# Sustav za automatizirano ocjenjivanje ispita

**Mutić, Domagoj**

**Master's thesis / Diplomski rad**

**2024**

*Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj:* **University of Zagreb, Faculty of Electrical Engineering and Computing / Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva**

*Permanent link / Trajna poveznica:* https://urn.nsk.hr/urn:nbn:hr:168:385363

*Rights / Prava:* In copyright / Zaštićeno autorskim pravom.

*Download date / Datum preuzimanja:* **2025-03-14**

*Repository / Repozitorij:*

FER Repository - University of Zagreb Faculty of Electrical Engineering and Computing repozitory


SVEUČILIŠTE U ZAGREBU
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

DIPLOMSKI RAD br. 540

# SUSTAV ZA AUTOMATIZIRANO OCJENJIVANJE ISPITA

Domagoj Mutić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

DIPLOMSKI RAD br. 540

# SUSTAV ZA AUTOMATIZIRANO OCJENJIVANJE ISPITA

Domagoj Mutić

Zagreb, lipanj 2024.

**SVEUČILIŠTE U ZAGREBU**
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

Zagreb, 4. ožujka 2024.

# DIPLOMSKI ZADATAK br. 540

| | |
|---|---|
| Pristupnik: | **Domagoj Mutić (0036523903)** |
| Studij: | Računarstvo |
| Profil: | Programsko inženjerstvo i informacijski sustavi |
| Mentor: | prof. dr. sc. Vedran Mornar |

Zadatak: **Sustav za automatizirano ocjenjivanje ispita**

Opis zadatka:

Ostvariti sustav za automatizirano ocjenjivanje ispita. U sustavu treba omogućiti generiranje obrazaca za odgovore, tisak, skeniranje ili fotografiranje te pohranu i automatsko ispravljanje rezultata. Omogućiti postavljanje pitanja s ponuđenim odgovorima i pitanja otvorenog tipa. Za prepoznavanje slike koristiti neki sustav otvorenog koda. Sustav, osim u komunikaciji s udaljenim poslužiteljem, treba raditi i bez povezanosti s internetom. Ugraditi odgovarajuću autentikaciju i autorizaciju. Posebnu pažnju posvetiti ergonomiji i jednostavnosti programskog sučelja. Sustav ostvariti koristeći neki od JavaScript programskih okvira, uz Electron ili PWA potporu radu bez povezanosti s internetom, Node.js na strani poslužitelja te SQLite bazu podataka, mogućnošću prijelaza i na neku drugu bazu.

Rok za predaju rada: 28. lipnja 2024.

# SADRŽAJ

# 1. Uvod

Tehnološki napredak u posljednjih nekoliko desetljeća donio je značajne promjene u mnogim područjima, no unatoč tome, određeni tradicionalni pristupi i dalje su prisutni. Jedan od takvih primjera je način provođenja i ispravljanja ispita, gdje se i dalje često koristi metoda pisanja olovkom i gumicom. Iako neki izražavaju skepticizam prema promjenama koje donosi moderna tehnologija, neosporno je da bi digitalizacija ovog procesa mogla značajno unaprijediti učinkovitost, posebno u segmentu ispravljanja ispita.

Ovaj diplomski rad bavi se razvojem aplikacije koja predstavlja hibridno rješenje, kombinirajući prednosti tradicionalnog pristupa s mogućnostima suvremenih tehnologija. Aplikacija omogućuje kreiranje i automatizirano ispravljanje ispita, dok istovremeno zadržava elemente klasičnog pisanja rukom. Ovakav model već je prisutan u nekim obrazovnim institucijama, poput fakulteta te na državnoj maturi, no cilj ove aplikacije je unaprijediti taj proces, čineći ga dostupnijim i fleksibilnijim. Dodatno, omogućuje se jednostavan razvoj novih tipova ispitnih pitanja.

U nastavku rada bit će prikazan pregled postojećih pristupa ispravljanja ispita, analizirani zahtjevi sustava te prikazano trenutno stanje aplikacije, analizirane primijenjene tehnologije i objašnjeni tehnički detalji implementacije. Također će biti predstavljeni potencijalni pravci za daljnji razvoj aplikacije i načini njene nadogradnje.
Izvorni kod projekta dostupan je na GitHub repozitoriju:
https://github.com/domagojmutic/examiner-diplomski-rad

# 2. Analiza postojećih rješenja

U razvoju sustava za automatizirano ocjenjivanje ispita, ključni je korak analiza postojećih rješenja koja su već implementirana u obrazovnim institucijama, a često i u širem kontekstu evaluacijskih sustava. Ova analiza pruža uvid u trenutačne tehnološke trendove, identifikaciju prednosti i nedostataka tih sustava, kao i mogućnosti za inovaciju. U ovom poglavlju ćemo razmotriti nekoliko poznatih pristupa i tehnika koje se koriste za ispravljanje ispita, te ih usporediti prema relevantnim kriterijima kao što su točnost ocjenjivanja, brzina obrade i mogućnosti prilagodbe.

## 2.1. Ručno ispravljanje

Ručno ispravljanje je i dalje standardna praksa u mnogim obrazovnim institucijama, osobito kada je riječ o složenijim tipovima ispita koji zahtijevaju evaluaciju kritičkog razmišljanja, analize i kreativnih rješenja. Ovakav pristup podrazumijeva pregledavanje i ocjenjivanje ispita od strane nastavnika ili ocjenjivača bez pomoći tehnologije, što često uključuje temeljitu analizu eseja, otvorenih pitanja i specifičnih zadataka.

Glavne prednosti ovakvog pristupa su prilagodba pojedinačnim odgovorima i kontekstualna evaluacija.
Ručno ispravljanje omogućava ocjenjivaču da uzme u obzir sve nijanse odgovora studenata, uključujući način izražavanja, originalnost i logičku povezanost argumenata. Takva fleksibilnost je od ključne važnosti kod eseja ili zadataka s kreativnim rješenjima. Nadalje, ljudski ocjenjivači mogu sagledati širi kontekst odgovora, uključujući relevantne aspekte koji možda nisu jasno definirani rubrikama ili uputama za ocjenjivanje. To omogućuje da se izbjegne kruto ocjenjivanje koje bi moglo biti problem kod automatiziranih sustava.

S druge strane neki od nedostataka su: subjektivnost, vrijeme i napor te neučinkovitost.

Ručno ocjenjivanje velikog broja ispita zahtijeva značajno vrijeme i trud, što može dovesti do zasićenja i umora ocjenjivača. To posebno dolazi do izražaja kod masovnih ispita s velikim brojem studenata, gdje se proces može protegnuti na duži period. Takav pristup je vrlo teško održiv u velikim obrazovnim sustavima koji zahtijevaju brze povratne informacije studentima.

## 2.2. Online sustavi za ocjenjivanje

S razvojem e-učenja, sve su popularniji online sustavi za ocjenjivanje ispita. Ovi sustavi obično nude mogućnost izrade digitalnih ispita s različitim tipovima pitanja – od višestrukog izbora, točno/netočno pitanja, pa do esejskih odgovora. Sustavi poput Moodle, Canvas i Blackboard široko su korišteni u obrazovnim institucijama širom svijeta.

Prednost ovakvog sustava je mogućnost automatizacije različitih vrsta pitanja. Ovakvi sustavi omogućuju automatizirano ocjenjivanje ne samo jednostavnih pitanja nego i složenih vrsta od kojih su neke vrlo teško izvedive na papiru. Također, mogu uključivati funkcionalnosti poput objave rezultata, slanja izvješća te izrade statistike.

Glavni nedostatak ovakvog provođenja ispita je sigurnost. Čak kada se ispiti pišu u kontroliranom okruženju i dalje postoji puno veća mogućnost za varanjem. Drugi problem predstavlja potreba za infrastrukturom. Potrebno je osigurati stabilnu internetsku vezu te računala za pristup ispitu.

## 2.3. Sustavi za skeniranje ispita

Sustavi koji koriste optičko prepoznavanje oznaka (engl. OMR - Optical Mark Recognition) već su dugo prisutni u obrazovnim institucijama. Ovi sustavi koriste posebne formulare s unaprijed definiranom strukturom, na kojima studenti označavaju svoje odgovore (obično višestruki izbor). S pomoću OMR tehnologije, ispitni listovi se skeniraju, a odgovori se automatski prepoznaju i ocjenjuju.

Neke od prednosti ovakvih sustava su: brzina i učinkovitost, točnost te objektivnost. OMR sustavi omogućuju brzu obradu velikog broja ispita, što je posebno korisno u institucijama s velikim brojem studenata. Nadalje takav oblik ispravljanja ispita se pokazao vrlo preciznim i točnim, a uz to je i objektivan. Ovakvi ispiti se uglavnom

sastoje od pitanja s ponuđenim odgovorom te tako uklanjaju faktor subjektivnosti pri ispravljanju.

Jedna od glavnih prednosti je ujedno i mana. Naime zbog ograničenog oblika pitanja javlja se i ograničena fleksibilnost. Pitanja koja su postavljena uvijek moraju moći biti prikazana kao pitanja s ponuđenim odgovorima.

## 2.4. Hibridni sustav

U suvremenom obrazovanju, često se koristi kombinacija ručnog i automatiziranog ispravljanja kako bi se maksimalno iskoristile prednosti obje metode. Primjerice, automatizirani sustavi mogu obraditi jednostavna pitanja s višestrukim izborom, dok se složenija pitanja i eseji prepuštaju ručnom ispravljanju. Ovakav pristup omogućuje nastavnicima da se fokusiraju na evaluaciju kritičkog mišljenja i kreativnih rješenja, dok sustavi preuzimaju rutinske zadatke ocjenjivanja.

Integracija ovih metoda smanjuje ukupno vrijeme ispravljanja, povećava točnost i pouzdanost ocjenjivanja, a ujedno omogućuje personalizirani pristup gdje je to potrebno. Tako se kombiniraju prednosti brzine i preciznosti automatizacije s ljudskom sposobnošću da prepozna i vrednuje složene odgovore.

Ovaj rad predstavlja upravo jedan ovakav hibridni sustav koji nudi i mogućnost daljnjeg proširenja na razne tipove pitanja.

# 3. Zahtjevi sustava

U ovom poglavlju se opisuju okvirni zahtjevi potrebni za izgradnju i funkcioniranje sustava za automatizirano ocjenjivanje ispita.

Kako bi aplikacija ispunila svoje ciljeve i bila dovoljno fleksibilna za različite scenarije, potrebno je pažljivo odabrati tehničke komponente koje će omogućiti brzinu, sigurnost i skalabilnost. Stoga je potrebno aplikaciju implementirati u tri dijela frontend, backend i sustav za pohranu podataka.

## 3.1. Funkcionalni zahtjevi

Sustav ima nekoliko ključnih funkcionalnosti koje su nužne za pravilno funkcioniranje i omogućavanje zadovoljavajućeg korisničkog iskustva. U nastavku su opisani funkcionalni zahtjevi sustava.

U ranim fazama sustav može podržavati jednog korisnika koji ima pristup svim podacima i funkcionalnostima, no dugoročno je potrebno omogućiti podršku za više korisnika. U budućim fazama razvoja, sustav treba osigurati različite korisničke uloge, uključujući administratore, profesore i studente.

Sustav mora omogućiti pohranu skeniranih obrazaca ispita te njihovo pregledavanje. Nakon što su obrasci pohranjeni, profesor može pregledati rezultate te izmijeniti ili nadopuniti odgovore, a sustav automatski generira izvještaj o rezultatima obrade.

Potrebno je omogućiti jednostavnu izradu ispita i pitanja. Profesoru mora biti omogućeno kreiranje različitih vrsta pitanja, kao što su višestruki izbor, točno/netočno, te pitanja otvorenog tipa.

Sustav mora omogućiti generiranje ispita i odgovarajućih obrazaca za odgovore u PDF

formatu. Kreirani ispiti i obrasci trebaju biti spremni za jednostavan tisak, čime se osigurava fizičko distribuiranje ispita studentima u papirnatom obliku.

Dohvat svih podataka iz sustava mora biti jednostavan i intuitivan. Potrebno je osigurati korisničko sučelje, čime se omogućava brz pristup svim relevantnim informacijama.

Ispravljanje obrazaca mora biti automatizirano, barem koliko je to moguće. Nakon ispravljanja mora se osigurati izvješće te omogućiti izmjene tog izvješća ako se za time ukaže potreba.

Poželjno je osigurati fleksibilnost i proširivost sustava kako bi podržao nove tipove pitanja u budućnosti.

## 3.2. Nefunkcionalni zahtjevi

Nefunkcionalni zahtjevi definiraju kvalitativne karakteristike sustava koje se odnose na performanse, skalabilnost, proširivost, kompatibilnost i sigurnost. Ovi zahtjevi osiguravaju da sustav za automatizirano ocjenjivanje ispita bude pouzdan, učinkovit i siguran u radu.

Sustav mora osigurati zadovoljavajuće performanse kako bi omogućio pravovremenu obradu velike količine podataka. Mora biti sposoban obraditi veliki broj zahtjeva, uključujući učitavanje skeniranih obrazaca, ispravljanje ispita i generiranje rezultata, unutar razumnog vremena. Odgovori sustava trebaju biti dovoljno brzi da ne ometaju tijek rada korisnika, čak i u slučaju većeg broja ispita ili studenata.

Potrebno je osigurati skalabilnost, kako bi podržao umjereno velik broj paralelnih korisnika. To podrazumijeva mogućnost istovremenog pristupa sustavu, bez narušavanja performansi. Ova skalabilnost omogućuje sustavu prilagodbu rastućim potrebama, uključujući veći broj studenata, institucija i ispita.

Sustav mora biti dizajniran tako da podržava jednostavne nadogradnje i proširenja. To znači da će u budućnosti biti moguće dodavati nove funkcionalnosti bez potrebe za značajnim promjenama postojeće infrastrukture. Proširivost sustava osigurava njegovu dugoročnu održivost i prilagodljivost različitim potrebama korisnika, kao što je doda-

vanje novih tipova pitanja ili integracija s drugim sustavima.

Nužno je omogućiti rad na različitim platformama, uključujući operativne sustave kao što su Windows, macOS i Linux. Također, kompatibilnost se mora proširiti na najčešće korištene preglednike (Chrome, Firefox, Edge, Safari). Ova fleksibilnost osigurava da korisnici mogu koristiti sustav bez obzira na hardversko ili softversko okruženje koje koriste, čime se olakšava pristup i upotreba sustava u različitim institucijama.

Zaštita osjetljivih podataka je vrlo bitna komponenta sustava. Sustav mora osigurati adekvatne mehanizme zaštite privatnosti korisnika i njihovih podataka. To uključuje zaštitu podataka o studentima, rezultatima ispita i drugim osjetljivim informacijama s pomoću odgovarajućih sigurnosnih protokola, poput SSL enkripcije. Također, poželjno je da sustav osigura sigurnosne kopije podataka i mehanizme za oporavak podataka u slučaju tehničkih problema ili sigurnosnih incidenata.

# 4. Korištene tehnologije

## 4.1. Frontend

Sučelje aplikacije ključna je stavka, budući da će većina korisnika pristupati aplikaciju upravo putem njega. Na tržištu postoji veliki broj okvira za razvoj korisničkih sučelja, a svaki od njih ima svoje prednosti i nedostatke. Među najpoznatijima su Vue.js, Angular, React i Svelte. Osim ovih osnovnih okvira, dostupni su i metaokviri koji proširuju njihove mogućnosti, poput Next.js za React, Nuxt.js za Vue i Quasar za Vue.

Za potrebe ovog rada odabrani su Vue.js i Quasar. Odluka je donesena zbog široke palete funkcionalnosti koje Quasar nudi, a ključna među njima bila je integracija s Electron paketom za razvoj aplikacija na više platformi. Iako je integracija s Electronom bila predviđena, zbog određenih ograničenja ta mogućnost nije u potpunosti iskorištena.

