SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 910

JEDNOSTRANIČNA WEB-APLIKACIJA ZA VIZUALIZACIJU PODATAKA U SUSTAVU EDGARU

Lovro Kovačić

Zagreb, lipanj 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 910

JEDNOSTRANIČNA WEB-APLIKACIJA ZA VIZUALIZACIJU PODATAKA U SUSTAVU EDGARU

Lovro Kovačić

Zagreb, lipanj 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 10. ožujka 2023.

ZAVRŠNI ZADATAK br. 910

Pristupnik:	**Lovro Kovačić (0036533938)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	prof. dr. sc. Igor Mekterović

Zadatak: **Jednostranična web-aplikacija za vizualizaciju podataka u sustavu Edgaru**

Opis zadatka:

Edgar je informacijski sustav koji se koristi za ispitivanje studenata na Fakultetu elektrotehnike i računarstva. U okviru nekog predmeta, tipično se provodi niz većih ili manjih ispita (domaćih zadaća, projekata, itd.) tijekom semestra putem kojih studenti skupljaju bodove koji im ulaze u konačnu ocjenu. Trenutno Edgar nudi različite vizualizacije uspješnosti ispita i pitanja, tablične i grafičke podatke o poretku i uspjehu studenata, čak i podatke u ponašanju (akcijama) studenata za vrijeme pisanja ispita. Vizualizacije su rađene koristeći D3.js i Apache ECharts knjižnice te vizualno donekle odudaraju. U okviru rada potrebno je proučiti načela vizualizacije podataka, zatim proučiti postojeće vizualizacije u Edgaru te predložiti poboljšanja i eventualne dodatne vizualizacije. Proučiti razvoj jednostraničnih web-aplikacija u Quasar radnom okviru koji se temelji na Vue.js radnom okviru. Proučiti kako integrirati D3.js knjižnicu u Quasar te u spomenutim tehnologijama implementirati predložene vizualizacije. Donijeti ocjenu ostvarenog pristupa te smjernice za budući razvoj.

Rok za predaju rada: 9. lipnja 2023.

Sadržaj

Uvod

Informacijski sustavi ključni su dio suvremenog obrazovanja jer pružaju studentima i nastavnicima razne alate koji olakšavaju i ubrzavaju provođenje nastave.

Provođenje ispita preko takvih sustava omogućuje automatsko ispravljanje koje znatno smanjuje količinu posla predavačima. Rezultati se brzo mogu unijeti u sustav, obraditi i prikazati studentima na brojne načine. Tako studenti mogu biti svjesni svog uspjeha u odnosu na kolege te nastavnici mogu dobiti puno bolju sliku o razini usvojenosti tema koje su se ispitivale. Automatsko provjeravanje plagijata također je jedna od mogućnosti koje nam takvi sustavi pružaju – ono smanjuje mukotrpno ručno pregledavanje ispita te povećava količinu pronađenih plagijata. Informacijski sustavi mogu nam unaprijediti i stil provođenja nastave u predavaonicama. Korištenjem upitnika, kvizova i drugim interaktivnih sadržaja pruža nam se jedinstvena prilika za uključivanje cijele dvorane studenata u razne aktivnost. Nastava tako postaje interaktivna i općenito ugodnija.

Jedan takav informacijski sustav je Edgar, koji se koristi na Fakultetu elektrotehnike i računarstva u Zagrebu. Edgar primarno dijelimo na 2 dijela: nastavnički i studentski. Studentski dio Edgara služi provođenju ispita, domaćih zadaća i projekata, omogućujući studentima da tijekom semestra skupljaju bodove koji ulaze u njihovu konačnu ocjenu. Nastavnička strana Edgara ima puno više funkcionalnosti, poput uređivanja i dodavanja ispitnih pitanja, uvid u studentske profile, detekciju plagijata te vizualizaciju rezultata provedenih provjera. Edgar trenutno nudi različite vrste vizualizacija, uključujući tablične i grafičke prikaze o poretku i uspjehu studenata, te vizualizacije njihovog ponašanja tijekom ispita.

Cilj ovog rada istražiti je načela vizualizacije podataka, analizirati postojeće vizualizacije unutar sustava Edgar i predložiti poboljšanja, kao i eventualne nove vizualizacije koje bi omogućile lakše korigiranje i prilagođavanje ispitnih materijala.

Ova analiza temeljiti će se na upotrebi tehnologija kao što su *Quasar* i *Vue.js*, te na integraciji s *D3.js* bibliotekom.

.

1. Korištene tehnologije

1.1. Jednostranične web-aplikacije (SPA)

Klasične višestranične web-aplikacije, MPA (eng. *Multi Page Application*), prilikom svakog klika korisnika na neku poveznicu šalju zahtjev web poslužitelju, koji zatim vraća novu web stranicu. Ovaj proces može biti spor i može smanjiti razinu korisničkog iskustva.

Jednostranične web-aplikacije, SPA (eng. *Single Page Application*), rješavaju ovaj problem. SPA je web aplikacija ili stranica koja komunicira s korisnikom dinamičkim ažuriranjem trenutne web stranice novim podacima s web poslužitelja, umjesto zadane metode web preglednika koja učitava potpuno nove stranice. Cilj su brži prijelazi koji web stranicu čine sličnijom izvornoj aplikaciji.

Umjesto osvježavanja cijele stranice, sav potreban HTML (eng. *HyperText Markup Language*), *JavaScript* i CSS (eng. *Cascading Style Sheets*) web preglednik ili preuzima s jednim učitavanjem stranice, ili se odgovarajući resursi dinamički učitavaju i dodaju na stranicu po potrebi, obično kao odgovor na radnje korisnika [9].

To omogućuje brže i ugodnije korisničko iskustvo, slično iskustvu kod stolnih (engl. *desktop*) ili mobilnih aplikacija. Također, smanjuje se opterećenje na server zbog smanjenog broja zahtjeva koji dolazi do njega.

Uz brojne prednosti, SPA također donose i određene izazove. Jedan od njih je implementacija optimizacije za tražilice (eng. *Search engine optimization*, SEO). SEO je vrlo važan marketinški proces optimiziranja sadržaja web stranica fokusiran na povećanju vidljivosti stranice na tražilicama. *Javascript* sadržaj SPA otežava programima za indeksiranje i pretraživanje interneta (eng. *web crawlers*) samostalno učitavanje HTML i CSS sadržaja ciljanih stranica. Također, često je početno učitavanje SPA dulje od MPA zbog veće količine informacija koje su potrebne klijentu

1.2. *Node.js*

Node.js je više-platformsko (eng. *cross-platform*) okruženje otvorenog koda (eng. *open-source*) koje omogućuje izvršavanje *JavaScript* koda na serveru.

Ključna prednost *Node.jsa* njegova je ne blokirajuća, asinkrona priroda. Za razliku od tradicionalnih serverskih okruženja koja obrađuju zahtjeve sinkrono, *Node.js* može obraditi mnogo zahtjeva paralelno bez blokiranja. To znači da može učinkovitije koristiti resurse, omogućujući bržu i skalabilniju izvedbu.

Node.js također koristi *V8 JavaScript* motor *(eng. engine)*, isti onaj koji koristi Google Chrome, što ga čini vrlo brzim.

Kroz *Node.js*, *JavaScript* je evoluirao iz jezika namijenjenog samo za klijentsku stranu do jezika koji se može koristiti za izgradnju sveobuhvatnih, tzv. *full-stack* aplikacija. To je dovelo do porasta popularnosti jezika *JavaScript*, kao i do razvoja cijelog ekosustava alata, biblioteka i okvira oko Node.js, uključujući popularni okvir *Express.js* kojeg koristimo za pokretanje servera naše aplikacije.

1.3. *Vue.js*

Vue.js je *JavaScript* razvojni okvir (eng. *framework*) za izgradnju korisničkih sučelja. Razvijen s fokusom na jednostavnost i fleksibilnost, Vue.js omogućava brz i učinkovit razvoj SPA i MPA.

Korištenje okvira *Vue.js* donosi brojne prednosti, uključujući čitljiviji kod, reaktivnost, modularnost te brz razvoj. Uz to, *Vue.js* podržava komponentno programiranje, što znači da se web aplikacija može podijeliti na manje, ponovno upotrebljive komponente. To povlači lakše održivu arhitekturu koda.

Odličan je za početnike zbog svoje podrazumijevane konfiguracije te relativno jednostavne sintakse. To ga također čini savršenim za primjenu na manjim projektima. Profesionalci ga vole zbog dobre skalabilnosti i fleksibilnosti u organizaciji koda i samoj konfiguraciji.

1.4. *Quasar*

Quasar je radni okvir visokih performansi baziran na okviru *Vue.js* koji pruža integrirani skup alata i komponenti za pomoć razvojnim timovima u brzoj izgradnji aplikacija. Ovaj skup alata uključuje predefinirane CSS klase, komponente i priključke (eng. *plugin*).

Quasar se posebno ističe svojom podrškom za izgradnju aplikacija koje se mogu pokrenuti na različitim platformama iz jedne baze koda. Ova značajka smanjuje kompleksnost i cijenu razvoja, a posebno održavanja programskog proizvoda. S jednim kodom možete izraditi

aplikacije za web, s podrškom za SSR (eng. *Server Side Rendering*) i PWA (eng. *Progressive Web Apps*), mobilne aplikacije koristeći *Cordova* ili *Capacitor*, *Electron* aplikacije za radnu površinu i čak razvijati aplikacije za mobilne operativne sustave kao što su *iOS* i *Android*.

Quasar sadrži bogatu kolekciju gotovih komponenti, koje se lako mogu integrirati u aplikaciju. To uključuje različite vrste komponenta za unos podataka (poput polja za unos teksta, prekidača, potvrdnih okvira), navigacijskih komponenta (poput traka s karticama, izbornika, alatnih traka), komponenta za prikaz podataka (poput tablica, kartica, popisa) i mnogih drugih. Posebno je korisna *Quasarova* mogućnost upotrebe podsustava za raspored elemenata na ekranu (eng. *Layout*), što omogućuje fleksibilno i responzivno raspoređivanje komponenti unutar aplikacije.

Quasar također pruža sustav za tematizaciju aplikacija. Tematizacija se odnosi na mogućnost promjene vizualnog izgleda i dojma aplikacije kroz definiranje kombinacija stilova i boja, omogućujući korisnicima da prilagode aplikaciju svojim individualnim preferencijama ili potrebama. Ovo nam omogućuje definiranje temi poput svijetle i tamne koje se zatim mogu koristiti kroz cijelu aplikaciju.

1.5. *D3.js* biblioteka

D3.js je *JavaScript* biblioteka za manipulaciju dokumenata na temelju podataka. Ova biblioteka koristi se za stvaranje interaktivnih vizualizacija podataka u web preglednicima [3].

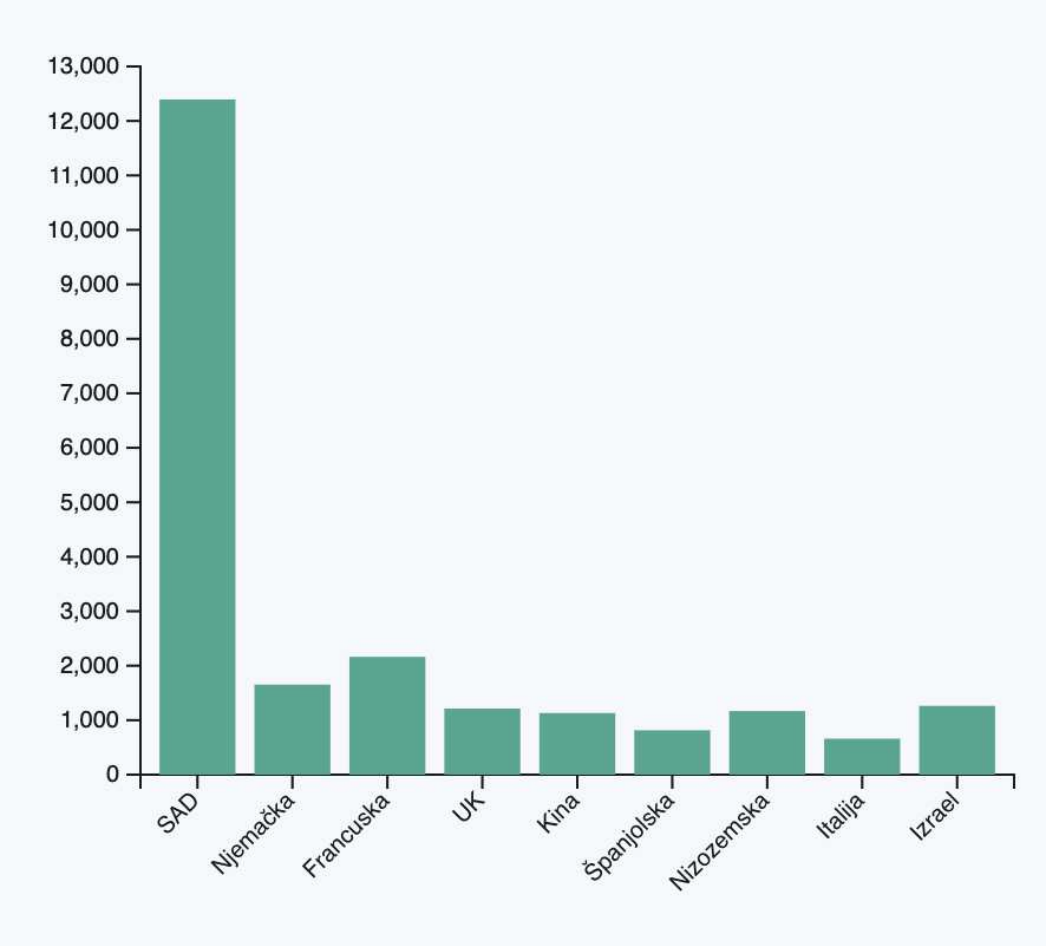
Vrlo je fleksibilna i moćna te omogućava stvaranje složenih i visoko prilagodljivih vizualizacija. Koristi web standarde poput SVG (eng. *Scalable Vector Graphics*), HTML i CSS, što omogućava široku kompatibilnost i fleksibilnost.

