Web aplikacije za generiranje i potpisivanje PDF dokumenata

Markovčić, Stjepan

Sveučilište Jurja Dobrile u Puli

Fakultet Informatike

STJEPAN MARKOVČIĆ

Web aplikacija za generiranje i potpisivanje PDF dokumenata

Diplomski rad

Pula, Rujan, 2022. godine

Sveučilište Jurja Dobrile u Puli

Fakultet Informatike

STJEPAN MARKOVČIĆ

Web aplikacija za generiranje i potpisivanje PDF dokumenata

Diplomski rad

JMBAG: 0303075497, redoviti student

Studijski smjer: Informatika

Kolegij: Izrada informatičkih projekata

Mentor: doc. dr. sc. Nikola Tanković

Pula, Rujan, 2022. Godine

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani ***Stjepan Markovčić***, kandidat za magistra ***Informatike*** ovime izjavljujem da je ovaj Diplomski rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Diplomskog rada nije napisan na nedozvoljeni način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

_____

U Puli, 14. Rujna 2022. godine

IZJAVA O KORIŠTENJU AUTORSKOG DJELA

Ja, ***Stjepan Markovčić*** dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj diplomski rad pod nazivom ***Web aplikacija za generiranje i potpisivanje PDF dokumenata*** koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama. Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, 14. Rujna 2022. godine

Potpis

_____

Sadržaj

1.Uvod

Bez obzira na sve veću digitalizaciju raznih procesa, još uvijek postoji velika potreba za popunjavanje razno raznih obrazaca. Kada se radi samo o jednom takvom obrascu nije toliki problem, ali ako se radi o više obrazaca(čak i samo 10) brzo dolazimo do problema.

Upravo taj se problem javljao na kolegiju „Stručna praksa“ gdje je za svakog studenta koji se prijavio na praksu trebao biti popunjen takav obrazac s podacima studenta.

Template engine je nastao kao rješenje tom problemu. *Template engine* je aplikacija koja omogućuje definiranje predložaka za generiranje PDF dokumenata. Korisnici mogu učitati svoj PDF dokument putem web sučelja te na njemu označiti područje koje žele popuni s podacima. Korisnici će na izbor imati nekoliko alata za kreiranje PDF predložaka te upravljanje određenim aspektima predložaka.

Kroz ovaj diplomski rad, opisat ćemo mogućnosti aplikacije, kako upravljati aplikacijom, korištene tehnologije, implementaciju aplikacije te slična programska rješenja koja rješavaju ovakvu vrstu problema.

Web aplikacije su danas češće korištene nego ikada jer korisnicima uvelike olakšavaju korištenje određenih usluga. Za razliku od desktop ili mobilnih aplikacija, web aplikacije nije potrebno instalirati da bi se koristile već je sve zadržano u internet pregledniku korisnika.

2.Korištene tehnologije

2.1.VueJs

VueJs je Javascript programski okvir(eng. *framework*) koji služi za izradu jednostavnih i/ili složenih korisničkih sučelja. VueJs je iz godine u godinu pri vrhu po korištenosti, razini zadovoljstva i generalnom interesu među programerima[6]. Iz tog razloga, često se uspoređuje sa AngularJs i ReactJs, za razliku od njih VueJs se smatra puno lakšim i jednostavnijim pogotovo za početnike.

VueJs programski okvir je reaktivan, što znači da kada se promjene pohranjeni podaci u Javascript dijelu, automatski se mijenja korisničko sučelje. Klasični primjer reaktivnosti je brojač. Korisničko sučelje se sastoji od jednog gumba i teksta. Ti elementi služe za povećavanje njegove vrijednosti te prikaz njegove vrijednost. Klikom na gumb, vrijednost pohranjena u JavaScript-u se povećava te se automatski mijenja tekst koji prikazuje vrijednost brojača. U klasičnom JavaScript-u, ovakve promjene bi trebale biti ručno izvršene kako bi se korisničko sučelje izmijenilo što dovodi do puno veće složenosti samog koda.

Za automatsko ažuriranje teksta koristi se deklarativno renderiranje. Deklarativno renderiranje djeluje tako da se proširuje standardna HTML sintaksa, što omogućuje da oblikujemo HTML koristeći JavaScript.

```
<h1>{{counter}}</h1>
```

Također, deklarativno renderiranje može koristiti uvjete(v-if za prikazivanje HTML elementa ili petlje(v-for) za renderiranje više HTML elemenata koristeći dane podatke.

```
<div v-if=„showElement“></div>

<div v-for=„element in  elementList“></div>
```

VueJs daje mogućnost za kreiranje komponenti koje se mogu koristiti na više mjesta kroz aplikaciju. Na ovaj način smanjujemo količinu koda koji se morao ponavljati. Osim toga, tim komponentama možemo proslijediti parametre koji će se koristiti u komponenti.

Primjer toga bila bi komponenta koja prikazuje listu elemenata. Elementi mogu biti proslijeđeni parametar koji se unutar komponenta renderira koristeći v-for.

2.1.1.Usporedba sa ReactJs-om

Sličnosti:

- Virtualni DOM(*Document Object Model*) u kojemu je svaki HTML element predstavljen kao JavaScript objekt. Ovaj pristup daje bolje performanse te omogućuje manipulaciju DOM-om na deklarativnom načinu.
- Korištenje komponente koje se mogu koristiti na više mjesta kroz aplikaciju.
- Korištenje *lifecycle* metode koje se koriste za promjene koje se događaju u komponenti.
- Aktivna i mnogobrojna zajednica programera koji pridonose daljnjem razvoju tehnologije

Razlike:

- React je biblioteka(eng. *library*) dok je Vue programski okvir(eng. *framework*) što znači da je na programeru da izabere prave alate dok Vue dolazi već s alatima za, primjerice, navigaciju(VueRouter).
- Sintaksa, React koristi JSX sintaksu dok Vue koristi sintaksu sličniju standardnom HTML-u. To znači da kada bi željeli renderirati listu elemenata, u React-u ćemo koristiti *map()* funkciju dok Vue koristi v-for direktivu
- Rukovanje stanjem aplikacije, React koristi *useState()* funkcije za mijenjanje neke vrijednosti dok se u Vue-u takva vrijednost može mijenjati direktno.

2.2.SASS/SCSS

SASS je stilski jezik koji omogućuju korištenje funkcionalnosti koje nisu dostupne u običnom CSS-u, kao što su: ugniježđeni stilovi, *mixin*, uvjetni operatori i mnoge druge stvari.

Takav SASS kod se zatim procesira u obični CSS kod. Svrha korištenja jezika kao što je SASS je da uvelike olakšava pisanje koda koji se može ponovno iskoristiti.

Primjerice, ako želimo iskoristi klasu na više mjesta u aplikaciji sa sitnim promjenama, tu klasu možemo napisati kao *mixin* i kao parametar proslijediti tu promjenu.

Slika 1 Primjer mixin-a u SCSS-u

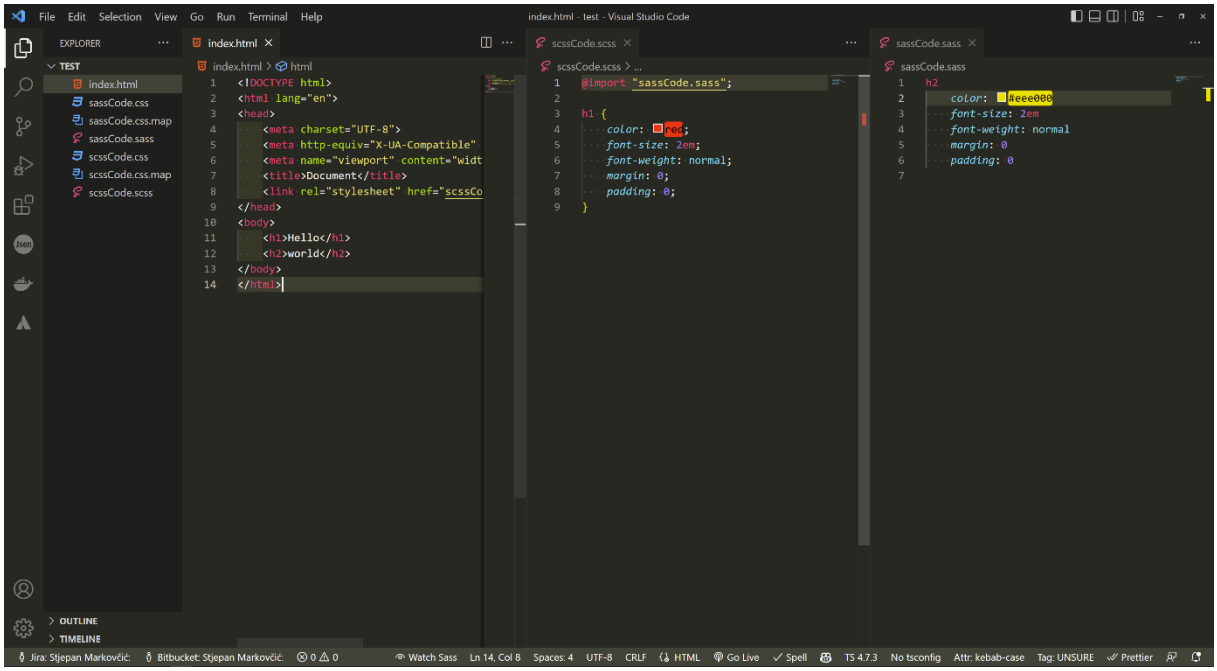
2.2.1 Razlika između SASS-a i SCSS-a

Jedina prava razlika između SASS-a i SCSS-a je njihova sintaksa. Dok sintaksa za SCSS nalikuje onoj na klasičan CSS, u SASS sintaksi nedostaju vitičaste zagrade te točka sa zarezom.

Rezultat ovih sintaktičkih razlika jest da je svaki validan CSS kod istovremeno i validan SCSS kod dok to nije slučaj za SASS. Zbog toga, implementacija SCSS-a u već postojeći projekt je vrlo laka i bezbolna te ne zahtijeva nikakve promjene u postojećem kodu.

Osim razlike u sintaksi još jedna razlika je u ekstenziji datoteke, *.scss* za SCSS i *.sass* za SASS.

Usprkos sintaktičkoj razlici, moguće je koristiti SASS kod unutar SCSS-a(i obratno) koristeći *@import* pravilo. Razlog tome je što se prilikom procesiranja SASS i SCSS koda, sav kod pretvara u standardni CSS.



Slika 2 Korištenje SCSS-a i SASS-a istovremeno

2.2.2 Usporedba SASS i LESS

Slična tehnologija SASS-u je LESS, LESS je također predprocesor za CSS koji proširuje CSS putem stvari kao što su *mixini*. Za razliku od SASS *mixina*, koji se dodaju u neku klasu koristeći @*include* pravilo, LESS koristi *mixin* kao običnu klasu(.ime_klase) unutar neke druge klase.

LESS, isto kao i SASS su nadskupovi CSS-a, što znači da sadrže sve funkcionalnosti CSS-a te ih proširuju s dodatnim stvarima kao što su već navedeni *mixini.* Osim *mixina* također postoje petlje, uvijeni operatori, varijable, funkcije te niz ostalih alata kojima se proširuje CSS.

Jedna od glavnih razlika između SASS-a i LESS-a jest jezik u kojemu su napisani. SASS je pisan u programskom jeziku Ruby dok je LESS pisan u Javascriptu.

Prilikom kreiranja nove varijable, SASS koristi znak $ dok LESS koristi znak @. Daljnje korištenje varijabli je identično, gdje se umjesto neke CSS vrijednosti postavlja određena varijabla. Najveća razlika je da LESS može u varijablu pohraniti CSS selektor dok to nije moguće u SASS-u.

```
.p1 {
   color:red;
}

.p2 {
   background : #64d9c0;
   .p1( );
}

.p3 {
   background : #LESS520;
  .p1;
}
```

Slika 3 Primjer mixina u LESS-u

2.3.AIOHTTP

AIOHTTP je asinkroni HTTP klijent/server za programski jezik Python. Omogućuje stvaranje web servera koji će poslužiti kao backend servis za aplikaciju. Definiramo rute za REST API(*Application Programming Interface*) na koje će korisnička aplikacija slati zahtjeve za izvođenje osnovnih CRUD(*Create, Read, Update, Delete*) operacija. Osim izvođenja CRUD operacija nad bazom podataka, CRUD operacije se izvode nad statičkim datotekama.