Također, jednako važan aspekt bio je i odabir biblioteke za komunikaciju između sučelja i poslužitelja. Danas se ta funkcionalnost može implementirati korištenjem Fetch API-a, koji je dio JavaScripta, no postoje biblioteke koje dodatno pojednostavljuju i proširuju njegove mogućnosti. Jedna od takvih biblioteka je Axios, koja je korištena i u ovom projektu. Uz Axios, korištena je i biblioteka Vue Query, koja dodatno olakšava funkcionalnosti poput keširanja podataka (caching).

### 4.1.1. Vue.js [10]

Vue.js je JavaScript okvir za izgradnju interaktivnih web korisničkih sučelja, kojeg je 2014. godine kreirao Evan You. Trenutno se koristi verzija 3 ovog okvira.

Jedna od ključnih prednosti Vue.js-a je njegova fleksibilnost u načinu korištenja. Može se lako integrirati u postojeće projekte, ali također omogućuje razvoj potpunih web

aplikacija. Uz to, podržava izradu stranica generiranih na poslužitelju (SSR), a može se implementirati i kao web komponenta na bilo kojoj web stranici.

Preporučeni način rada s Vue.js-om je korištenje jedinstvenih datoteka komponenata (Single File Components, SFC), koje omogućuju pisanje HTML, JavaScript i CSS koda unutar iste datoteke. Ova značajka olakšava razvoj, no Vue omogućuje i razdvajanje tih elemenata u zasebne datoteke, ako je to potrebno.

U radu s Vue.js-om trenutno postoje dva pristupa za pisanje koda: 'Options API' i 'Composition API'. Oba pristupa omogućuju postizanje istog ponašanja, ali se razlikuju u načinu organizacije i upravljanja reaktivnošću. 'Options API' se oslanja na koncept instance komponente te apstrahira dio reaktivnosti, dok 'Composition API' pruža veću fleksibilnost u organizaciji koda, ali zahtijeva veću odgovornost programera za upravljanje reaktivnošću pojedinih objekata.

Vue.js, u kombinaciji s proširenjima kao što su Pinia i Vue Router, nudi cjelovito i fleksibilno rješenje za različite potrebe. Od izrade malih, izoliranih komponenti do velikih produkcijskih web aplikacija, Vue.js pruža skalabilan i prilagodljiv okvir koji zadovoljava širok raspon zahtjeva u razvoju web sučelja.

**Slika 4.1:** Vue.js logo [13]

### 4.1.2. Quasar [7]

Quasar je moćan okvir koji, u kombinaciji s Vue.js-om, značajno olakšava i ubrzava razvoj web stranica i aplikacija. Pruža podršku za različite tipove aplikacija, uključujući jednostruke stranice (SPA - Single Page Application), aplikacije renderirane na poslužitelju (SSR - Server-Side Rendered Application), te progresivne web aplikacije (PWA - Progressive Web App). Osim toga, omogućava razvoj mobilnih aplikacija putem Capacitor platforme te aplikacija za računala koristeći Electron.

Jedna od ključnih prednosti Quasara je njegova jednostavna integracija Vue.js-a u različite vrste aplikacija. Uz to, Quasar nudi širok spektar gotovih, prilagodljivih kompo-

nenti, što značajno ubrzava razvoj korisničkog sučelja. Također, okvir dolazi s ugrađenim klasama za stiliziranje koje, slično Bootstrapu, pojednostavljuju dizajniranje i prilagodbu izgleda stranice.

Dodatno, Quasar pruža podršku za izradu višejezičnih stranica te u potpunosti podržava RTL (Right-to-Left) orijentaciju, čime omogućava razvoj stranica i aplikacija na jezicima koji se čitaju s desna na lijevo.


**Slika 4.2:** Quasar logo [7]

### 4.1.3. Axios [1]

Axios je HTTP klijent temeljen na obećanja (promises), koji je kompatibilan s Node.js okruženjem i web preglednicima. U preglednicima koristi XMLHttpRequests, dok u Node.js-u koristi nativni HTTP modul.

Axios pruža niz korisnih značajki koje olakšavaju rad s HTTP zahtjevima. Neke od tih značajki uključuju postavljanje vremenskog ograničenja za zahtjeve, mogućnost otkazivanja zahtjeva, automatsku serijalizaciju JSON podataka, podršku za slanje Multipart form podataka te enkodiranje URL-ova.


**Slika 4.3:** Axios logo [1]

### 4.1.4. Vue Query [9]

Vue Query je implementacija TanStack Query biblioteke prilagođena za Vue.js. TanStack Query je moćna biblioteka koja olakšava dohvat podataka, keširanje, te upravljanje i sinkronizaciju podataka sa serverom. Pruža implementacije za sve veće i popularne JavaScript okvire, dok je originalno razvijena za React.

Vue Query značajno pojednostavljuje rad s podacima, rješavajući brojne izazove pri njihovom dohvaćanju. Neke od funkcionalnosti koje nudi uključuju sprječavanje du-

plih zahtjeva za istim resursom, automatsku obnovu zastarjelih podataka te učinkovito keširanje.


**Slika 4.4:** TanStack logo [9]

## 4.2. Backend

Poslužitelj je neizostavan dio svake web stranice. U mnogim slučajevima, njegova glavna uloga svodi se na dostavljanje HTML, CSS i JavaScript koda, dok se većina logike odvija na strani klijenta, odnosno u web pregledniku. No, u slučaju web aplikacija, poslužitelj često preuzima puno važniju ulogu, omogućujući složene funkcionalnosti kao što su autentifikacija, obrada podataka i interakcija s bazama podataka.

Poslužiteljski se kod danas može pisati u mnogim programskim jezicima, od kojih su neki prikladniji za određene zadatke, ovisno o specifičnim zahtjevima projekta. Uobičajeno je koristiti okvire koji pojednostavljuju izradu poslužitelja. Među popularnim jezicima i okvirima su: C#, Java (Spring Boot), Golang, Python (Flask, Django), te Node.js s JavaScriptom (Express, Fastify).

Uz odabir jezika i okvira, ključan je i arhitekturni stil poslužitelja. Jedan od najraširenijih pristupa je REST (Representational State Transfer), koji omogućuje strukturiranu komunikaciju između klijenta i poslužitelja putem API-ja. Osim REST-a, danas se sve više koriste i drugi pristupi, poput GraphQL-a, gRPC-a i SOAP-a, ovisno o specifičnim potrebama aplikacije i zahtjevima korisnika.

Ovisno o jeziku i okviru, često su potrebne dodatne funkcionalnosti koje se implementiraju putem paketa ili proširenja. Neke od tih funkcionalnosti uključuju dokumentiranje API-ja, validaciju podataka, generiranje dokumenata i obradu teksta.

Za ovaj rad odabrani su TypeScript (JavaScript proširen tipovima podataka) i Fastify okvir. Ova kombinacija odabrana je s dugoročnim planom da se poslužiteljski kod (ili njegov dio) može integrirati unutar Electron aplikacije, čime bi se omogućila velika

prenosivost aplikacije na različite operativne sustave. Fastify je izabran jer najpopularniji Node.js okvir, Express, u vrijeme odabira nije imao ažuriranja gotovo dvije godine, što je potaknulo sumnje o njegovoj budućnosti. Fastify se pokazao kao dostojna zamjena s dodatnim značajkama korisnim za ovaj projekt.

Arhitekturni stil koji je primijenjen u ovom radu je REST, iako nije implementiran do razine 3 prema Richardsonovom modelu zrelosti.

Od ostalih korištenih paketa, odabrani su: Zod za validaciju objekata (ulaznih i izlaznih podataka), OpenCV za analizu slika te Puppeteer za generiranje PDF dokumenata.

### 4.2.1. REST API

REST (Representational State Transfer) je arhitekturni stil koji definira način komunikacije između udaljenih računala, odnosno klijenata i poslužitelja. Postavlja skup pravila koja aplikacije moraju slijediti kako bi se smatrale Restful sustavima ili aplikacijama. [19]

Jedna od ključnih značajki REST-a je strogo razdvajanje klijenta i poslužitelja. To znači da klijent pristupa poslužitelju putem dobro definiranih točaka pristupa (putanja) na kojima može očekivati određene resurse. Ova razdvojenost omogućuje paralelan i nezavisan razvoj klijenta i poslužitelja. Poslužitelj se može nadograđivati sve dok ne naruši postojeće točke pristupa ili strukturu podataka koji se vraćaju klijentu. Dobra praksa je uključivanje verzije API-ja u URL putanju kako bi se spriječilo narušavanje funkcionalnosti klijenta u slučaju promjena. [19]

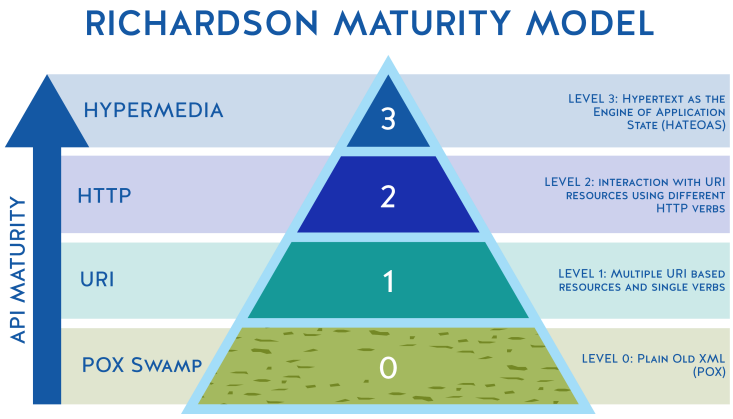
Restful sustavi ne bi trebali pohranjivati stanje klijenta. Klijent, prilikom slanja zahtjeva, mora dostaviti sve potrebne informacije za obradu. Ovaj pristup omogućuje veću učinkovitost, posebno u sustavima koji trebaju podržavati velik broj klijenata. [19]

REST se može kategorizirati prema Richardsonovom modelu zrelosti, koji REST sustave dijeli na četiri razine zrelosti.
Level 0 opisuje sustav s jednom jedinom krajnjom točkom koja koristi samo jednu HTTP metodu, najčešće POST. U ovom slučaju, zahtjev sadrži sve potrebne informacije o akciji koja se želi izvršiti. Takvi sustavi se ne smatraju pravim REST sustavima. [17]

Level 1 dodaje više URL adresa, pri čemu svaka adresa predstavlja specifičan resurs. Svaki resurs ima jedinstveni URI (Uniform Resource Identifier), no sustav i dalje koristi samo jednu metodu (uglavnom POST). Ova razina uvodi "imenice" u sustav, s obzirom na to da se resursi identificiraju po nazivima. [17]

Level 2 uvodi korištenje različitih HTTP metoda (GET, POST, PUT, PATCH, DELETE itd.), čime omogućuje preciznije upravljanje resursima. Na ovoj razini, svaki resurs je jedinstveno identificiran i nad njim se mogu obavljati CRUD operacije (kreiranje, čitanje, ažuriranje i brisanje). Ova razina uvodi "glagole" u REST API, budući da različite metode označavaju različite akcije nad resursima. [17]

Level 3 predstavlja najvišu razinu zrelosti, koja je rijetko u potpunosti implementirana. Ova razina uključuje HATEOAS (Hypermedia As The Engine Of Application State), gdje svaki resurs uz podatke vraća i poveznice na druge resurse i akcije koje se nad tim resursima mogu izvršiti. Cilj HATEOAS-a je omogućiti korisniku ili klijentu da, bez proučavanja vanjske dokumentacije, dobije sve potrebne informacije iz odgovora poslužitelja, čime se olakšava navigacija unutar sustava. [17]




**Slika 4.5:** Prikaz Richardsonovog modela zrelosti [12]

### 4.2.2. OpenAPI (Swagger) [5]

OpenAPI je standardizirana specifikacija koja definira način dokumentiranja HTTP API-ja. Njena glavna svrha je olakšati komunikaciju između ponuditelja API-ja i njegovih korisnika, omogućujući brz i učinkovit prijenos znanja o funkcionalnostima API-ja. Dokumentacija se piše u JSON ili YAML formatu te je potpuno neovisna o

programskom jeziku koji se koristi za razvoj poslužitelja. U kombinaciji s odgovarajućim alatima i proširenjima, OpenAPI omogućuje vizualni prikaz i testiranje API-ja, što znatno ubrzava razvoj i implementaciju.

Prvotna verzija ove specifikacije bila je poznata pod nazivom Swagger, kojeg je razvila tvrtka SmartBear Software. Tvrtka je kasnije prepustila razvoj specifikacije organizaciji The OpenAPI Initiative, koja djeluje pod okriljem Linux Foundation. Danas, ova organizacija i dalje razvija i unaprjeđuje OpenAPI specifikaciju, koja je postala industrijski standard za dokumentiranje API-ja.


**Slika 4.6:** The OpenAPI Initiative logo [5]

### 4.2.3. Fastify [3]

Fastify je napredni okvir za razvoj brzih i učinkovitih poslužitelja u Node.js okruženju. Dizajniran je s naglaskom na jednostavnu proširivost putem sustava dodataka. Dio dodataka razvija i održava tim Fastify-a, dok mnoge druge kreira i održava zajednica koja se okuplja oko ovog okvira.

Inspiriran okvirima kao što su Express i Koa, Fastify se danas ističe kao jedan od najbržih poslužiteljskih okvira u Node.js ekosustavu. Njegova proširivost ostvaruje se kroz sustav dodataka (engl. plugins), dekoratora i kuka za upravljanje događajima na poslužitelju. Jedna od značajnih prednosti Fastify-a je njegova podrška za JSON Schema, koja omogućava precizno definiranje strukture podataka, pomaže u izradi dokumentacije te kontrolira slanje i primanje podataka i objekata.

Dodatno, Fastify nudi nativnu podršku za TypeScript, što olakšava razvoj poslužitelja pisanih u ovom jeziku. Također, sadrži integrirani sustav za bilježenje (logging) koji omogućuje praćenje i analizu rada poslužitelja tijekom njegovog izvršavanja.


**Slika 4.7:** Fastify logo [3]

### 4.2.4. Zod [11]

Zod je napredna biblioteka za deklaraciju i validaciju tipova u JavaScript/TypeScript kodu. Može se koristiti u Node.js okruženju ili u web pregledniku. Uz proširenje za Fastify, omogućava automatsku validaciju svih zahtjeva i odgovora poslanih s poslužitelja.

Zod je dizajniran kao TypeScript-first biblioteka, što znači da je prvenstveno namijenjena za rad s TypeScriptom, ali se može koristiti i s običnim JavaScriptom.

Zod također nudi podršku za napredne tipove kao što su unije, rekurzivni tipovi, i prilagođene sheme, što ga čini vrlo fleksibilnim i moćnim alatom za validaciju podataka.

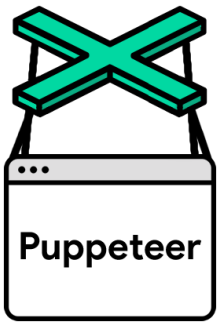
**Slika 4.8:** Zod logo [11]

### 4.2.5. Puppeteer [6]

Puppeteer je moćna JavaScript biblioteka za kontrolu web preglednika, omogućujući upravljanje preglednicima Chrome i Firefox. Često se koristi za testiranje ili prikupljanje podataka s weba (engl. crawling).

Puppeteer nudi mnoge postavke pri pokretanju instance, a jedna od najznačajnijih je pokretanje u bezglavom načinu. To znači da se instanca može pokrenuti u pozadini bez prikaza na ekranu.

Puppeteer pruža brojne mogućnosti koje olakšavaju razvoj, a u nekim slučajevima može biti i centralni dio funkcionalnosti aplikacije. Gotovo sve što korisnik može napraviti na stranici, može i ova biblioteka. Tako se može koristiti za pritisak gumba ili popunjavanje formi. Još jedna korisna mogućnost je generiranje vremenske linije učitavanja stranice, što može biti korisno pri detekciji uzroka sporog učitavanja. Puppeteer također nudi mogućnost pohrane stranice u obliku PDF-a, pri čemu je ispis jednak onome koji bi dobili da se stranica ispisuje ručno.