D3.js pruža brojne alate i funkcionalnosti, uključujući alate za rad s bojama, oblicima, skalama, uzorcima, interpolacijom, transformacijama, prijelazima i puno više. Ideja iza biblioteke omogućiti je korisnicima modularan pristup te potpunu kontrolu prilikom razvijanja vizualizacija.

Iako je vrlo moćan alat, njegova složenost također je njegova najveća mana. Biblioteka se teško usvaja te zahtjeva dobro razumijevanje tehnologija poput SVG-a i programiranja u jeziku *Javascript*. U većini slučajeva nije potrebno korištenje ovakvog alata jer već postoje gotova rješenja koja su puno pristupačnija.

1.5.1. Primjer korištenja *D3.js* biblioteke

Slika 1.1 prikazuje jednostavan primjer korištenja biblioteke [10]. Pomoću njega približit ćemo osnovne funkcionalnosti *D3.js*.



Slika 1.1 Jednostavan primjer stupčastog grafa izrađenog pomoću *D3.js* biblioteke [10]

Za prikaz tog grafikona treba nam HTML i JS (eng. *Javascript*) dokument. Slika 1.2 prikazuje korišteni HTML dokument, a Slika 1.3 korišteni JS kod.

barChartExample.html > div#my_dataviz

```html
<!DOCTYPE html>
<meta charset="utf-8" />

<!-- Učitavanje D3.js biblioteke sa službenog mjesta -->
<script src="https://d3js.org/d3.v6.js"></script>
<script type="module" src="barChartExample.js"></script>

<!-- Stvaranje div-a u kojeg ćemo smjestiti grafikon -->
<div id="my_dataviz"></div>

```

Slika 1.2 Sadržaj HTML dokumenta korištenog za izradu jednostavnog stupčastog grafikona

barChartExample.js > ...

```javascript
import data from './podatci-za-graf.json' assert {type: 'json'};

// postavljanje dimenzija i margina grafikona
const margin = {top: 30, right: 30, bottom: 70, left: 60},
    width = 460 - margin.left - margin.right,
    height = 400 - margin.top - margin.bottom;

// umetanje svg elementa u body
const svg = d3.select("#my_dataviz")
  .append("svg")
    .attr("width", width + margin.left + margin.right)
    .attr("height", height + margin.top + margin.bottom)
  .append("g")
    .attr("transform", `translate(${margin.left},${margin.top})`);

// X os
const x = d3.scaleBand()
  .range([ 0, width ])
  .domain(data.map(d => d.Country))
  .padding(0.2);
svg.append("g")
  .attr("transform", `translate(0, ${height})`)
  .call(d3.axisBottom(x))
  .selectAll("text")
    .attr("transform", "translate(-10,0)rotate(-45)")
    .style("text-anchor", "end");

// Y os
const y = d3.scaleLinear()
  .domain([0, 13000])
  .range([ height, 0]);
svg.append("g")
  .call(d3.axisLeft(y));

// Stupci
svg.selectAll("mybar")
    .data(data)
    .enter()
  .append("rect")
    .attr("x", d => x(d.Country))
    .attr("y", d => y(d.Value))
    .attr("width", x.bandwidth())
    .attr("height", d => height - y(d.Value))
    .attr("fill", "#69b3a2")
```

Slika 1.3 Sadržaj JS dokumenta korištenog za izradu jednostavnog stupčastog grafikona

`d3.select("#my_dataviz").append("svg")` stvara SVG element unutar prvog `div` elementa kojemu je id jednak `my_dataviz`. SVG se koristi za crtanje grafičkih elemenata.

`D3.scaleBand()` funkcija je koja se koristi za kreiranje kategorijske skale. To je vrsta skale koja se koristi kada podaci na osi nisu kontinuirani, već kategorijski, poput imena zemalja u ovom primjeru. Korištenje `range()` i `domain()` funkcija određuje raspon vrijednosti na osi.

`d3.axisBottom()` stvara funkciju koja generira donju os (x os) za naš grafikon. Kada tu funkciju proslijedimo metodi `call()` koja se primjenjuje na element (u ovom slučaju, grupu elemenata, ili `g`), donja os generira se unutar tog elementa. Oznake na osi generirane su na temelju domene definirane u skali.

`d3.scaleLinear()` funkcija je koja se koristi za kreiranje linearnih skala. To je vrsta skale koja se koristi kada su podaci kontinuiran, poput običnih brojeva.

`d3.selectAll()` funkcija je koja odabire sve elemente određene vrste u dokumentu. Ako takvi elementi ne postoje, vraća se prazan odabir. Ova metoda često se koristi u kombinaciji s `data()` i `enter()` metodama kako bi se povezala skupina podataka s odabranim elementima. Na primjer, u kodu prikazanom na grafici Slika 1.3, `enter()` stvara novi `rect` element za svaki podatak unutar `data`.

`attr()` je metoda koja se koristi za postavljanje ili dobivanje atributa određenog elementa. U našem primjeru postavljamo atribute `x`, `y`, `width`, `height` i `fill` elementima `rect`.

1.6. Integracija tehnologija *Vue.js*, *Quasar* i *D3.js*

Integracija ovih tehnologija nije se pokazala izazovnom. *Vue.js* omogućava stvaranje komponenti koje se mogu ponovno koristiti, a *D3.js* vizualizacije mogu se lako ugraditi u te komponente. Ove komponente se potom mogu koristiti unutar Quasar aplikacije gdje god su potrebne te omogućuju izgradnju složenih korisničkih sučelja.

Slika 1.4 prikazuje izgled prijašnjeg primjera unutar *Vue.js* komponente.

Slika 1.4 Primjer integracije tehnologija *D3.js*, *Vue.js* i *Quasar*

Možemo primijetiti da umjesto `.html` i `.js` dokumenata sada radimo samo s jednim `.vue` dokumentom. Unutar njega možemo definirati `<script>`, `<template>` i `<style>` dijelove.

`<template>` je zamijenio HTML dio koda. Unutar njega definirali smo `div` sa referencom na objekt `my_dataviz`.

Unutar `<script>` dijela stavili smo sav potreban JS kod.

Korištenjem ugrađene Vue.js funkcije `onMounted()` pozivamo funkciju `drawGraph(data)` nakon što je komponenta renderirana.

Ovakva komponenta sada se može koristiti na bilo kojoj stranici unutar *Quasar* aplikacije kao što je vidljivo na grafici Slika 1.5.

frontend > src > pages > ExamplePage.vue > ...

```
1  <template>
2    <q-page class="flex flex-center">
3      <BarChartExample />
4    </q-page>
5  </template>
6
7  <script setup>
8  import BarChartExample from "src/components/BarChartExample.vue";
9  </script>
10
```

Slika 1.5 Primjer uporabe definirane *Vue.js* komponente unutar stranice *Quasar* aplikacije

2. Vizualizacija podataka

Vizualizacija podataka grafički je prikaz informacija i podataka. Koristeći vizualne elemente poput grafikona, dijagrama i mapa, alati za vizualizaciju podataka omogućuju pristupačan način za vidjeti i razumjeti trendove, iznimke i obrasce u podacima. Osim toga, pruža izvrstan način za zaposlenike ili tvrtke da prezentiraju podatke publici koja nema nužno duboku tehničku pozadinu [1].

2.1. Tipovi grafikona

Različiti tipovi vizualizacija mogu se koristiti ovisno o vrsti podataka i informacijama koje želimo prenijeti.

Primjerice, stupčasti grafikoni (eng. *bar charts*) odlični su za usporedbu kvantitativnih vrijednosti između različitih skupina, dok se linijski grafikoni (eng. *line charts*) najčešće koriste za prikaz trendova i promjena tijekom vremena.

Dijagrami raspršenosti (eng. *scatter plots*) idealni su za prikazivanje odnosa između dvije varijable, a histogrami (eng. *histograms*) omogućuju vizualizaciju distribucije podataka. Toplinske karte (eng. *heat maps*) oblik su grafikona koji odlično prikazuju kompleksnije odnose između varijabla čije se vrijednosti obično kreću u nekom realnom rasponu brojeva. Tortni grafikoni (eng. *pie charts*) također su vrlo popularna opcija. Dobri su za prikazivanje podataka koji čine cjelinu, no u sljedećem poglavlju osvrnuti ćemo se na to zašto nisu preferirani.

2.2. Odlike dobre vizualizacije

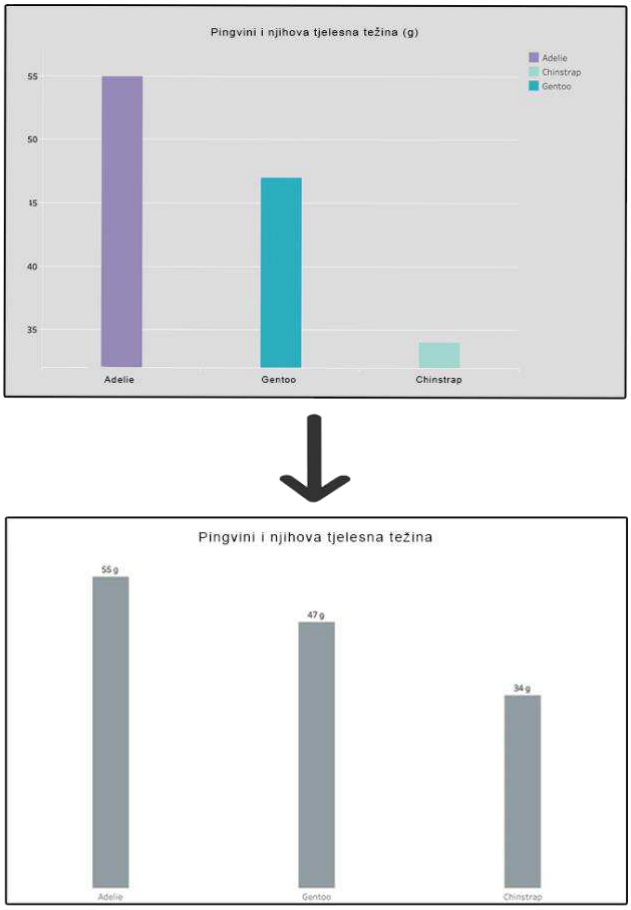
Važno je napomenuti da dizajn dobre vizualizacije nije jednostavan te u njega treba uložiti puno vremena i razmišljanja. Jedan od slavnih primjera loše vizualizacije poznati je tortni graf. Iako vrlo čest u praksi, problematičan je jer ljudi imaju poteškoća s percepcijom kutova i površine. Stoga je bolje koristiti vrste grafikona koji koriste veličine koje ljudi bolje percipiraju, poput duljine i debljine . U ovom poglavlju osvrnuti ćemo se na 5 smjernica koje nam pomažu u osmišljavanju i izradi funkcionalnih vizualizacija [4].

2.2.1. Pokaži podatke

Po Tufteu [4], temeljno pravilo vizualizacije jasno je i nedvosmisleno prikazivanje podataka. Svaka vizualizacija trebala bi pružiti jasnu sliku o tome što podaci predstavljaju. Korisnici bi na prvi pogled trebali moći prepoznati ključne informacije koje se prenose. Ilustracije, boje i oblici trebaju se koristiti da bi se pojačao utjecaj tih podataka, ali nikad da bi se zamaskirali ili preuveličali.

2.2.2. Maksimiziranje omjera podataka i tinte

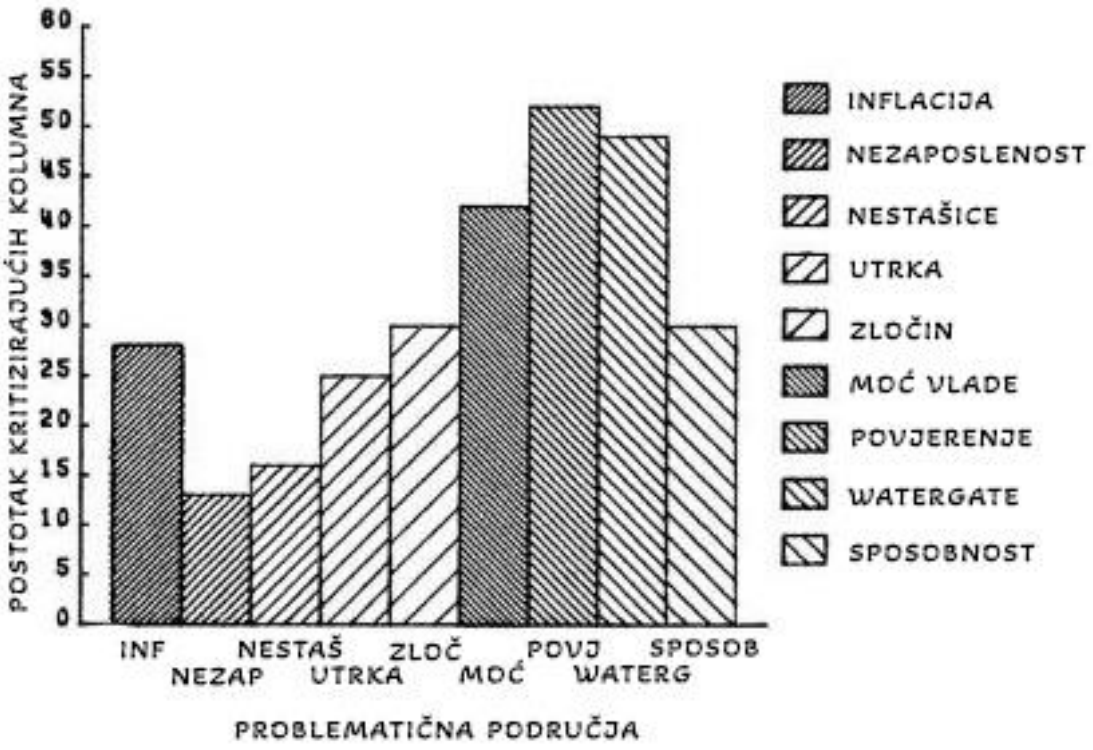
Ovaj koncept govori o efikasnosti vizualizacije. Ideja je minimizirati manje bitne vizualne elemente (ne-podatak tintu) i maksimizirati one koji prenose stvarne informacije (podatak tintu). Na primjer, mreža koja se koristi za pomoć u čitanju grafikona trebala bi biti što suptilnija kako ne bi ometala same podatke te boje na grafikonima treba koristiti rezervirano, s jasnim ciljem. Slika 2.1 prikazuje primjer gdje smo grafikon učinili puno jasnijim micanjem nepotrebnih elemenata.