Osim za izradu backend-a, AIOHTTP također može poslužiti za izvršavanje asinkronih HTTP zahtjeva u sklopu klijentske aplikacije.

Slanje takvih asinkronih zahtjeva daje mogućnost puno bržeg rada aplikacije. Razlog tome jest što za razliku od sinkronog programiranja, gdje se pošalje zahtjev za dohvaćanje resursa s interneta i čeka se izvršenje tog zahtjeva, asinkrono programiranje ne čeka da se zahtjev izvrši.

Umjesto čekanja na rezultat asinkronog zadatka, vraća se obećanje da će se zadatak izvršiti. Dobiveno obećavanje može kasnije biti izvršeno ili odbijeno ovisno o krajnjem rezultatu samog asinkronog zadatka. Na ovaj način znatno se ubrzava izvođenje zadataka koji su sačinjeni od mnogobrojnog niza dugotrajnih procesa.

Primjer takvog zadatka bilo bi dohvaćanje podataka putem API-a sa zadanom listom ključeva. Svaki od ključeva u listi definira jedan od resursa koje je potrebno dohvatiti te za dohvaćanje prolazimo kroz tu listu i dohvaćamo jedan po jedan resurs.

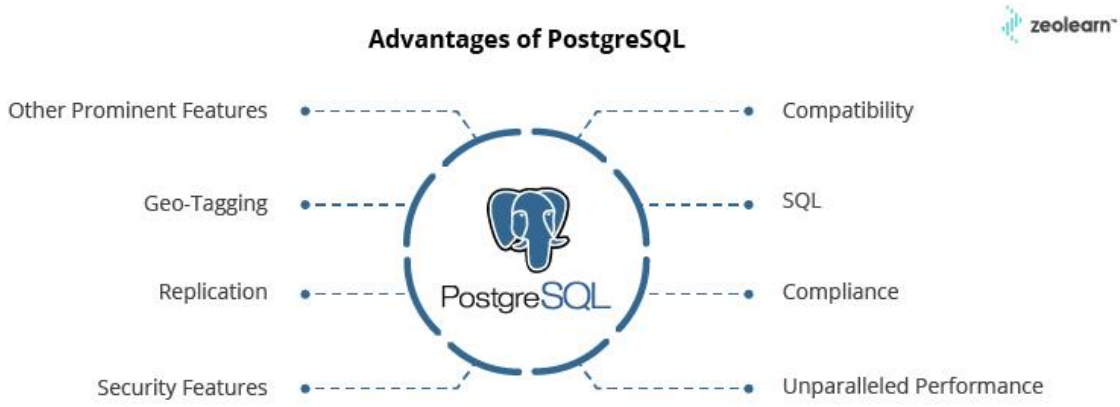
2.4.PostgreSQL

PostgreSQL je objektno orijentirana baza podataka bazirana na otvorenom kodu(eng. *open source*). Prvi put se pojavljuje 1987. godine te je kroz sve te godine dobio reputaciju zbog svoje velike razine otpornosti, integriteta te robusnosti svojih alata. Upravo zbog toga se danas koristi kao primarna baza podataka za mnoge web, mobilne, geoprostorne i analitičke aplikacije.

Podaci se pohranjuju kao unutar tablica koje sadrže primarne i strane ključeve za spajanje s ostalim tablicama putem relacijskih veza. Osim primarnih i stranih ključeva, tablice također sadrže stupce, svaki od kojih označava podatak unutar tablice. Također, podaci pohranjeni u jednom stupcu mogu biti samo jednog tipa podatka koji je određen za taj stupac. PostgreSQL podržava vrlo velik broj tipova podataka, od onih osnovnih kao što su boolean i integer do prostornih tipova podataka koji se koriste u obradi geografskih podataka.

PostgreSQL se koristi u bankarskim sustavima, sustavima za procjenu rizika, multiplikacijskim repozitorijima podataka, proizvodnji, sustavima poslovne inteligencije i kao primarna baza za mnoge poslovne aplikacije. Primjer nekih velikih kompanija i aplikacija koje koriste PostgreSQL su: Apple, Reddit, Instagram, Twitch...

PostgreSQL prati klasičnu SQL sintaksu koju možemo naći u mnogim drugim SQL bazama poput MySQL. Također PostgreSQL je usklađen sa ACID(*Atomicity, Consistency, Isolation, Durability*) pravilima.



Slika 4 Prednosti PostgreSQL-a, Izvor: https://www.zeolearn.com/magazine/reasons-to-enrol-for-a-postgresql-course

2.4.1.Usporedba s MongoDB

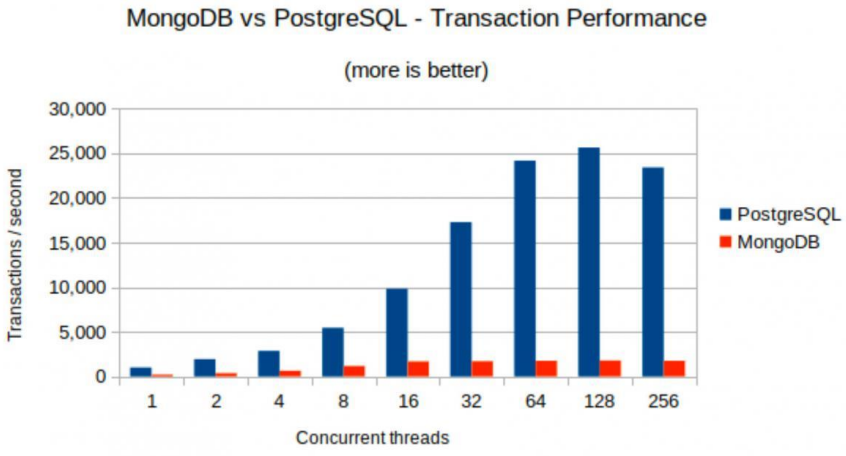
Osim PostgreSQL baze podataka postoji još velik broj vrlo popularnih baza podataka koje se vrlo često koriste. Jedna od takvih baza je MongoDB.

Za razliku od PostgreSQL-a, MongoDB je *NoSQL* baza podataka. NoSQL baze podataka su često jednostavnije u usporedbi sa SQL bazama jer se u njima podaci pohranjuju kao JSON dokumenti unutar kolekcija. Za razliku od toga, PostgreSQL pohranjuje podatke u retke unutar tablica s relacijskim vezama između tim tablica.

Umjesto SQLa, MongoDB koristi BSON(Binary JSON) koji omogućuje korištenje nekih tipova podataka koji nisu podržani standardni JSON ne podržava kao primjerice vremenski podaci.

MongoDB se zasniva na distribuiranoj arhitekturi što znači da različite komponente funkcioniraju kroz više različitih platformi u kolaboraciji za rješavanje nekog zadatka. Prednost ovakve distribuirane arhitekture je također u tome što MongoDB može rasti kroz više različitih platformi dok je PostgreSQL, koji je baziran na monolitnoj arhitekturi, ograničen na performanse platforme na kojoj se nalazi.

U testiranju performansi i transakcija između PostgreSQL-a i MongoDB-a utvrđeno je da je PostgreSQL 4-15 puta brži. Testiranje je provedeno od strane OnGres tvrtke, koja se specijalizira za pružanje softvera i usluga za baze podataka[7].



Slika 5 Usporedba performanski PostgreSQL-a i MongoDB-a, Izvor: https://www.enterprisedb.com/news/new-benchmarks-show-postgres-dominating-mongodb-varied-workloads

3.Implementacija projekta

Projekt možemo podijeliti u dvije cjeline, frontend dio i backend dio projekta. Ova dva dijela projekta se nalaze u zasebnim repozitorijima na Githubu.

Repozitorij za frontend dio projekta: *https://github.com/mstjepan28/bpmn-admin*

Repozitorij za backend dio projekta: *https://github.com/mstjepan28/pdfBuilderService*

3.1.Implementacija frontend dijela projekta

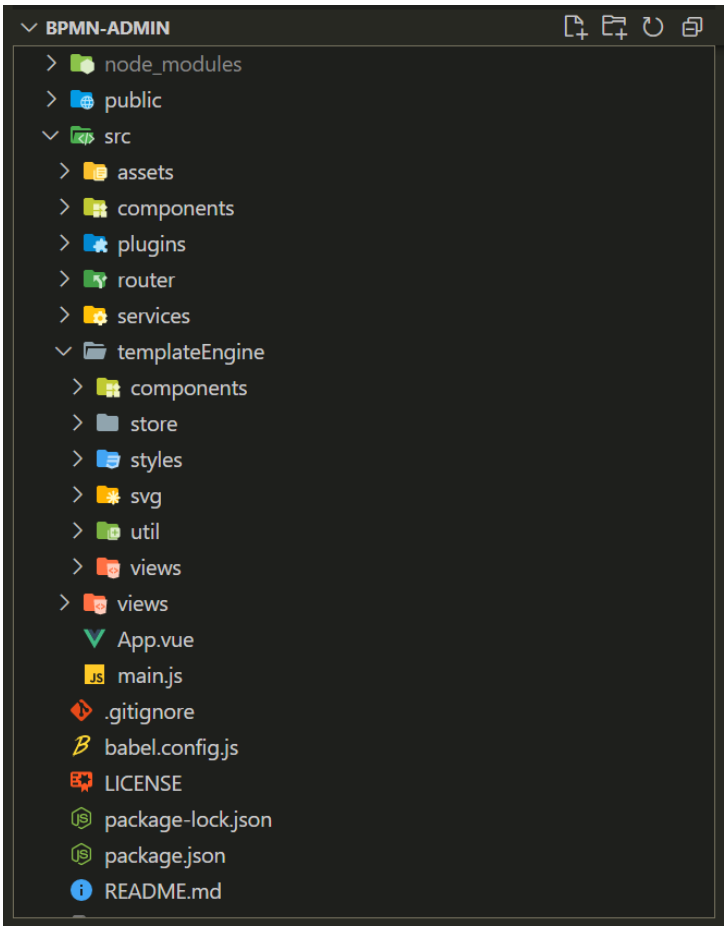
Frontend dio projekta implementiran je koristeći primarno VueJs programski okvir. Osim VueJs-a korištena je još nekolicina biblioteka za olakšavanje implementiranja određenih dijelova projekta.

Primjerice, korištena je biblioteka *InteractJS* koja olakšava implementaciju logike za pomicanje te promjenu veličine elemenata. Osim biblioteke *InteractJS* također su korišteni *DayJS* koji služi za upravljanje i manipulaciju vremenskim podacima te *AxiosJS* koji služi za kreiranje i slanje HTTP zahtjeva.

3.1.1.Struktura projekta

Ova aplikacija kreirana je i razvijana s idejom integracije u veći sustav na kojemu radi više ljudi te svi rade odvojene aplikacije. Zbog toga, da bi se izbjegli problemi prilikom integracije, sve se odvaja u zasebni modul. Taj modul pohranjuje sve stilove, komponente, ekrane te ostale dijelove aplikacije, drži ih odvojenim od ostatka sustava. Na ovaj način nastoje se spriječiti konflikti koji se mogu javiti kao posljedica integracije više projekata u konačni sustav.

Iako su svi dijelovi aplikacije odvojeni u zasebni modul, taj modul još ima minimalni doticaj s vanjskim sustavom. Modul dotiče vanjski sustav kroz inicijalizaciju VueX storea te definiranje navigacije.



Slika 6 Struktura frontend dijela projekta

3.1.2.Stvaranje nove selekcije

DOM(*Document Object Model*) je struktura podataka koje predstavlja HTML dokument. Koristeći JavaScript, možemo upravljati DOM-om na razne načine. Primjerice: dodavanje i uklanjanje HTML elemenata, promjena stila, te čitanje i pisanje teksta iz HTML elemenata. U ovoj aplikaciji se upravo ova tri način DOM manipulacije koriste za ostvarivanje željenih funkcionalnosti.