**Slika 4.9:** Puppeteer logo [6]

## 4.3. Baza podataka

Moderne aplikacije često pohranjuju i poslužuju velike količine podataka. Kako bi se te radnje odvijale što brže i efikasnije, uglavnom se koriste baze podataka.

Danas se baze podataka mogu podijeliti na SQL i NoSQL baze. SQL baze su starije i imaju kruću strukturu, dok NoSQL baze predstavljaju novije pristupe pohrane podataka.

U svijetu SQL baza postoji daljnja podjela, ali nije toliko značajna kao kod NoSQL baza. Sve SQL baze implementiraju osnovni standard na sličan način, barem što se tiče krajnjeg korisnika. Razlike se javljaju u načinu pohrane i nekim manje jasno definiranim područjima specifikacije prilikom pohrane i upravljanja podacima. Neke od popularnijih SQL baza su PostgreSQL, MySQL i SQLite.

Za ovaj rad korištena je SQLite baza podataka, odnosno njena implementacija za NodeJS, 'better-sqlite3'. Ova baza je izabrana zbog svoje prenosivosti i jednostavnosti. Većina drugih baza zahtijeva predinstaliranu infrastrukturu na računalu ili serveru koji ih pokreće, dok SQLite to ne zahtijeva. SQLite pohranjuje svoje podatke u jednu da-

toteku, čime ostvaruje značajnu prenosivost. Budući da je jedan od ciljeva aplikacije bila prenosivost i lakoća instaliranja/pokretanja, upravo je ova baza podataka izabrana kao zadana prilikom nove instalacije.

### 4.3.1. SQLite [8]

SQLite je baza podataka izvorno napisana u programskom jeziku C. Ova baza se pokreće u zasebnom procesu te je samodostatna, što znači da ne zahtijeva poseban server. Unatoč svojoj jednostavnosti, zadržava ključne karakteristike SQL baza, poput podrške za transakcije i 'write-ahead log'.

Podaci u SQLite bazi pohranjuju se u običnu datoteku na disku. Ta datoteka sadrži sve informacije o tablicama i pohranjenim podacima, eliminirajući potrebu za vanjskom komunikacijom. Dodatno, datoteka je prenosiva između različitih računala i operativnih sustava, bez problema s razlikama u arhitekturi poput little i big endiana.

SQLite je jedna od najrasprostranjenijih SQL baza podataka, a također i baza podataka općenito. Koristi se na mobilnim uređajima te unutar brojnih mobilnih i računalnih aplikacija.


**Slika 4.10:** SQLite logo

## 4.4. Electron [2]

Electron je okvir za izradu računalnih aplikacija koristeći web tehnologije poput HTML-a, JavaScript-a i CSS-a. Ovaj okvir omogućuje ugrađivanje Chromiuma i NodeJS-a u binarnu datoteku aplikacije, čime se postiže mogućnost pokretanja istog JavaScript koda na svim većim operativnim sustavima.

Zahvaljujući načinu na koji funkcionira, moguće je razvijati aplikacije u bilo kojem JavaScript okviru te ih potom pokrenuti i pretvoriti u računalne aplikacije koristeći Electron.

Jedan od problema koji se javlja kod Electrona je prevođenje nativnih paketa. Većina paketa može se prevesti korištenjem alata 'Electron rebuild', no određeni dio još uvijek ne može. To je jedan od razloga zašto Electron još nije implementiran u ovaj projekt.

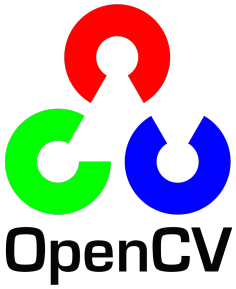
Kao alternative Electronu postoje Tauri, koji je nešto mlađi projekt, te Cordova i Capacitor za izradu mobilnih aplikacija.

**Slika 4.11:** Electron logo [2]

## 4.5. OpenCV [16]

OpenCV je biblioteka otvorenog koda koja sadrži stotine algoritama vezanih za računalni vid. Prva verzija bila je napisana u programskom jeziku C, dok je druga i aktualna verzija napisana u C++. Biblioteka se može koristiti u Pythonu, C++-u, Javi, Androidu, iOS-u te u JavaScriptu. Konkretno za JavaScript, postoji mogućnost izrade iz izvornog koda ili korištenja već izgrađene biblioteke, kao i korištenja koda prevedenog u WASM (Web Assembly) format.

OpenCV pruža mnoge moćne alate za analizu i obradu slika i videozapisa. Omogućuje analizu geometrije iz više točaka (pogleda) te prepoznavanje objekata i barkodova na slikama i u videozapisima.




**Slika 4.12:** OpenCV logo

# 5. Arhitektura sustava

U ovom poglavlju bit će opisana arhitektura sustava te način interakcije različitih dijelova aplikacije. Također, bit će objašnjeni neki algoritmi i važniji dijelovi koda.

Arhitektura cijelog sustava najbolje se može opisati kao troslojna arhitektura. Slojevi arhitekture prate logične cjeline: Frontend predstavlja prezentacijski sloj, Backend sloj poslovne logike, a Baza podataka podatkovni sloj.

Backend se može smatrati slojevitim, ali također ima karakteristike modularne arhitekture. Osim REST sučelja, backend uključuje module za generiranje PDF-a te skeniranje i obradu obrazaca. Dodatno, backend je proširiv putem sustava proširenja (engl. plugins).

## 5.1. Model podataka

Aplikacija upravlja s četiri osnovna tipa podataka i dva izvedena tipa. Glavni tipovi podataka su: Ispit, Predmet, Pitanje i Student. Izvedeni tipovi podataka su Instanca ispita, koja se temelji na vezi s ispitom, te Odgovor na instancu ispita, koji se oslanja na instancu ispita i studenta.

### 5.1.1. Ispit

Ispit je jedan od kompleksniji tipova. Sadrži iduća polja:

**id** - jedinstveni identifikator ispita

**name** - naziv ispita, opcionalni parametar

**subjectIds** - popis predmeta kojima ispit pripada, opcionalni parametar

**questionIds** - popis pitanja u ispitu

**configs** - objekt koji sadrži opcije ispita, trenutno podržava polje **questions** koje izmjenjuje podatke pojedinog pitanja.

**tags** - popis oznaka za ispit

```
export interface Exam {
  id: string;
  name?: string | null;
  subjectIds?: string[] | null;
  questionIds: string[];
  configs: {
    questions?: {
      [key: string]: {
        questionObject: { [key: string]: any };
        answerObject: { [key: string]: any };
      };
    };
    [key: string]: unknown;
  };
  tags: string[];
}
```

**Slika 5.1:** TypeScript definicija tipa ispita

### 5.1.2. Predmet

Predmet služi više kao organizacijski nego kao funkcionalni objekt. Sadrži iduća polja:

**id** - jedinstveni identifikator predmeta

**name** - naziv predmeta

**questionIds** - popis pitanja koja pripadaju predmetu

**tags** - popis oznaka za predmet

```
export interface Subject {
  id: string;
  name: string;
  questionIds: string[];
  tags: string[];
}
```

**Slika 5.2:** TypeScript definicija tipa predmeta

### 5.1.3. Pitanje

Pitanje je jedan od najbitnijih objekata, uz stalna polja sadrži i polja koja služe kako bi se ostvarila proširivost tipova pitanja. Sadrži iduća polja:

**id** - jedinstveni identifikator pitanja

**text** - tekst pitanja

**type** - identifikator tipa pitanja, određuje tip pitanja koji je dinamički učitan

**questionObject** - objekt kojeg određuje tip pitanja, specifičan za svaki tip

**answerObject** - objekt kojeg određuje tip pitanja, specifičan za svaki tip

**tags** - popis oznaka za pitanja

```
export interface Question {
  id: string;
  text: string;
  type: string;
  questionObject: { [key: string]: any };
  answerObject: { [key: string]: any };
  tags: string[];
}
```

**Slika 5.3:** TypeScript definicija tipa pitanja

### 5.1.4. Student

Student je vrlo jednostavan objekt. U budućnosti je vjerojatno proširenje tipa kako bi se omogućilo generiranje obrazaca za nepostojeće tj. lažne osobe. Sadrži iduća polja:

**id** - jedinstveni identifikator studenta

**firstName** - ime studenta

**lastName** - prezime studenta

**studentId** - identifikator studenta bitan za organizaciju, npr. JMBAG, opcionalni parametar

**tags** - popis oznaka za pitanja

```
export interface Student {
  id: string;
  firstName: string;
  lastName: string;
  studentId?: string | null;
  tags: string[];
}
```

**Slika 5.4:** TypeScript definicija tipa studenta

## 5.2. Frontend

Frontend predstavlja prezentacijski sloj aplikacije, odgovoran za prikaz sadržaja i interakciju s korisnicima. Razvijen je korištenjem VueJS-a, točnije Quasar okvira za VueJS, uz TypeScript kao programski jezik, a sastoji se od pet glavnih stranica i jedne stranice za grešku prilikom odlaska na nepostojeću rutu. Sve stranice dijele jedan layout koji se primjenjuje na cijelu aplikaciju, a za navigaciju se koristi Vue Router.

Sadržaj na stranicama uglavnom je definiran komponentama, dok se manji dio koda nalazi direktno u datotekama stranica. Većina funkcionalnosti ostvaruje se putem interaktivnih dijaloga koji se otvaraju pritiskom na gumbe smještene uz određene cjeline ili objekte.

### 5.2.1. Komunikacija s backendom

Za komunikaciju se koriste Axios u kombinaciji s Vue Query (TanStack Query). Adresa backenda se nalazi u datoteci **quasar.config.js** unutar objektu build.env.API. Ta adresa se postavlja kao zadana za Axios klijenta u datoteci **src/boot/axios.ts**. Vue Query koristi zadane postavke.

Adrese i funkcije za pozive definirane su u direktoriju **src/api**, dok se sami pozivi tih funkcija nalaze unutar komponenti.

Vue Query nudi opciju invalidacije podataka putem sustava oznaka. Oznake su većinom definirane na sljedeći način: [tip_objekta, id_objekta]. Tako bi na primjer dohvat pitanja za ispit izgledao: ['questions', examId], a mijenjanje i dohvat pitanja na glavnom izborniku bi imalo oznaku ['questions'].

Još jedan oblik komunikacije s backendom je dohvat izgleda instance ispita. Ispit, odnosno pregled ispisa, prikazuje se s pomoću IFrame elementa. U tom elementu zapravo se prikazuje stranica modula za izradu PDF-a koji se nalazi na backendu. Na sličan način dohvaćaju se i slike ispravljenih obrazaca. To su img elementi s izvorom slike koji pokazuje na backend.

### 5.2.2. Proširivost

Proširivost se odnosi na mogućnost dodavanja novih tipova pitanja. Trenutno proširivost nije implementirana na frontendu, ali postoje preduvjeti za njenu buduću implementaciju. U ovom potpoglavlju razmatraju se teoretski tehnički zahtjevi za takvu implementaciju.

Proširivost se postiže definiranjem tipa pitanja i pripadajućih komponenti. Komponente su definirane putem Web Components API-ja. Takva komponenta mora prihvatiti objekt pitanja i na temelju njega prikazati željeni izgled. Uz to, komponenti se pružaju dva objekta u koja može pohranjivati sve potrebne podatke za generiranje pitanja, izgleda odgovora te izgleda na obrascu za odgovore. Komponenta se dohvaća sa servera putem putanje .../plugins/questions/:tip_komponente/:tip_komponente-component.js te se zatim dinamički učitava u sučelje koristeći funkciju import().

Za frontend je potrebno osigurati više prikaza, koji bi mogli biti prikazani na podstranici za pitanje koja trenutno ne postoji.

Potrebno je osigurati prikaz prilikom izrade pitanja. Taj prikaz trebao bi biti interaktivan i definirati sadržaj objekta questionObject i answerObject. Također, dobro je da definira ispravan odgovor i eventualni sustav bodovanja za taj tip pitanja. Komponenta bi morala objavljivati događaj prilikom izmjene nekog od polja te u tom događaju poslati objekte questionObject i answerObject.

Nadalje, potrebno je prikazati pitanje u popisu. Taj prikaz ilustrira odabrane opcije za pitanje. Na primjeru pitanja s više ponuđenih odgovora, prikaz bi uključivao sve odgovore, pri čemu je točan odgovor označen zelenom bojom, '*' označava odgovor koji se uvijek prikazuje, a bez obzira na odabrani broj ponuđenih odgovora, prikazuju se svi. Također, može se osigurati prikaz razlike u samom tipu pitanja, npr. lista odgovora ima oznake A, B, C... ako su ponuđeni odgovori uvijek u tom redoslijedu, a točku ako su u nasumičnom poretku.

Uz navedene prikaze, potrebno je osigurati prikaz u samom ispitu te prikaz na obrascu za odgovore. Budući da su ta dva prikaza relevantnija za backend, bit će opisani u tom poglavlju.

## 5.3. Backend

Backend je razvijen koristeći Fastify okvir za NodeJS, uz TypeScript kao programski jezik.

Backend se može podijeliti na jezgreni dio, koji sadrži Fastify poslužitelj, i module. Moduli su razvijani kao zasebni projekti, a kompajlirani kod je potom kopiran u odgovarajući direktorij unutar backend projekta.

Korišten je SSL, a certifikati se nalaze u direktoriju cert. Prilikom stavljanja aplikacije u produkciju, ove datoteke potrebno je zamijeniti. Trenutne datoteke su privremene i služe za olakšavanje prvog pokretanja aplikacije. Uz to, privatni ključ je distribuiran uz certifikat, što ga čini nesigurnim.

### 5.3.1. Komunikacija s frontendom

Poslužitelj nema direktnu komunikaciju s frontendom. On pruža REST API koji može konzumirati frontend aplikacija, ali i bilo koji drugi sustav.

API je dostupan na putanji /api, a prilikom pristupa toj adresi prikazuje se dokumentacija API-ja u skladu s OpenAPI standardom.

Osnovni objekti dostupni su putem sljedećih putanja: /api/subjects, /api/exams, /api/questions i /api/students. Na tim se putanjama mogu izvršavati CRUD operacije (kreiranje, čitanje, ažuriranje i brisanje) koristeći HTTP metode: GET, POST, PUT, PATCH i DELETE.

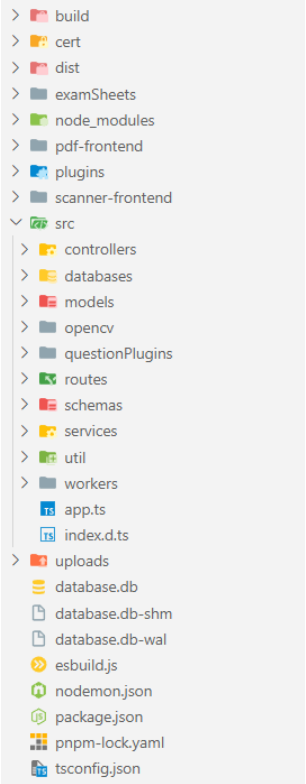
Objektima drugog reda i listama uglavnom se može pristupiti putem očekivanih putanja koje slijede identifikator objekta prvog reda. Tako se oznakama pristupa putem adrese /api/:objekt/:id/tags, a instancama ispita putem /api/exam/:id/instances.

Osim REST API-ja, sustav omogućuje pristup dodatnim modulima i adresama za prijenos slika obrazaca.