Slika 2.1 Primjer maksimiziranja omjera podataka i tinte [5]

2.2.3. Brisanje tinte koja ne prikazuje podatke

Ovo pravilo nadograđuje se na prijašnje te se odnosi na uklanjanje svih nepotrebnih elemenata koji ne pružaju nikakve dodatne informacije grafikonu (eng. *chartjunk*). Sve što je nepotrebno ili odvlači pažnju od ključnih podataka treba biti uklonjeno. Česti primjer fenomena *chartjunk* je Moiré efekt, gdje se dizajner igra s različitim grafičkim uzorcima kako bi stvorio dojam vibracije u dijagramu. Moiré vibracije stvaraju distrakciju u dizajnu, a čitatelj često mora prebacivati pogled naprijed-nazad između legende i grafa, što stvara "fiziološki tremor u oku" [8]. Slika 2.2 prikazuje nam primjer takve vizualizacije.



Slika 2.2 Prikaz Moiré vibracije na grafikonu prepunom *chartjunka*

2.2.4. Jasno, detaljno i temeljito označavanje

Bez obzira koliko je sama vizualizacija intuitivna, bez adekvatnog označavanja, korisnici mogu biti zbunjeni. Oznake, naslovi, potpisi, legende i objašnjenja trebaju biti jasni i informativni. Korisnici bi trebali moći razumjeti vizualizaciju čak i ako nemaju prethodno znanje o temi.

2.2.5. Revizija i uređivanje

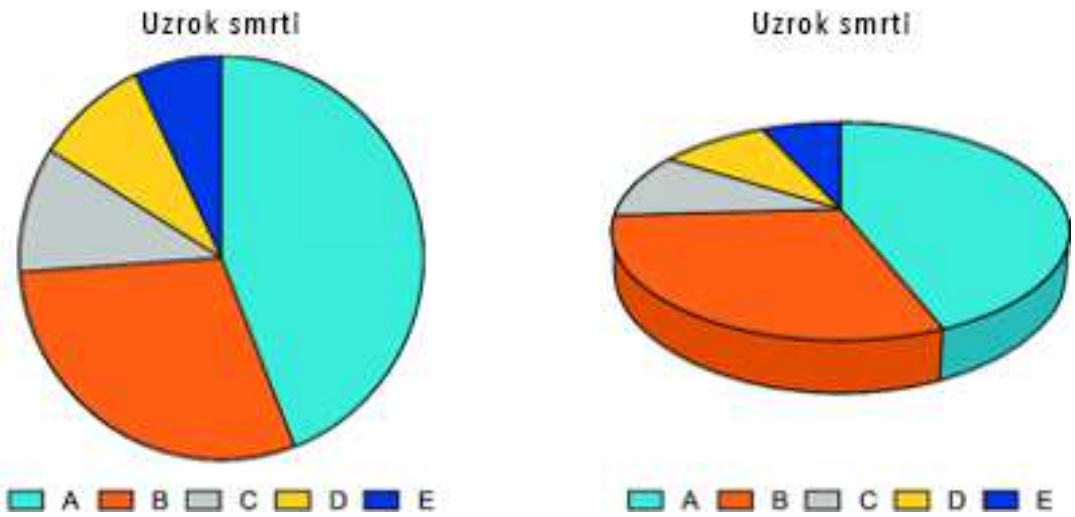
Svaka vizualizacija treba biti dobro promišljena i pažljivo oblikovana kako bi se osiguralo da je što učinkovitija. To uključuje skupljanje povratne informacije od korisnika i ažuriranje vizualizacija primarno prema njihovim potrebama. Bez revizije i uređivanja nikada nećemo doći do rafiniranog proizvoda, što se može primijeniti i na područje puno šire od samih vizualizacija.

2.3. Česte pogreške i savjeti prilikom izrade grafikona

Uzimajući u obzir sva prethodno spomenuta pravila te tipove grafikona, navodimo nekoliko čestih pogrešaka i savjeta prilikom izrade vizualizacija.

2.3.1. Izbjegavati 3D prikaze

Korištenje 3D prikaza u vizualizaciji podataka često može stvoriti nepotrebnu komplikaciju i dovesti do pogrešne interpretacije podataka. Ljudski mozak nije dobar u točnoj percepciji volumena, osobito kada su u pitanju apstraktni objekti poput onih prikazanih u 3D grafovima. Slika 2.3 odlično demonstrira kako uvođenje treće dimenzije može poremetiti odnos cjelina na grafikonu. Na grafikonu lijevo jasno je da je područje A skoro duplo zastupljenije od područja B, dok se na desnom grafikonu taj odnos potpuno izgubi te se zastupljenost područja A i B čini skoro jednaka [8].



Slika 2.3 Primjer distorzije odnosa cjelina uvođenjem treće dimenzije [7]

2.3.2. Odsijecanje y osi

Jedna od čestih pogrešaka u izradi grafikona pogrešno je prikazivanje y osi. Prilikom usporedbe velikih brojeva mnogi pokušavaju, slučajno ili zlonamjerno, isključiti niže vrijednosti kako bi bolje pokazali razlike u brojevima.

Ovo može biti vrlo obmanjujuće jer naizgled povećava razlike između primjera što se odlično vidi na grafici Slika 2.4. Važno je znati gledate li na relativne ili apsolutne razlike između vrijednosti. [8]

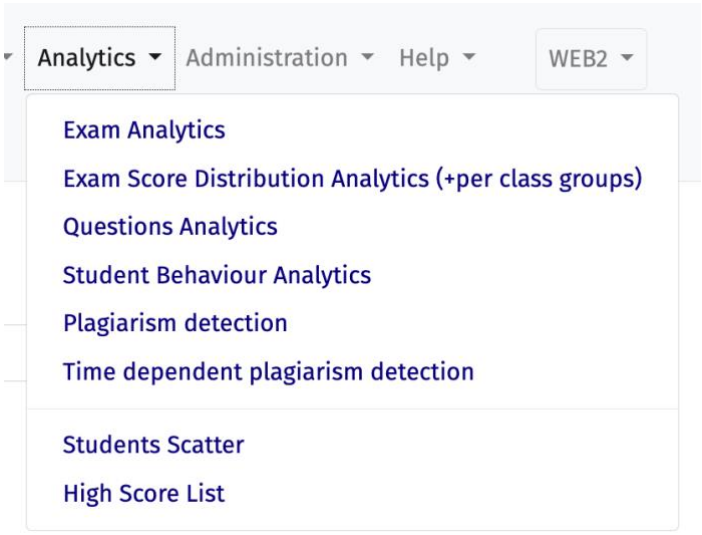
Slika 2.4 Primjer utjecaja odsijecanja y osi na razlike u podatcima

2.3.3. Jednostavnost

Jednostavnost je često ključ uspješnog prikaza podataka. Složeni grafovi mogu preplaviti gledatelja s previše informacija dok jednostavni grafovi omogućuju lakše prepoznavanje ključnih informacija. Ovo ne znači da graf ne smije prikazivati složene podatke, već da dizajn i prikaz trebaju biti što jednostavniji i jasniji. To može uključivati izbjegavanje nepotrebnih elemenata dizajna, korištenje jasnih i razumljivih oznaka i legende, odabir paleta boja koje olakšavaju razlikovanje između različitih dijelova grafikona te razdvajanje grafova s puno komponenta na više odvojenih cjelina.

3. Analiza trenutnog rješenja

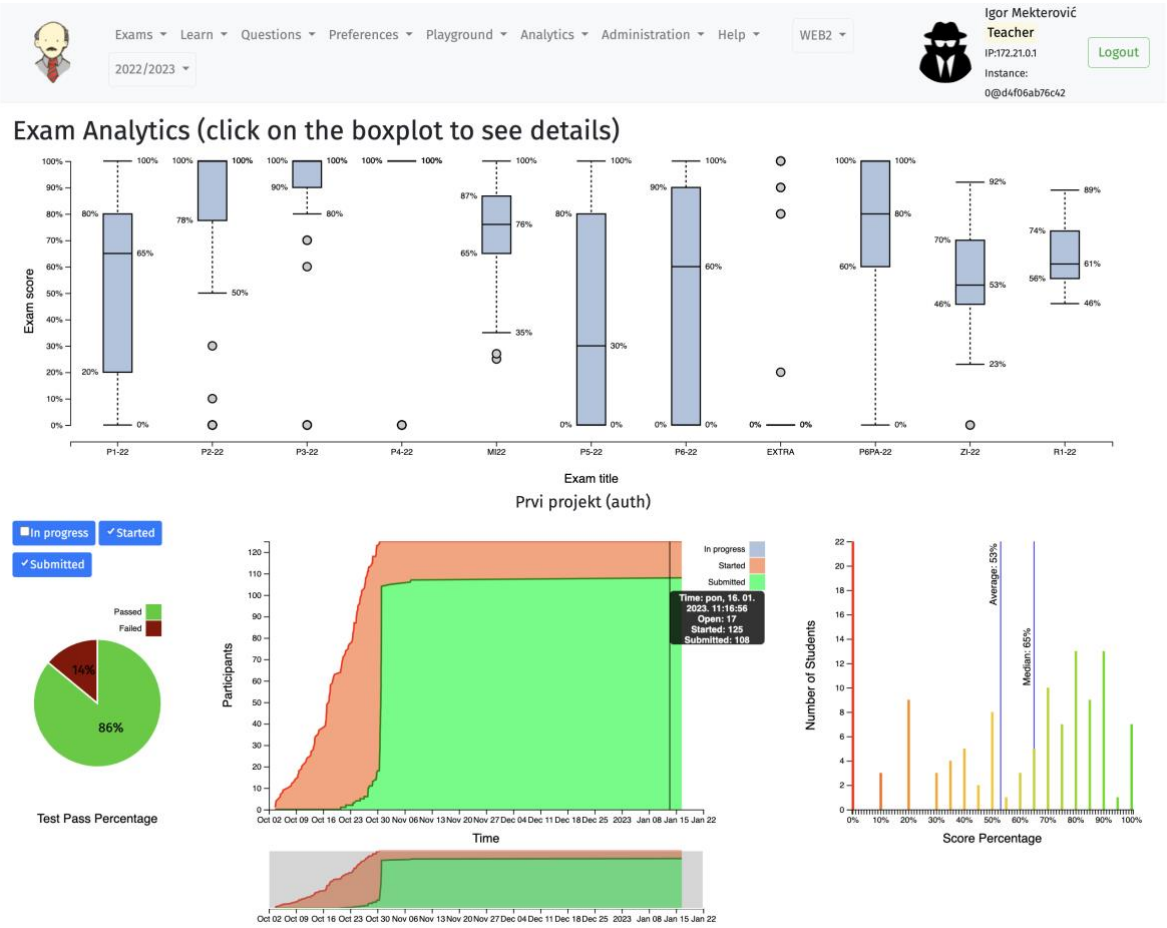
U okviru rada fokusiramo se na stranicu *Analytics* u Edgaru. Ona je samo jedan od dijelova Edgara te se na njoj nalaze nama najzanimljivije vizualizacije. Slika 3.1 prikazuje padajući izbornik sa svim pod-stranicama. U ovom radu osvrnut ćemo se na *Exam Analytics*, *Exam Score Distribution Analytics*, *Questions Analytics*, *Student Scatter* i *High Score List*, koje ćemo zasebno proučiti, komentirati te predložiti poboljšanja.



Slika 3.1 Padajući izbornik stranice Analytics

3.1. *Exam Analytics*

Exam Analytics pod-stranica je koja nam prikazuje rezultate svih ispita odabranog kolegija unutar željene akademske godine. Na stranici možemo primijetiti 4 vizualizacije - kutijasti grafikon (eng. *boxplot, box and whiskers plot*), tortni grafikon, prostorni grafikon (eng. *area chart*) te histogram. Način navigacije te korištenja stranice je sljedeći: klikom na željeni pravokutnik kutijastog dijagrama odabiremo provjeru znanja (ili bilo kakvu drugu zadaću/vježbu) za koju želimo vidjeti dodatne podatke. Klikom na pravokutnik učitavaju se ostale 3 vizualizacije. Slika 3.2 pokazuje nam raspored i izgled stranice.



Slika 3.2 Izgled stranice *Exam Analytics* nakon odabira ispita

3.1.1. *Test Pass Percentage*

Ovaj tortni grafikon pokazuje nam postotak prolaznosti studenata na provjeri. Ovo je jedan od najboljih primjera kako pravilo iskoristiti ovakav grafikon – broj kategorija najmanji je moguć (prolaz i pad) te time olakšava percepciju njihovog odnosa. Na samim odjeljcima ispisan je točan postotak koji taj odlomak predstavlja što nam također pomaže u interpretaciji podataka.

Grafikon je lijepo animiran te se učitava kružnim pokretima odjeljaka. Prelaskom miša preko njegovih „komada“ oni se povećaju te posvijetle, što vizualizaciju čini zanimljivijom korisniku.

Mana ove interakcije je da ona ne otkriva nikakve nove informacije o grafikonu. Bilo bi korisno kada bismo vidjeli točan broj studenata koji su pali, a koji prošli provjeru, u obliku info-oblačića (eng. *tooltip*) na način kako je to izvedeno u ostalim vizualizacijama. Uz to, analiza navika korisnika (ponajviše nastavnika i asistenata) otkriva nam da puno njih nikada ne definira prag na svojim ispitima prilikom njihove izrade. Tada ova vizualizacija gubi potpuni značaj jer će biti ili potpuna zelena ili potpuno crvena.

Treća mana grafikona pozicija je legende – nalazi se sa strane. Jedna od glavnih odlika tortnog grafa njegova je jednostavnost koja nam omogućava, pogotovu u ovakvim situacijama, da legendu stavljamo direktno na graf. Tako sprječavamo pogledavanje lijevo-desno od grafikon do legende.

3.1.2. Prostorni graf stanja ispita

Drugi graf s lijeve strane prostorni je graf koji nam pokazuje na koji način se tijekom vremena mijenjaju stanja u kojima se nalaze specifične instance ispita. Instanca ispita predstavlja studentov ispit kojeg rješava – koliko studenata je pristupilo ispitu, toliko imamo instanci.

