organizacije

Ta vrsta uporabnika lahko dodaja kartice svoji organizaciji, pregleduje zahteve za kartico in dodaja uporabnike svoji organizaciji, da ti postanejo dijaki. Pregleduje tudi ali je kartica verodostojna na podlagi zapisa v podatkovni bazi in podatkov, ki jih izlušči s branjem generirane QR kode, če ta dijak pripada njegovi organizacij.



Slika 10: Pogled administratorja organizacije

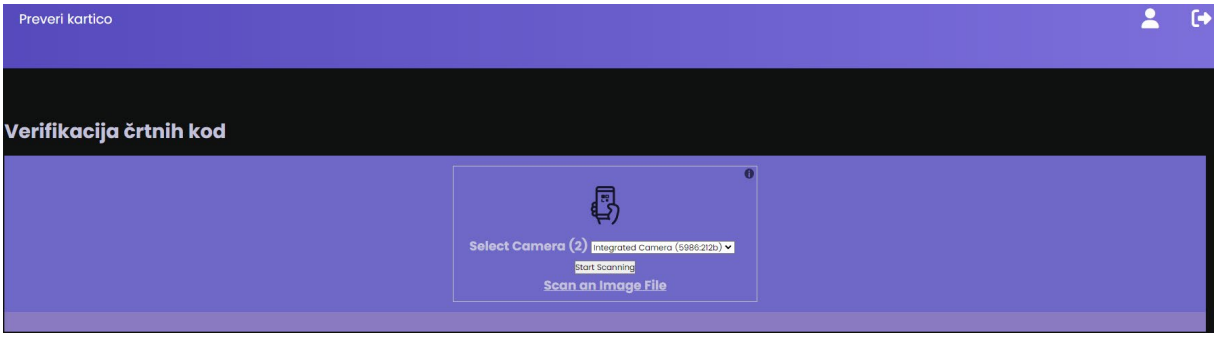
4.6.5 Administrator sistema

Administrator sistema ima funkcionalnost, ki lahko upravlja ter dodaja organizacije in upravlja s partnerji, ki pregledujejo istovetnost kartice.

Slika 11: Pogled administratorja

4.6.6 Partner

Želeli smo tudi, da bi lahko določene osebe pregledovale, ali je kartica, ki jo dijak prikazuje, veljavna ali ne. S tem namenom smo ustvarili nov tip uporabnika, imenovan partner. Ta pregleduje z integriranim bralnikom črtnih kod, ali je prikazana kartica veljavna ali ne. Ta bi lahko bil uporabljen v namene, da bi dokazali recimo v kinu, da smo še kar dijaki neke šole.



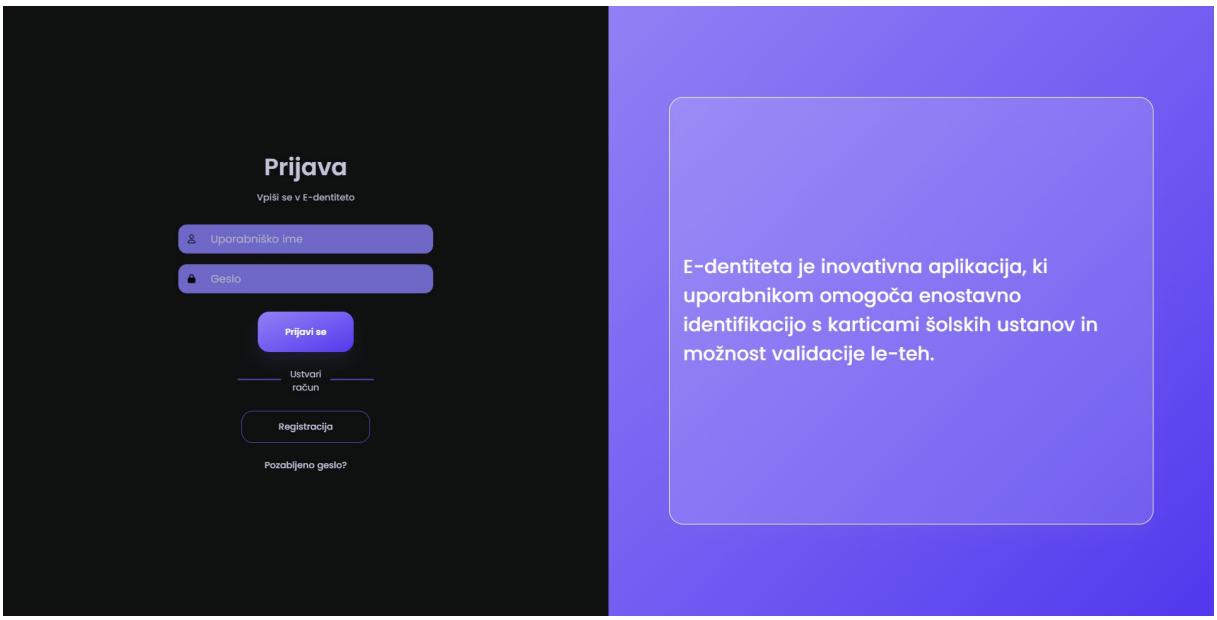
Slika 12: Pogled partnerja

4.7 Predstavitev funkcij naše aplikacije

Ker smo zelo na hitro predstavili uporabniške poglede in njihove funkcije, se nam zdi pravilno, da vsako komponento naše aplikacije predstavimo še malo podrobneje.