Modul za generiranje PDF dokumenata dostupan je na adresi /template. S obzirom na to da je implementiran kao web-aplikacija koja koristi Vue.js, puna putanja izgleda

ovako: /template/index.html#/.... Završni dio adrese odnosi se na navigaciju unutar Vue.js aplikacije koristeći Vue Router. Modul za skeniranje nalazi se na putanji /scanner i implementiran je kroz dvije (+jedna) osnovne HTML stranice. Podržane putanje su /scanner/empty.html za obradu pitanja, /scanner/analyser.html za analizu ispita te /scanner/index.html. Posljednja stranica namijenjena je skeniranju dokumenata uživo putem mobitela ili kamera, no trenutno nije funkcionalna i ne podržava izravno slanje slika na poslužitelj.

Zadnje pristupne točke uključuju: /upload/exams/responses za prijenos ispunjenih obrazaca, /images za preuzimanje slika ispravljenih obrazaca te /plugins za dohvat proširenja.




**Slika 5.5:** Organizacija backenda

### 5.3.2. Model podataka

Fastify poslužitelj strogo kontrolira primanje i slanje podataka. Uz pomoć Zod validatora, svaki zahtjev se temeljito provjerava, čime se osigurava da je nemoguće poslati objekt neispravnog oblika. Definicije validatora nalaze se u datoteci /src/models i ekvivalentne su modelima navedenim ranije. Iz validatora se također povlači definicija

tipa za TypeScript.

Validator pretpostavlja da će identifikatori objekata biti tipa cuid2. U slučaju promjene ili korištenja drugog izvora podataka, može doći do problema ako se taj tip ne promijeni u samom modelu objekta.

### 5.3.3. REST API

Početna točka za analizu REST API-a u kodu je direktorij /src/routes. U njemu se nalazi datoteka i direktorij api. U datoteci je definirano ponašanje u slučaju pogreške te su učitane daljnje rute koje se nalaze u api direktoriju.

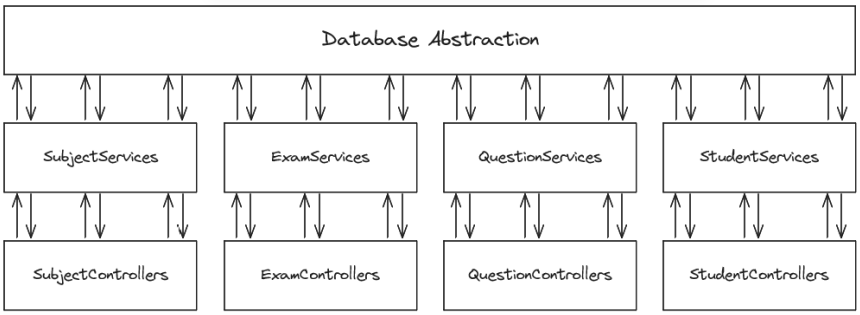
Registracija putanja organizirana je prema glavnim funkcionalnim cjelinama aplikacije, odnosno objektima. Primjerice, jedna datoteka sadrži putanje vezane uz pitanja, dok je u drugoj definirana logika za ispite.

Sheme za putanje su definirane u direktoriju /src/schemas i također su strukturirane prema objektima. One služe kao temelj za izradu OpenAPI dokumentacije te za validaciju dolaznih zahtjeva i odgovora. Ovaj pristup osigurava da poslužitelj uvijek šalje ispravne objekte, dok se neočekivani podaci automatski odbacuju. Uz definiciju putanja i objekata, svaka shema sadrži i kratki opis.

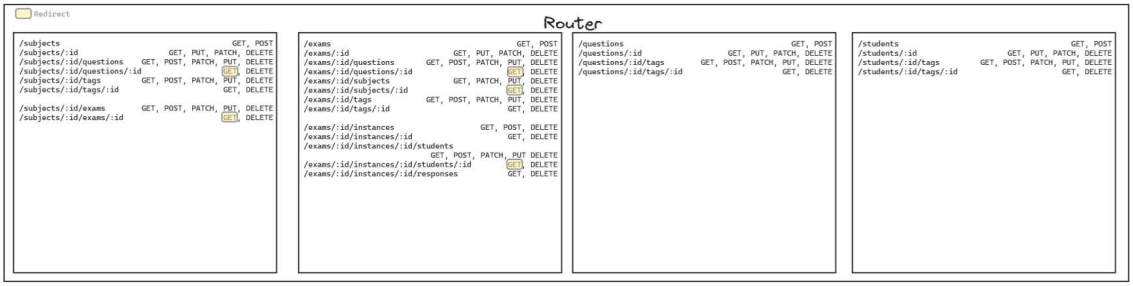
Upravljači (engl. controllers) se nalaze u direktoriju /src/controllers i zaduženi su za prihvat i slanje objekata. Oni predstavljaju ulaznu točku za obradu zahtjeva te izlaznu točku za povrat odgovora. Struktura koda u upravljačima odgovara strukturi definicija shema. Bitno je napomenuti da se upravljači ne bave poslovnom logikom, već taj zadatak prepuštaju uslugama (engl. services).

Usluge su smještene u direktoriju /src/services i organizirane su prema objektima. One implementiraju poslovnu logiku aplikacije te komuniciraju s bazom podataka putem poznatog sučelja.

Važno je napomenuti da se rute, sheme i upravljači preslikavaju jedan na jedan te se ne koriste izvan API sučelja. S druge strane, usluge su fleksibilnije i mogu se pozivati iz bilo kojeg dijela aplikacije, bez potrebe da budu vezane za specifične putanje.




**Slika 5.6:** Prikaz iz planiranja REST API-a - arhitektura



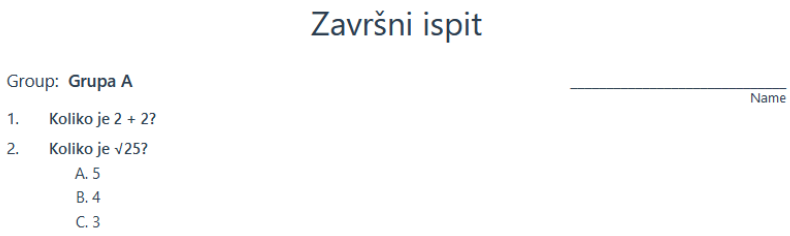

**Slika 5.7:** Prikaz iz planiranja REST API-a - putanje

### 5.3.4. Modul za generiranje PDF-a

Za generiranje PDF dokumenata koriste se alati Puppeteer i Vue.js stranica. Stranica je razvijena kao zaseban modul, a nakon prevođenja njezin se kod smješta u direktorij /pdf-frontend.

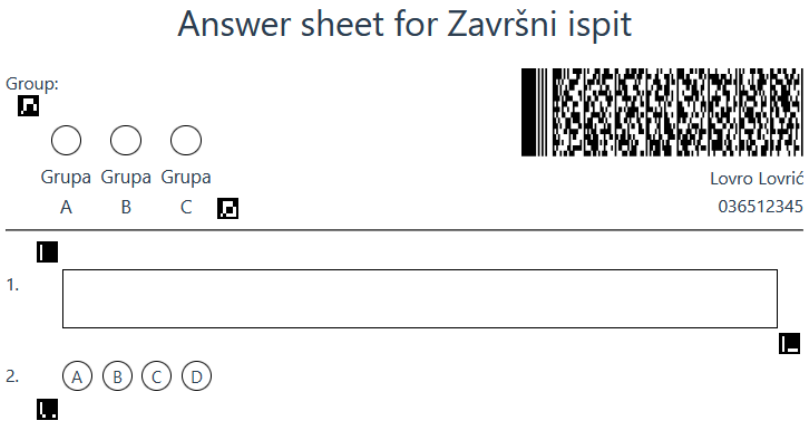
Web stranica podržava tri tipa putanja:

Putanja '/exams/:examId/instances/:instanceId/groups/:groupName' služi za prikaza ispita, odnosno instance ispita. Stranica prikazuje osnovne informacije kao što su naslov ispita, grupa te prostor za upis imena studenta. Također, dinamički dohvaća sva pitanja i njihov izgled za određenu instancu ispita putem proširenja. Svakom pitanju prosljeđuje se objekt 'question', koji uključuje objekte questionObject i answerObject, prilagođene specifičnoj vrsti pitanja. Uz to, instanci ispita dodaje se nasumično generiran broj, modificiran ovisno o nazivu grupe. Taj broj omogućava varijacije među grupama, primjerice, u redoslijedu ponuđenih odgovora. Algoritam za generiranje nasumičnog broja jednostavno pretvara naziv grupe u numeričku vrijednost i zbraja je s originalnim brojem.




**Slika 5.8:** Prikaz stranice za generiranje ispita

Na putanji ’/exams/:examId/instances/:instanceId/sheets/:studentId’ se omogućava dohvat obrasca za određenog studenta. Obrazac sadrži naslov, odabir grupe, PDF417 barkod, kao i ime te organizacijski identifikacijski broj studenta. Ispod ovih podataka nalazi se niz pitanja, koja trenutno zauzimaju cijelu širinu stranice, bez mogućnosti raspodjele na više stranica. Izgled područja za odgovore, kao i kod pitanja, definiran je putem web komponenti koje se dohvaćaju preko backend proširenja. Obrazac sadrži i ArUco oznake koje služe za prepoznavanje područja pitanja, što omogućuje prisustvo različitih tipova pitanja na istom obrascu. Oznake su raspoređene naizmjenično lijevo-desno, pri čemu prva oznaka nosi identifikator 0, dok oznake za grupu nose identifikatore 1000 i 1001. Barkod sadrži JSON objekt u tekstualnom obliku, a uključuje identitet studenta, ispita i instance ispita.
Za generiranje ArUco oznaka korištena je biblioteka ’js-aruco-2’ [4], dok je za generiranje barkoda biblioteka ’bwip-is’ [14].




**Slika 5.9:** Prikaz stranice za generiranje obrasca

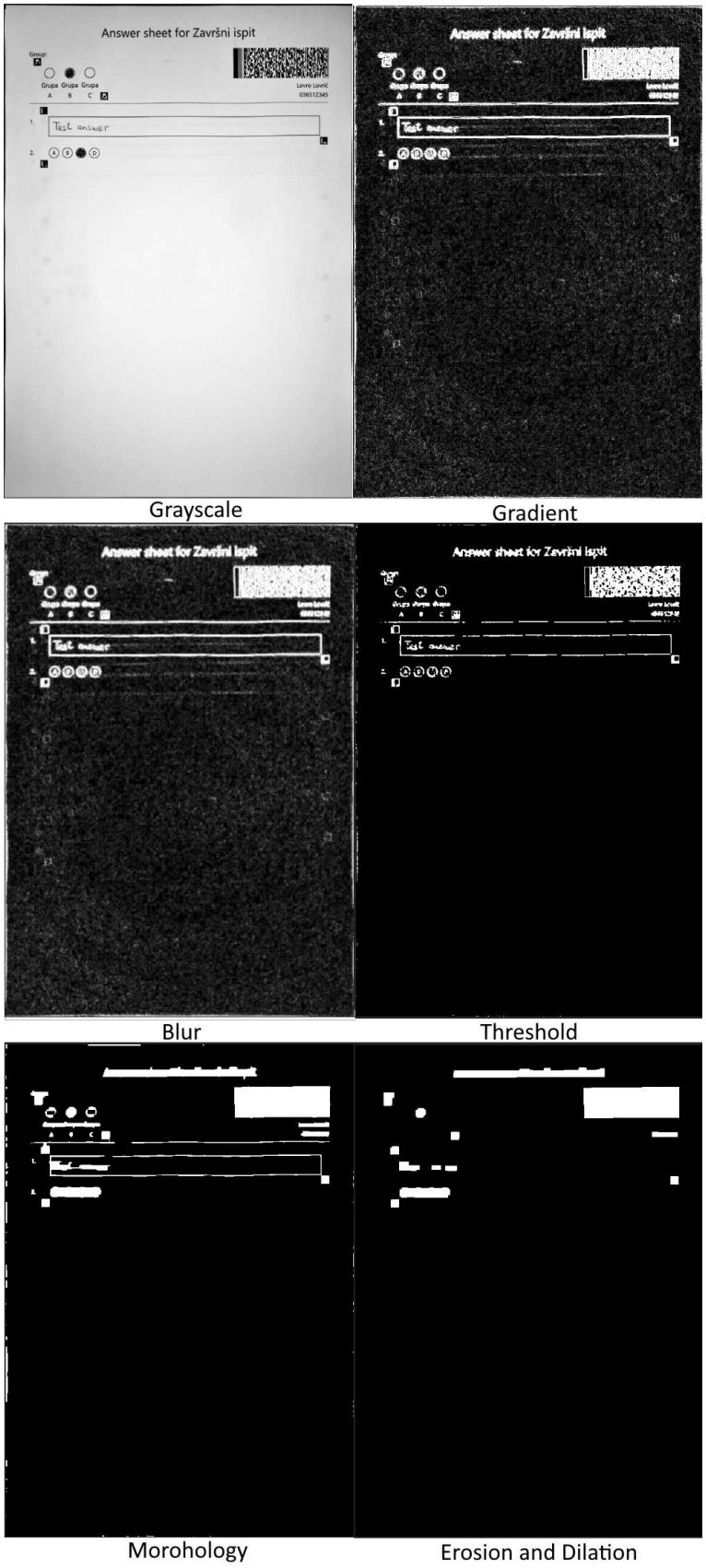
Zadnja dostupna putanja je '/exams/:examId/instances/:instanceId/sheets'. Na toj putanji se povezuju obrasci za sve studente dodane u instancu ispita. Prilikom prikaza na webu, obrasci izgledaju zgusnuto, no prilikom generiranja PDF-a, zahvaljujući CSS svojstvu break-after: page na div elementu nakon svake instance obrasca, oni će se pravilno rasporediti na zasebne stranice.

Backend modul, osim pristupa ispitima i obrascima u web formatu, nudi i mogućnost preuzimanja PDF dokumenata izravno. Putanje putem kojih je to moguće su: .../instances/:instanceId/students/:studentId.pdf, .../instances/:instanceId/students.pdf i .../instances/:instanceId-:group. Ove putanje koriste Puppeteer u pozadini kako bi pretvorile web stranicu u PDF datoteku. Prilikom kreiranja PDF-a, Puppeteer je konfiguriran da generira A4 stranice s marginom od 1 cm sa svih strana.

### 5.3.5. Modul za analizu obrazaca

Modulu za analizu obrasca se nalazi na putanji /scanner/... Taj modul omogućava analizu obrasca te razdvajanje i ispravljanje pojedinih pitanja. Postoji i putanja do mobilnog skenera, ali on nije u potpunosti implementiran.

Na putanji /scanner/analyser.html se nalazi alat koji analizira obrazac. On koristi OpenCV za analizu slike. Prvi korak u analizi je otkrivanje barkoda.




**Slika 5.10:** Prikaz koraka algoritma za otkrivane pozicije barkoda

Algoritam za otkrivanje barkoda ima dva dijela: otkrivanje pozicije i skeniranje. Kako bi se otkrila pozicija barkoda primjenjuje se idući algoritam: [15]:

- Slika se smanji na 70% originalne veličine
- Pretvori sliku u prikaz sivim tonovima
- Na sliku se primjeni gradient
- Doda se zamućenje slike
- Slika se pretvori u crno bijelu s određenom početnom i završnom vrijednosti sive boje
- Primjeni se određeni algoritam morfologije
- Primjene se erozija i dilatacija
- Pronađu se konture
- Unutar pozicije kontura na unaprijeđenoj slici se provjerava da li postoji barkod

Za poboljšanje slike koje olakšava pronalazak barkoda primjenjuje se algoritam:

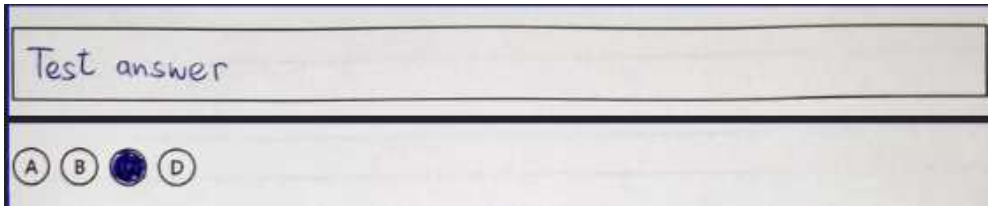
- Pretvori sliku u prikaz sivim tonovima
- Uvećaj sliku 2 puta kako bi barkod bio veći
- Izoštri sliku
- Pretvori sliku u crno bijelu

Sadržaj barkoda se postavlja u HTML element s identifikatorom 'barcodeData', dok se slika barkoda nalazi u Canvas elementu s identifikatorom 'barcodeImg'.