Graf prikazuje 3 površine:

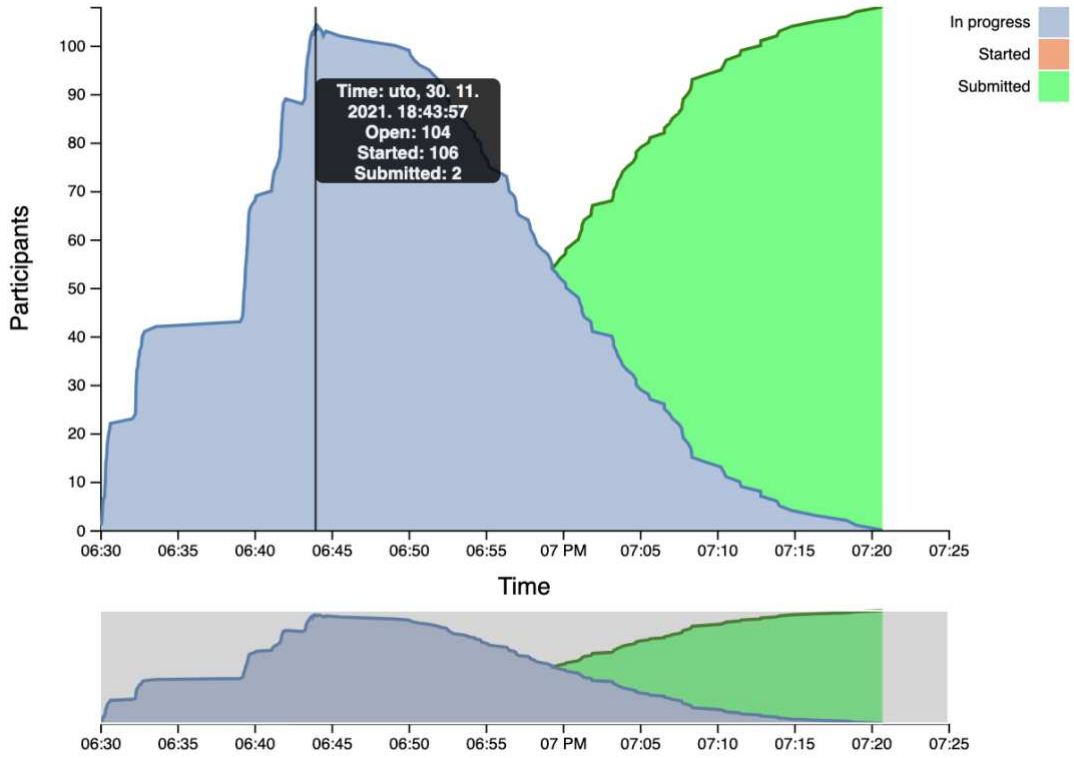
1. *Started*
2. *In Progress*
3. *Submitted*

Sumbitted nam pokazuje koliko studenata je ispit predalo, *In Progress* koliko studenata još uvijek rješava ispit (što znači da su ga pokrenuli, no da je vrijeme predaje nedefinirano) te *Started* koliko je ukupno studenata pristupilo provjeri, tj. pokrenulo ispit. Taj broj nikada se ne smanjuje te je ekvivalentan zbroju predanih ispita i ispita koji još uvijek traju.

Grafikon je interaktivne prirode te korisniku omogućava približavanje i micanje po vremenskoj osi, smanjujući i povećavajući rezoluciju nje same. Na taj način moguće je promatrati vrlo specifične trenutke.

Graf je lijepo izveden te nam omogućuje grubu procjenu težine ispita. Ako se broj studenata koji su predali ispit krene strmo povećavati pred sam kraj pisanja, i broj studenata koji nisu nikada službeno predali nije zanemariv, to je dobar pokazatelj da bi ispit bilo dobro provjeriti i ponovno procijeniti njegovu složenost ili vremensku zahtjevnost. S druge strane, ako već na pola vremena pisanja ispita pola studenata predaje ispit, možemo se zapitati je li ispit dovoljno izazovan ili je pak u igri neka druga varijabla (ako kolokvij nema prag, a student je skupio dovoljno bodova za prolaz, motivacija za pisanjem ispita se gubi). Slika 3.3 pokazuje izgled grafa za MI (međuispit) predmeta WEB2 (Napredni razvoj programske potpore za Web). Na njemu je vidljivo da su svi studenti ispit predali prije kraja te da su neki ispit predali već nakon 20 minuta pisanja.

Osim toga, nomenklatura info-oblačića odudara od legende grafikona. Na njemu nailazimo na nazive *Open*, *Started* i *Submitted* kategorija, dok se na legendi nalaze nazivi *In Progress*, *Started* i *Submitted*. To čini graf zbunjujućim i nezgodnim za shvatiti prilikom prvog susreta s njim.



Slika 3.3 Prostorni graf MI-a kolegija *Napredni razvoj programske potpore za Web* iz 2021. godine

Grafikon ima mogućnost označavanja specifičnih dijelova koje želimo prikazati označavanjem *toggle* gumba s njegove lijeve strane. Pozicija tih gumba nije dobra – teško je shvatiti da su vezani za prostorni graf, već izgleda kao da su vezani uz tortni graf zbog toga što se nalaze direktno iznad njega.

Zbog preklapanja površina bilo bi idealno da je smanjenja njihova neprozirnost (eng. *opacity*) vrijednost kako bi se u svakom trenu sve površine bile u cijelosti vidljive.

Korisni dodatak grafu bio bi način da povežemo vrijeme predaje s ostvarenim rezultatom. Tako bi mogli vidjeli jesu li studentu koji su brzo predali dobro riješili ispit.

3.1.3. Histogram rezultata

Histogram je odličan grafikon za prikaz raspodjela. Graf također prikazuje 2 vertikalne linije koje predstavljaju medijan (eng. *median*) te srednju vrijednost (eng. *average*).

Stupcima su pridijeljene boje gradijenta koji s lijeva na desno prelazi iz crvene u zelenu boju. Ovo je odlična ideja jer je gledatelju odmah jasno koje rezultate smatramo dobrim, a koje lošim.

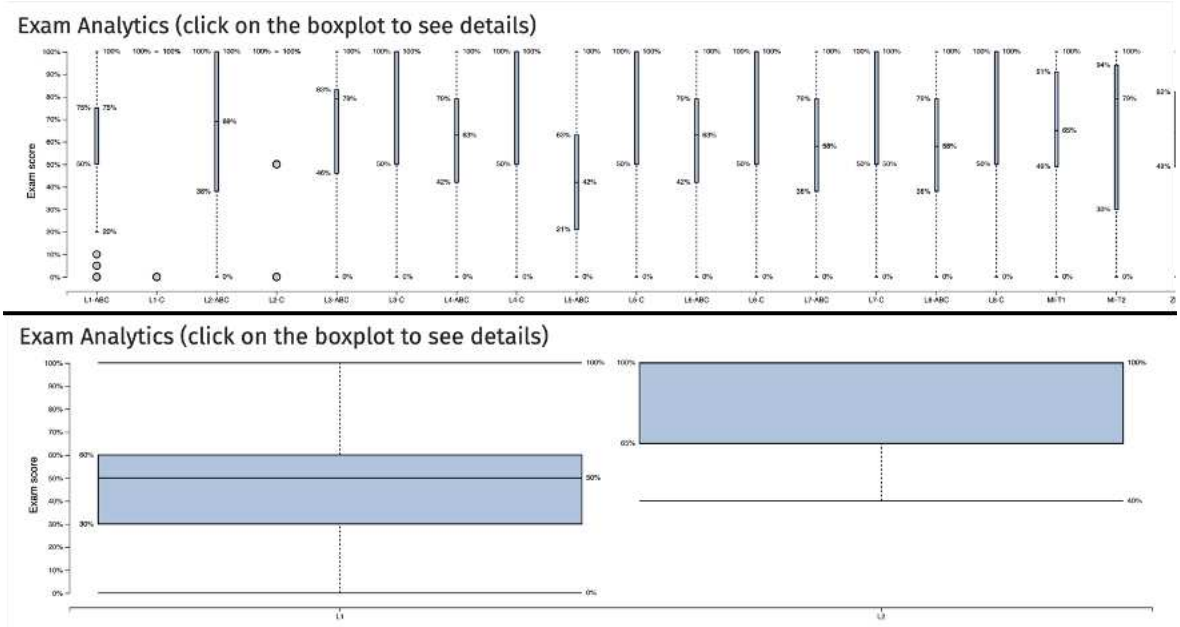
Ovaj graf relativno je redundantan zbog postojanja tehnički naprednijeg i bolje izvedenog histograma na sljedećoj stranici koju ćemo analizirati. Ovom histogramu nedostaje mogućnost mijenjanja rezolucije na x osi. Također, kada prikazujemo manji skup podataka graf nije vizualno privlačan,

3.1.4. Kutijasti dijagram ispita

Glavni dio *Exam Analytics* stranice upravo je sam kutijasti dijagram koji prikazuje sve ispite kroz akademsku godinu na tom kolegiju. Dijagram ima sve uobičajene elemente poput brkova (eng. *whiskers*), odudarajućih vrijednosti (eng. *outliers*) te samih interkvartlilnih raspona s medijanom. Prelaženjem mišem preko određenih stupaca dodatno nam pokazuje puno imena kolegija te njegov ID (eng. *identification*).

Reaktivnost grafa vrlo je bitan aspekt svake vizualizacije podataka koju koriste korisnici – nakon što se odabere detaljniji pregled rezultata klikom na jedan od stupaca, sve što se pokaže puni je naziv te provjere. Korisnički iskustvo bilo bi bolje da se također posvijetli odabrani pravokutnik ili da se na neki drugi način implicira na koju smo provjeru kliknuli.

Glavni nedostatak ovog grafa leži u njegovoj orijentaciji. Graf lista sve ispite horizontalno, s lijeva na desno, i kao takav je ograničen veličinom korisnikovog ekrana. Zbog toga pravokutnici koje koristi za prikaz interkvartilnog raspona nisu fiksne širine nego se mijenjaju ovisno o širini ekrana. Ovakav pristup dovodi nas do vrlo neujednačenih prikaza, što je vidljivo na grafici Slika 3.4.

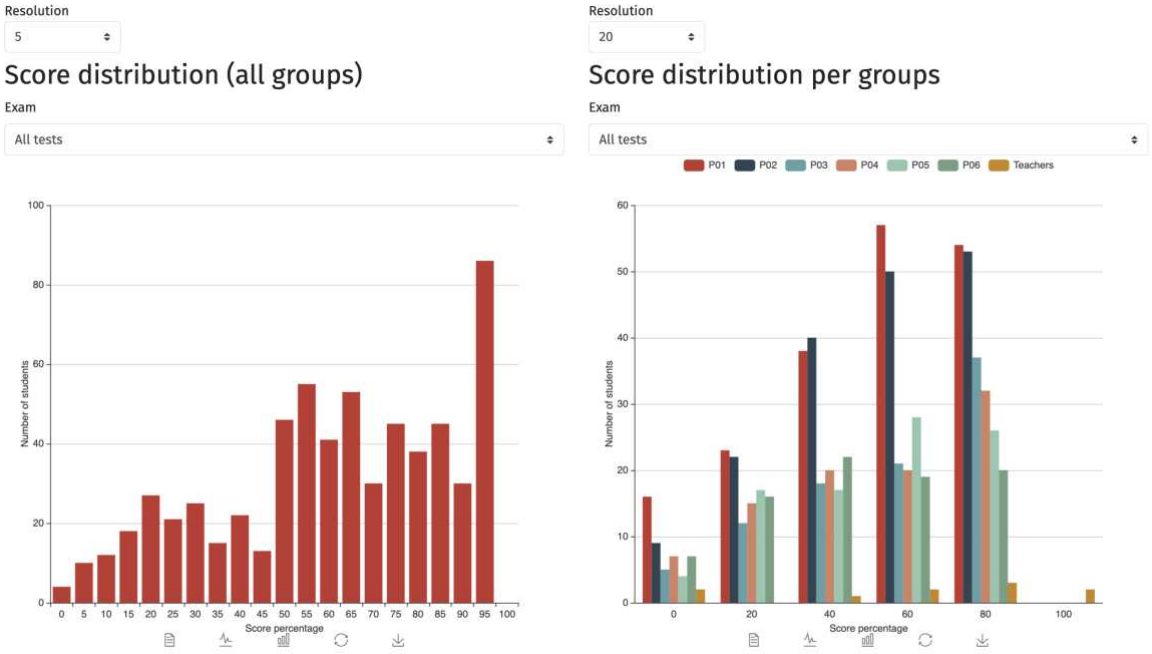


Slika 3.4 Primjer vizualnog odudaranja *boxplota* kolegija *Uvod u Programiranje* (gore) i *Algoritmi i Strukture Podataka* (dolje)

3.2. *Exam Score Distribution Analytics*

Ova stranica fokusira se na prikaz distribucije rezultata ispita. Slika 3.5 pokazuje nam dva grafikona. Lijevi prikazuje distribuciju rezultata svih studenata dok desni uzima grupe u obzir. Svaki od ovih grafikona može se prikazati u obliku tabličnog prikaza te kao linijski ili stupčasti grafikon. Tako možemo dobiti najbolje od svake od tih vizualizacija te nismo ograničeni na samo jednu.

Grafovi imaju drukčiju estetiku nego gore diskutirani, vidljivo je da su napravljeni s drukčijim alatima. Svaki ima mogućnost prilagođavanja rezolucije x osi te biranja specifičnih ispita čije rezultate želimo pokazati.



Slika 3.5 Grafovi stranice *Exam Score Distribution Analytics*

Grafovi slijede sve glavne tehnike oblikovanja vizualizacija te pružaju odlično korisničko iskustvo.

3.3. *Questions Analytics*

Ova stranica mijenja fokus na analizu individualnih pitanja. Po otvaranju stranice vidimo veliki dijagram raspršenosti. On nam pokazuje odnos između postotka riješenosti pitanja te broja studenata koji su to pitanje rješavali. Veličina točkice odgovara broju različitih pitanja koja su se našla na istoj poziciji u grafu.

Odabirom točkice pokazuju se dodatni podatci o tom pitanju te imamo mogućnost odlaska na stranicu za analizu tog specifičnog pitanja. Slika 3.6 pokazuje stanje stranice nakon odabira pitanja.