Stvaranje nove selekcije u trenutnom predlošku vrši se putem kreiranja novog HTML div elementa kojemu se zatim mijenja stil ovisno o poziciji miša. Ovisno o početnoj i novoj poziciji miša, izračunava se širina i visina selekcije.

Širina i visina se odnosi na *width* i *height* stilske atribute koji opisuju svaki HTML element. Manipulacijom tih vrijednosti kao i nekolicine ostalih stilskih atributa, upravljamo HTML elementima.

Širina i visina selekcije se mijenjaju slušanjem pomaka miša. Na svaki pomak miša izvršava se funkcija koja ažurira širinu i visinu selekcije. Podaci koji se ažuriraju su oni stilski, koji i određuju širinu i visinu elementa, te pozicijski podaci koji na kraju opisuju visinu i širinu selekcije.

Ovdje se javlja problem ako želimo mišem krenuti u lijevo ili prema gore od početne točke selekcije. Kada prvi put kliknemo na neku točku na kanvasu, ta točka postaje izvorište selekcije te selekcija raste prema dolje te u desno. Točnije y-koordinata raste prema dolje dok x-koordinata raste u desno.

Zbog toga, ukoliko miš pomaknemo u lijevo ili više od početne točke, tada bi širina i visina selekcije bile negativne što je krivo. Da bi riješili ovaj problem, provjeravamo ako je širina ili visina negativna, ukoliko je to slučaj postavljamo vrijednost stilskih atributa *left* i *top* na širinu i visinu.

Razlog tome jest što na ovaj na ovaj način pomičemo samu selekciju u onom smjeru u kojemu bi širina ili visina bila negativna. U slučaju negativne širine selekciju pomičemo u desno, dok u slučaju negativne visine selekciju pomičemo prema dolje u odnosu na početnu točku selekcije. Uz to, prilikom postavljanja visine i širine, koristimo njihove apsolutne vrijednosti upravo iz razloga što mogu biti negativne.

```
function drawSelection(event) {
  let width  = event.offsetX - positionData.x
  let height = event.offsetY - positionData.y
  const selection = thisRef.selectionCreation.selection;

  selection.style.left = width < 0? `${width}px` : "";
  selection.style.top = height < 0? `${height}px`: "";

  positionData.width = Math.abs(width);
  positionData.height = Math.abs(height);

  selection.style.width  = `${positionData.width}px`;
  selection.style.height = `${positionData.height}px`;
}
```

Slika 7 Kod zadužen za crtanje selekcija

Nakon ponovnog klika na kanvasu, završava crtanje selekcije te se više se sluša pomak miša. Nova selekcija pohranjena je u listu svih selekcija kao Javascript objekt koji sadrži sve informacije vezene uz tu selekciju.

Još jedan problem koji se ovdje javlja je kada želimo započeti novu selekciju unutar postojeće selekcije. Pošto je crtanje selekcije dozvoljeno jedinu na kanvasu, klik na selekciju se neće računati kao klik na kanvas iz razloga što je selekcija zaseban element.

Kao rješenje toga, isključujemo *pointer evente* svih trenutnih selekcija. To znači da klik na bilo koju od selekcije se ne registrira kao klik na selekciju već kao klik na sljedeći element koji leži ispod selekcije, a to je kanvas.

Nakon isključivanja opcije za crtanje selekcija, *pointer eventi* svih selekcija se ponovno uključuju.

```
togglePointerEvents(parentElement, boolValue){
  Array.from(parentElement.children).forEach(child => {
    return child.style.pointerEvents = boolValue? "auto": "none";
  })
}
```

Slika 8 Funkcija za isključivanje/uključivanje pointer evenata

Novoj selekciji moguće je ponovno promijeniti širinu i visinu kao i obrisati ju. Brisanje selekcije uklanja HTML element koji predstavlja tu selekciju, zajedno sa Javascript objektom koji sadrži sve informacije pohranjene o selekciji.

3.1.3.Promjena veličine i položaja selekcije

Kod promjene veličine selekcije također se koriste slične DOM manipulacije vezane uz promjenu stila. Dohvaća se trenutna pozicija elementa koja se prethodno pohranila u HTML atributu kao „data-x“ i „data-y“ te se zajedno sa novom pozicijom miša koristi za izračunavanje nove pozicije selekcije. Novu poziciju miša dobivamo kao podatak iz događaja zajedno sa podatkom o novoj veličini selekcije.

Događaj iz kojega dobivamo podatke naziva se *resizeMove* događaj te je dio InteractJS biblioteke koja olakšava implementaciju pomicanja i promjena veličina HTML elemenata koji na sebi imaju predefiniranu klasu. U ovom slučaju selektor je „*.draggable*“ što znači da će svaki element koji na sebi ima klasu *draggable* biti podložan promjeni veličine te pomicanju.

InteractJS nam također omogućuje postavljanje restrikcija na određene elemente, neke od kojih su:

- Zadržavanje unutar roditeljskog elementa, koristeći ovu restrikciju nije moguće pomaknuti niti proširiti selekciju izvan kanvasa koristeći miš.
- Postavljanje minimalne i maksimalne veličine selekcije, putem ove restrikcije postavljamo minimalnu veličinu selekcije na 15 piksela
- Restrikcije za promjenu veličine samo iz određenih rubova, primjerice ako želimo omogućiti samo širenje selekcije prema gore u pravilo *edges* postavili bi vrijednost atributa *top* kao *true* a *left*, *right* i *bottom* kao *false*
- „Snapping“ selekcija na mjesto na kanvasu tijekom pomicanja, omogućuje preciznije pomicanje selekcija prilikom pomicanja jer svaki put kad se selekcija pomakne i pusti, automatski „snappa“ na točniju poziciju

```
interaction(){
  const thisRef = this;

  interact('.draggable')
  .resizable({
    inertia: false,

    // resize from all edges and corners
    edges: { left: true, right: true, bottom: true, top: true },

    modifiers: [
      interact.modifiers.restrictEdges({
        outer: 'parent'
      }),

      // minimum size
      interact.modifiers.restrictSize({
        min: { width: this.minElementSize, height: this.minElementSize }
      })
    ],

    listeners: {
      move: (event) => thisRef.resizeHandler(event)
    },
  })
  .draggable({
    inertia: false,

    modifiers: [
      interact.modifiers.snap({
        targets: [
          // Snap elements in place when dragged
          interact.snappers.grid({ x: 10, y: 10 })
        ],
        range: Infinity,
        relativePoints: [ { x: 0, y: 0 } ]
      }),

      // Keeps the element inside the parent
      interact.modifiers.restrictRect({ restriction: 'parent' })
    ],

    listeners: {
      move: (event) => thisRef.dragMoveListener(event),
    },
  })
}
```

Slika 9 Definiranje postavki za rad sa InteractJS bibliotekom

```
resizeHandler(event){
  const target = event.target;

  // Get the previous coordinates of the object
  let x = (parseFloat(target.getAttribute('data-x')) || 0)
  let y = (parseFloat(target.getAttribute('data-y')) || 0)

  // translate when resizing from top or left edges
  x += event.deltaRect.left;
  y += event.deltaRect.top;

  let width = event.rect.width;
  let height = event.rect.height;

  target.style.width = `${width}px`;
  target.style.height = `${height}px`;

  target.style.transform = `translate(${x}px, ${y}px)`;

  this.updatePositionData(target, {
    x: parseInt(x),
    y: parseInt(y),
    width: parseInt(width),
    height: parseInt(height)
  })

  target.setAttribute('data-x', x);
  target.setAttribute('data-y', y);
}
```

Slika 10 Kod zadužen za promjenu veličine selekcija

Promjena pozicije selekcije je jednostavnija od promjene veličine selekcije pošto ne moramo voditi računa o svim smjerovima u kojima se pozicija može proširivati te smanjivati.

Najkompleksniji dio promjene pozicije, ograničavanje selekcije unutar kanvasa već je odrađen od strane InteractJS biblioteke što značajno pojednostavljuje ovaj dio.

Izvlačimo HTML atribute data-x i data-y koji označavaju trenutnu poziciju selekcije te na njih zbrajamo trenutnu poziciju miša. Nove vrijednosti za x i y koordinatu koristimo za postavljanje vrijednosti *transform* stilskog svojstva koje određuje konačnu poziciju elementa na kanvasu.

Također ažuriramo pozicijske podatke s novim x i y vrijednostima te HTML atribute koji sadrže x i y vrijednost.

Ova funkcija se poziva na svaki pomak miša dok je desni gumb miša pritisnut.

```
dragMoveListener(event){
  const target = event.target;

  // keep the dragged position in the data-x/data-y attributes
  const x = (parseFloat(target.getAttribute('data-x')) || 0) + parseInt(event.dx)
  const y = (parseFloat(target.getAttribute('data-y')) || 0) + parseInt(event.dy)

  // update position data of object
  this.updatePositionData(target, {x, y})
  target.style.transform = `translate(${x}px, ${y}px)`

  // update the position attributes
  target.setAttribute('data-x', x)
  target.setAttribute('data-y', y)
}
```

Slika 11 Kod zadužen za promjenu pozicije selekcija

3.1.4.Strukturiranje podataka

U aplikaciji, glavni podaci s kojima se radi su vezani za sam predložak. Podaci o predlošku se pohranjuju u JSON formatu te izgledaju ovako:

```
{
  "id": "id predloška",
  "pdfDimensions": "dimenzije konačnog PDF dokumenta u pikselima",
  "name": "ime predloška",
  "selectionList": [
    {
      "id": "id selekcije koji se koristi samo na korisničkom dijelu aplikacije",
      "elementRef": "referenca na HTML element",
      "positionData": {
        "x": "X koordinata HTML elementa u odnosu na lijevi rub",
        "y": "Y koordinata elementa u odnosu na gornji rub",
        "width": "širina elementa HTML element",
        "height": "visina elementa HTML elementa"
      },
      "type": "tip podatka koji se popunjava u ovom području",
      "variable": "ime varijable u koju se popunjava podatak",
      "isStatic": "da li je element statički ili ne",
      "staticContent": "statički sadržaj elementa",
      "internalComponent": "interna komponenta zadužena za drag-and-drop slika"
    },
  ],
  "variableList": [
    {
      "name": "ime varijable",
      "type": "tip podatka ove varijable",
      "example": "primjer varijable",
      "description": "opis varijable"
    }
  ]
}
```

Slika 12 Struktura predloška u JSON obliku s opisom svih vrijednosti

ID predloška se ne generira prilikom spremanja u bazu podataka, već se dohvaća sa servera i koristi kod učitavanja PDF dokumenta nad kojim će se predložak izraditi.

Prilikom učitavanja i slanja PDF dokumenta, generirani ID se šalje zajedno sa dokumentom i koristi se kao ime pod kojim se pohranjuje kao statički dokument na

serveru. Ukoliko korisnik zatim odustane od izrade predloška, na server se šalje zahtjev za brisanje statičkih dokumenata vezanih za taj predložak.

Od svih tih podataka, najbitniji su oni pohranjeni unutar „*selectionList*“ liste. Ti podaci predstavljaju selekcije koje se nalaze na predlošku, točnije opisuju sve karakteristike selekcije.

Kada će se ovi podaci koristiti za generiranje PDF dokumenata, podaci o svakoj selekciji će služiti kao instrukcije za kreiranje sadržaja PDF dokumenta.

3.1.5.Odabir, preuzimanje i brisanje dokumenata

Nakon stvaranja predloška te generiranja PDF dokumenata putem njega, korisniku je omogućeno preuzimanje posljednjih deset generiranih dokumenata u svrhu provjere i pregleda ako je sve prošlo uredu tijekom generiranja. Osim pregleda, korisniku je dano na mogućnost i brisanje odabranih dokumenata.