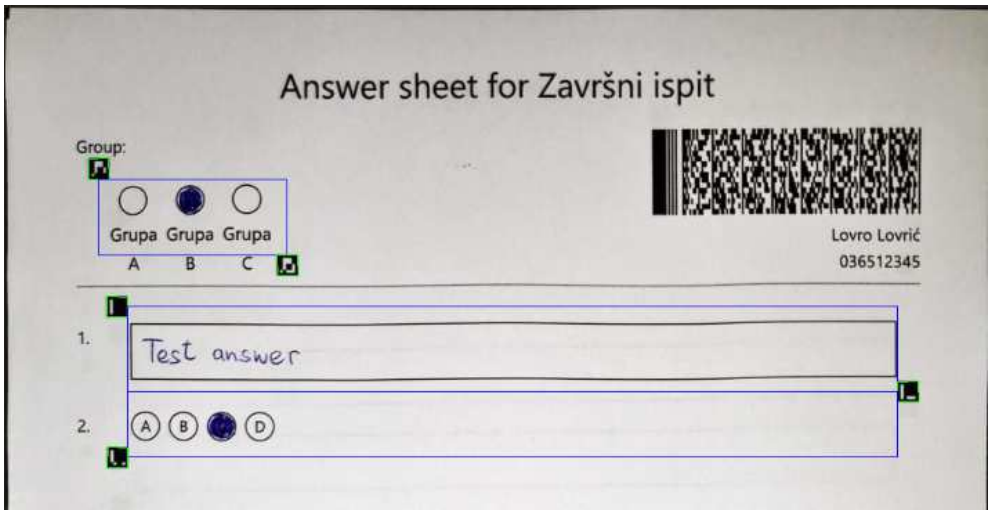
Nakon otkrivanja barkoda, pokreće se algoritam detekcije ArUco oznaka. Za poboljšanje slike se opet koristi OpenCV dok se za prepoznavanje ArUco oznaka koristi js-aruco-2 [4] paket. Algoritam se sastoji od idućih koraka:

- Izoštri sliku
- Uvećaj sliku 2 puta
- Pretvori sliku u crno bijelu
- Koristeći js-aruco-2 pronađi sve oznake
- Sortiraj oznake po identifikatoru i ukloni duplikate
- Za svaku oznaku, ako postoji iduća i ima id za jedan veći, odredi centar (lijevi ili desni)
- Kopiraj sadržaj od prve točke do druge točke u zasebnu sliku

– Ako oznaka ima identifikator 1000, pokreni prepoznavanje grupe




**Slika 5.11:** Prikaz otkrivenih pitanja




**Slika 5.12:** Prikaz ArUco oznaka i zona interesa

Nakon ovog koraka, sustav dodaje slike pojedinog pitanja i to pod identifikatorom 'questionNNImg' gdje je NN redni broj pitanja. Taj se broj također određuje s pomoću ArUco oznaka, gornja oznaka je id-1, a donja (druga) je upravo redni broj pitanja.

Zadnji korak kojega vrši analiza obrasca je prepoznavanje grupe. Taj algoritam je isti kao i algoritam prepoznavanja odgovora kod ABC tipa pitanja te će biti opisan samo tu [18]:

– Pretvori sliku u tonove sive boje

– Pretvori sliku u crno bijelu

– Pronađi konture

– Ako je kontura veća ili jednaka od 20 u visinu i širinu te ako je omjer visine i širine oko 1 dodaj konturu u listu

– Sortiraj listu dobivenih kontura od lijevo prema desno

- Koristeći konturu stavi masku nad crno-bijelu sliku iznad pojedinog odgovora
- Prebroji broj bijelih pixela, ako je više od pola bijelo, označi odgovor kao označen

Rezultat ove detekcije se zapisuje u HTML element pod identifikatorom 'groupSubtitle' i predstavlja redni broj grupe (index) koji se kasnije može pretvoriti u njen naziv.

Grafike prikazane u ovom dijelu se mogu dobiti postavljanjem opcije 'showWork' u modulu za analizu te ponovnim prevođenjem koda.

Druga aktivna putanja je scanner/empty.html, koja služi za analizu pojedinog pitanja. Otvaranjem te stranice prikazuje se prazna HTML stranica koja odmah započinje učitavanje OpenCV biblioteke. Nakon što se biblioteka učita, ona se javno izlaže na window objektu i postavlja se HTML element s oznakom done. Kada se pojavi ta oznaka, backend dodaje skriptu za analizu pitanja.

Ta skripta je još jedan dio koji proširivo pitanje mora implementirati. Skripta, koja se dinamički učitava, izlaže na window objektu funkciju analyzeImage koja prima sliku u formatu data URL-a, objekt pitanja i sjeme za generiranje pitanja. Očekuje se da funkcija vrati objekt koji sadrži informacije o ispravljenom pitanju. Tijekom izvođenja, funkcija može pristupiti OpenCV biblioteci pozivom javne cv varijable.
Trenutno postoje dva tipa pitanja: tekstualno, koje vraća prazan objekt, i 'ABC', koje provodi analizu sličnu onoj za otkrivanje grupe.

Za otvaranje stranice i dodavanje koda zadužen je Puppeteer.

### 5.3.6. Proširivost

Backend sustav u potpunosti podržava proširivost, kako za nove tipove pitanja, tako i za različita spremišta podataka.

Prilikom izrade novog tipa pitanja, za backend, potrebno je definirati web komponentu koja će prikazivati izgled pitanja u tiskanom obliku i na obrascu. Osim toga, važno je osigurati i funkcionalnost za automatsko ispravljanje pitanja.

Pri korištenju web komponenti, komponenta prima objekt question, koji predstavlja pitanje, putem svojstva istog naziva. Objekt se prenosi u tekstualnom formatu te ga

je potrebno parsirati s pomoću JSON.parse() metode. Uz objekt pitanja, prosljeđuje se i broj sjemena (seed) ispita, koji omogućava nasumične varijacije između različitih grupa i instanci ispita.

Očekuje se da definirane komponente koriste HTML oznake u formatu: question-:type:-sheet-module za prikaz na obrascu te question-:type:-display za prikaz na ispitu. Ove komponente mogu biti napisane koristeći bilo koji frontend okvir, pod uvjetom da ih je moguće učitati iz .js datoteke. Na primjer, komponente za tekstualna i ABC pitanja implementirane su koristeći Svelte okvir.

Drugi ključni dio koji je potrebno implementirati je ispravljanje ispita. Za to je potrebno definirati funkciju analyzeImage() na window objektu. Ova funkcija koristi globalnu varijablu cv, koja pruža pristup OpenCV biblioteci. Kao argumenti funkcija prima sliku u obliku data URL-a, objekt question i broj sjemena (seed). Kao izlaz se očekuje objekt koji sadrži informacije o odgovoru na pitanje, a taj se objekt zatim pohranjuje i kasnije poslužuje putem API putanja .../answers. Funkcija mora biti smještena u datoteci imageAnalyser.js unutar direktorija za proširenje. Iako je moguće koristiti dodatne biblioteke, one moraju biti kompatibilne s pregledničkim okruženjem i integrirane u jednu datoteku.

Više informacija o proširivosti vezanoj uz spremišta podataka bit će obrađeno u poglavlju "Baza podataka".

## 5.4. Baza podataka

Baza podataka u ovoj aplikaciji podrazumijeva bilo koji sustav ili pristup sustavu koji implementira funkcije propisane s apstraktnom klasom definiranom u datoteci IDatabase.ts. Zadani i pokazni primjer jedne takve baze je SQLite implementacija koja je napisana za potrebe ovog rada.

### 5.4.1. IDatabase

Datoteka se nalazi u sklopu backenda, a kopija je dostupna i u modulu baze podataka. Na vrhu datoteke definirani su tipovi podataka, dok se ispod njih nalazi definicija apstraktne klase koju je potrebno implementirati. Na dnu se nalazi mapa uzroka grešaka koje se mogu baciti putem opcije cause objekta Error, te će biti ispravno proslijeđene

krajnjem korisniku preko backenda.

Apstraktna klasa sastoji se od niza funkcija čija imena započinju očekivanom radnjom, a završavaju objektom nad kojim se radnja izvršava. Argumenti funkcija su uglavnom intuitivni. Argument replace odnosi se na objekte i liste unutar krovnog objekta. Očekivano ponašanje je da se bez oznake replace objekti samo nadopunjuju, dok se s oznakom u potpunosti zamjenjuju.

Sve što nije specificirano u datoteci IDatabase.ts ostaje na izbor implementatoru. U klasu se mogu dodati druge funkcije, interna pohrana može sadržavati objekte koji izgledaju drugačije, ali prilikom poziva vraća objekt u skladu s očekivanjima.

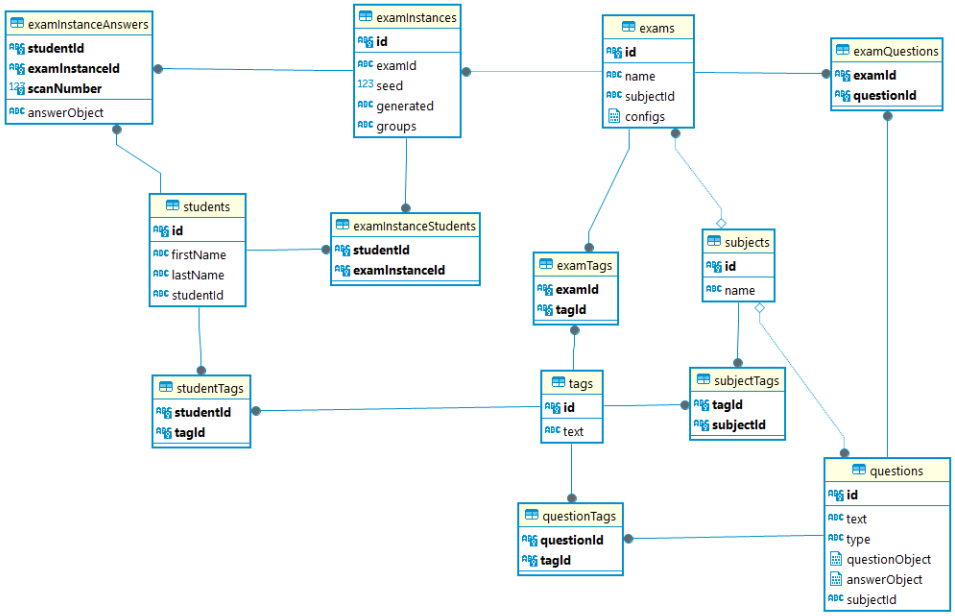
Još jedna bitna stvar je da, ako je potrebno, konekcija sa spremištem bude ostvarena u konstruktoru te pohranjena unutar objekta. U backendu će se koristiti jedna instanca objekta koja će biti inicijalizirana pri pokretanju backenda, a sva ostala komunikacija će se odvijati putem poznatih funkcija.

### 5.4.2. SQLite implementacija

SQLite je odabran kao zadana baza podataka jer ne zahtijeva dodatnu infrastrukturu. Prilikom implementacije korišten je paket better-sqlite3.

Better-sqlite3 je sinkrona implementacija SQL baze podataka, stoga svaka funkcija ima sinkronog dvojnika s nazivom koji završava na ...Sync. Unutar zadanih funkcija, sinkroni dvojnici su omotani u obećanje (Promise). Osim traženih funkcija, implementirane su i pomoćne funkcije za povezivanje i razdvajanje ispita i pitanja te ispita i predmeta.

Jedna od odluka koja utječe na funkcioniranje krajnjeg proizvoda je pohranjivanje samo jednog predmeta uz ispit. U praksi to znači da se uzima u obzir samo prvi predmet iz liste. Iako ovo nije nužno očekivano ponašanje, pokazuje kako baza podataka, odnosno spremište, može utjecati na konačni proizvod.




**Slika 5.13:** Prikaz veza unutar SQLite baze podataka

Prikazana baza je samo ogledna i može služiti kao inspiracija, ali je ujedno i funkcionalna baza podataka.

### 5.4.3. Alternativne implementacije

Ovaj pristup pohrani podataka je izabran kako bi omogućio maksimalnu fleksibilnost. Tako bi na primjer umjesto u bazi podataka podaci mogli biti spremljeni u datotečnom sustavu. To naravno ne bi bio optimalan pristup, ali je svakako moguć.
Možda realnija i vjerojatnija primjena bi bila spajanje dijela funkcija na vanjsku uslugu. Ako ustanova već posjeduje bazu ili server kojemu može pristupiti preko API sučelja, podaci o studentima bi mogli dolaziti od tamo te bi ih implementirana funkcija samo prilagodila u očekivani izgled. Ako takav servis ne podržava stvaranje novih studenata onda bi pri pokušaju stvaranja implementacija mogla bacati notImplementedError (501) grešku i ignorirati takve zahtjeve.
Iz prošlog primjera je vidljivo da podaci mogu dolaziti iz različitih izvora čak i drugih API sučelja sve dok se poštuje izgled definiran sučeljem IDatabase.

# 6. Opis aplikacije

Aplikacija je ostvarena kroz tri dijela: sučelje (frontend), poslužitelj (backend) i bazu podataka.

Backend predstavlja središnju i ključnu komponentu koja povezuje bazu podataka s frontendom. To se postiže korištenjem REST API-ja i sustava proširenja. Sustav je dizajniran tako da bude fleksibilan, omogućujući jednostavnu nadogradnju ili zamjenu pojedinih dijelova. Dodatna prednost je modularnost sustava, koji se sastoji od više modula i podržava proširivost tipova pitanja.

Kako bi korisničko iskustvo bilo što ugodnije, sustav nudi napredne mogućnosti označavanja i pretraživanja podataka (ispita, pitanja, predmeta itd.). Ovaj pristup omogućava maksimalnu fleksibilnost i prilagodbu povezivanja podataka prema potrebama korisnika.

## 6.1. Frontend

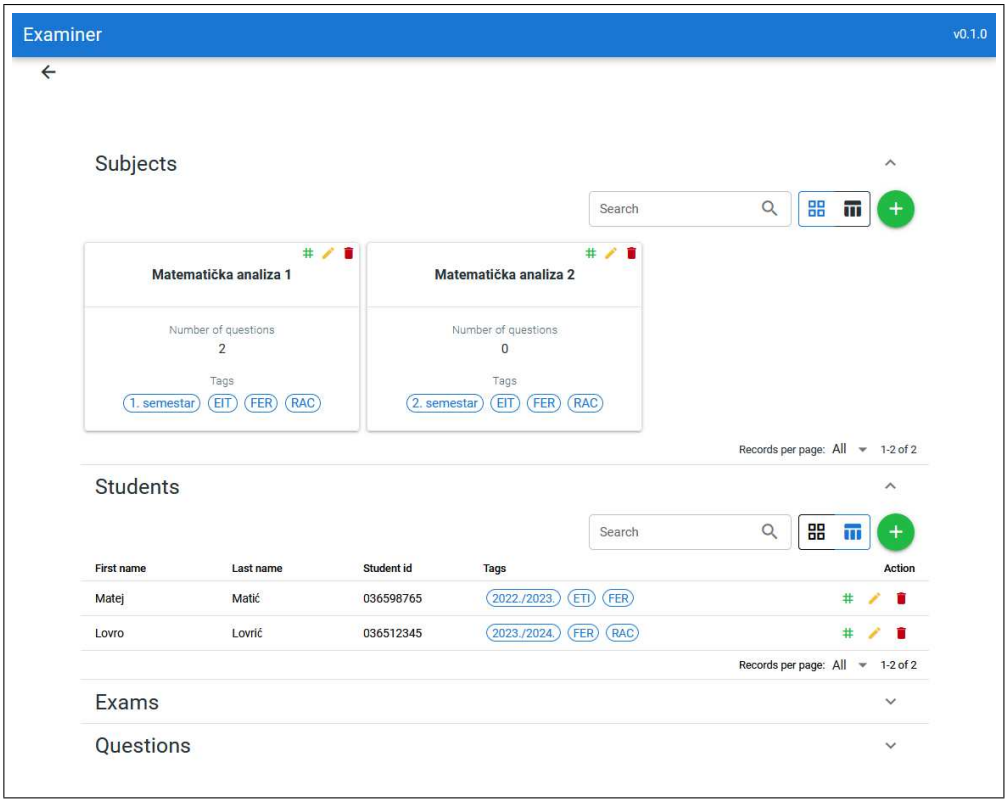
Frontend je sučelje koje organizira i pojednostavljuje pozive prema backendu. On ne sadržava dodatne informacije već pokušava na smisleni način organizirati resurse koji se dohvaćaju Restful pozivima prema backendu.