4.7.1 Prijavno okno

Naša prva komponenta je prijavno okno. Na tem oknu se nahajajo tri glavne komponente. Prva funkcija je prijava, ki samo preveri, ali v podatkovni bazi obstaja zapis, ki se ujema z vnesenimi podatki. Naslednja funkcija je registracija, s katero se oseba registrira v naš sistem in s tem postane najpreprostejši tip uporabnika – uporabnik. Zadnja, najzanimivejša funkcija, pa je pozabljeno geslo. Le-to smo implementirali na zelo poseben način. Da bi se izognili nevarnosti pridobivanja uporabniških imen in e-poštnih naslovov, samo zasnovali funkcijo tako, da ta pregleda, če uporabnik z vpisom v polje obstaja ali ne. V obeh primerih bo uporabnik videl samo sporočilo, da je na pripadajoč e-naslov bila poslana zahteva za ponastavitev gesla. Kar pa zamolčimo uporabniku je to, da če vnos v podatkovni bazi ne obstaja, mu tega ne povemo, a samo izpišemo, da je zahteva za ponastavitev gesla bila poslana.



Slika 13: Struktura prijavnega okna

4.7.2 Urejanje profila

Naslednja funkcija naše rešitve je urejanje profila. Tu uporabnik vidi svoje podatke in izbrane podatke spreminja. Uporabniku tu pustimo spremeniti samo ime in priimek, saj se drugi podatki (načeloma) ne spreminjajo.

Slika 14: Profilno okno

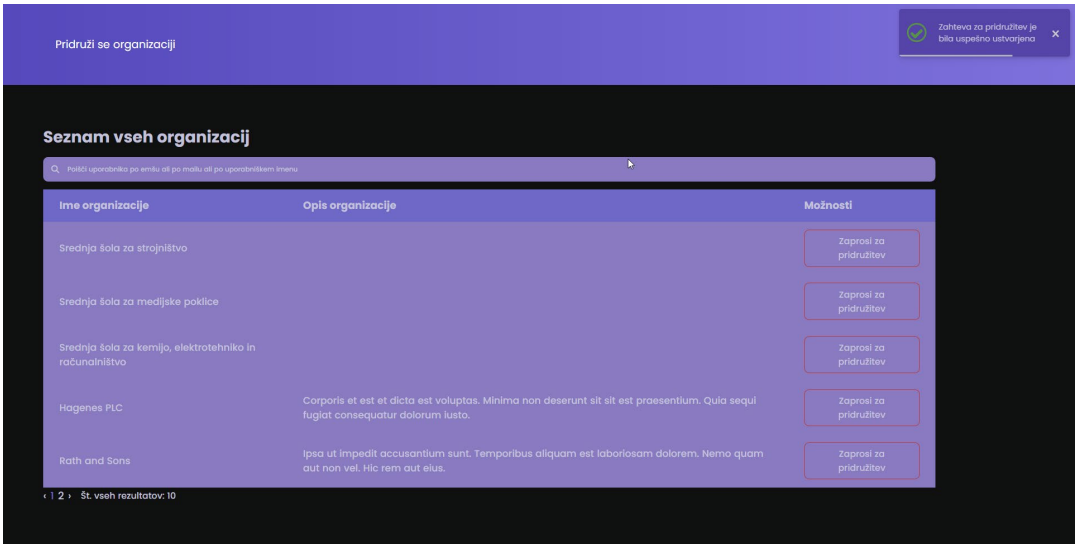
4.7.3 Obvestila

Naslednji pomemben gradnik naše aplikacije je obvestilno okno. Tu so prikazana obvestila, ki se nanašajo na našo nalogo, recimo kot administrator organizacije moramo potrditi pridružitev k organizaciji s strani neke osebe.

Slika 15: Primer obvestila

4.7.4 Pridružitev organizaciji

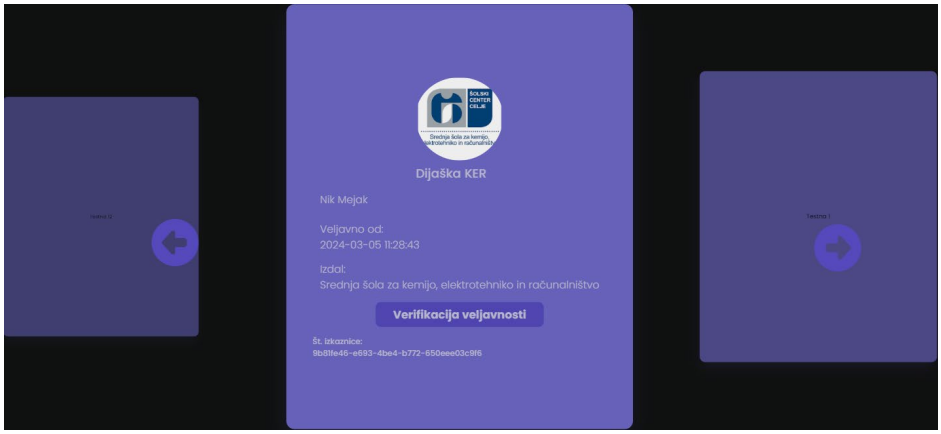
Ta funkcija pripada uporabniku, saj le-ta ne more narediti nič drugega, kot pa se pridružiti organizaciji, pri kateri bo šele potem imel pravico zaprositi za kartico.