Preuzimanje i brisanje dokumenata vrši se putem izbornika koji se nalazi u modalu. Prilikom otvaranja ovog izbornika, šalje se HTTP zahtjev za dohvaćanje dokumenata s backenda koji se zatim prikazuje unutar izbornika, točnije prikazuju se imena dokumenata.

```
async fetchFiles() {
  if(!this.templateId) {
    return;
  }

  this.selectedFilesList = [];

  try{
    const res = await axios.get(`${this.baseUrl}/template/${this.templateId}/file`);
    this.fileList = res.data;
    this.isAllSelected = false;
  }catch(e){
    this.fileList = []
    console.log(e);
  }
}
```

Slika 13 Funkcija koja dohvaća popis dokumenata

Prije samog preuzimanja/brisanja dokumenta, korisnik klikom na instancu određenog dokumenta dodaje taj dokument u listu odabranih dokumenata, ukoliko se dokument već nalazio u listi izbacuje ga se iz iste i poništava se njegov odabir.

```
fileSelected(selectedFile) {
  this.selectedFilesList.push(selectedFile);
  this.isAllSelected = this.selectedFilesList.length === this.fileList.length;
}

fileDeselected(deselectedFile) {
  this.selectedFilesList = this.selectedFilesList.filter(file => file !== deselectedFile);
  this.isAllSelected = false;
}
```

Slika 14 Funkcije za dodavanje i uklanjanje dokumenta s liste odabranih dokumenata

Također, prilikom označavanja elementa s liste provjerava se ako su svi elementi označeni. Ako jesu, gumb odgovoran za odabir svih elemenata postaje aktivan te njegova funkcija postaje uklanjanje svih odabranih dokumenata.

Svakom odabranom dokumentu dodaje se CSS klasa koja stilizira komponentu na način da joj daje drugačiju pozadinsku boju i obrub u svrhu razlikovanja odabranih dokumenata.

Nakon što su svi dokumenti odabrani, korisniku je dano na izbor njihovo brisanje ili preuzimanje, u oba slučaja lista sa odabranim dokumentima se putem HTTP zahtjeva šalje na backend gdje se zahtjev obrađuje.

Ukoliko se radi o brisanju, nakon slanja HTTP zahtjeva i dobivanja odgovora o uspješnom izvršavanju brisanja, prikazuje se poruka o uspješnom brisanju dokumenata, dokumenti se filtriraju iz lokalne liste i zatvara se modal sa popisom dokumenata.

Ako je korisnik odabrao preuzimanje dokumenata, HTTP zahtjev vraća odabrane dokumente te se oni preuzimaju. Preuzima se na način da se stvori novi *anchor* element(<a>) na koji se dodaju podaci za preuzimanje zajedno sa imenom dokumenta koji će se preuzeti. Ime ovisi o tome ako je odabran jedan ili više dokumenata. U slučaju da je odabran samo jedan, ime je, jednostavno, ime tog

odabranog dokumenta. Ako se radi o više dokumenata, ime je ID odabranog predloška po kojemu su dokumenti generirani. *Anchor* element se spaja na tijelo HTML dokumenta te se na njemu programski izvršava klik što pokreće preuzimanje. *Anchor* element se na posljetku briše jer više nije potreban.

```
async downloadSelectedFiles() {
  if(this.selectedFilesList.length === 0) {
    return;
  }

  try{
    const response = await axios.post(`${this.baseUrl}/template/${this.templateId}/file`,
      {
        fileList: this.selectedFilesList
      },
      {
        responseType: "blob"
      }
    );

    const fileName = this.selectedFilesList.length === 1 ? this.selectedFilesList[0] : `${this.templateId}.rar`;

    const blob = new Blob([response.data], { type: "application/octet-stream" });

    const fileLink = document.createElement('a');
    fileLink.href = window.URL.createObjectURL(blob);
    fileLink.setAttribute('download', fileName);

    document.body.appendChild(fileLink);
    fileLink.click();
    document.body.removeChild(fileLink);
  }catch(e) {
    console.log(e);
    this.openResponsePopup("error", "Error while downloading files");
  }

  this.closeWrapperModal();
},
```

Slika 15 Funkcija za preuzimanje dokumenata

3.1.6.Komunikacija s backend dijelom projekta

Za komunikaciju sa serverom koristimo *Axios* paket koji omogućuje komunikaciju s RESTful API-om. *Axios* se koristi za sve HTTP zahtjeve koji se izvršavaju na serveru.

Komunikacija s backend dijelom projekta vrši se prilikom izvršavanja osnovnih CRUD(*Create Read Update Delete*) operacija nad predlošcima. Primjerice, na ekranu koji prikazuje listu svih stvorenih predložaka šaljemo HTTP zahtjev za dohvat

svih predložaka. Dohvaćeni predlošci se zatim sortiraju i filtriraju po zadanim uvjetima te prikazuju korisniku.

```
async fetchTemplates() {
  try{
    const response = await axios.get(`${this.baseURL}/template`);
    this.templateList = response.data;

    this.sortTemplates('updated_at');
    this.searchTemplates();
  }catch(error){
    console.log(error)
  }

  this.isLoading = false;
}
```

Slika 16 Funkcija za dohvat predložaka putem HTTP zahtjeva

Nakon što su svi predlošci dohvaćeni i prikazani korisnik bira jedan od tih predložaka te se taj odabrani predložak dohvaća putem njegovog ID-a.

```
async fetchTemplateById(templateId){
  try{
    const response = await axios.get(`${this.baseURL}/template?id=${templateId}`);
    this.template = response.data;
  }catch(error){
    console.log(error)
  }
}
```

Slika 17 Funkcija za dohvaćanje individualnog predloška

Prilikom kreiranja novog te ažuriranja postojećeg predloška koristimo *multipart/form-data* enkodiranje za slanje podataka. To činimo iz razloga što osim slanja podataka o predlošku, šaljemo i PDF dokument koji će služiti kao baza za

generiranje svih PDF dokumenata. Kod ovog slanja, podaci o predlošku, zajedno sa PDF dokumentom šaljemo kao BLOB(*Binary Large Object*) podatke.

```
async saveTemplate(){
  const template = this.formatTemplate();

  const blobTemplate = new Blob([ JSON.stringify(template) ], { type: "application/json" });
  const blobPdfFile = new Blob([ this.pdfTemplateBuffer ], { type : 'application/pdf'})

  let data = new FormData();
  data.append("template", blobTemplate, "template")
  data.append("pdfTemplate", blobPdfFile, "pdfTemplate.pdf")

  if(this.isNewTemplate){
    await this.createTemplate(data);
  }
  else{
    await this.updateTemplate(data);
  }
}
```

Slika 18 Enkodiranje podataka u multipart/form-data

Prije nego što podatke o predlošku enkodiramo u multipart/form-data, formatiramo ih tako da će daljnji rad s njima biti lakši. Razlog tome je što su do sada podaci prilagođeni potrebama frontenda te sadrže stvari kao što su reference na HTML elemente i VueJS komponente.

Također, potrebno je normalizirati pozicijske podatke selekcija, inače će selekcije na generiranim PDF dokumentima biti pogrešno pozicionirane zbog razlike veličina kanvasa na korisničkom sučelju te PDF dokumenta koji će se generirati.

Da bi riješili ovaj problem, x koordinatu i širinu selekcije dijelimo s ukupnom širinom kanvasa dok y koordinatu i visinu selekcije dijelimo s visinom kanvasa. Kada će se ove vrijednosti koristiti za generiranje PDF dokumenata, pomnožit će se s visinom i širinom dokumenta koji se stvara. Rezultat toga biti će precizno pozicioniranje generiranih podataka.

3.2.Implementacija backend dijela projekta

Backend dio projekta implementiran je u programskom jeziku Python koristeći AIOHTTP za web server te PonyORM za komunikaciju s bazom podataka.

3.2.1.Komunikacija s bazom

Za komunikaciju s bazom koristi se Pony ORM pomoću kojega se izvršavaju upiti(eng. *query*) nad bazom. Također, koristeći PonyORM definiramo tablice unutar baze zajedno sa svim njihovim svojstvima poput označavanja određenih polja tablice kao opcionalnih ili određivanje tipa podatka nekog polja.

```python
class Template(DB.Entity):
  id = PrimaryKey(str, auto=False)
  base_template_id = Optional(str, nullable=True)
  pdf_dimensions = Required(Json)

  created_by = Required(str)
  created_at = Required(str)
  updated_at = Required(str)

  name = Required(str, unique=True)
  variable_list = Required(Json)

  selection_list = Required(Json)
```

Slika 19 Definiranje tablice predloška koristeći PonyORM

Upiti se prije izvršavanja dohvaćaju s korisničkog zahtjeva te obrađuju. Primjerice prilikom korisničkog zahtjeva za ažuriranje podataka o predlošku, provjerava se nalazi li se dano ime predloška već u bazi. Ako se nalazi, upit za ažuriranje se ne izvršava i korisniku se vraća greška.

```python
@db_session
def addTemplate(template):
  Template(
    id = template["id"],
    base_template_id = template["baseTemplateId"],
    pdf_dimensions = template["pdfDimensions"],

    created_by = template["createdBy"],
    created_at = template["createdAt"],
    updated_at = template["updatedAt"],

    name = template["name"],
    variable_list = template["variableList"],

    selection_list = template["selectionList"]
  )
```

Slika 20 Funkcija za dodavanje novog predloška u bazu

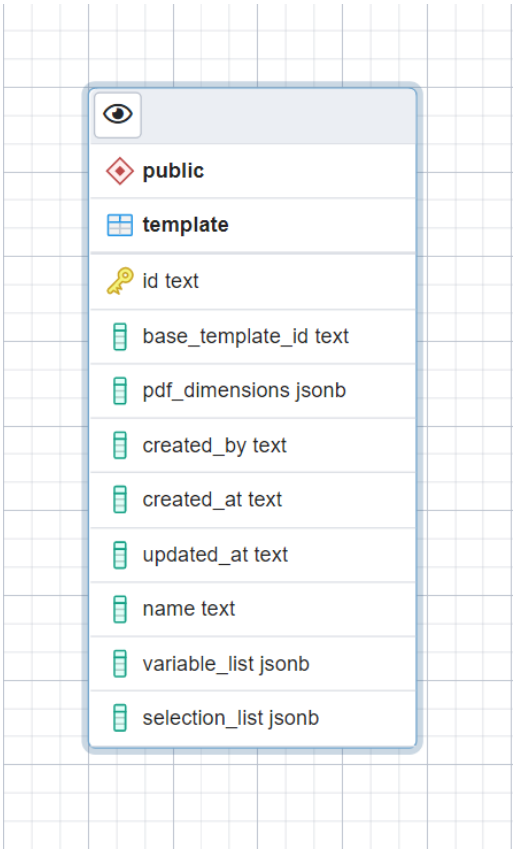
```python
@db_session
def updateTemplate(templateId, updatedTemplate):
  template = Template.get(id=templateId)

  template.base_template_id = updatedTemplate["baseTemplateId"]
  template.pdf_dimensions = updatedTemplate["pdfDimensions"]
  template.name = updatedTemplate["name"]
  template.selection_list = updatedTemplate["selectionList"]
  template.variable_list = updatedTemplate["variableList"]
  template.updated_at = str(datetime.now())
```

Slika 21 Funkcija za ažuriranje već postojećeg predloška

3.2.2.Struktura podataka u bazi

Pošto se svi podaci s kojima rukujemo odnose na predložak, koristimo samo jednu tablicu u bazi.