Slika 3.6 Stranica *Questions Analytics*

Sama vizualizacija odlična je te nam omogućava brzo uočavanje pitanja koje odskaču od većine. Na taj način možemo se posvetiti tim iznimkama i proučiti zašto je došlo do odstupajućih rezultata.

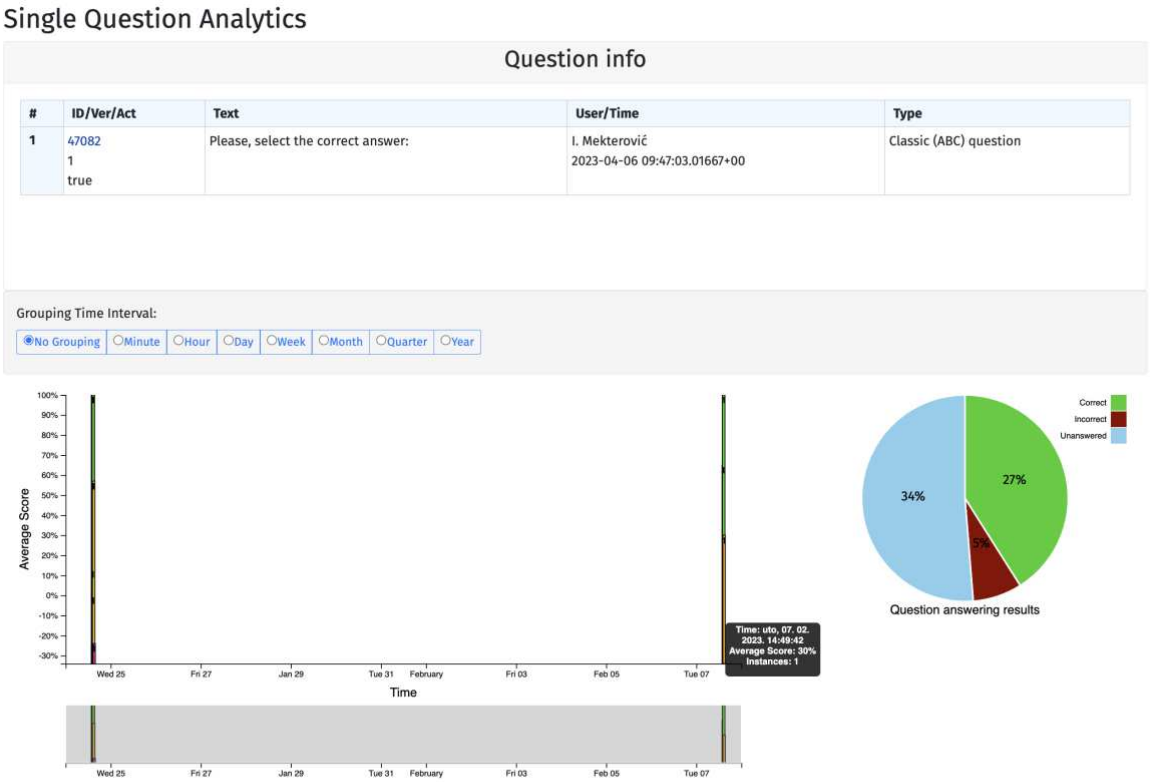
Kada se stranica prvi put prikaže, točkice su pobojane gradijentom koji prelazi iz zelene boje u crvenu kako riješenost pitanja opada, no odabirom načina filtriranja pitanja gradijent se obrne, što nam pokazuje Slika 3.7.

Slika 3.7 Izmijenjen gradijent nakon odabira filtriranja

No ni jedan od ove dvije vrste gradijenta ne prenosi korisniku postotak riješenosti kojemu se nadamo da će pitanja imati. Idealno, htjeli bi da svako pitanje ima riješenost između 40% i 70%. Zbog toga bi trebao biti korišten gradijent koji ide prema sredini.

3.3.1. *Single Question Analytics*

Klikom na gumb *Analytics* na dnu stranice preusmjereni smo na stranicu za analizu individualnih pitanja, kao što možemo vidjeti na grafici Slika 3.8.



Slika 3.8 Stranica za detaljniju analizu pojedinih pitanja

Na stranici vidimo stupčasti graf s pomičnom vremenskom trakom te tortni graf. Tortni graf ovdje se također ažurira mijenjanjem vremenskog isječka.

Stupčasti graf pokazuje nam kada se i koliko instanci ovog pitanja pokrenulo u sustavu Edgar, dok nam tortni grafikon predočuje riješenost.

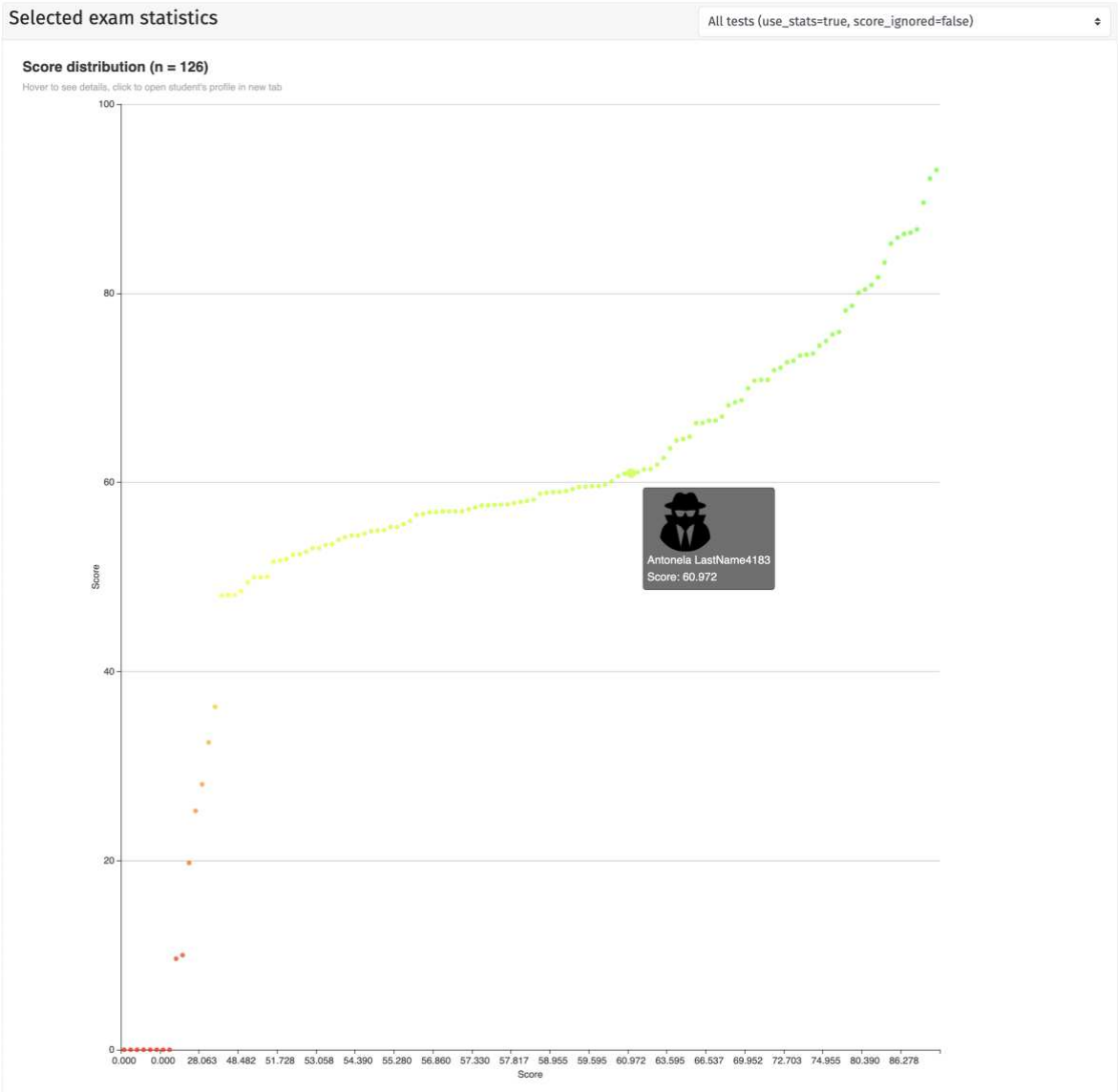
Problem s tortnim grafikonom je da nema odsječak za parcijalna rješenja. Ako pobliže pogledamo grafiku Slika 3.8 možemo primijetiti da zbroj postotaka na njemu nije 100%. Zbog toga se podatci pitanja u kojima postoji parcijalno rješenje pogrešno vizualiziraju.

3.4. *Students Scatter*

Na ovoj stranici možemo uočiti samo jedan raspršeni grafikon. Koristi se isti gradijent za bojanje točkica kao i u prijašnjim grafovima. Kao što Slika 3.9 prikazuje, na x i y osi imamo broj bodova koje su studenti skupili. U gornjem desnom kutu uočavamo padajući izbornik s kojim možemo odabrati koji ispit ulazi u prikaz bodova.

Bilo bi odlično uvesti mogućnost označavanja kombinacije ispita za koje želimo vidjeti zbrojene rezultate, umjesto da ih moramo odabrati ili sve ili samo jedan.

Obje osi prikazuju iste podatke te su označene sa *Score,* ali stope podataka na osima drukčije su. Y os uvijek prikazuje podatke od 0 do 100, gdje naznačuje svako povećanje od 20. S druge strane, x os adaptira se skupu podataka koje prikazuje te naznačuje povećanja različitih iznosa. Zbog ovog razloga grafikon nije kvadratnog oblika te može biti zbunjujući korisniku.

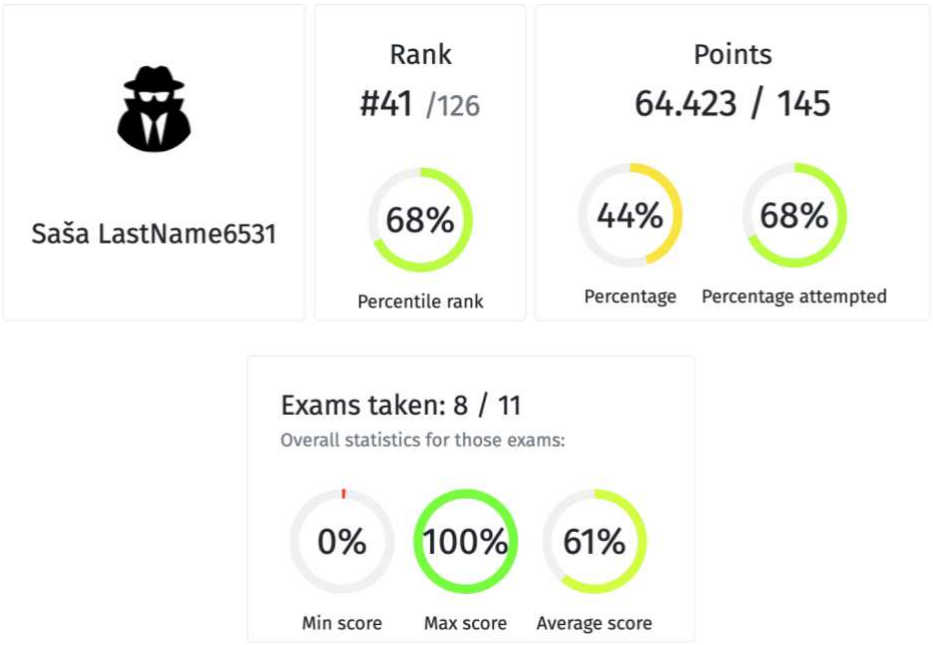


Slika 3.9 *Student Scatter* raspršeni grafikon za praćenje uspjeha studenata

Osim toga, vizualizacija je korisna te nam omogućuje brz pregled rubnih slučajeva ukupnog broja bodova studenata. Odabirom određene točkice preusmjereni smo na stranicu gdje saznajemo više o performansama studenta.

3.5. *Student Analytics*

Pri vrhu stranice imamo niz metrika koje rangiraju studenta unutar generacije. Možemo vidjeti njegov rang, broj bodova te količinu ispita koje je pisao, kao što je vidljivo na grafici Slika 3.10.



Slika 3.10 Rangiranje studenta po broju bodova te ostale statistike

Ove statistike lijep su pokazatelj studentima o njihovom trenutnom uspjehu u usporedbi s cijelom generacijom (ovaj pogled dostupan je i na studentskoj strani Edgara, ali student naravno pristupa samo svojem profilu).

Ispod generalnih statistika imamo detaljniji pregled riješenosti odabranog ispita te riješenosti svih dosadašnjih ispita što možemo vidjeti na grafici Slika 3.11.

Slika 3.11 Vizualizacija uspjeha studenta i generacije na ispitima na stranici *Student Analytics*

Prvi puta uočavamo funkcionalni spoj dvije vrste grafova: stupčasti i linijski. Na ovaj način imamo detaljan pregled uspjeha na svakom individualnom pitanju ili ispitu, dok pomoću linijskih grafova pratimo kako se s tim rezultatima mijenja naš rang i prosjek bodova.