### 6.1.1. Stranice

Aplikacija se sastoji od glavnog izbornika na kojem su prikazani popisi predmeta, studenata, ispita i pitanja, s osnovnim informacijama o svakom od njih. Svaki popis može biti prikazan u tabličnom obliku ili kao niz kartica, pri čemu oba prikaza pružaju ekvivalentne informacije. U zaglavlju prikaza nalazi se polje za pretraživanje i gumb za

dodavanje novog objekta.

Uz podatke o pojedinom objektu nalaze se gumbi za uređivanje, izmjenu oznaka i brisanje objekta. Pritiskom na redak ili karticu otvara se stranica s detaljima o objektu. Trenutno je ta funkcionalnost implementirana za ispite i predmete, dok za studente i pitanja specifični detalji još nisu dostupni.

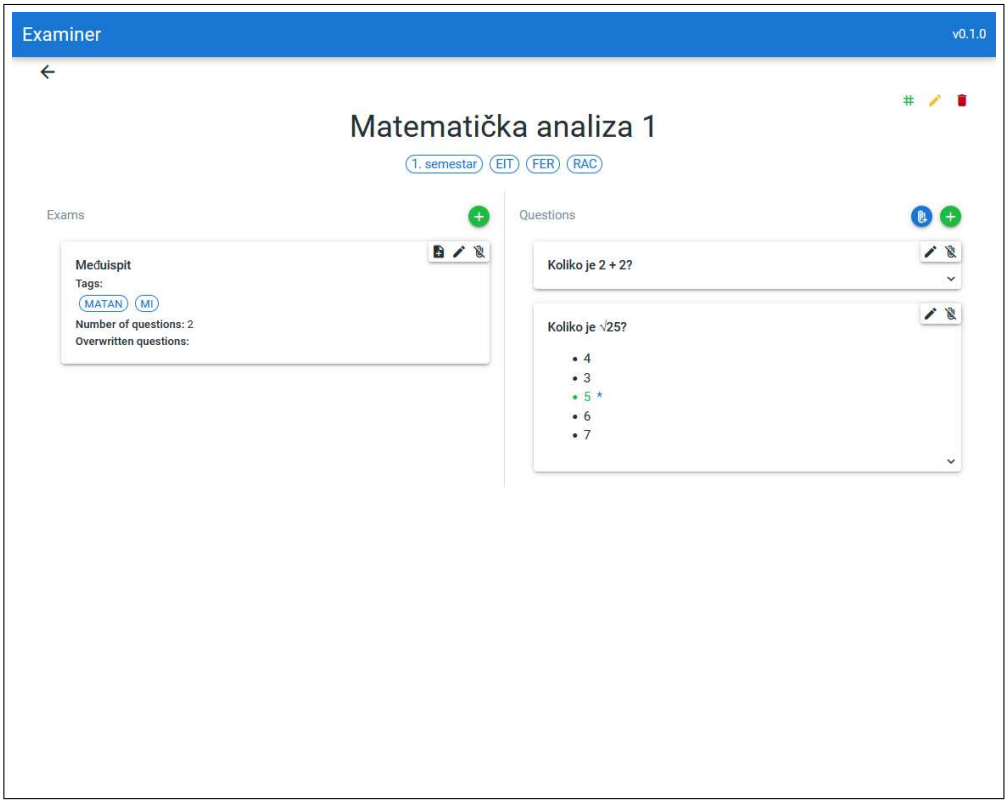



**Slika 6.1:** Prikaz početnog ekrana

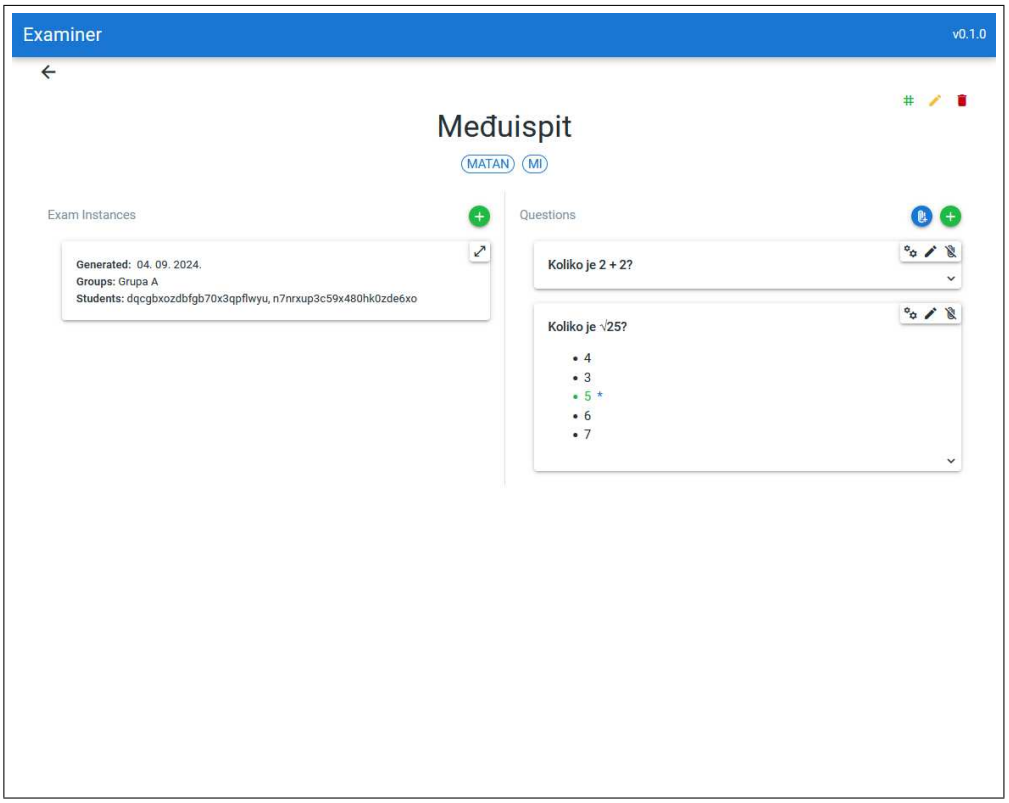
Stranica za prikaz detalja sastoji se od naslova u zaglavlju, koji prikazuje ime ispita ili predmeta, odnosno tekst formiran iz identifikatora objekta. Ispod naslova nalazi se popis oznaka, dok se u gornjem desnom kutu nalaze akcijski gumbi. Ovi gumbi imaju istu funkcionalnost kao i gumbi unutar tablice ili kartice na početnom zaslonu.

Tijelo stranice podijeljeno je na dva dijela. U slučaju predmeta, to su popis ispita i popis pitanja, dok su u slučaju ispita to popis instanci ispita i popis pitanja. Ovi dijelovi su prilagodljive veličine. Na vrhu svakog dijela nalaze se akcijski gumbi koji se odnose na popis tog dijela. Dostupne akcije uključuju dodavanje novog objekta i upravljanje postojećim vezama.

Svaki popis sadrži listu kartica koje prikazuju osnovne informacije o objektu. Za prikaz detalja postoji gumb u donjem desnom kutu kartice. U gornjem desnom kutu nalaze se akcijski gumbi za pojedinu karticu, a njihove funkcionalnosti ovise o tipu objekta. Mogu uključivati izmjenu objekta, uklanjanje veze s objektom, stvaranje nove instance (u slučaju ispita) i otvaranje prikaza detalja.




**Slika 6.2:** Prikaz ekrana detalja o predmetu

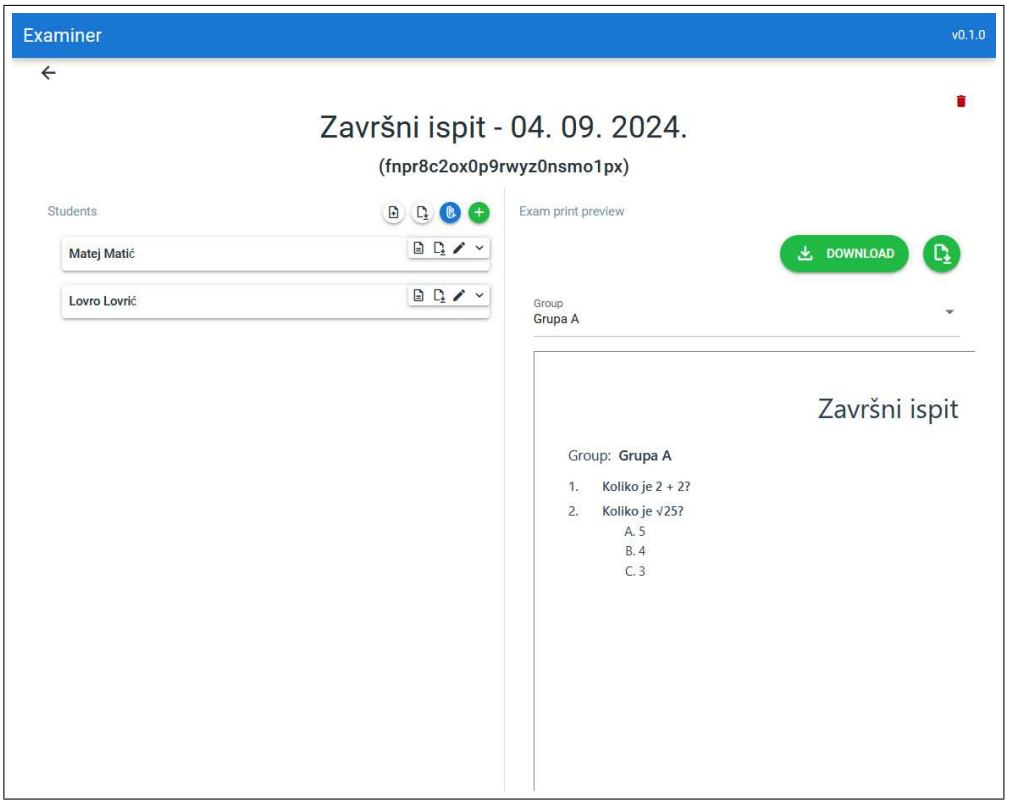



**Slika 6.3:** Prikaz ekrana detalja o ispitu

Uz navedene stranice, postoji i podstranica za prikaz instance ispita. Na njoj se u lijevom dijelu nalazi popis studenata pridijeljenih navedenoj instanci, dok se u desnom dijelu nalazi prikaz ispita za ispis. U blizini prikaza postoji mogućnost odabira grupe te akcije za ispis (preuzimanje PDF dokumenta za ispis) odabrane grupe i svih grupa u instanci ispita.

Popis studenata sadrži akcije za preuzimanje svih obrazaca za studente i prijenos ispunjenih obrazaca. Pojedini student nudi opciju preuzimanja njegovog obrasca i pregleda učitanih rješenja.

Akcije za prijenos ispunjenih obrazaca omogućuju predaju obrasca iste osobe više puta kako bi se popunila polja koja eventualno nisu prepoznata.




**Slika 6.4:** Prikaz ekrana detalja o instanci ispitu

Zadnja stranica je stranica za prikaz rezultata, kojoj se pristupa pritiskom gumba akcije kod pojedinog studenta.

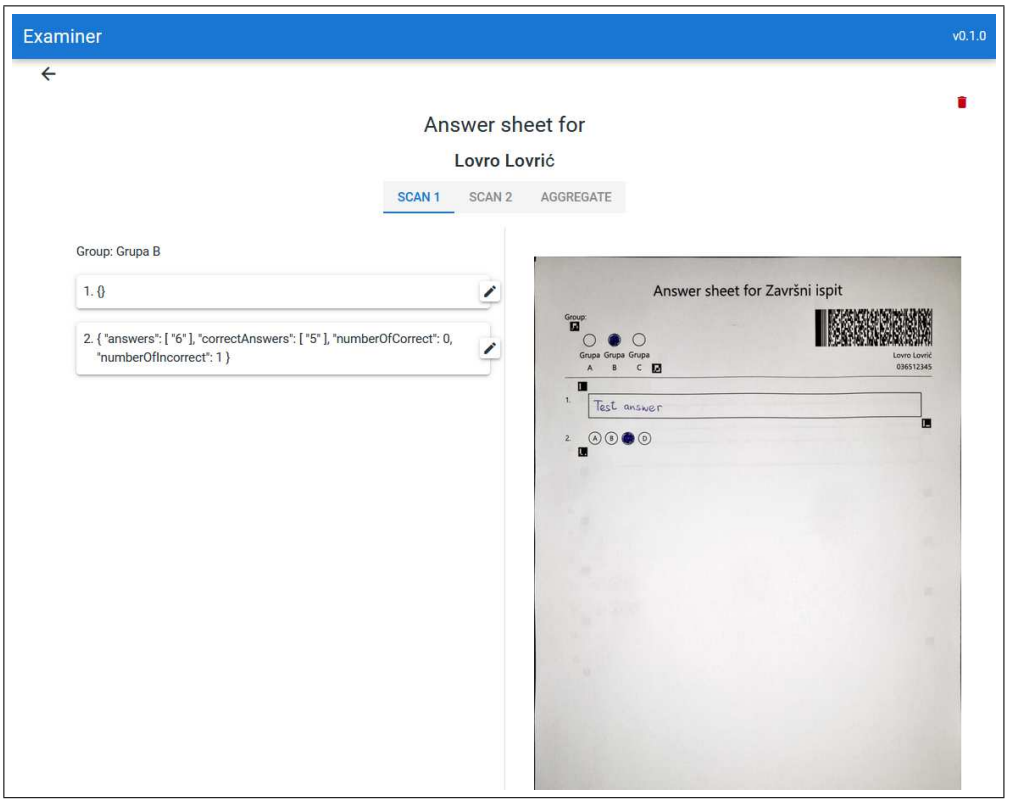
Na toj stranici prikazana je skenirana (prenesena) slika obrasca s desne strane te rezultati ispravljanja s lijeve strane. Iznad tih cjelina postoji mogućnost odabira skeniranog dokumenta te, u slučaju više dokumenata, odabira agregata. Agregat objedinjuje rezultate svih skeniranja i koristan je u slučaju neuspješnog prepoznavanja pojedinih pitanja.

Ispod slike obrasca nalazi se prikaz pregleda ispisa ispita uz mogućnost odabira grupe. Izgledom je isti kao i pregled na stranici instance ispita, a služi kao podsjetnik na zadana pitanja i ponuđene odgovore u ispitu.

Uz rezultate se nalazi gumb koji pokreće akciju uređivanja pojedinog odgovora. Ova funkcionalnost omogućava dodavanje rezultata pitanja koja nije moguće skenirati ili izmjenu u slučaju primijećene pogreške.

U gornjem desnom kutu nalazi se gumb koji pokreće brisanje odabranog skupa od-

govora.