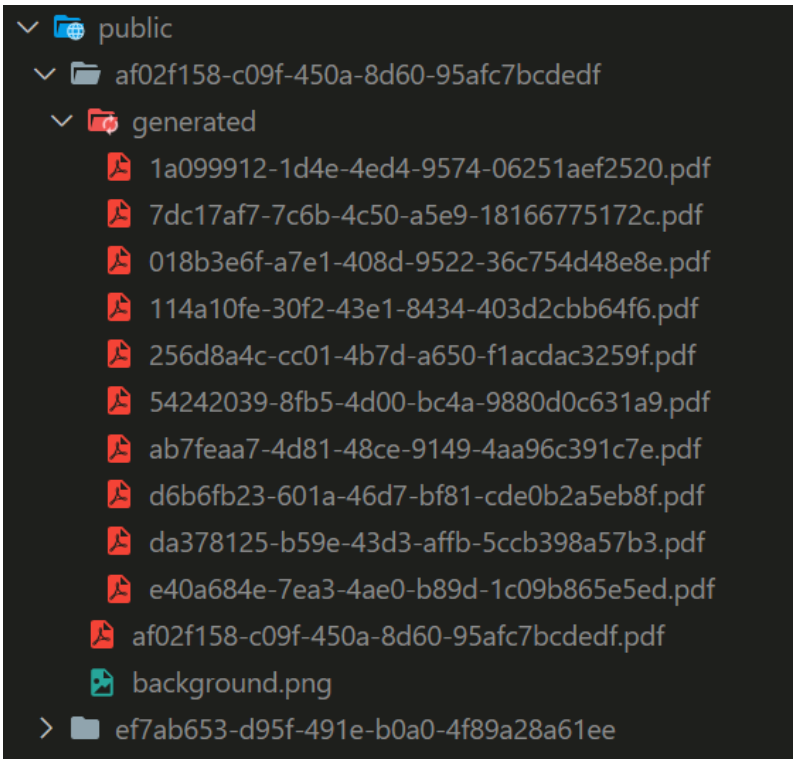
Slika 22 Tablica predloška unutar baze podataka

Optimizacija koja je ovdje moguća jest izdvajanje *„variable_list*“ i *„selection_list“* podataka u zasebne tablice koje bi bile preko stranog ključa povezane sa glavnom tablicom. Na taj način bi granulirali sve podatke o predlošku. Mana ovog pristupa je da bi kompleksnost upita na bazu porasla, a ne bi vidjeli nikakav stvarni benefit od toga. Pošto se *„variable_list“* i *„selection_list“* uvijek dohvaćaju sa svim ostalim podacima o predlošku, jednostavnije je sve spremati u jednu tablicu.

Jedan podatak koji je vezan uz predložak, a koji ne pohranjujemo u bazu je PDF dokument povrh kojega se popunjava predložak. U bazi pohranjujemo samo naziv dokumenta, odnosno *null* u slučaju da korisnik nije učitao dokument.

3.2.3.Rukovanje statičkim datotekama

Prilikom učitavanja PDF dokumenta povrh kojega se popunjava predložak, dokument se sprema kao statička datoteka. Taj dokument, kao i svi ostali statički dokumenti vezani uz predložak spremaju se u direktorij čiji je naziv ID predloška.



Slika 23 Struktura statičkih datoteka na serveru

Na taj način povezujemo sve dokumente vezane uz predložak sa njim. Neki od dokumenata koji se također pohranjuju i vežu uz predložak su svi generirani PDF dokumenti po danom predlošku te slika koja se koristi za prikazivanje predloška na korisničkom sučelju.

Generirani PDF dokumenti se pohranjuju u poddirektorij pod nazivom „*generated*“ dok se pozadinska slika predloška sprema pod nazivom „*background.png*“

Prije bilo kakvog spremanja dokumenta, prvi korak je provjera ako već postoji direktorij za dani predložak. Ukoliko ne postoji stvara se novi i u njega se sprema dani dokument.

```
def templateDirectoryExist(templateId):
  filePath = f"public/{templateId}"
  if not os.path.exists(filePath):
    os.makedirs(filePath)

  return filePath
```

Slika 24 Funkcija zadužena za provjeru i stvaranje datoteke

Prije spremanja PDF dokumenta potrebno ga je izdvojiti iz HTTP zahtjeva u kojemu su podaci zapisani kao *multipart/form-data*. Za izdvajanje PDF dokumenta iz HTTP zahtjeva koristimo čitač za *multipart/form-data* koji je dio AIOHTTP biblioteke.

Pomoću tog čitača prolazimo kroz svaki pojedini dio zahtjeva te provjeravamo koja je vrsta sadržaja pohranjena u trenutnom dijelu zahtjeva. Vrsta sadržaja se može iščitati iz zaglavlja zahtjeva, točnije iz „*Content-Type*“ atributa zaglavlja. Moguće je da je riječ o „*application/json*“ ili o „*application/pdf*“ tipu sadržaja te ukoliko je riječ o „*application/pdf*“ znamo da se radi o PDF dokumentu.

Nakon što smo utvrdili koji dio zahtjeva sadržava PDF dokument, dekodiramo ga te ga pohranjujemo u Python rječnik koji sadrži sve dekodirane dijelove forme.

Ukoliko je tip sadržaja trenutnog dijela zahtjeva *„application/json“* tada ga dekoriramo kao JSON objekt te provjeravamo o kojim podacima je riječ i podatke pohranimo pod određeni ključ unutar rječnika.

Ako je sljedeća vrijednost u čitaču *multipart/form-data* zahtjeva None znamo da smo prošli kroz sve dijelove i tu prestajemo s obrađivanjem zahtjeva.

```python
async def multipartFormReader(request):
  reader = aiohttp.MultipartReader.from_response(request)
  requestData = {}

  while True:
    # Get next form part
    part = await reader.next()

    if part is None:
      break

    elif part.headers['Content-Type'] == "application/json":
      byteArrayResult = await part.read(decode=True)

      jsonValue = byteArrayResult.decode('utf8').replace("'", '"')
      jsonData = json.loads(jsonValue)

      formPartName = part.headers["Content-Disposition"].split(" ")[2]

      if formPartName == 'filename="template"':
        requestData["template"] = jsonData
      elif formPartName == 'filename="pdfDimensions"':
        requestData["pdfSize"] = jsonData
      elif formPartName == 'filename="templateInfo"':
        requestData["templateInfo"] = jsonData

    # Get the PDF template
    elif part.headers['Content-Type'] == "application/pdf":
      decoded = await part.read(decode=True)
      requestData["pdfTemplate"] = decoded if len(decoded) > 4 else None

  return requestData
```

Slika 25 Funkcija za čitanje multipart/form-data zahtjeva

3.2.5.Generiranje PDF dokumenata

Generiranje PDF dokumenata započinje API pozivom. Pozivamo rutu koja je odgovorna za generiranje PDF dokumenata te u tijelu zahtjeva šaljemo podatke u obliku liste JSON objekata. Točan predložak čije podatke želimo koristiti se dobiva iz URL zahtjeva koji je formatiran kao *„/template/{name}/file/generate“*.

```python
@routes.post("/template/{name}/file/generate")
async def generateFiles(request):
  templateName = request.match_info.get("name")

  template = database.getTemplateByName(templateName)
  data = await request.json()

  fileGenerator(template, data)
  fileList = fileHandler.getGeneratedFiles(template["id"])

  return web.json_response(fileList)
```

Slika 26 Ruta koja se poziva za pokretanje generiranja dokumenata

Iz samog URL-a zahtjeva čitamo ime predloška. Pošto je ime svakog predloška jedinstveno, koristimo ime predloška za pretraživanje baze i dohvaćanje svih podataka o predlošku.

```python
@db_session
def getTemplateByName(templateName):
  template = Template.get(name=templateName)
  return template.to_dict() if template else None
```

Slika 27 Funkcija za dohvat predložak po imenu iz baze

Prilikom generiranja novih PDF dokumenata koristimo *reportlab* biblioteku(eng. *library*) koja sadrži veliki broj funkcija za stvaranje i rad s PDF dokumentima.

Pomoću ID-a predloška provjeravamo postoji li PDF dokument povrh kojega se trebaju generirati podaci. Ukoliko takav PDF dokument ne postoji, podaci će biti generirani povrh zamjenskog, praznog PDF dokumenta. Također, podaci o predlošku također sadržavaju podatke o visini i širini PDF dokumenta povrh kojega će podaci biti generirani. U slučaju da taj PDF dokument ne postoji, širina zamjenskog dokumenta je 596 piksela dok je visina 842 piksela.

Širina i visina dokumenta potrebna je iz razloga što su pozicijski podaci svih selekcija u predlošku u postocima. Zbog toga, x-koordinata govori koliko je posto(od ukupne širine dokumenta) određena selekcija udaljena od lijevog ruba, dok y-koordinata govori koliko je posto(od ukupne visine dokumenta) selekcija udaljena od gornjeg ruba. Ista stvar vrijedi i za širinu i visinu same selekcije.

Broj konačnih PDF dokumenata ovisi o broju JSON objekata koji su poslani u tijelu HTTP zahtjeva, petljom prolazimo kroz svaki od tih objekata te generiramo dokument koristeći dane podatke.

Prilikom generiranja pojedinog PDF dokumenta stvaramo novu instancu kanvasa na koji će se postavljati sav generirani sadržaj. Kao parametre prilikom inicijalizacije kanvasa predajemo:

- međuspremnik(eng. *buffer*) bajtova koji predstavlja sam dokument koji generiramo
- veličinu dokumenta kojega generiramo
- *bottomup* s vrijednošću 0, ovime određujemo da će se vrijednosti y koordinate rasti prema dolje

Iz podataka o predlošku dohvaćamo listu svih selekcija te prolazimo kroz svaku pojedinu selekciju. Iz selekcije dohvaćamo statički sadržaj ili ako je riječ o dinamičkom sadržaju, dodjeljujemo sadržaj ovisno o varijabli koju selekcija sadrži na sebi. Koristimo ime varijable da bi iz podataka po kojima se generira dokument pročitamo dani sadržaj za određenu varijablu. Ukoliko varijabla ne postoji u danim podacima, trenutna selekcija se preskače.

```
def getContent(item, selection):
  if selection["staticContent"]:
    content = selection["staticContent"]
  else:
    try:
      content = item.get(selection["variable"]["name"])
    except:
      content = None

  return False if content == None else str(content)
```

Slika 28 Funkcija za dohvaćanje sadržaja određene selekcije

Ovisno o tipu podatka selekcije, generiranje sadržaja se odvija drugačije. Zbog toga provjeravamo tip sadržaja selekcije te pozivamo određenu funkciju koja će se pobrinuti za generiranje tog određenog tipa sadržaja.

Nakon što je stvoren sadržaj svih selekcija, sadržaj međuspremnika čitamo kao PDF dokumente te ga spajamo s kopijom prve stranice PDF dokumenta kojega smo prethodno učitali. Ovdje dolazi do izražaja mana da je moguće generiranje samo jednostraničnih dokumenata, zbog toga koristimo samo prvu stranicu.

Novostvoreni PDF dokument zatim spremamo u direktoriji vezan za predložak, unutar poddirektorija „*generated*“ te na kraju korisniku vraćamo odgovor na inicijalni HTTP zahtjev koji sadrži listu imena svih generiranih PDF dokumenata.

```python
def fileGenerator(templateData, generateWithData):
  pdfTemplate = readPdfTemplate(templateData)

  pdfSizeDict = templateData["pdf_dimensions"]
  pdfSize = (pdfSizeDict["width"], pdfSizeDict["height"])

  pdfmetrics.registerFont(TTFont('Arial', 'Arial.ttf'))

  # for every object in the data array
  for item in generateWithData:
    packet = io.BytesIO()
    can = canvas.Canvas(packet, pagesize=pdfSize, bottomup=0)

    can.setFillColorRGB(0, 0, 0)
    can.setFont("Arial", FONT_SIZE)

    # for every selection made on the template
    for selection in templateData["selection_list"]:
      content = getContent(item, selection)
      position = normalizePositionData(selection["positionData"], pdfSize)

      if not content:
        continue

      if(selection["type"] == "singlelineText"):
        handleText(can, position, content)
      elif(selection["type"] == "paragraph"):
        handleParagraph(can, position, content)
      elif(selection["type"] == "image"):
        handleImage(can, position, content)

    can.save()

    packet.seek(0)
    newPdf = PdfFileReader(packet)

    templatePage = getPageCopy(pdfTemplate.getPage(0), pdfSize)
    templatePage.mergePage(newPdf.getPage(0))

    output = PdfFileWriter()
    output.addPage(templatePage)

    pdfSavePath = f"public/{templateData['id']}/generated/{str(uuid.uuid4())}.pdf"

    outputStream = open(pdfSavePath, "wb")
    output.write(outputStream)
    outputStream.close()
```

Slika 29 Kod za generiranje dokumenata

3.2.6.Dohvaćanje generiranih dokumenata

Nakon generiranja PDF dokumenata, na korisničkom sučelju je omogućen odabir do 10 generiranih datoteka. Na taj način korisnik može pregledati ako su dokumenti pravilno generirani ili je iz nekog razloga došlo do problema prilikom generiranja dokumenata.