Slika 16: Funkcija pridružitvi organizacije

4.7.5 Kartice dijaka

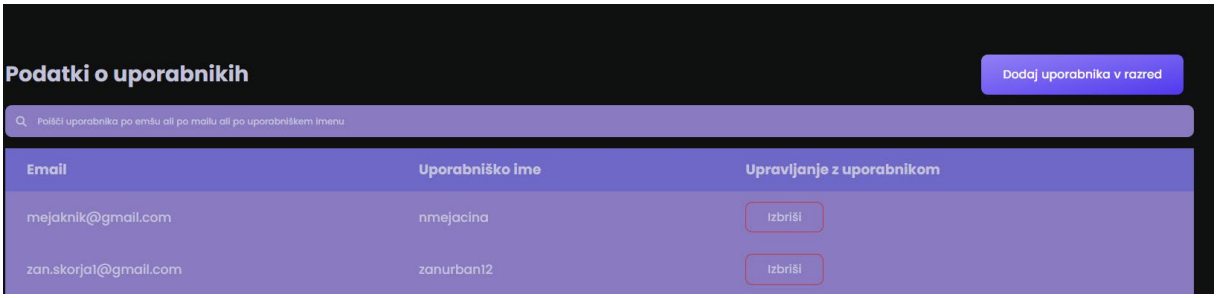
Ta pogled uporabniškemu tipu dijak omogoča, da vidi svoje kartice ali zaprosi za nove kartice. Omogoča pa mu tudi, da izkaže, da je njegova kartica veljavna, s prikazom črtne kode. Kartice lahko iz svojega profila tudi poljubno briše.



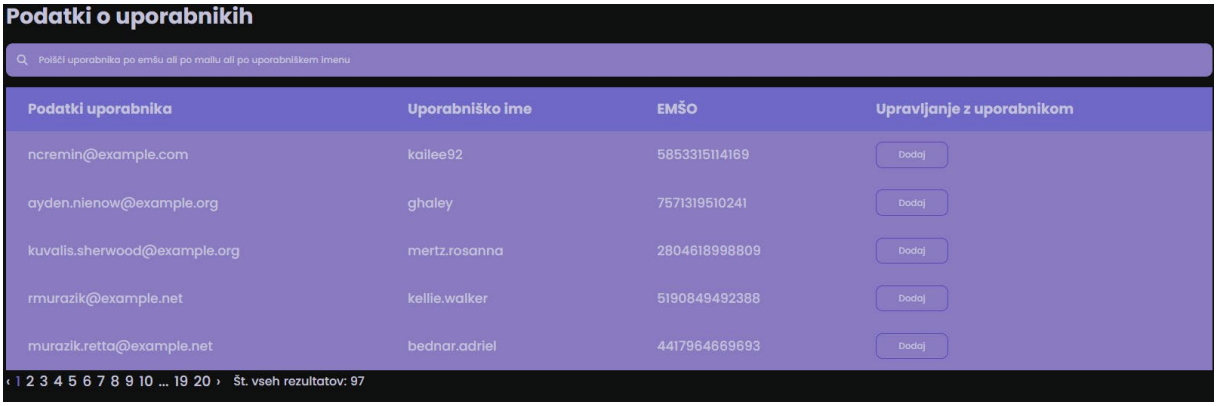
Slika 17: Prikaz kartic dijaka

4.7.6 Dijaki določenega oddelka

Naslednja funkcija naše aplikacije je, da ima profesor pregled, kateri dijaki pripadajo njegovemu razredu, jih izbriše iz svojega oddelka ter jih dodaja v svoj razred.



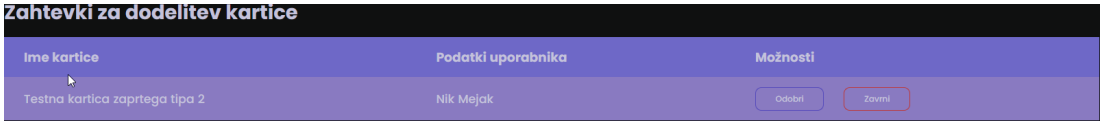
Slika 18: Seznam dijakov v profesorjevem oddelku