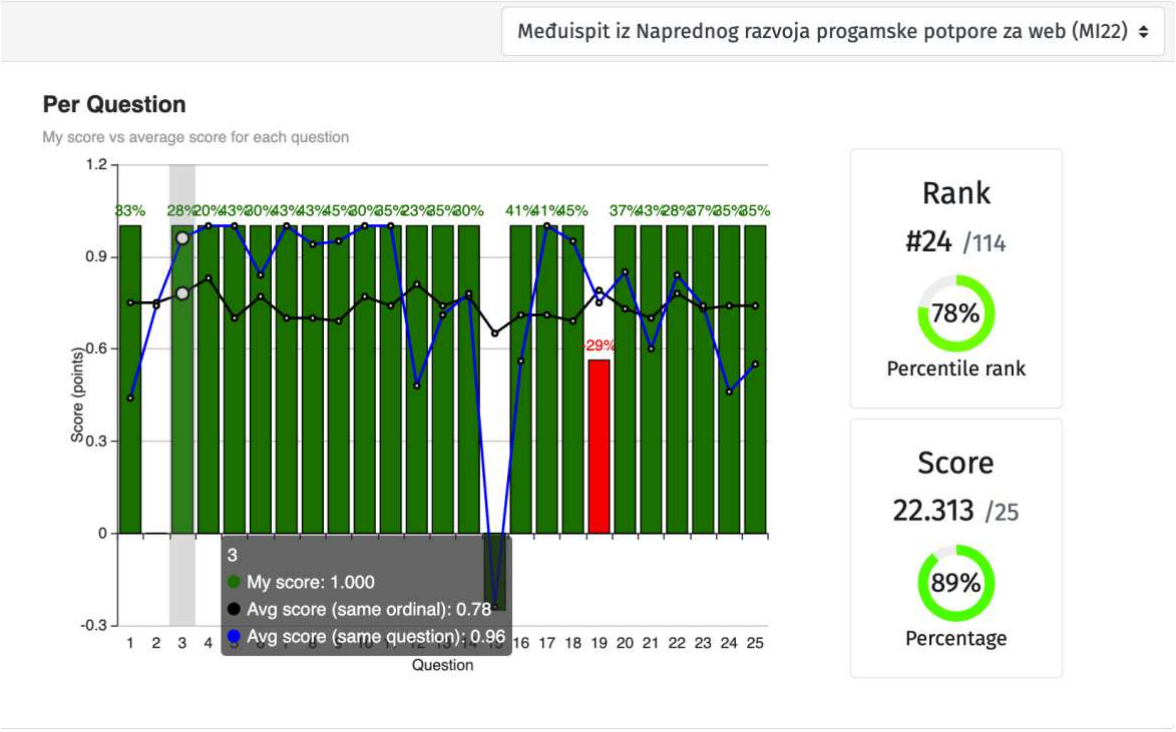
Prelaskom miša preko stupaca dobijemo sažeto napisane iznose oba linijska te stupčastog grafikona, što pomaže odrediti njihov odnos te dodatno pojašnjava cijeli grafikon.

Na donjem grafikonu moguće je odabrati koji linijski graf želimo prikazati, što na gornjem nije. Pošto su grafovi identični, oba bi trebala imati takvu funkciju.

Problem se pojavi kada na manjem ekranu selektiramo ispit koji ima veći broj pitanja, kao što je vidljivo na grafici Slika 3.12, iako nam čak i u tom slučaju info-oblačić s konkretnim podatcima omogućuje snalaženje.

Svaki stupac na svojem vrhu ima postotak, no taj postotak ne označuje broj bodova koji on predstavlja, već za koliko je studentov broj bodova bolji ili lošiji od prosjeka. Ovo je zanimljiva ideja s prilično lošom implementacijom, jer je teško shvatiti što ti brojevi znače.

U najmanju ruku bilo bi dobro staviti objašnjenje sa strane da korisnici znaju što se događa te da ne moraju nagađati.



Slika 3.12 Izgled grafikona na *Student Analytics* stranici kada ga promatramo na manjem ekranu

3.6. *High Scores List*

Konačno, imamo listu studenta poredanih po trenutnom uspjehu na predmetu. Ovdje se pokazuju isti podatci kao i na prethodno raspršenom dijagramu, no u standardnom, preglednom formatu tablice.

4. Rješenje

Rješenje predstavlja lokalnu implementaciju *Exam Analytics* stranice te rješava neke od prije spomenutih problema.

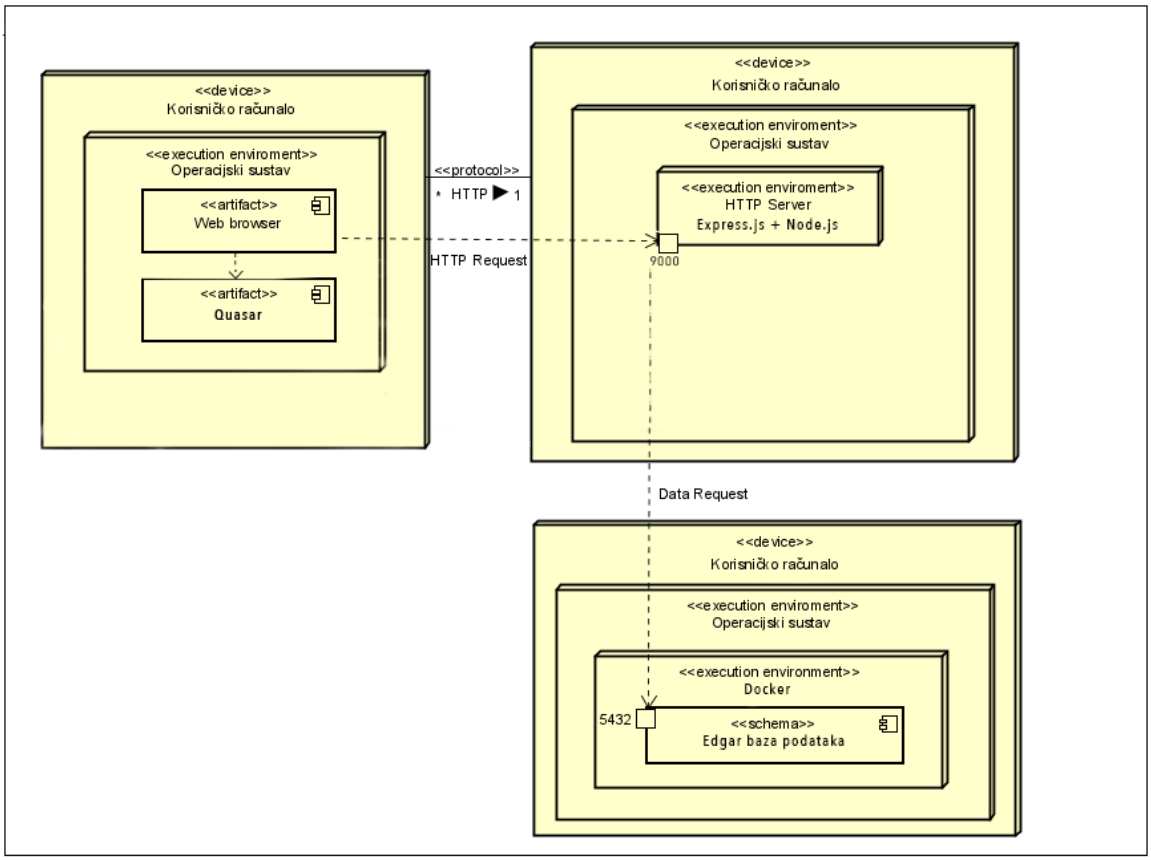
4.1. Model

Korisničko sučelje izrađeno je pomoću radnog okvira *Quasar* i jezika *Vue.js,* dok je pozadinski servis napisan pomoću *Javascript* programskog jezika koji se izvan web preglednika pokreće pomoću *Node.jsa*. *Također* koristimo *Express.js*, radni okvir za razvoj web-aplikacija napravljen upravo za *Node.js.*

Pozadinski servis slijedi principe REST (eng. *Representational state transfer*) API-a (eng. *Application programming interface*). On je u suštini podskup servisa koji koristi Edgar s nekoliko novih krajnjih točaka (eng. *endpoint*).

Za bazu podataka (eng. *database*) koristimo *PostgreSQL* relacijsku bazu podatka (eng. *relational database*). Korištena je Edgarova baza podataka i to je jedina direkta poveznica s projektom Edgar, odnosno ovaj rad izveden je kao zasebna, samostojeća aplikacija.

Slika 4.1 prikazuje dijagram koji opisuje strukturu lokalne implementacije.



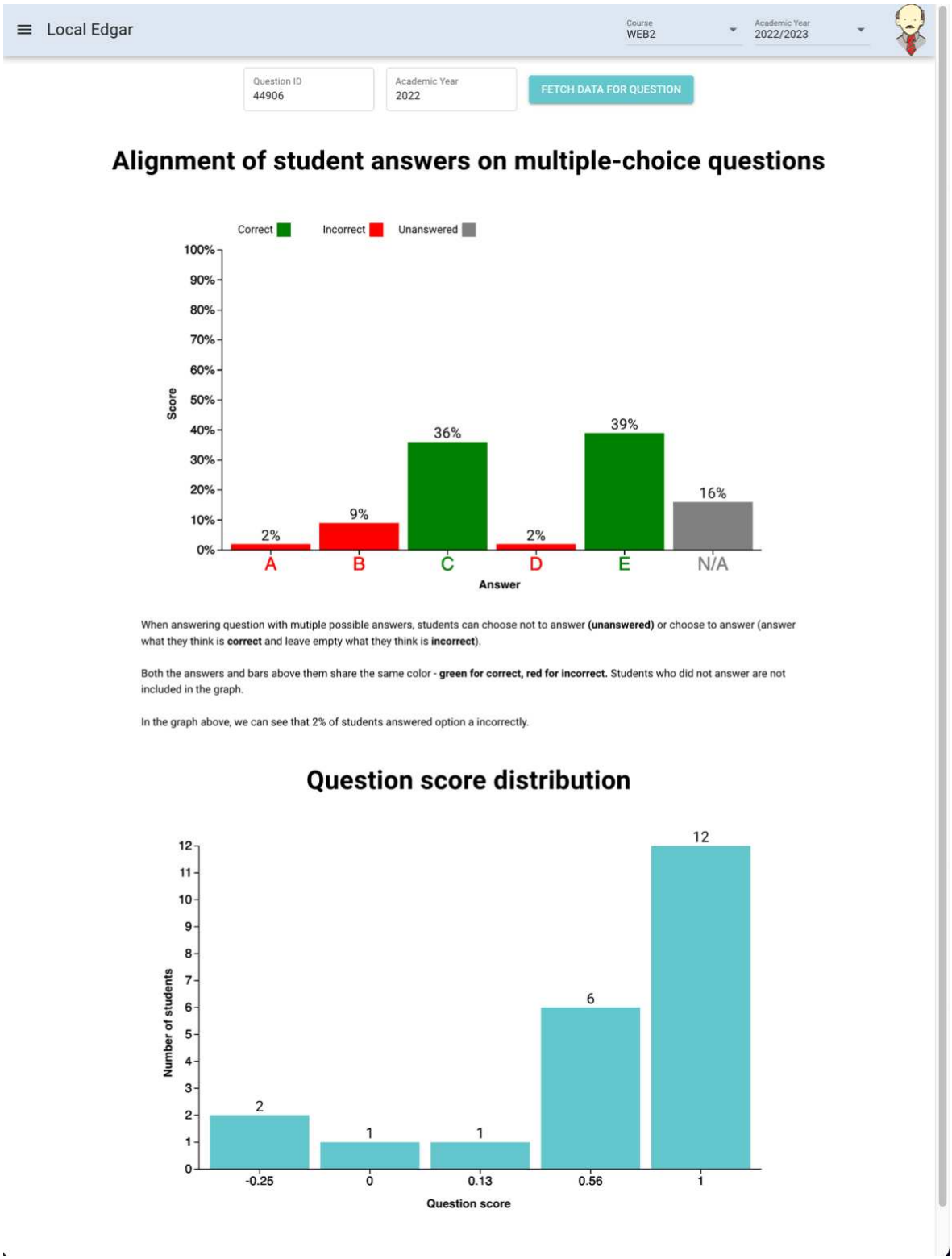
Slika 4.1 Arhitektura rješenja

4.2. Analiza implementiranih vizualizacija

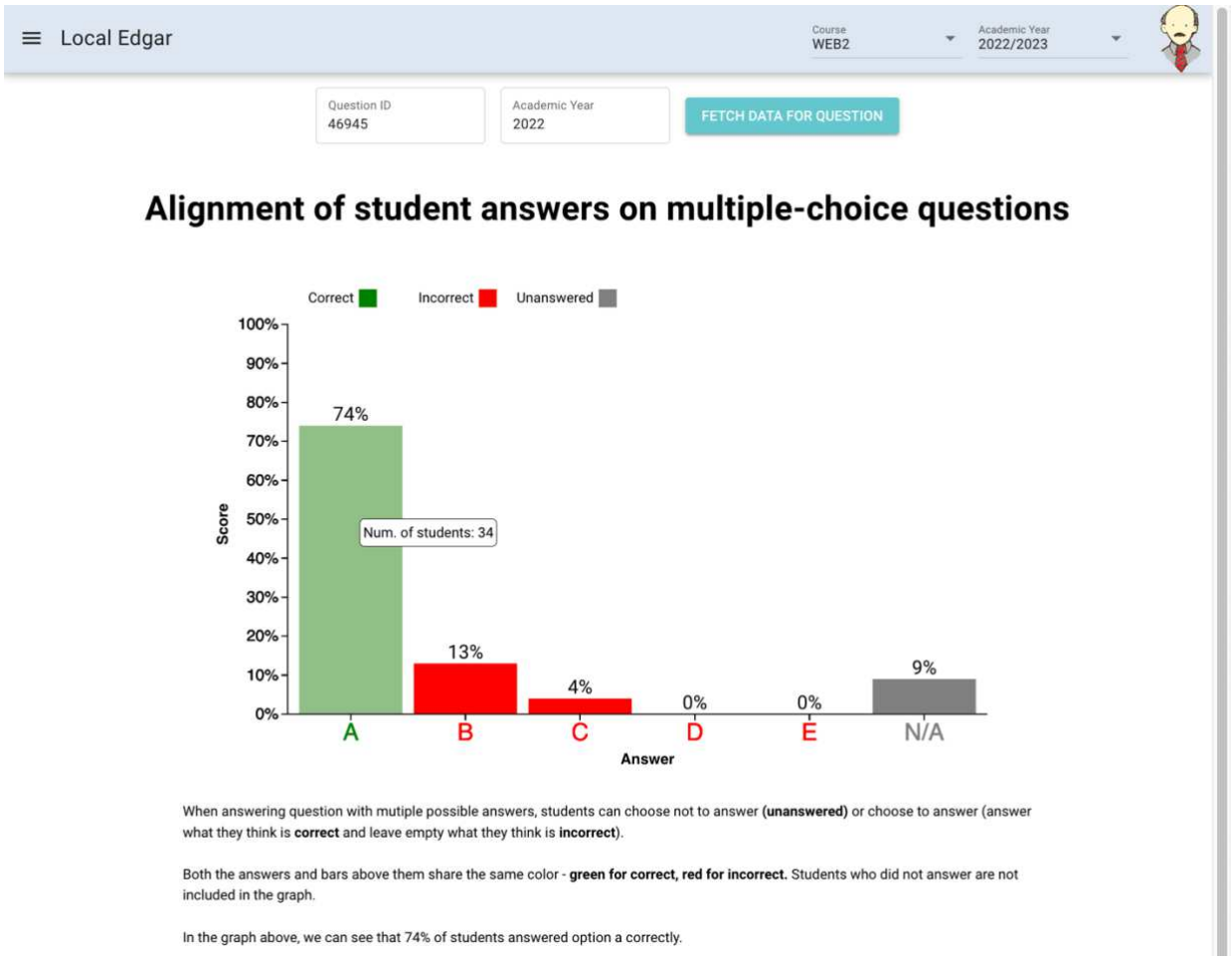
Rješenje obuhvaća dvije stranice – *Individual Question Analytics* i *Exam Analytics*. Obje stranice proučit ćemo i komentirati u nastavku.