Ukoliko je došlo do nekakvih problema i dokumenti nisu pravilno generirani, korisniku je dano na izbor brisanje tih dokumenata.

Prilikom dohvaćanja generiranih dokumenata provjeravamo ako direktorij unutar kojega se dokumenti spremaju uopće postoji te ako ne postoji stvaramo novi i vraćamo praznu listu kao odgovor na zahtjev.

Iz direktorija čitamo listu generiranih dokumenata te ih sortiramo na način da najnoviji dokumenti budu na početku. Iz te liste izdvajamo prvih deset dokumenata, te ih vraćamo kao odgovor na zahtjev.

Brisanje dokumenata se vrši na način da se u tijelu HTTP zahtjeva šalje lista svih naziva dokumenata koji će biti obrisani. Unutar direktorija koji je vezan za taj određeni predložak pronalazimo direktorij s generiranim dokumentima te prolazimo kroz svaki od dokumenata. Ukoliko se naziv dokumenta nalazi u listi dokumenata za brisanje, brišemo taj dokument.

```
def deleteGeneratedFiles(templateId, fileList):
  filePath = f"./public/{templateId}/generated"

  curWrkDir = os.getcwd()
  os.chdir(filePath)

  for file in os.listdir():
    if file in fileList:
      os.remove(file)

  os.chdir(curWrkDir)
```

Slika 30 Funkcija za brisanje dokumenata

U slučaju da korisnik želi preuzeti određene dokumente, u HTTP zahtjevu se također šalje lista s imenima dokumenata, ovaj puta s imenima dokumenata za preuzimanje. Ukoliko je riječ samo o jednom dokumentu, korisniku se vraća samo PDF dokument, dok u slučaju da je riječ o više dokumenata, ti dokumenti se arhiviraju te tako arhivirani šalju korisniku. Zbog toga prije svega provjeravamo ako se radi o samo jednom ili više odabranih dokumenata te ovisno o tome obrađujemo zahtjev.

Prije samog arhiviranja dokumenata definiramo ime arhiviranog dokumenta kao ID odabranog predloška s ekstenzijom zip. Koristeći Python paket *ZipFile* stvaramo novi zip dokument u koji pohranjujemo sve odabrane PDF dokumente.

Odabrane PDF dokumente pronalazimo unutar poddirektorija „*generated*“ direktorija vezanog za određeni predložak. Prolaskom kroz sve dokumente provjeravamo ako se određeni dokument nalazi u listi odabranih dokumenata te ako se nalazi arhiviramo ga u zip dokument. Na kraju vraćamo generirani zip dokument kao odgovor na zahtjev.

```
def zipTemplateFiles(templateId, fileList):
  filePath = f"./public/{templateId}/generated"
  zipFileName = f"../{templateId}.zip"
  zipFilePath = f"./public/{templateId}/{templateId}.zip"

  curWrkDir = os.getcwd()
  os.chdir(filePath)

  with ZipFile(zipFileName, 'w') as newZipFile:
    for file in os.listdir():
      if file in fileList:
        newZipFile.write(file)

  os.chdir(curWrkDir)

  return web.FileResponse(zipFilePath)
```

Slika 31 Funkcija za arhiviranje odabranih dokumenata

4.Korisničke upute

4.1. Sučelje za stvaranje predloška

Korisničko sučelje za stvaranje i uređivanje predložaka podijeljeno je u tri dijela:

- Prvi dio – odnosi se na lijevi stupac koji sadrži alate vezane uz izradu predloška
- Drugi dio – kanvas na kojemu se izrađuje sam predložak
- Treći dio – odnosi se na desni stupac koji sadrži alate vezane za trenutno odabranu selekciju

Slika 32 Dijelovi korisničkog sučelja

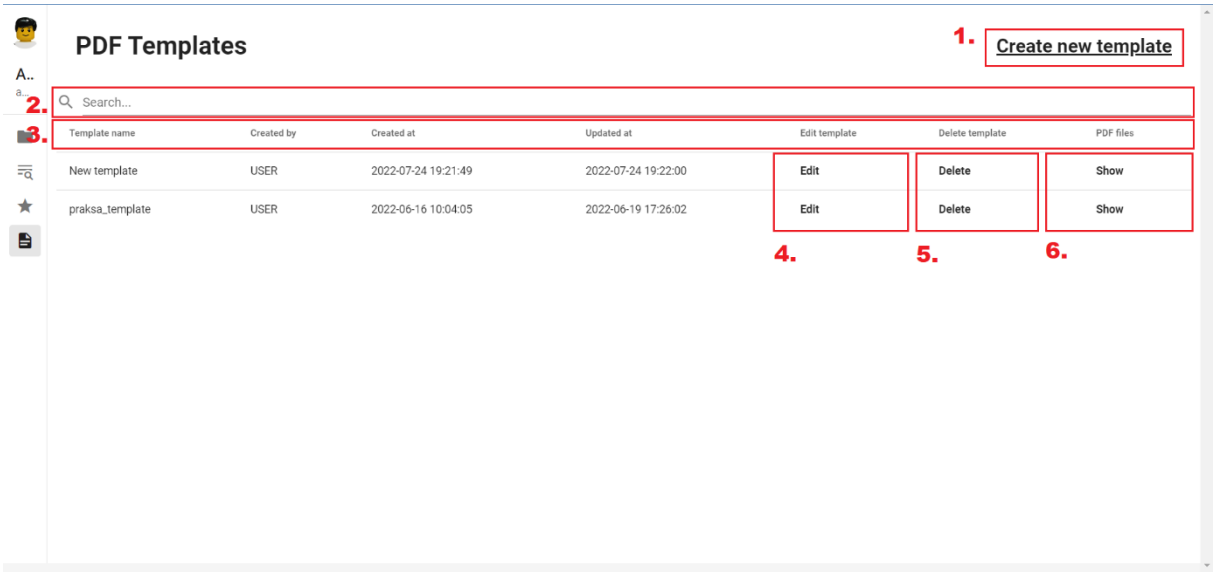
Glavni alati kojima korisnik ima pristup:

1. Unos imena predložaka - ime mora biti jedinstveno iz razloga što se koristi za identificiranje predloška putem API poziva prilikom generiranja PDF dokumenata. Ukoliko je uneseno ime zauzeto, ispisuje se odgovarajuća poruka korisniku. Prilikom klika na gumb za spremanje, polje za unos i gumb nestaju te se na njihovom mjestu pojavljuje samo tekst s novo unesenim imenom. Za ponovno prikazivanje polja dovoljno je kliknuti na ime.
2. Gumb za crtanje selekcija – gumb koji uključuje mogućnost crtanja novih selekcija na kanvasu. Crtanje nove selekcije se vrši tako da korisnik klikne na kanvas te odvuče miš na neku drugu poziciju unutar kanvasa, ponovnim klikom završava crtanje selekcije. Gumb ostaje uključen nakon kreiranja selekcije što je iskazano plavom bojom gumba koja označava aktivno stanje.
3. Gumb za spremanje predloška – gumb za spremanje predloška služi za pohranjivanje promjena stvorenih na predlošku. Prilikom klika na gumb, podaci se obrađuju te se šalju na server za spremanje u bazu. Unutar gumba se prikazuje indikator da se proces spremanja izvršava, indikator nestaje kada se dobije povratna informacija sa servera.
4. Unos PDF dokumenta – služi za učitavanje PDF dokumenta koji služi kao podloga za kasnije generiranje svih PDF dokumenta. Dokument se učita preko *drag-and-drop*-a te se nakon učitavanja korisniku prikazuje poruka ako je dokument uspješno učitan. Primjerice ukoliko korisnik pokuša učitati dokument koji nije PDF, ispisuje se poruka koja obavještava korisnika o krivom formatu dokumenta.
5. Gumb za prikaz pregled podataka – kada je ovaj gumb uključen, unutar svih selekcija koje sadrže varijable se prikazuju primjeri tih varijabli. Ponovnim klikom na ovaj gumb, svi prikazani podaci za pregled se uklanjaju.
6. Polje za dodavanje varijable – unosimo naziv varijable koju želimo dodati te klikom na gumb ili pritiskom tipke „*Enter*“ dodajemo novu varijablu. Dodano ime varijable služi kao identifikator te ga nije moguće mijenjati. Jedini način za promjenu imena varijable je stvaranje nove varijable i brisanje postojeće.
7. Lista varijabli – prikaz svih varijabli dodanih u predložak. Klikom na jednu od njih otvara se izbornik za uređivanje ili brisanje te odabrane varijable.

8. Unos pozicijskih podataka – omogućuje pregled i izmjenu pozicijskih podataka odabrane selekcije. Koordinate *x* i *y* govore koliko je selekcija pomaknuta od gornjeg lijevog kuta kanvasa dok *w* i *h* označavaju širinu i visinu selekcije u pikselima. Prilikom promjene vrijednosti, vrši se validacija nad unesenim vrijednostima te ukoliko odstupaju od zadanih parametara, postavljaju se na maksimalnu/minimalnu prihvatljivu vrijednost.
    a. Minimalna vrijednost za koordinate je 0 dok je za širinu i visinu postavljena na 15 piksela.
    b. Minimalna vrijednost za širinu i visinu selekcije je postavljena na 15 piksela
    c. Maksimalna vrijednost za *x* koordinatu je ukupna širina kanvasa u pikselima
    d. Maksimalna vrijednost za *y* koordinatu je ukupna visina kanvasa u pikselima
    e. Maksimalnu širinu selekcije dobivamo oduzimanjem x koordinate selekcije od ukupne širine kanvasa
    f. Maksimalnu visinu selekcije dobivamo oduzimanjem y koordinate selekcije od ukupne visine kanvasa
9. Gumb za onemogućavanje micanja selekcije – ukoliko isključimo ovaj gumb, odabrana selekcija se više neće moći pomicati.
10. Gumb za promjenu vrste sadržaja – kada je selekcija kreirana, njen sadržaj je početno postavljen da bude statičan. Ukoliko je ovaj gumb uključen, sadržaj postaje dinamički.
11. Odabir sadržaja – ukoliko je sadržaj selekcije dinamičan, odabiremo varijablu iz padajuće liste. Ukoliko je sadržaj selekcije ipak statičan, na mjestu padajuće liste dolazi izbornik za odabir tipa podatka te polje za unos sadržaja.
12. Gumb za brisanje selekcije – klikom na gumb briše se odabrana selekcija.

4.2.Sučelje za pregled i upravljanje predlošcima

Svi stvoreni predlošci prikazani su u obliku tablice koja sadrži osnovne informacije o predlošcima kao i kontrole za upravljanje njima.