Slika 19: Seznam uporabnikov, ki ne pripadajo nobenemu oddelku in organizaciji

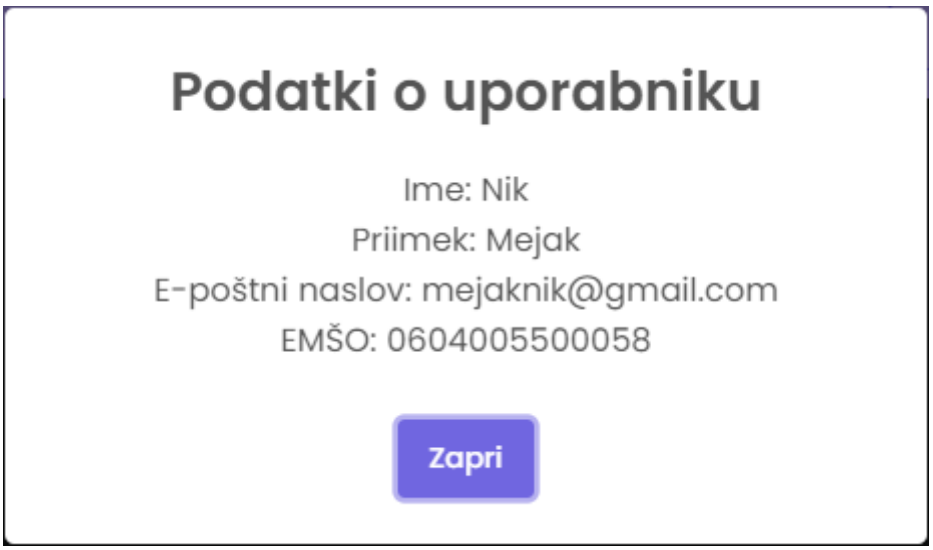
4.7.7 Zahtevki za dodelitev kartic

V tem oknu profesor sprejema, ali ima kakšno zahtevo za kartico s strani svojega dijaka, če je ta zaprtega tipa, kar pomeni, da mora uporabnik poslati zahtevo za dodelitev kartice. Če ima organizacija več kartic, se to okno pokaže samo v primeru, da je ena izmed kartic zaprtega tipa.



Slika 20: Primer zahteve za kartico

Če bi profesor želel videti še druge podatke o dijaku, to preprosto naredi s klikom na dijakovo ime in priimek.

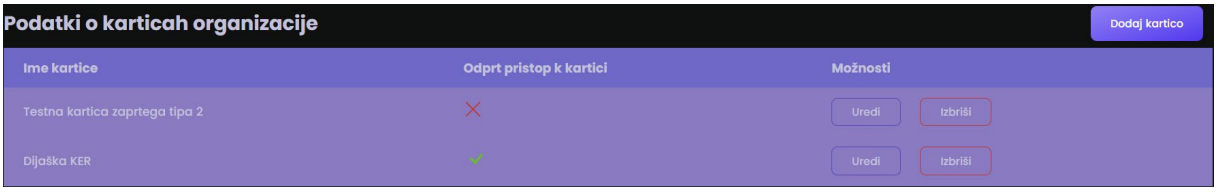


Slika 21: Primer podatkov dijaka

Tako kot profesor ima upravljalec organizacije tudi pregled nad zahtevami za kartico. Razlika je v tem, da ta vidi zahtevke za kartico vseh dijakov, ki pripadajo organizaciji in ne samo od posameznega oddelka.

4.7.8 Pregled kartic organizacije

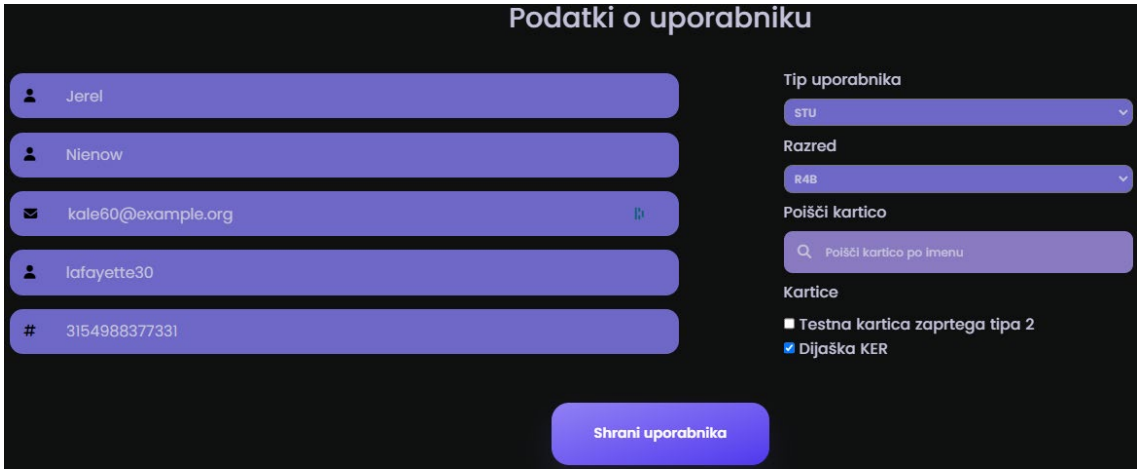
Administrator organizacije tudi pregleduje, katere kartice njegova organizacija izdaja. Kartice lahko organizaciji tudi doda.



Slika 22: Prikaz kartic organizacije

4.7.9 Uporabniki organizacije

Upravljalec organizacije ima funkcijo za pregledovanje, kateri uporabniki pripadajo organizaciji, jih izbriše iz organizacije, jim spreminja ime, priimek, e-naslov, uporabniško ime, EMŠO, oddelek in kartice, ki dijaku pripadajo.