4.2.1. Implementacija stranice *Individual Question Analytics*

Svrha ove stranice analiza je odgovora na specifične ABCD pitalice s jednim ili više točnih odgovora. Kao što možemo vidjeti na grafici Slika 4.2, stranica sadrži dvije vizualizacije: *Alignment of student answers on multiple-choice questions* i *Question score distribution*. Prva vizualizacija govori nam koliko studenata je pogrešno ili točno odgovorilo na svaki od ponuđenih odgovora. Slika 4.3 prikazuje primjer pitalice s jednim točnim odgovorom.



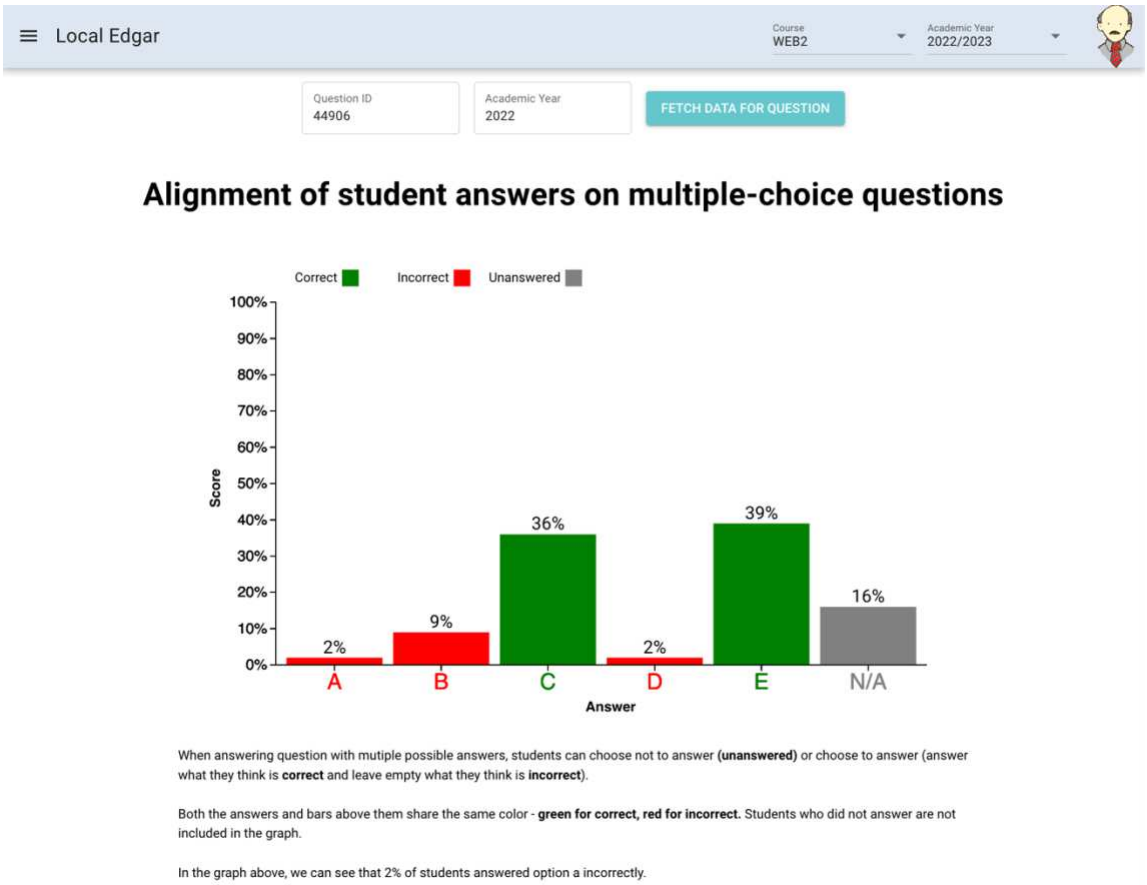
Slika 4.2 Izgled stranice *Individual Question Analytics*



Slika 4.3 Primjer grafikona za pitalicu s jednim točnim odgovorom

U računanje postotaka uzimali smo u obzir sve studente koji su pristupili pitanju te je zbog toga zbroj stupaca u stupčastom grafikonu jednak 100%. Sivi stupac označen sa 'N/A' odgovara postotku studenata koji nisu odgovorili na pitanje. Možemo primijetiti kako stupci i oznake na x osi dijele istu boju koja označava je li taj odgovor točan, netočan ili student nije odgovorio na pitanje.

Kod pitalica s više ponuđenih točnih odgovora logika je ista, jedino što zbroj postotaka stupaca neće biti jednak 100% zbog mogućnosti zaokruživanja više odgovora, što je vidljivo na grafici Slika 4.4.



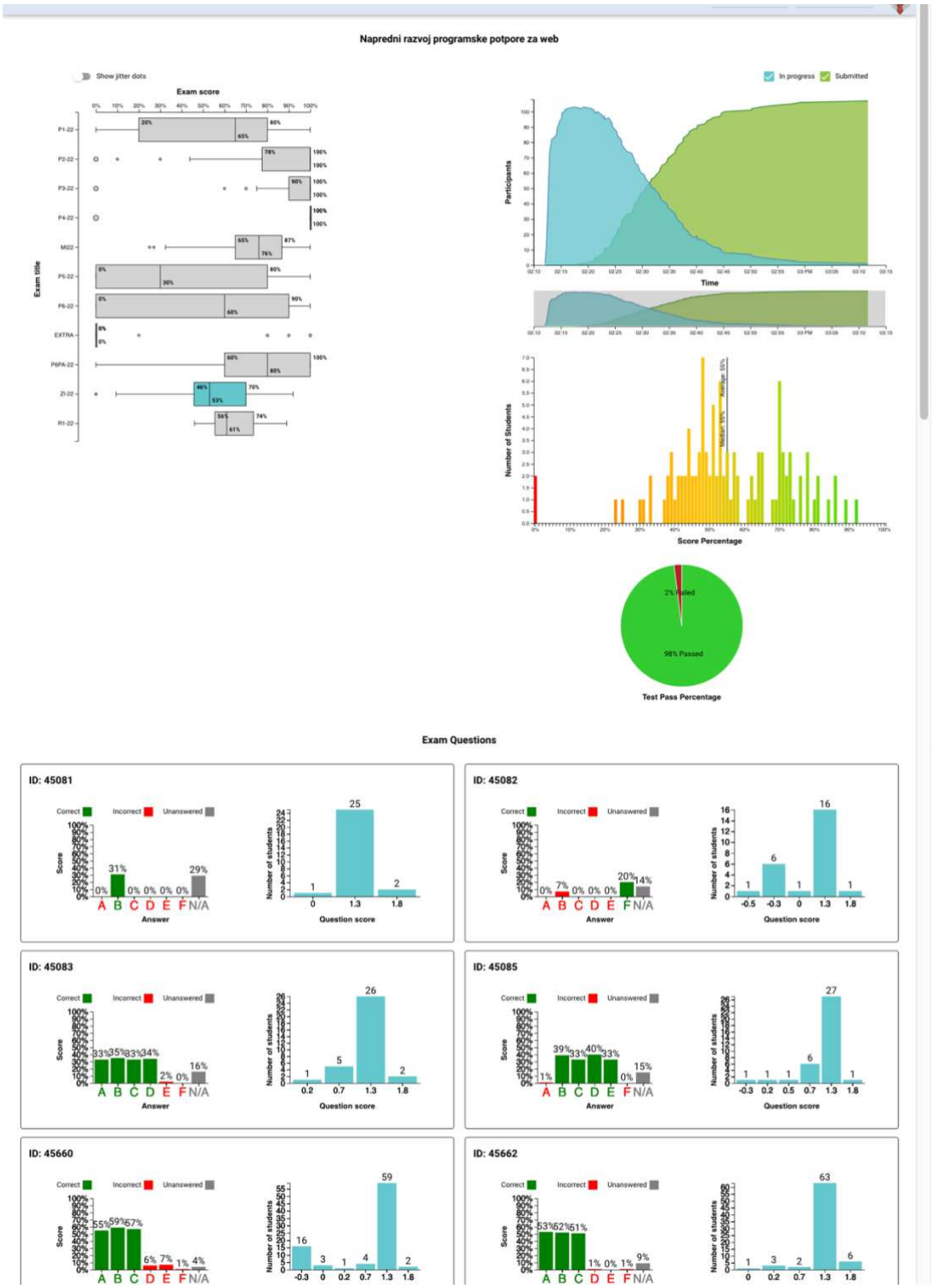
Slika 4.4 Primjer grafikona za pitalicu s više točnih odgovora

Drugi grafikon, *Question score distribution*, pokazuje nam koliko studenata je ostvarilo određeni broj bodova na pitanju. Prednost ovog grafikona je što je primjenjiv na sva pitanja, a ne samo ABCD pitalice, što ćemo iskoristiti na *Exam Analytics* stranici.

4.2.2. Implementacija stranice *Exam Analytics*

Exam Analytics sadržajno je vrlo sličan istoimenoj stranici na Edgaru, no dizajn je osvježen.

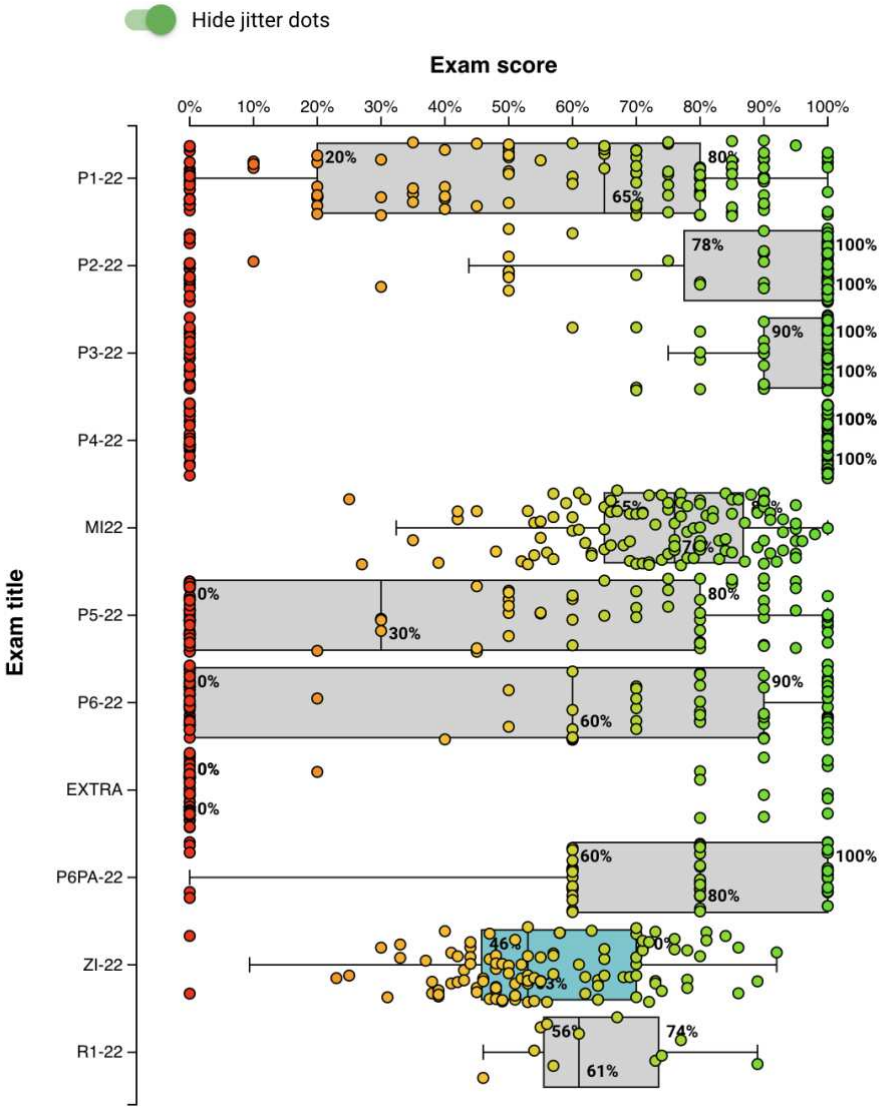
Slika 4.5 pokazuje nam izgled stranice kada odaberemo neku provjeru na *boxplotu*.



Slika 4.5 Izgled lokalne implementacije stranice Exam Analytics

Boxplot koji pokazuje sve provjere okrenut je vertikalno kako bi mogli postići fiksnu visinu individualnih pravokutnika. Ovakav dizajn omogućava puno veću konzistentnost između prikaza raznih kolegija te nam daje veću kontrolu nad općenitim izgledom. Odabrana provjera također je osjenčana.

Jedan od većih nedostataka *boxplota* kao grafikona udaljavanje je od stvarne raspodjele podataka. *Outliers* i *whiskers* pomažu u stjecanju boljeg dojma o skupu podataka (eng. *dataset*) ali nekad ni oni nisu dovoljni. Redizajnirani *boxplot* ima još jedan pogled, a to je pogled s prikazanim raspršenim podatcima (eng. *jitter dots*) koji je prikazan na grafici Slika 4.6.