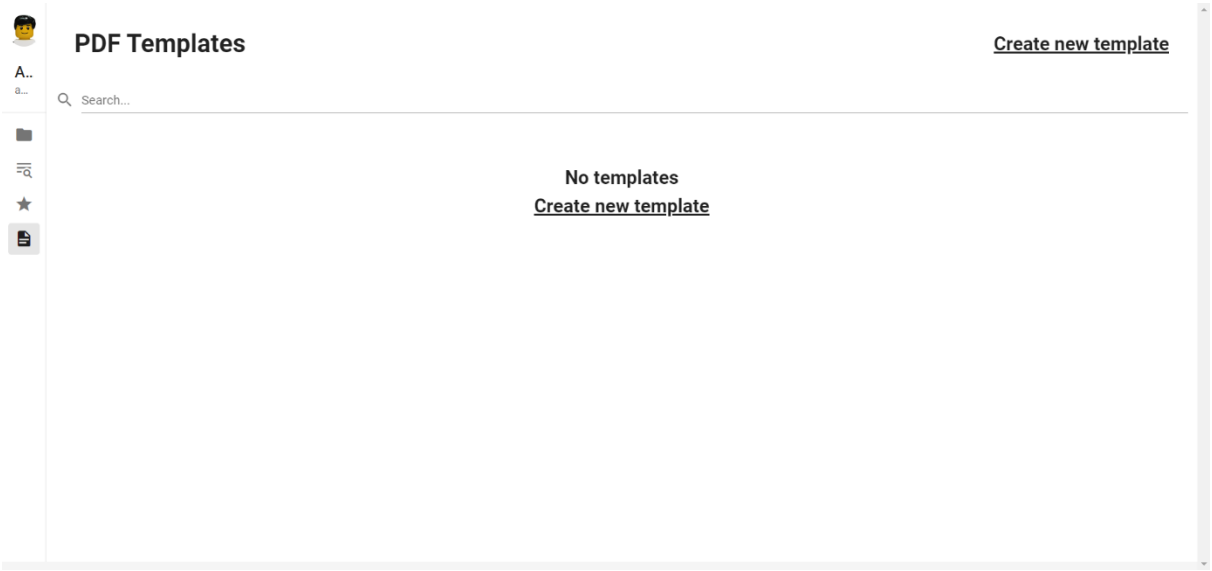
Slika 33 Dijelovi sučelja za pregled i upravljanje predlošcima

Dijelovi na koje je podijeljeno sučelje za pregled i upravljanje predlošcima:

1. Poveznica na sučelje za stvaranje novog predloška
2. Polje za pretragu predlošcima, predlošci se pretražuju po njihovom imenu te putem *fuzzy* pretraživanja. Ova vrsta pretraživanja omogućuje da se uneseno ime predloška ne podudara u potpunosti sa stvarnim imenom predloška, ali će svejedno biti prihvaćeno.
3. Imena stupaca tablice, klikom na „*Template name*“, „*Created by*“, „*Created at*“ ili „Updated at“ možemo sortirati sadržaj tablice, jednim klikom sadržaj se sortira uzlazno dok se ponovnim klikom na istu vrijednost sortira silazno. Tablica je inicijalno sortirana po „*Updated at*“ vrijednostima na način da su posljednje ažurirani predlošci na vrhu.
4. Gumb koji otvara odabrani predložak u sučelju za izradu i uređivanje predložaka

5. Gumb za brisanje odabranog predloška. Klikom na ovaj gumb od korisnika se traži potvrda o brisanju da ne bi došlo do slučajnog brisanja predloška.
6. Gumb za otvaranje izbornika za pregled i upravljanje dokumentima generiranim po odabranom predlošku.

Ukoliko ne postoji niti jedan predložak, umjesto tablice se prikazuje poruka da nije pronađen niti jedan predložak zajedno sa poveznicom na sučelje za stvaranje novog predloška.



Slika 34 Sučelje za pregled i upravljanje predlošcima u stanju bez predložaka

4.3.Varijable predloška

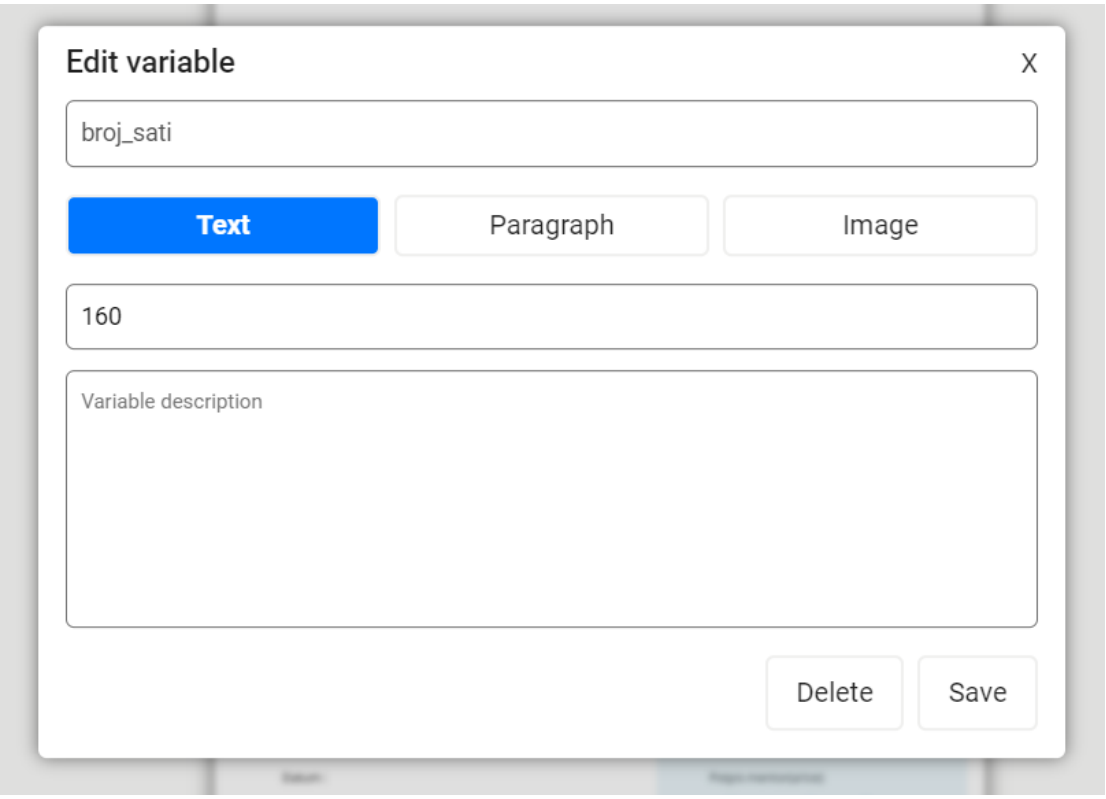
Varijable služe za uparivanje dinamičkog sadržaja sa stvorenom selekcijom. Ukoliko je sadržaj selekcije označen kao dinamički, korisniku se daje na izbor dodavanja varijable na tu selekciju. Prije samog dodavanja varijable korisnik mora dodati varijablu.

Dodavanje varijable se vrši dodavanjem imena varijable u polje za unos varijabli te pritiskom gumba za dodavanje. Novoj varijabli je zadan tip podatka „text“, to možemo promijeniti otvaranje izbornika za uređivanje varijable.

Izbornik se otvara klikom na ime varijable u listi. Unutar izbornika nam je ponuđen odabir tipa podatka varijable te također unos primjera vrijednosti i opis varijable ali nije omogućeno mijenjanje naziva varijable.

Za promjenu naziva varijable, korisnik mora dodati novu varijablu sa željenim nazivom te po izboru izbrisati trenutnu. Unutar izbornika također možemo dodati opis varijable te primjer vrijednosti varijable(za slike unosimo *url* slike).

Ako smo unijeli primjer vrijednosti varijable te zatim tu varijablu postavili na jednu od selekcija, klikom na gumb „*Preview enabled*“ možemo vidjeti kako će selekcija izgledati sa danim podacima. Ovo također vrijedi za slike, gdje će se slika prikazati na selekciji.



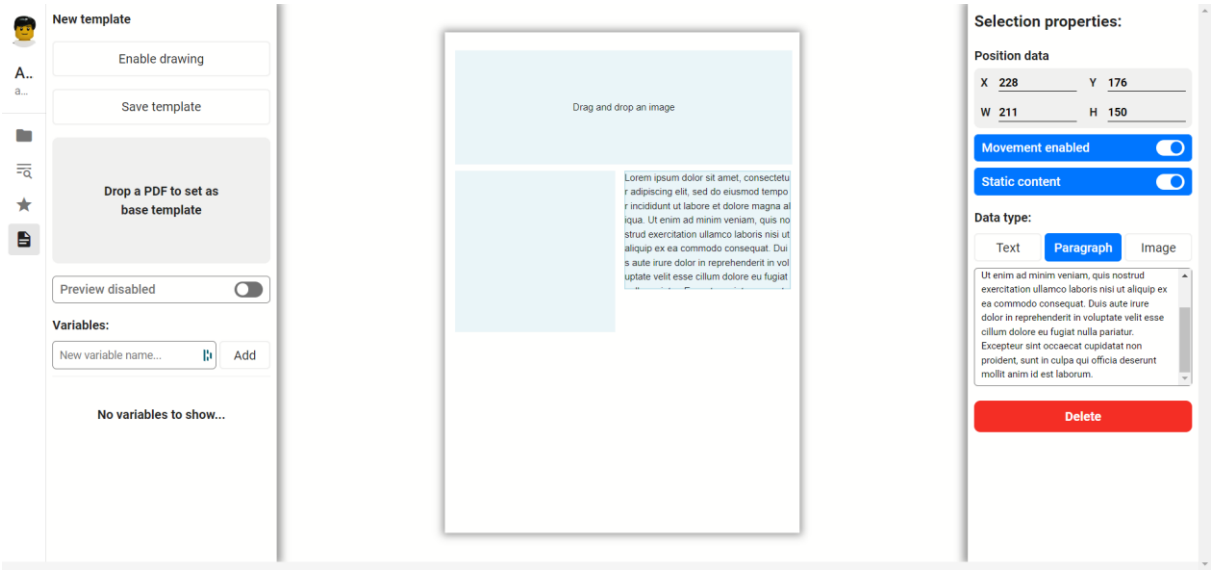
Slika 35 Izbornika za uređivanje varijabli

4.4.Statički sadržaj

Statički sadržaj se odnosi na sadržaj koji pridodajemo selekciji te se taj isti sadržaj koristi za generiranje svih konačnih PDF dokumenata. Sadržaj je zadan kao statički kod svake nove selekcije sa tipom podatka postavljenim na jednolinijski tekst. Tip podatka možemo promijeniti u paragraf i sliku te na taj način mijenjati konačno generiranje sadržaja.

Ukoliko tip selekcije postavimo kao slika, unutar selekcije se pojavljuje poruka za učitavanje slike kroz *drag-and-drop*. Slika učitana na ovaj način zauzima potpunu veličinu selekcije čak i ako mijenjamo veličinu. Treba napomenuti da ukoliko mijenjamo tip podatka, a već smo dodali statički sadržaj, taj sadržaj će biti izgubljen.

Primjer ovako postavljenog statičkog sadržaja bio bi potpis. Potpis mora biti na svakom generiranom dokumentu isti te se mora prikazivati na istome mjestu. To možemo postići tako da na mjestu na kojemu se treba naći potpis postavimo selekciju čiji je tip slika te učitamo sliku potpisa u „.png“ formatu zbog transparentne pozadine.



Slika 36 Uređivanje statičkog sadržaja

4.5.Spremanje i uređivanje predloška

Nakon izrade cijelog predloška, postavljanja dinamičkog i statičkog sadržaja te postavljanje svih ostalih postavki predloška, klikom na „*Save template*“ gumb predložak se sprema u bazu te ako je spremanje uspješno, na ekranu se pojavljuje poruka da je predložak uspješnog spremljen.

Nakon spremanja predloška, taj predložak, zajedno sa svim ostalim se nalazi u listi koja omogućuje pregled, uređivanje i brisanje stvorenih predložaka.

Ista stvar vrijedi i za ažuriranje već postojećeg predloška kojega sada uređujemo, klikom na gumb za spremanje, predložak se ažurira te se korisniku prikazuje poruka da je predložak ažuriran.

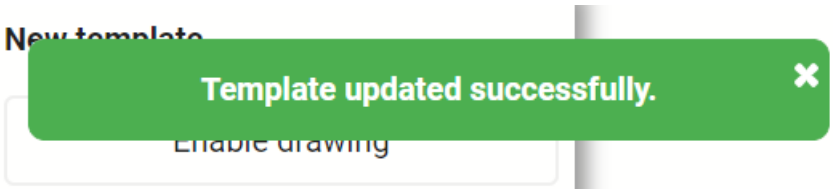
Predložak se može uređivati tako da na ekranu koji sadrži listu sa svim predlošcima, odaberemo jedan od njih klikom na gumb „*Edit*“ nakon čega se otvara sučelje za izradu predloška sa predloškom u onom stanju u kojemu je bio prilikom posljednjeg klika na gumb za pohranu.

Svaka promjena koja se desi nakon klika na gumb za spremanje neće biti odražena prilikom ponovnog otvaranja predloška. Ovdje dolazi do izražaja mana aplikacije, a to je da se korisnika ne upozori da ima nespremljenih promjena na predlošku prilikom zatvaranja ili osvježavanja sučelja.