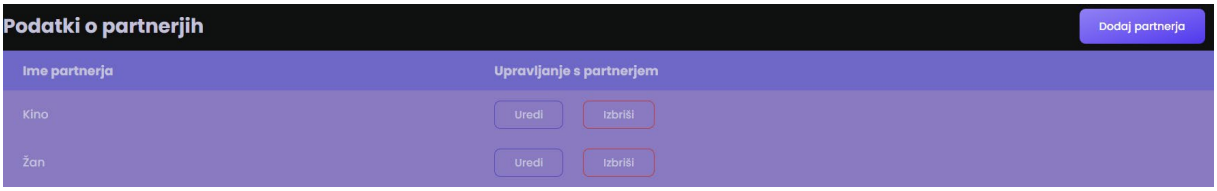


Slika 23: Spreminjanje podatkov o uporabniku

4.7.10 Pregled organizacij in partnerjev

Med zadnjimi funkcijami naše rešitve sta še dve funkciji sistemskega administratorja. Ta lahko dodaja organizacije, jih briše in jim popravlja podatke ter identično dela tudi s partnerji aplikacije.

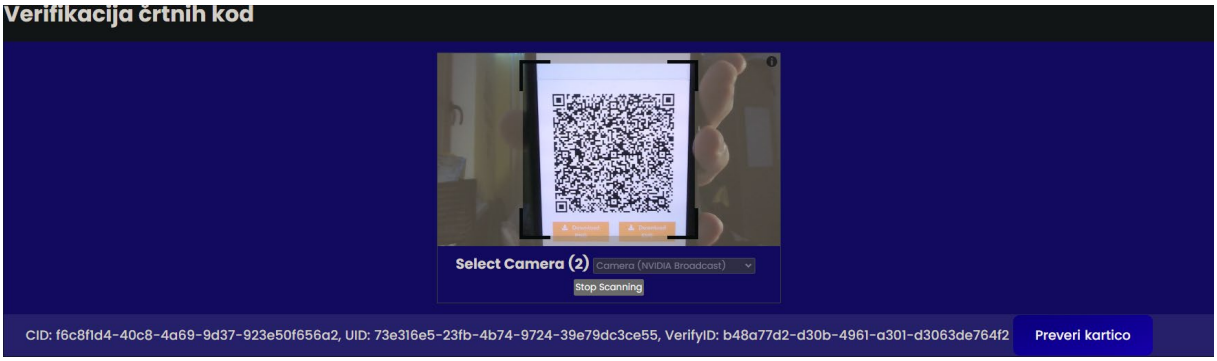
Slika 24: Slika seznama organizacij



Slika 25: Slika seznama partnerjev

4.7.11 Verifikacija črtnih kod

Zadnja funkcija, ki jo ponuja naša rešitev, je možnost pregledovanja, ali je kartica, ki nam jo prikazuje dijak, res vnesena v podatkovno bazo pri določeni organizaciji. Ta funkcija znatno zmanjša možnost ponarejanja kartic, saj se ob kliku dijaka na gumb verifikacije kartice ustvari nov zapis v podatkovni bazi, ki je veljaven samo 2 minuti.

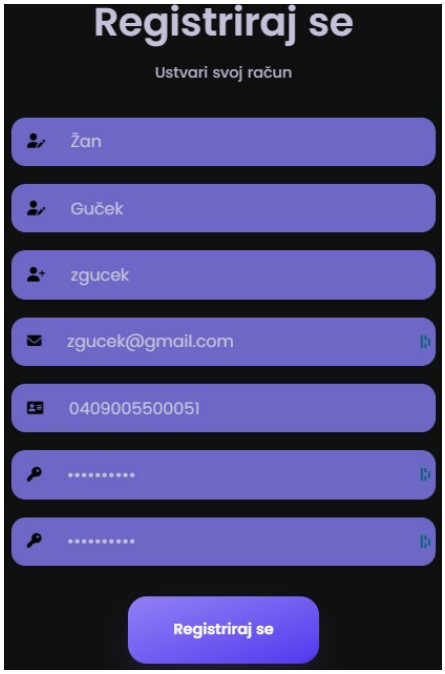


Slika 26: Prikaz vgrajenega črtnega bralnika QR kod

4.8 Praktični prikaz delovanja aplikacije

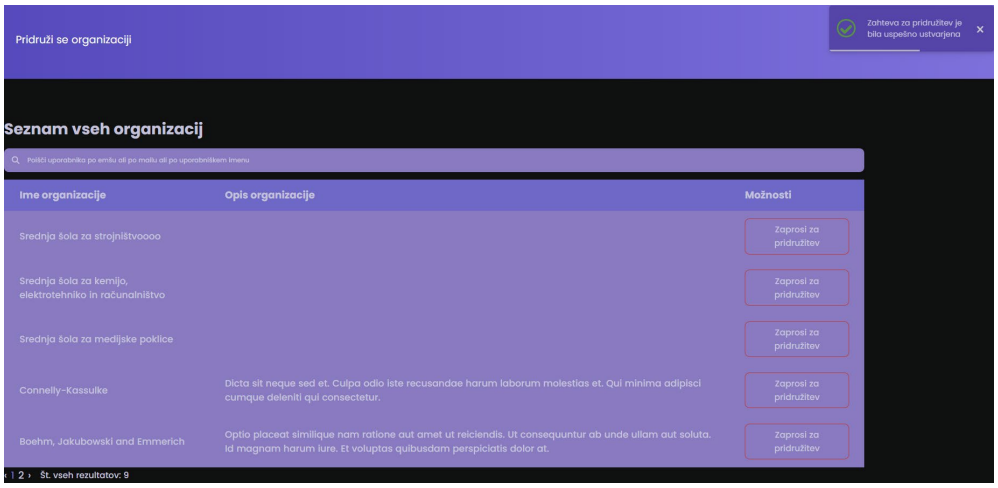
Da bo delovanje naše aplikacije razumljivejše, bomo predstavili, kako naša aplikacija deluje na praktičnem primeru, vse od registracije pa do vseh funkcij, ki smo jih predstavili.