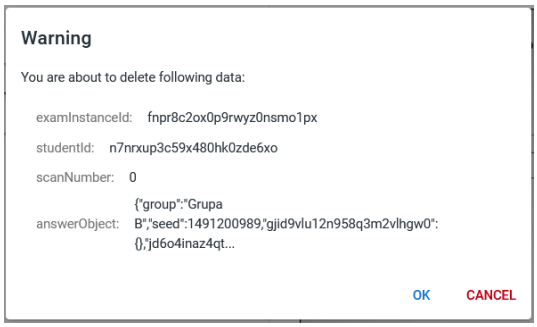



**Slika 6.5:** Prikaz ekrana rezultata studenta na ispitu

### 6.1.2. Dijalozi

Sva dodavanja, izmjene i brisanja podataka odvijaju se pritiskom na gumb akcije. Takve radnje otvaraju dijalog koji je potrebno popuniti i potvrditi.

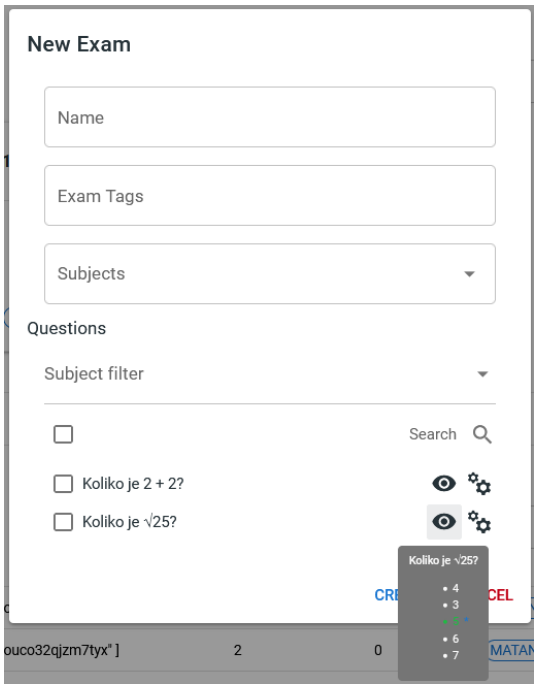
Brisanje pojedinog objekta uvijek otvara dijalog upozorenja. U tom dijalogu prikazuju se sva polja objekta označenog za brisanje te se pruža mogućnost odustajanja od navedene radnje.




**Slika 6.6:** Prikaz dijaloga za brisanje

Prilikom stvaranja objekata potrebno je popuniti ponuđena polja u otvorenom dijalogu. Polja se razlikuju ovisno o vrsti objekta. Jednostavniji primjeri su stvaranje studenta i predmeta, koji zahtijevaju popunjavanje tekstualnih polja i eventualno dodavanje oznaka objektu. S druge strane, stvaranje pitanja, ispita ili instance ispita ima nešto složenije sučelje.

Prilikom stvaranja ispita nudi se mogućnost odabira predmeta kojima ispit pripada te pitanja koja se nalaze u ispitu. Ponuđena pitanja moguće je filtrirati prema predmetu uz koji se vežu, tekstu samog pitanja ili oznakama koje se nalaze uz pitanje. Uz svako pitanje postoje dva gumba: pregled pitanja, koji prelaskom miša prikazuje izgled pitanja, i konfigurator. Konfigurator omogućuje prilagodbu postavki pitanja za pojedini ispit.

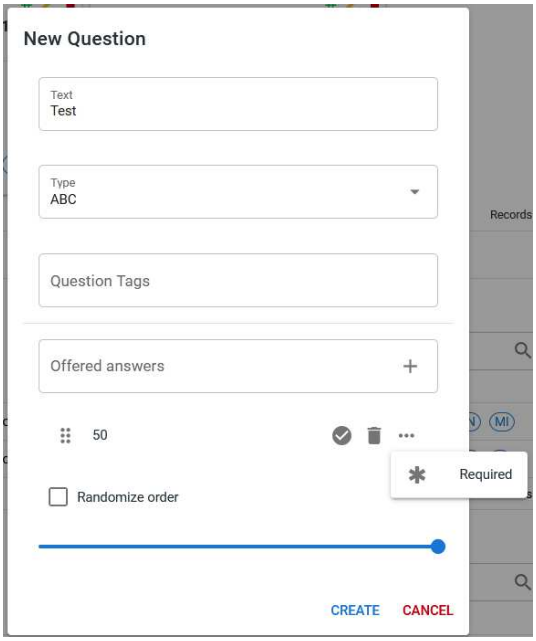



**Slika 6.7:** Prikaz dijaloga za stvaranje novog ispita

Stvaranje pitanja nudi osnovna polja (iznad crte) koja uključuju tekst pitanja, tip pitanja i oznake. Prilikom odabira tipa, ispod crte pojavljuju se dodatne opcije specifične za taj tip pitanja. Trenutno postoje dva tipa: 'ABC' i 'Text'. Njihove konfiguracije su trenutno fiksne, ali postoje preduvjeti za učitavanje specifičnih konfiguracija sa servera, što bi omogućilo potpunu proširivost aplikacije barem u pogledu tipova pitanja.

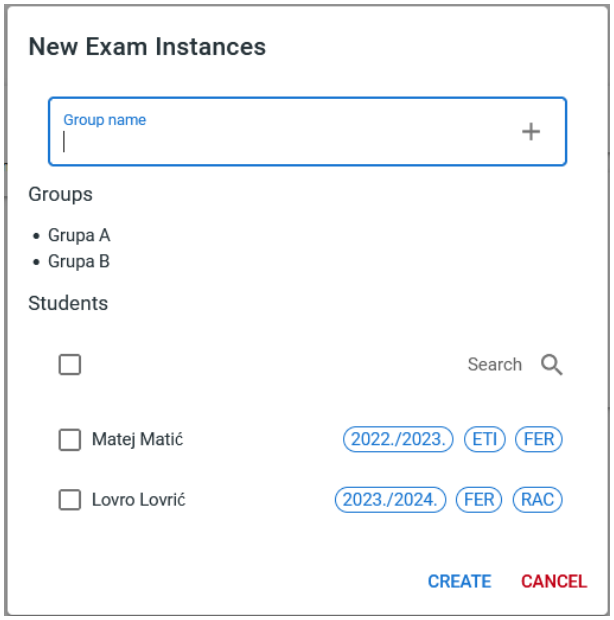
Tip 'Text' nema specifične konfiguracije, dok tip 'ABC' omogućuje dodavanje po-

nuđenih odgovora, označavanje točnih odgovora, nasumičan poredak i broj ponuđenih odgovora prilikom generiranja pitanja. Uz to, 'ABC' pitanja nude mogućnost odabira ponuđenih odgovora koji se moraju prikazati te promjenu redoslijeda ponuđenih odgovora povlačenjem uz lijevi rub.



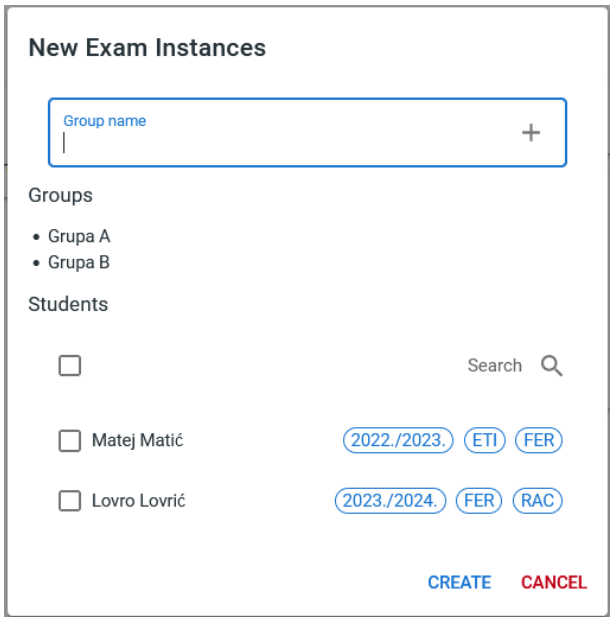

**Slika 6.8:** Prikaz dijaloga za stvaranje novog pitanja

Prilikom stvaranja instance ispita otvara se dijalog koji omogućuje dodavanje grupa i odabir studenata za tu instancu. Popis studenata može se pretraživati prema imenu studenta ili oznaci koja se nalazi uz studenta. Popis grupa kasnije se ne može mijenjati (barem ne kroz aplikaciju), dok je popis studenata promjenjiv. Potrebno je dodati barem jednu grupu, a važno je da svaka grupa ima jedinstveno ime jer se ono koristi prilikom generiranja pseudo-slučajnog sjemena za pitanja.




**Slika 6.9:** Prikaz dijaloga za stvaranje nove instance ispita

Uz navedene dijaloge, postoji još jedan dijalog za upravljanje vezama. Taj dijalog sadrži pretraživu listu objekata i dolazi u dvije varijante: upravljač veza pitanja i upravljač veza studenata. Vezama s pitanjima može se pristupiti sa stranice predmeta i ispita, dok se upravljanje studentima odnosi na instancu ispita.




**Slika 6.10:** Prikaz dijaloga za stvaranje nove instance ispita

## 6.2. Backend

Backend je jedini fiksni dio aplikacije. On obuhvaća Restful API, učitavanje i komunikaciju s bazom podataka preko dobro poznatog sučelja, učitavanje proširenja te logiku za obradu skeniranog obrasca.

### 6.2.1. REST API

API implementira GET, PUT, PATCH, POST i DELETE metode za sve glavne objekte. Ti objekti su predmeti (subjects), ispiti (exams), pitanje (questions) i učenici (students). Uz navedene objekte, postoje i izvedeni kao što su instanca ispita (exam instance) i rezultati ispita (exam results).
Sve metode prate REST načela:

- GET se koristi za dohvat podataka.
- PUT zamjenjuje objekt na danoj putanji s objektom iz svog tijela.
- PATCH nadopunjava, tj. mijenja samo dijelove objekta na danoj putanji s dijelove objekta iz tijela poruke.
- POST stvara novi objekt s novim jedinstvenim identifikatorom.
- DELETE briše objekt koji se nalazi na danoj putanji

Sve metode za izmjenu podataka ili stvaranje novih podataka vraćaju izmijenjeni tj. stvoreni objekt. Metoda za brisanje vraća rezultat radnje u obliku boolean varijable koja predstavlja uspješnost akcije.

### 6.2.2. Moduli

Uz RESTful API, na serveru se nalazi još nekoliko modula. Tim modulima može se pristupiti pozivom na njihovu URL adresu, a dolaze u obliku web stranica (HTML + JavaScript)

Modul za generiranje PDF dokumenata nalazi se na putanji '.../template/index.html#...' Taj modul generira HTML stranice prema parametrima u adresi iza '#'.Kasnije se te stranice pretvaraju u PDF dokumente. Korisničko sučelje također koristi ovaj modul za prikaz predispisa ispita.

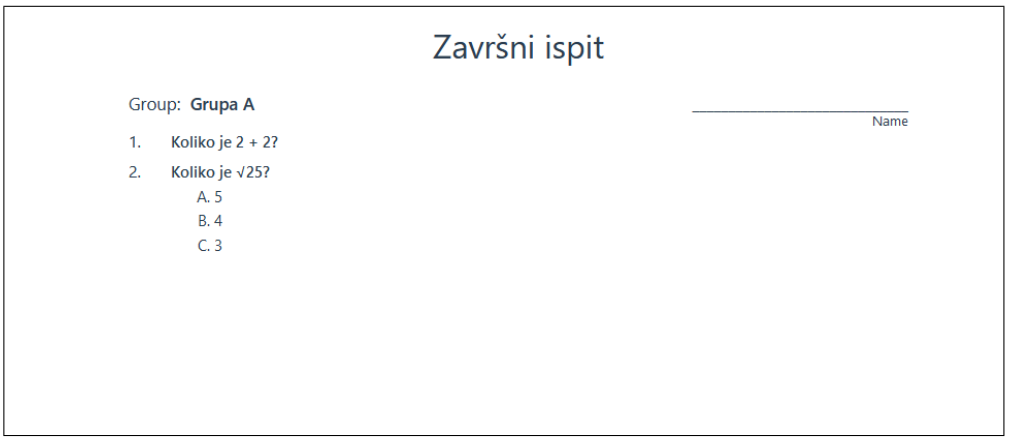
Trenutno su podržane 3 putanje unutar ovog modula:

– .../template/index.html#/exams/:ispit-id/instances/:instanca-id/groups/naziv-grupe

– .../template/index.html#/exams/:ispit-id/instances/:instanca-id/sheets

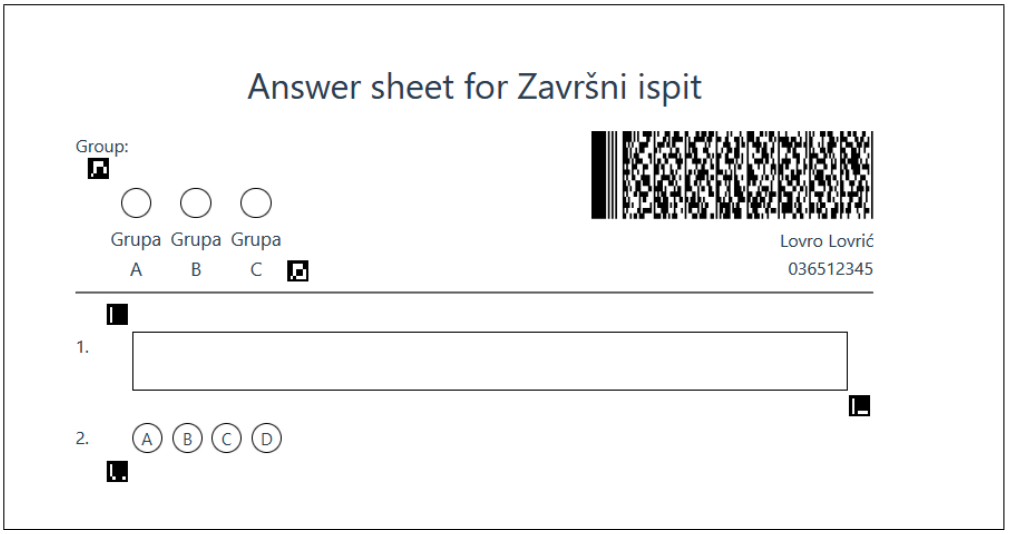
– .../template/index.html#/exams/:ispit-id/instances/:instanca-id/sheets/:student-id

Prva putanja prikazuje generirani ispit. Pseudoslučajni broj koji određuje karakteristike pitanja modificira se s nazivom grupe te je stoga bitno da je različit u slučaju više grupa.