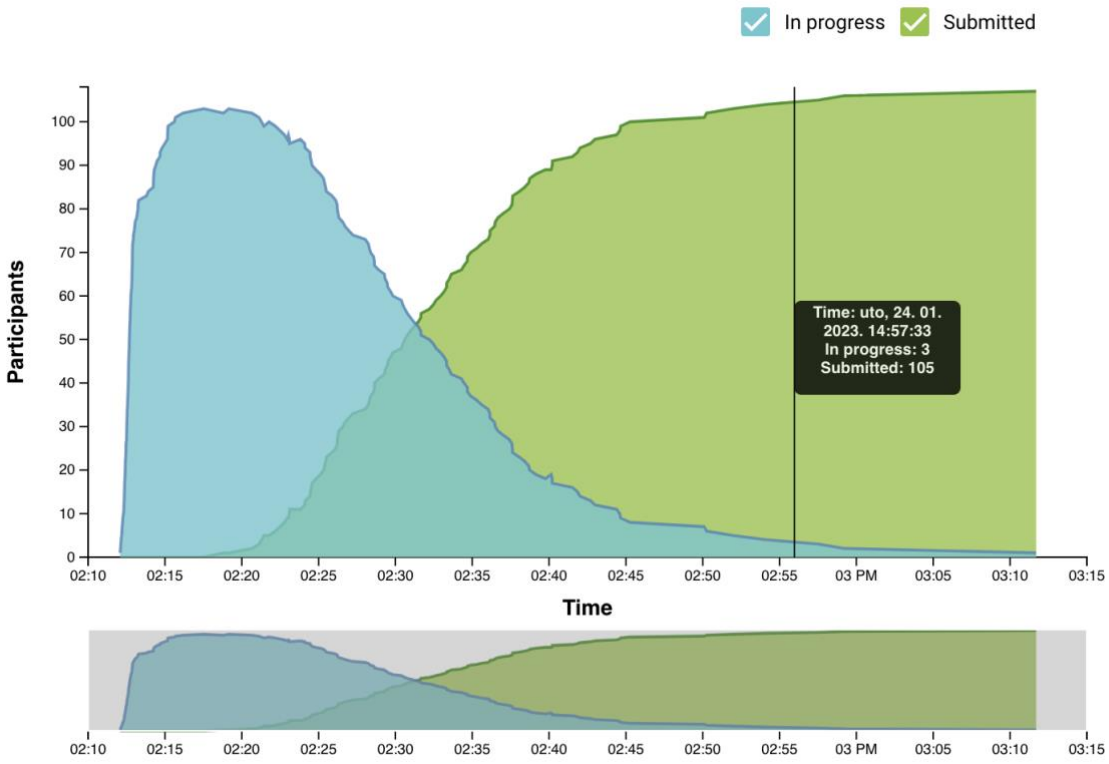


Slika 4.6 *Show jitter dots* pogled *boxplota*

Svaka točkica predstavlja jednu instancu ispita. Točkice su razmještene unutar visine zasebnih pravokutnika s određenim šumom kako bi bilo lakše raspoznati točnu količinu.

Prostorni dijagram za praćenje aktivnosti studenata tijekom ispita ostao je većinom nepromijenjen. Svaka površina sada ima smanjenu neprozirnost (eng. *opacity*) kako bi bilo lakše pratiti što se događa kada prikazujemo više površina. Uklonjena je cijela *Started*

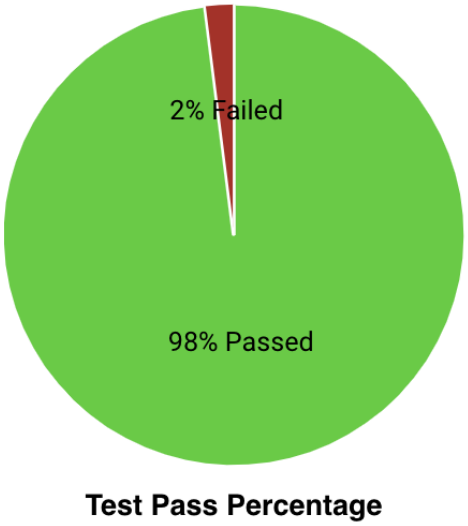
površina, usklađeno nazivlje info-oblačića te legenda pojednostavljena, kao što je vidljivo na grafici Slika 4.7.



Slika 4.7 Prikaz dorađenog prostornog grafikona implementirane *Exam Analytics* stranice

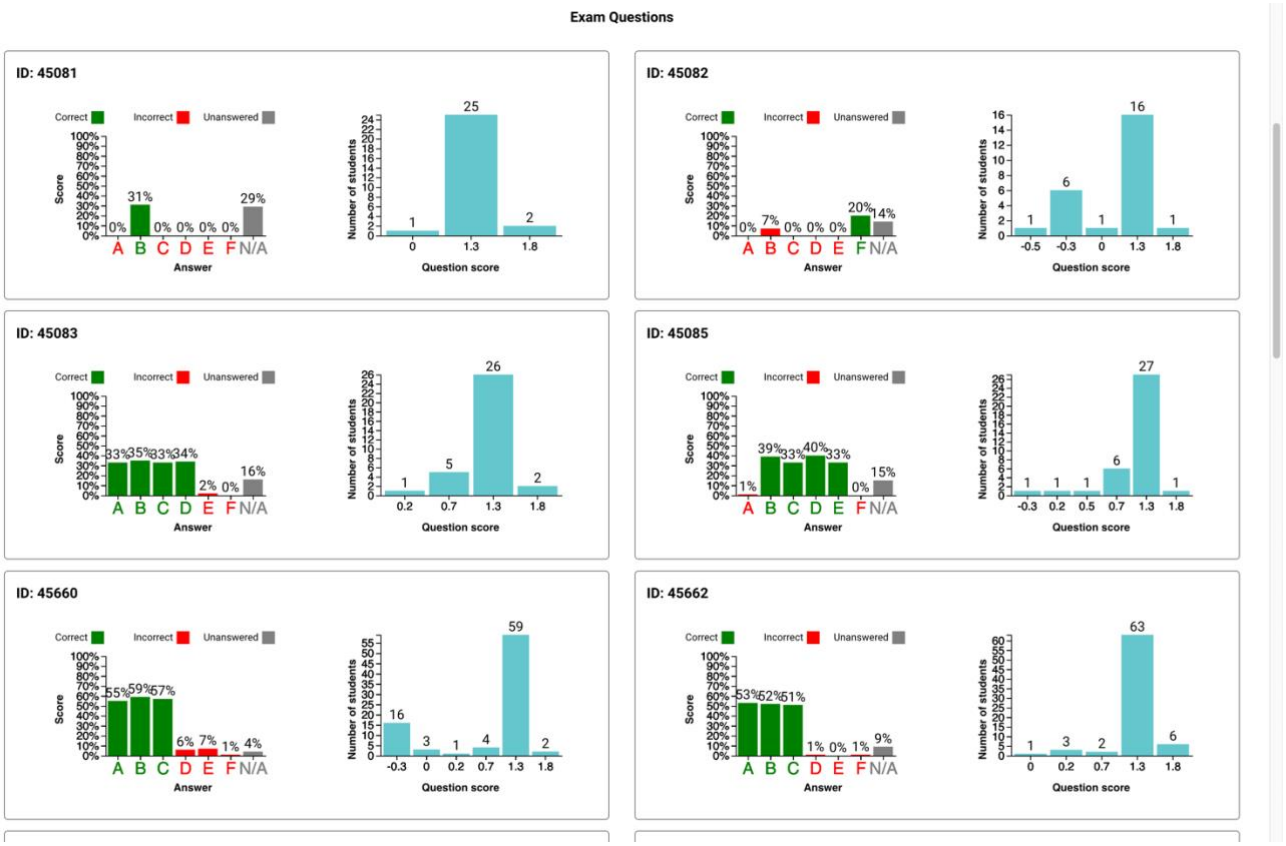
Histogram je ostao identičan, jedina promjena je povećanje njegove širine zbog naknadnog prostora što se oslobodio na ekranu uvođenjem novog poretka elemenata. Sada su stupci malo širi i izgledom privlačniji, iako rješenje je daleko od idealnog.

Jedina promjena kod tortnog grafa micanje je legende koja je sada ukomponirana u sam grafikon. Tako smo smanjili nepotreban *chartjunk* te omogućili osobama s problemom raspoznavanja boja čitanje grafikona. Promjene su vidljive na grafici Slika 4.8.



Slika 4.8 Tortni grafikon na implementiranoj stranici *Exam Analytics*

Exam Questions novi je dodatak stranici. Prikazujemo *Question score distribution* stupčasti grafikon svakog pitanja koje se pojavilo na odabranoj provjeri. Uz to, za svaku ABCD pitalicu također prikazujemo i *Alignment of student answers on multiple-choice questions* stupčasti dijagram. Tako omogućavamo brzi pregled uspješnosti odgovaranja studenata na pitanja te laganu identifikaciju najlošije riješenih pitanja. Uz svako pitanje prikazan je i njegov ID pomoću kojega ga lagano možemo pronaći u sustavu te detaljnije analizirati. Slika 4.9 pobliže pokazuje *Exam Questions* dio stranice.



Slika 4.9 *Exam Questions* dio implementirane stranice *Exam Analytics*

5. Zaključak

U informacijskom sustavu Edgar postoje brojne vizualizacije koje koriste nastavnicima i studentima za analizu rezultata postignutih na ispitima, domaćim zadaćama te laboratorijskim vježbama. Osim pregleda rezultata Edgar omogućuje uvid u podatke o poretku pa čak i u podatke o ponašanju studenata prilikom pisanja ispita.

Kroz ovaj završni rad provedena je analiza postojećih vizualizacija te su predložena poboljšanja. Fokus je bio na *Exam Analytics* stranici te analizi individualnih pitanja. Sve implementacije pisane su u *Quasar* radnom okviru koristeći *D3.js* biblioteku za izradu vizualizacija.

Redizajnom Edgarove *Exam Analytics* stranice ispravljeni su neki od problema koje je prijašnja implementacija imala. Kutijasti grafikon sada puno bolje izgleda pri većim i manjim količinama podataka te pruža opciju promatranja stvarne raspodjele podataka. Općenito je stranica ljepša i nudi ugodnije korisničko iskustvo.

Exam Questions odjeljak koji prikazuje grafikone individualnih pitanja vrlo je funkcionalan dodatak, no usporava stranicu kada je u pitanju velik broj pitanja – za njega bi trebalo implementirati paginaciju (eng. *pagination*) kako se ne bi narušilo korisničko iskustvo.

Individual Questions Analytics stranica omogućuje puno detaljniju analizi individualnih pitanja na ispitu, što može pomoći nastavnicima u korigiranju težine provjera. Mana trenutne implementacije nepovezanost je ponuđenih odgovora s njihovim tekstom te s tekstom samog pitanja. Zbog toga nastavnik, nakon što je uočio problematično pitanje, treba otići na njegov pregled na potpuno drukčiji dio Edgara. To narušava korisničko iskustvo te usporava proces analize.

D3.js kao biblioteka vrlo je zahtjevna za učenje i korištenje no bila je dobar odabir za ovaj rad zbog fleksibilnosti koju pruža. Bilo je moguće napraviti svaku željenu promjenu te nije bilo potrebno raditi kompromise.

Literatura

[1] Tableau Software, What Is Data Visualization? Definition, Examples, And Learning Resources. Poveznica: https://www.tableau.com/learn/articles/data-visualization; Pristupljeno: 13. lipnja 2023.

[2] Filip Tobiasz, Using Vue: Pros and Cons, (2022, travanj). Poveznica: https://thecodest.co/blog/pros-and-cons-of-vue/; Pristupljeno: 13. lipnja 2023.

[3] Mike Bostock, D3 Data-Driven Documents, (2021), Poveznica: https://d3js.org/; Pristupljeno: 13. lipnja 2023.

[4] Tufte, Edward R., The Visual Display of Quantitative Information. Cheshire: Graphics Press, 2001.

[5] Mala Deep, Little Known Ways to Make your Data Visualization Awesome, (2020, listopad). Poveznica: https://towardsdatascience.com/little-known-ways-to-make-your-data-visualization-awesome-890d71b6e365; Pristupljeno: 13. lipnja 2023.

[6] Rajesh Sigdel, Improve Your Visualization Skills Using Tufte's Principles of Graphical Design, (2020, rujan). Poveznica: https://medium.com/nightingale/improve-your-visualization-skills-using-tuftes-principles-of-graphical-design-3a0f40a53a2c; Pristupljeno: 13. lipnja 2023.

[7] In, Junyong & Lee, Sangseok. (2017). Statistical data presentation. Korean Journal of Anesthesiology. 70. 267. 10.4097/kjae.2017.70.3.267.

[8] Maarten Grootendorst, Misleading Graphs …and how to fix them!, (2021, ožujak). Poveznica: https://towardsdatascience.com/misleading-graphs-e86c8df8c5de; Pristupljeno: 14. lipnja 2023.

[9] Wikipedia, Single-page application (2023, lipanj). Poveznica: https://en.wikipedia.org/wiki/Single-page_application; Pristupljeno: 17. lipnja 2023

[10] Yan Holtz, Most basic barplot in d3.js (2018). Poveznica: https://d3-graph-gallery.com/graph/barplot_basic.html; Pristupljeno: 18. lipnja 2023.

Jednostranična web-aplikacija za vizualizaciju podataka u sustavu Edgar

Sažetak

Ovaj rad bavi se analizom i unaprjeđenjem vizualizacija podataka u informacijskom sustavu Edgar. Cilj rada je proučavanje postojećih vizualizacija, istraživanje osnovnih načela vizualizacije podataka i predlaganje poboljšanja i novih vizualizacija, koristeći tehnologije poput radnog okvira *Quasar* i *Vue.js* te biblioteke *D3.js*. Cilj je pružiti nova saznanja i smjernice koje mogu unaprijediti vizualizacije unutar Edgarovog informacijskog sustava, pružajući korisnicima bolje iskustvo i omogućujući lakše korigiranje i prilagođavanje ispitnih materijala.

Ključne riječi:

Web aplikacija, Vizualizacija podataka, Edgar, SPA, Vue.js, Quasar, D3.js

Single page web application for data visualization in the Edgar system

Abstract

This paper deals with the analysis and improvement of data visualizations in the Edgar information system. The aim of the paper is to study existing visualizations, explore the basic principles of data visualization, and propose improvements and new visualizations, using technologies such as the Quasar framework, Vue.js, and the D3.js library. The goal is to provide new insights and guidelines that can enhance visualizations within Edgar's information system, offering users a better experience and enabling easier correction and adaptation of exam materials.

Keywords

Web application, Data visualization, Edgar, SPA, Vue.js, Quasar, D3.js