Na začetku naše aplikacije si mora uporabnik ustvariti račun, če mu ta ni dodeljen s strani organizacije. To preprosto naredi z oknom za registracijo.



Slika 27: Primer registracije

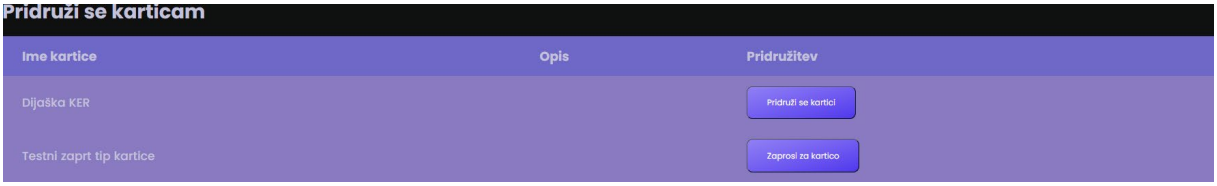
Če so vsi podatki pravilni in še ne obstaja kakšen uporabnik z enakim e-naslovom, uporabniškim imenom ali EMŠO, se uporabnik prijavi v naš sistem. Ob prijavi uporabnik vidi vse svoje podatke, ki jih je vnesel ob registraciji in pa zavihek za pridružitev organizaciji. Dokler se ne pridruži organizaciji, uporabnik ne more zahtevati kartice. Ko zahtevo odda, ga v organizacijo doda lahko samo administrator organizacije, ki prejme obvestilo za zahtevo pri pridružitvi k organizaciji.



Slika 28: Uspešno oddana zahteva za pridružitev

Slika 29: Obvestilo o zahtevi za pridružitev

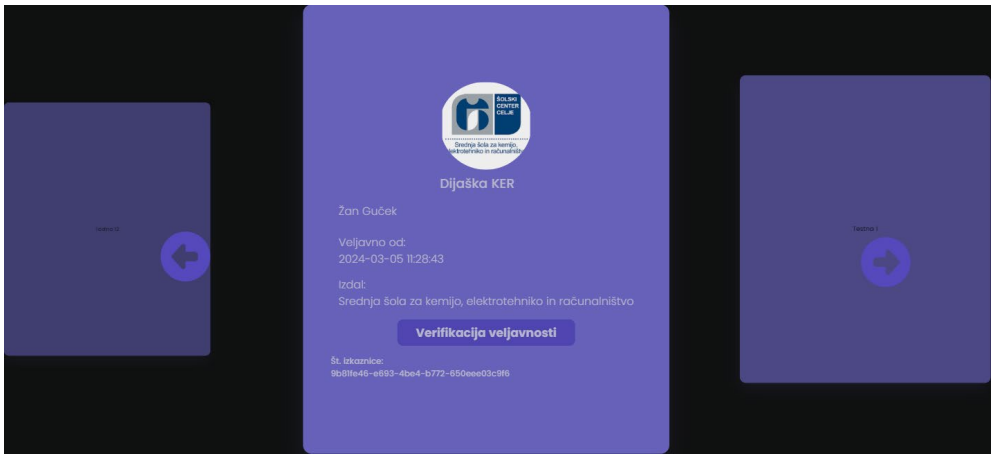
Ko je uporabnik uspešno potrjen s strani upravljalca organizacije, ali pa je dodan s strani profesorja k svojemu razredu, dijak sedaj vidi svoje dodeljene kartice ali pa te zahteva oziroma se jim pridruži, glede na to, ali se ima možnost kartici samo pridruži ali mora poslati prej prošnjo.



Slika 30: Primer dodajanja kartic v svoj račun

Ob uspešni pridružitvi k kartici oziroma odobreni zahtevi za kartico, dijak svoje kartice vidi pod zavihkom moje kartice.

Slika 31: Primer zahtevka za kartico

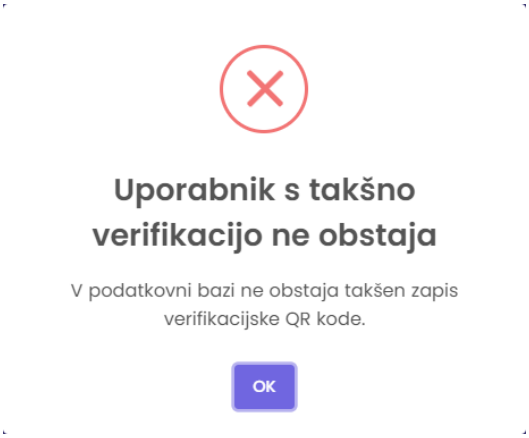


Slika 32: Kartice dijaka

Ko ima dijak končno kartice, z njimi izkazuje pripadnost neki organizaciji oziroma, da je uporabnik res vnesen v naš sistem.

Slika 33: Prikaz uspešne verifikacije

Ob primeru, da takšnega verifikacijskega zapisa s QR kodo ne najdemo v podatkovni bazi, se nam izpiše naslednje sporočilo.