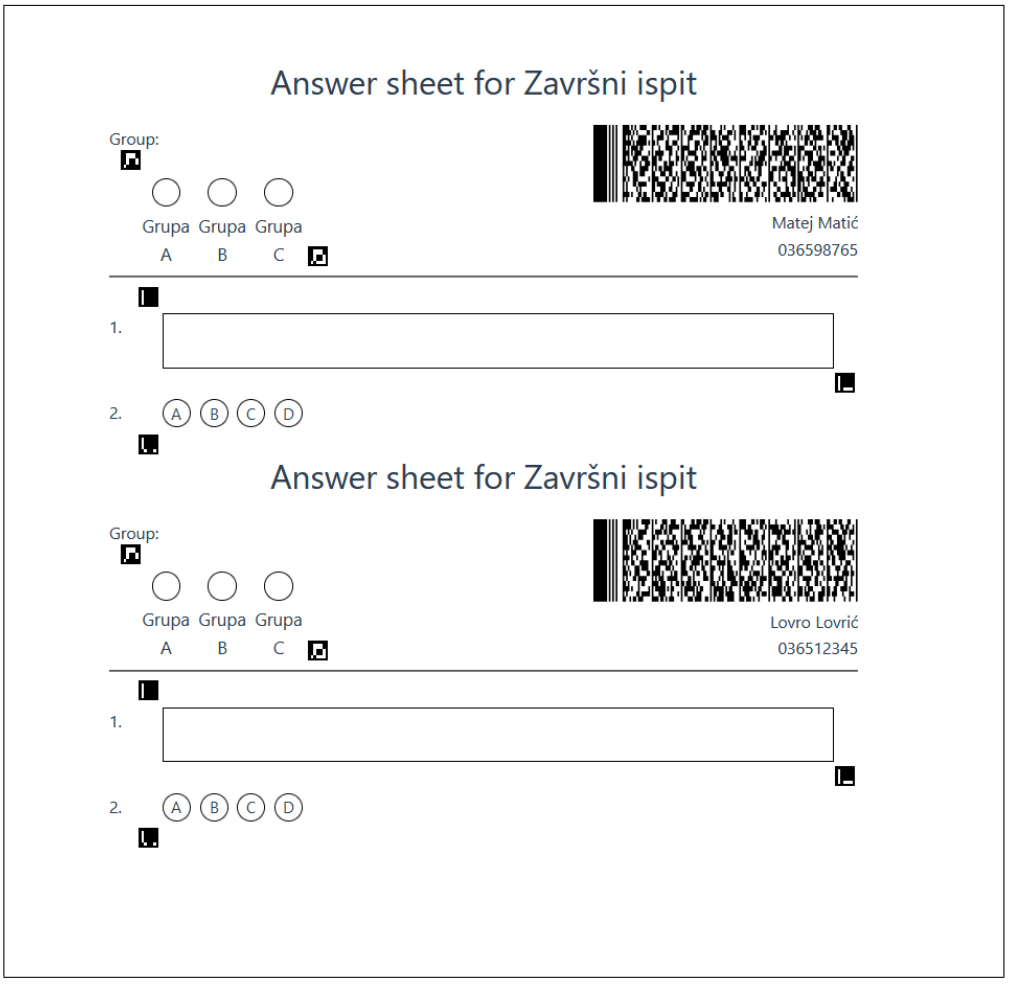



**Slika 6.11:** Prikaz modula za generiranje PDF-a - ispit

Druga putanja dohvaća sve studentske obrasce, a zadnja putanja dohvaća pojedini obrazac za studenta.



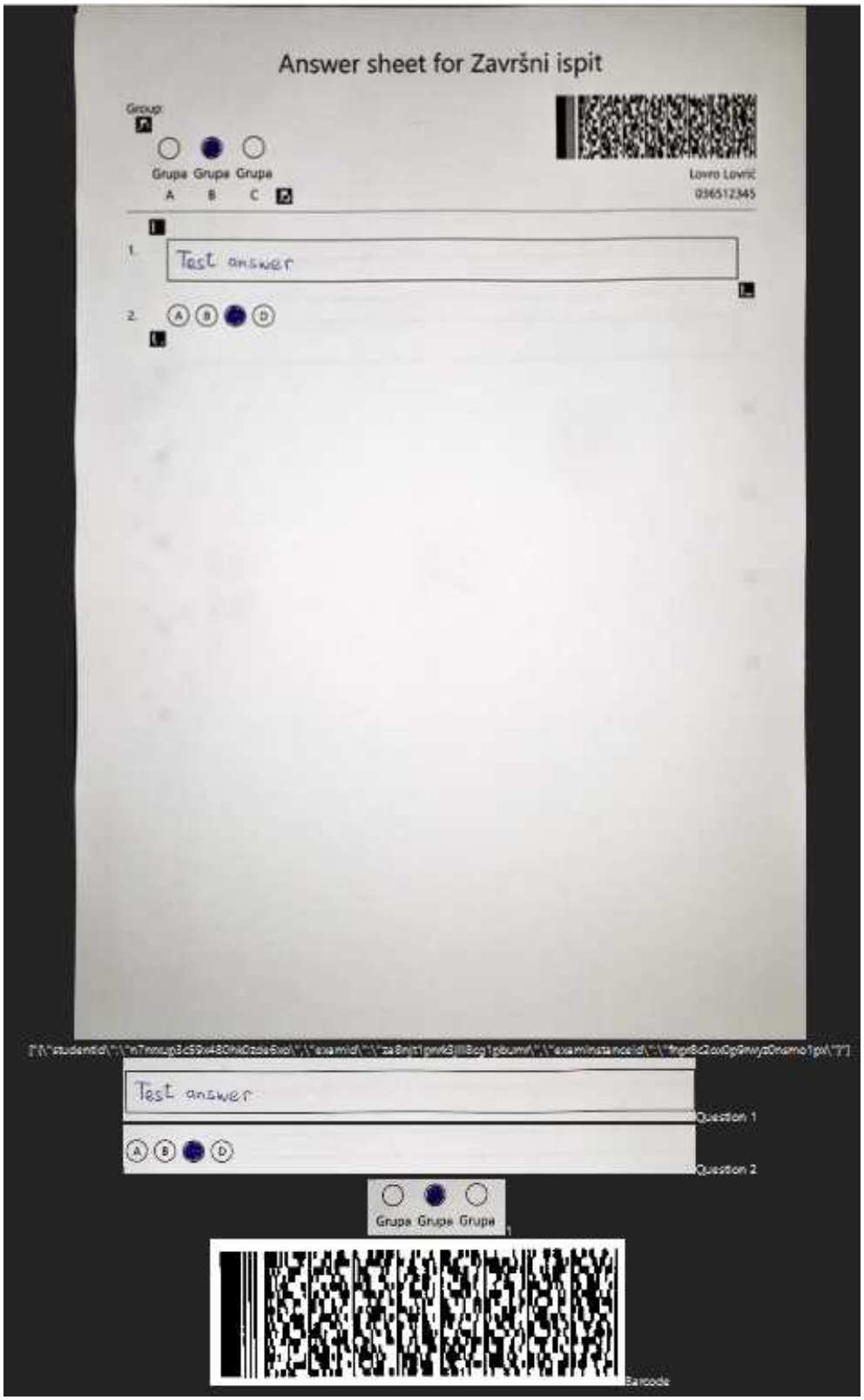

**Slika 6.12:** Prikaz modula za generiranje PDF-a - jedan obrazaca




**Slika 6.13:** Prikaz modula za generiranje PDF-a - više obrazaca

Modul za prijenos, odnosno prihvaćanje slika obrazaca, sastoji se od samo jedne putanje: '.../upload'. Na toj putanji prihvaćaju se 'multipart form' POST zahtjevi u kojima se datoteka mora nalaziti u polju 'file'. Ovaj modul zatim, u suradnji s modulom za analizu slike, obrađuje dobivenu datoteku. Korisnik dobiva odgovor '202 Accepted'. Trenutno nije moguće pratiti status analize prenesenog dokumenta, ali to je potencijalna nadogradnja za budućnost.

Modul za analizu slike obrađuje sliku te iz nje izvlači podatke poput pitanja, grupe i osnovnih informacija o obrascu. Pristupa mu se putem putanje '.../scanner/...'. Iako nije predviđen pristup ovom modulu izvan servera, tehnički nije ni zabranjen. Analiza samog obrasca i razdvajanje na pojedina pitanja odvija se na putanji '.../scanner/analyser.html#:slika-id'.




**Slika 6.14:** Prikaz modula za skeniranje (.../analyser.html) nakon izvršene analize

### 6.2.3. Proširenja

Proširenja aplikacije moguća su u obliku novih tipova pitanja. Server u potpunosti podržava ovu vrstu nadogradnje, dok sučelje još uvijek nema implementiranu podršku. Trenutno dostupni tipovi pitanja implementirani su upravo s pomoću sustava proširenja.

Proširenje trenutno mora definirati izgled polja za odgovor na obrascu i prikaz pitanja na ispitu. U budućnosti će biti potrebno definirati i izgled na sučelju prilikom definiranja pitanja te prikaz pitanja u listama na korisničkom sučelju.

## 6.3. Baza podataka

Baza podataka je najlakše zamjenjivi dio sustava. Tehnički, to ne mora biti baza podataka, podaci mogu biti pohranjeni u bilo kojem formatu. Važno je samo da objekt koji dohvaća podatke odgovara zadanom sučelju. Ovakav pristup omogućava povezivanje s postojećom bazom podataka ili nekim drugim sustavom.

Aplikacija dolazi upakirana sa SQLite bazom podataka, koja je pohranjena u datoteci uz izvršni kod. Ova implementacija služi kao ogledni primjer (ne nužno optimalni) za druge implementacije. Predviđeno je da organizacija koja postavlja sustav može sama odabrati bazu podataka i implementirati sučelje za komunikaciju koje njoj odgovara.

# 7. Budući rad

U ovom kratkom poglavlju su navedene neke od ideja za daljnji razvoj i unaprjeđenje aplikacije. Ideje su navedene u nasumičnom poretku:

- Omogućiti proširivost tipova pitanja na frontendu
- Osposobiti integraciju s Electronom
- Dodati još pokaznih primjera spremišta podataka
- Dodati autentifikaciju i autorizaciju u aplikaciju
- Dodati limitiranje količine podataka u jednom zahtjevu
- Dodati metapodatke u definiciju spremišta podataka kako bi se frontend mogao prilagoditi (primjer s ispitom i više predmeta)
- Osposobiti mobilni skener za prepoznavanje dokumenata
- Omogućiti ispis obrazaca na više stranica/stupaca
- Dodati sustav bodovanja i ocjenjivanja
- Dodati "lažne" studente za ispis obrazaca za unaprijed nepoznate osobe
- Dodati mogućnost odabira jezika
- Unaprijediti postojeće tipove pitanja (npr. dodatne linije za odgovor kod tekstualnih pitanja, pokušaj analize rukopisa te pohrana otkrivenog teksta...)
- Dodati nove tipove pitanja (npr. slijepa karta, povezivanje pojmova...)
- Poboljšati prikaz rezultata ispita i dodati skupni prikaz
- Dodati prikaz rezultata studentima
- ...

# 8. Zaključak

Zaključak ovog diplomskog rada usmjeren je na razvoj sustava za automatizirano ocjenjivanje ispita, pri čemu je glavni cilj u velikoj mjeri postignut. Sustav je osmišljen kao korisno rješenje za manje obrazovne ustanove ili pojedince koji trebaju efikasan način provedbe ispita s brzim ispravljanjem. Omogućava pohranu pitanja, njihovu raspodjelu po predmetima i ispitima, kao i generiranje ispita te obrazaca prema definiranim kriterijima. Dodatno, sustav nudi automatsko ispravljanje putem prijenosa slika obrazaca te pohranu rezultata ispita u bazu podataka.

Aplikacija se sastoji od tri glavne komponente: korisničkog sučelja (frontend), poslužitelja (backend) i baze podataka. Njezina modularna struktura omogućuje fleksibilne nadogradnje, kao i jednostavnu zamjenu pojedinih dijelova sustava. Poslužiteljski dio uključuje različite module, kao i proširenja za tipove pitanja, dok sustav dolazi s SQLite bazom podataka, koju je moguće zamijeniti prema potrebama korisnika.

Posebna pažnja posvećena je funkcionalnostima označavanja i pretraživanja podataka poput ispita, pitanja i predmeta. Korištenjem oznaka (tagova) omogućuje se jednostavna organizacija i brza pretraga unutar sustava, uz podršku za ponovno korištenje oznaka, čime se olakšava rad korisnicima.

Iako je implementacija sustava opsežna, nedostaju ključne funkcionalnosti autentikacije i autorizacije, koje su zbog vremenskih i opsežnih ograničenja izostale iz trenutne verzije. Međutim, sustav je postavljen na stabilne temelje, što ostavlja prostor za daljnji razvoj i implementaciju novih značajki.

Zaključno, iako je sustav još uvijek u ranoj fazi, nudi značajan potencijal za daljnje unaprjeđenje. Mogućnosti nadogradnje i ideje za budući razvoj navedene su u radu, otvarajući put za daljnju optimizaciju i širenje funkcionalnosti.

# LITERATURA


[1] Getting Started | Axios Docs, . URL https://axios-http.com/docs/intro.

[2] Build cross-platform desktop apps with JavaScript, HTML, and CSS | Electron, . URL https://electronjs.org/.

[3] Fast and low overhead web framework, for Node.js | Fastify, . URL https://fastify.io/.

[4] JS-ARUCO2, . URL https://damianofalcioni.github.io/js-aruco2/.

[5] What is OpenAPI?, . URL https://www.openapis.org/what-is-openapi.

[6] Puppeteer | Puppeteer, . URL https://pptr.dev/.

[7] Quasar Framework - Build high-performance VueJS user interfaces in record time, . URL https://quasar.dev/.

[8] SQLite Home Page, . URL https://www.sqlite.org/index.html.

[9] Overview | TanStack Query Vue Docs, . URL https://tanstack.com/query/latest/docs/framework/vue/overview.

[10] Vue.js, . URL https://vuejs.org/.

[11] TypeScript-first schema validation with static type inference, . URL https://zod.dev/.

[12] Rest depiction. https://devopedia.org/richardson-maturity-model Richardson Maturity Model. Source: Sandoval. 2018.

[13] Vue.js logo. Evan You, https://github.com/yyx990803, CC BY 4.0 <https://creativecommons.org/licenses/by/4.0>, via Wikimedia Commons.

[14] bwip-js, Kolovoz 2024. URL https://www.npmjs.com/package/bwip-js.

[15] M. Asad Ali. pyxploiter/Barcode-Detection-and-Decoding, Listopad 2017. URL https://github.com/pyxploiter/Barcode-Detection-and-Decoding.

[16] Jones Esteban Munch. OpenCV: OpenCV modules, Veljača 2021. URL https://docs.opencv.org/4.10.0/index.html.

[17] Lokesh Gupta. Richardson Maturity Model, Ožujak 2018. URL https://restfulapi.net/richardson-maturity-model/.

[18] Adrian Rosebrock. Bubble sheet multiple choice scanner and test grader using OMR, Python, and OpenCV, Listopad 2016. URL https://pyimagesearch.com/2016/10/03/bubble-sheet-multiple-choice-scanner-and-test-grader-using-omr-python-and-opencv/.

[19] Codecademy Team. What is REST? URL https://www.codecademy.com/article/what-is-rest.

**Sustav za automatizirano ocjenjivanje ispita**


**Sažetak**

Ovaj diplomski rad bavi se razvojem sustava za automatizirano ocjenjivanje ispita, što predstavlja značajan korak naprijed u modernizaciji obrazovnog procesa. Kroz detaljan opis komponenti sustava, uključujući izgled baze podataka, poslužiteljsko sučelje te korisničko sučelje za pregled i analizu rezultata, prikazana je cjelokupna arhitektura i funkcionalnost sustava. Daljnja istraživanja i razvoj mogu dodatno poboljšati funkcionalnosti sustava, uključujući integraciju s drugim obrazovnim alatima i prilagodbu specifičnim potrebama različitih obrazovnih institucija. Ovaj rad postavlja temelje za buduće inovacije u području hibridnog ocjenjivanja i pruža smjernice za daljnji razvoj i implementaciju sličnih sustava.



**Ključne riječi:** generiranje ispita ; automatizirano ispravljanje ; analiza slike ; analiza obrazaca ; aplikacija ; fastify ; quasar ; vuejs ; sqlite3 ; opencv.js; aruco


**System for automated evaluation of exams**


**Abstract**

This thesis deals with the development of a system for automated exam grading, which represents a significant step forward in the modernization of the educational process. The overall architecture and functionality of the system are presented through a detailed description of the system components, including the database structure, server interface, and user interface for reviewing and analyzing results. Further research and development can additionally enhance the system's functionalities, including integration with other educational tools and adaptation to the specific needs of different educational institutions. This work lays the foundation for future innovations in the field of hybrid grading and provides guidelines for further development and implementation of similar systems.



**Keywords:** exam generation ; automated correction ; image analysis ; bubble sheet analysis; application; fastify ; quasar ; vuejs ; sqlite3 ; opencv.js; aruco