Slika 34: Neuspešno sekeniranje črtne kode

5 PREDSTAVITEV REZULTATOV RAZISKOVALNE NALOGE

Od samega začetka projekta smo želeli doseči predvsem eno stvar – odzivnost in stabilnost naše aplikacije tudi pri veliko uporabnikih ter njeno varnost. Z uporabo Laravel ogrodja za čelni in zaledni del smo odzivnost aplikacije zagotovo izboljšali, vprašanje pa je za koliko.

Našo aplikacijo smo primerjali z lanskoletno aplikacijo, ki je imela podobno idejno zasnovo, vendar je bila postavljena na Vue.js čelnem delu in PHP zaledju. Aplikaciji smo, zaradi doseganja čim bolj relevantnih rezultatov, preizkusili v istem omrežju, na istem strežniku (4vCPU jedra, 6GB RAM pomnilnika, 100GB NvME SSD diska) in pri enakem številu vnosov v podatkovno bazo (10 000).

Ugotovitve kažejo, da se je izboljšal reakcijski čas prav vsake CRUD (create, read, update, delete oziroma ustvari, preberi, posodobi in izbriši) operacije. To je opazno že pri sami prijavi v sistem, saj se je čas, od poslane zahteve pa do prikaza začetne strani, izboljšal iz 240 ms na 130 ms, hkrati pa se je zaradi uporabe spremenljivk seje in zapisa v podatkovni bazi ter uporabe funkcij za nadzor nad zahtevami povečala varnost pred vdori v sistem in krajo podatkov.

Zaradi uporabe Redis podatkovne baze se je izboljšala tudi obremenjenost strežnika, saj smo eliminirali ponavljajoče se poizvedbe in tako uspeli prihraniti okoli 30 ms na vsako poizvedbo, saj Redis vrednosti vrača direktno iz RAM pomnilnika, čas od zahteve do rezultata pa redko preseže 2 ms.

Pridobivanje profilnih podatkov iz podatkovne baze se je izboljšalo za 60 ms, tako zdaj sistem za pridobivanje podatkov in njihov prikaz porabi slabih 30 ms, medtem ko jih je prejšnji sistem porabil 3-krat toliko, torej 90 ms.

Z uporabo funkcij, kot je paginacija, pa smo izboljšali tudi obremenjenost strežnika pri pridobivanju velikih setov podatkov, saj so le-ti zdaj razdeljeni v manjše podsete, po katerih se uporabnik lahko sprehaja z menjavo zavihka tabele v aplikaciji. To je povprečen čas pridobivanja 5 zapisov iz baze in paginacijo ostalih postavilo na 33 ms, medtem ko je bila ta številka pri prejšnji aplikaciji, zaradi neuporabe paginacijskih metod in velikih setov podatkov, med 300 ms in 500 ms.

Prednost pri uporabi Laravel Blade ogrodja se pokaže tudi pri nalaganju same strani, saj je ta čas občutno krajši kot pri Vue.js, saj mora le-ta še hidrirati UI na strani klienta, medtem ko Laravel Blade klientu pošlje že generirano HTML stran, kar pomeni, da je pripravljena takoj, ko jo brskalnik prejme.

Tako smo po analizi rezultatov optimizacije zelo zadovoljni, saj se, v normalnih omrežnih pogojih, aplikacija odziva za okoli 60 %–150 % hitreje, odzivni časi pa so zanemarljivo majhni in v redkih primerih presežejo 100 ms.

6 PREDSTAVITEV REZULTATOV HIPOTEZE

Na podlagi prejšnjega poglavja smo hipoteze potrdili ali ovrgli.

Prvo hipotezo, ki se je glasila: »Aplikacija bo imela ob enakih pogojih za najmanj 40 % boljšo odzivnost,« smo potrdili. Naši aplikaciji se je že samo pri prijavnem oknu zmanjšal odzivni čas za kar 45 %. Pri vračanju podatkov iz podatkovne baze pa se je, z uporabo Redis podatkovne baze, odzivni čas zmanjšal za kar 90 %. Z implementacijo paginacije pa smo še izboljšali odzivnost naše aplikacije. S tem smo za prikaz večjega seta podatkov zmanjšali odzivnost za kar 89 % pa vse do 93 %.

Drugo hipotezo, ki smo jo zapisali in pravi: »Z uporaba Laravel ogrodja v kombinaciji s procesorjem za predloge Laravel Blade bo boljša za odzivnost aplikacije kot pa uporaba Laravel zalednega sistema in uporaba ogrodja Vue.js za čelni del aplikacije,« smo delno potrdili. Pri prejšnji raziskovalni nalogi smo, kot že zapisano, uporabili PHP za zaledni sistem in Vue.js za čelni sistem. Menimo, da uporaba Laravela za zaledni sistem in Vue.js za čelni del naše aplikacije ne bi bistveno pohitrila.

Predzadnje hipoteze, ki se glasi: »Uporaba knjižnice Auth0, v kombinaciji s SSL certifikatom, bo poskrbela za dovolj dobro varnost naše aplikacije,« pa na žalost nismo mogli potrditi. Pri snovanju raziskovalne naloge smo menili, da bomo lahko to funkcijo preprosto integrirali v našo aplikacijo. Ampak se nismo zavedali, da dostop do Auth0 pridobimo le, če smo podjetja, saj bi potrebovali račun preverjanja našega podjetja. Zaradi tega naše hipoteze nikakor nismo mogli potrditi. Kljub temu pa ima naša aplikacija zelo visoko raven varnosti. Vse te smo podrobno že predstavili v naši raziskovalni nalogi.

Zadnjo hipotezo, ki pravi: »Uporaba predpomnjena uporabnikov in nekaterih podatkov v Redis podatkovni bazi bo izboljšala odzivni čas aplikacije za najmanj 2 0% v primerjavi z uporabo Eloquent ORM poizvedb v SQL podatkovno bazo,« smo z velikim zadovoljstvom tudi potrdili, saj smo z uporabo Redis podatkovne baze prihranili zelo veliko časa. Iz okoli 30 ms na odziv smo prišli na pičle 2 ms odziva strežnika. S tem smo hitrost odziva strežnika zmanjšali za kar 90 %.

7 ZAKLJUČEK

V zaključku naše raziskovalne naloge bi želeli izpostaviti pomembnost pridobljenega znanja skozi razvoj naše aplikacije.

V tem projektu smo pridobili in poglobili dragocene izkušnje ter znanje o uporabi PHP-ja, ki je močno orodje za razvoj spletnih aplikacij. S tem smo poglobili razumevanje o naprednih funkcijah jezika PHP, sintaksi in načinu uporabe, kar nam je omogočilo učinkovito pisanje naše spletne aplikacije.

Z uporabo Laravela smo pridobili vpogled v napredne koncepte razvoja spletnih aplikacij, kot so usmerjanje, kontrolerji, modeli, pogledi, migracije in sejanje podatkov. Prav tako smo se dodobra seznanili s principi MVC arhitekture in njenih številnih prednosti, tako iz DX vidika kot iz vidika stabilnosti ter odzivnosti aplikacije.

Pri uporabi Laravel Blade ogrodja smo pobliže spoznali nove načine doseganja reaktivnosti z uporabo PHP jezika. Prav tako smo podrobneje analizirali prednosti SSR pristopa v primerjavi z JavaScript tehniko hidracije UI ter si še dodatno razširili obzorja tehnologij spletnega razvoja.

Zaradi lažje distribucije smo se bili primorani spopasti tudi z orodjem, ki večini razvijalcev še vedno predstavlja sivo cono – Docker. Po nekaj neuspelih poskusih smo uspeli ustvariti Docker sliko, ki služi kot razvojno in produkcijsko okolje naše aplikacije. S tem smo se dodobra spoznali s prednostmi in slabostmi vzpostavitve takšnih rešitev ter njihovo integracijo na spletne strežnike.

Tako smo mnenja, da smo skozi to raziskovalno nalogo pridobili ogromno znanja in novih izkušenj, ki nam bodo zagotovo v pomoč v nadaljnjem razvoju kariere. Prav tako smo se naučili novih konceptov pri sodelovanju v ekipah na večjih projektih, kar je v očeh večjih podjetji na današnjem trgu dela še kako pomembno in zaželeno.

V prihodnosti bi želeli našo aplikacijo nadgraditi še na več nivojih. Med glavnimi stvarmi bi zagotovo bila implementacija Auth0, da bi še izboljšali varnost naše aplikacije. Uporabniško izkušnjo aplikacije pa bi lahko dodelali še bolj z uporabo raznih ogrodji, ki bi naši aplikaciji dodali še animacije.

Prav tako bi radi v prihodnosti našo aplikacijo preizkusili v realnem okolju. V prihodnosti bi bili zelo veseli sodelovanja s kakšno organizacijo, pri kateri bi tudi preizkusili, kako se naša aplikacija odnese v realnem okolju.

VIRI IN LITERATURA

BizConnect, Inc. (brez datuma). BizConnect. Pridobljeno 20. februar 2024 iz https://bizconnectus.com/

Huang, D. (12. januar 2018). [Learning Laravel] Let's Blade. Pridobljeno 26. februar 2024 iz https://medium.com/@dannyhuang_75970/learning-laravel-lets-blade-fc3f47a8f6a9

ID123 Inc. (brez datuma). ID123. Pridobljeno 20. februar 2024 iz https://www.id123.io/

ISIC Association. (brez datuma). International Student Identity Card. Pridobljeno 20. februar 2024 iz https://www.isic.org/app/

JetBrains s.r.o. (brez datuma). PhpStorm. Pridobljeno 20. februar 2024 iz https://www.jetbrains.com/phpstorm/

Kinsta Inc. (17. november 2023). What Is GitHub? A Beginner's Introduction to GitHub. Pridobljeno 20. februar 2024 iz https://kinsta.com/knowledgebase/what-is-github/

Kopf, B. (brez datuma). The Power of Figma as a Design Tool. Pridobljeno 13. februar 2024 iz https://www.toptal.com/designers/ui/figma-design-tool

Laravel Holdings Inc. (brez datuma). Laravel. Pridobljeno 20. februar 2024 iz https://laravel.com/

Ramos, M. (21. december 2023). Understanding Laravel Blade and How To Use It. Pridobljeno 26. februar 2024 iz https://kinsta.com/blog/laravel-blade/

Redis Ltd. (brez datuma). Redis. Pridobljeno 20. februar 2024 iz https://redis.io/