4.6.Povratne informacije o procesu

Nakon određenih radnji, primarno onih vezanih za komunikaciju sa serverom, korisniku se prikazuje povratna informacija o uspješnosti.

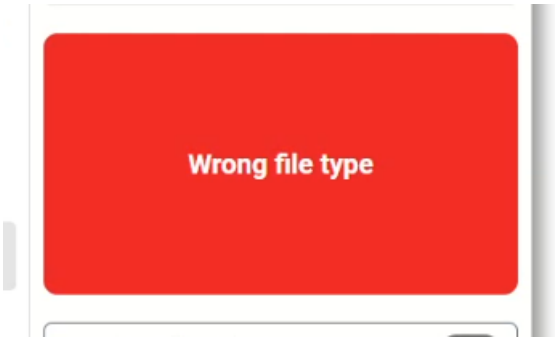
Primjerice, jedna od radnji za koju dobivamo povratnu informaciju je spremanje i ažuriranje predloška. Klikom na gumb „*Save template*“ započinje spremanje/ažuriranje predloška, unutar gumba se pojavljuje indikator izvođenja procesa koji nestaje nakon što je proces izvršen. Nakon što je proces spremanja završen, prikazuje se modal s porukom o uspješnom ili neuspješnom spremanju. Također, boja pozadine ovisi o rezultatu, crvena je indikator da je nešto pošlo po zlu dok je zelena pozadina indikator da je sve uredu.



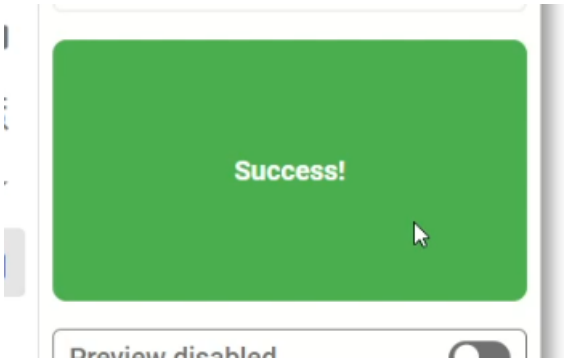
Slika 37 Povratna poruka o uspješnom generiranju dokumenata

Još jedno mjesto na kojemu se prikazuje povratna informacija korisniku je prilikom učitavanja PDF dokumenta koji služi kao podloga za generiranje dokumenata i učitavanja slike kao statičkog sadržaja selekcije.

Ovdje prije samog učitavanja datoteka provjeravamo format datoteka, to jest ako se stvarno radi o PDF dokumentu i slici. U ovom slučaju poruka se ispisuje unutar *drag-and-drop* zone koja služi za učitavanje dokumenta.



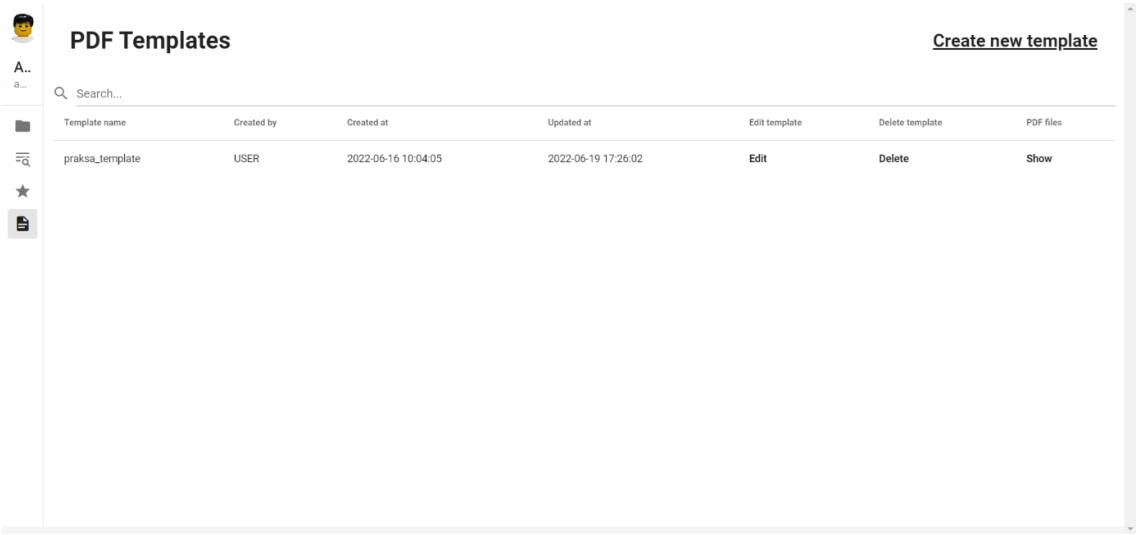
Slika 38 Povratna poruka o krivom formatu unesene datoteke



Slika 39 Povratna poruka o uspješnom učitavanju datoteke

4.7.Pregled predložaka

Nakon kreiranja i spremanja novog predloška, taj predložak se pojavljuje u listi svih predložaka zajedno sa informacijama poput datuma i vremena kreiranja te ažuriranja.



Slika 40 Popis stvorenih predložaka

Klikom na bilo koji od naziva stupaca u tablici s predlošcima sortira sve predloške po tom atributu. Primjerice, klikom na „Name“ svi predlošci se sortiraju abecedno, uzlazno po imenu. Ponovnim klikom na istu vrijednost predlošci se sortiraju silazno. Prilikom inicijalnog učitavanja stranice predlošci su sortirani po datumu ažuriranja, gdje je predložak koji je posljednji ažuriran na vrhu.

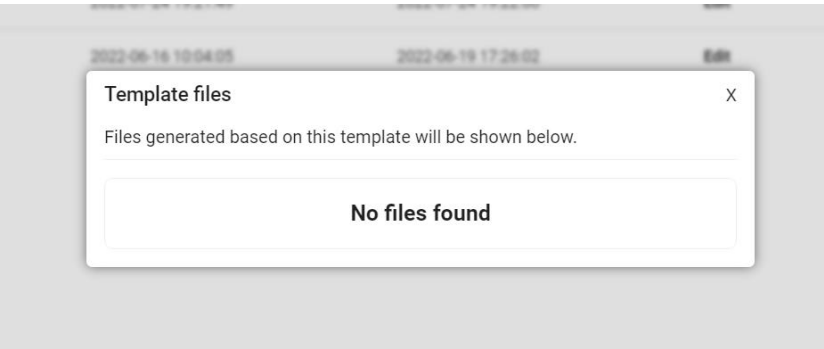
Također, u ovome pregledu omogućeno je brisanje pojedinih predložaka kao i prikaz nekoliko PDF dokumenata koji su generirani koristeći ovaj predložak.

Klikom na gumb „Show“ otvara se modal koji prikazuje imena deset generiranih PDF dokumenata, iz ovog izbornika korisnik može odabrati bilo koji broj ponuđenih PDF dokumenata te ih ili preuzeti ili obrisati.

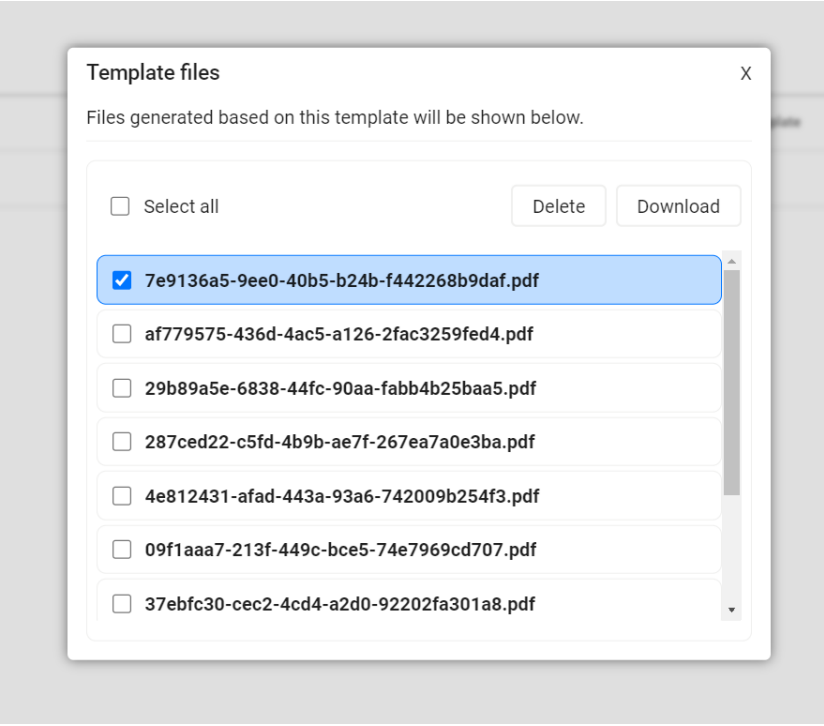
Ukoliko korisnik odluči preuzeti odabrane dokumente, format u kojemu se oni preuzmu ovisi o broju odabranih dokumenata, ako je odabran samo jedan dokument preuzima se kao PDF dokument, u suprotnome ako je odabrano više od jednog

dokumenta oni se preuzimaju u arhiviranom, točnije u *rar* formatu. Ime datoteke koja sadrži arhivirane PDF dokumente je ID predloška iz kojega su dokumenti generirani.

U slučaju da iz odabranog predloška nije generiran niti jedan PDF dokument, korisniku se ispisuje odgovarajuća poruka da niti jedan dokument nije pronađen.



Slika 41 Predložak bez generiranih dokumenta



Slika 42 Predložak s generiranim dokumentima

4.8.Ostale funkcionalnosti

Osim već navedenih funkcionalnosti, postoji još nekoliko različitih funkcionalnosti koje služe da bi korisniku olakšale izradu predloška:

- Kopiraj i zalijepi - korisniku je omogućeno kopiranje selekcija. Kopiranje i lijepljenje se vrši pritiskom tipki CTRL i C, to jest CTLR i V u kombinaciji. Ovime se kopira selekcija koja je trenutno odabrana kao i sav njen sadržaj. Ukoliko se kopira selekcija koja u sebi sadrži sliku kao statički sadržaj, slika se neće kopirati. Selekcija stvorena na ovaj način je za 10 piksela pomaknuta prema dolje i ulijevo.
- Pomicanje selekcije putem tipkovnice – ukoliko korisnik želi određenu selekciju pomaknuti precizno, te u tome ne uspijeva koristeći miš, na mogućnosti mu je korištenje tipki sa strelicama. Pritiskom jedne od tipki odabrana selekcija se pomiče u odabranu stranu za 5 piksela.
- Promjena veličine selekcije putem tipkovnice – na isti način na koji se selekcija može pomaknuti putem tipkovnice, odabranoj selekciji možemo i promijeniti veličinu. Pritiskom tipke SHIFT te zatim jedne od tipki sa strelicama mijenjamo veličinu odabrane selekcije. Strelice za gore i lijevo smanjuju, dok strelice za desno i dolje povećavaju veličinu selekcije.
- Isključivanje crtanja selekcije putem tipkovnice – umjesto pritiska gumba za isključivanje crtanja selekcija, korisniku je na mogućnosti pritisak tipke ESC koja čini istu stvar.

5.Zaključak

Kroz posljednjih nekoliko godina, a pogotovo zbog pojave COVID virusa, sve veći naglasak se stavlja na digitalizaciju. Digitalizacija nastoji za svakodnevne procese stvoriti digitalno rješenje, primjer toga su PDF dokumenti koji su danas u velikoj većini zamijenili obične papirnate dokumente.

Problem kod PDF dokumenta je što se oni još uvijek moraju popunjavati ručno, te ako se javlja potreba za velikom količinom tih dokumenata popunjenih s različitim podacima dolazimo do problema.

Cilj ovog diplomskog rada bio je rješavanje upravo ovog problema kroz web aplikaciju. Web aplikacija pruža izradu predloška koji se koristi za automatsko generiranje PDF dokumenata koristeći proslijeđene podatke.

Aplikacija je podijeljena na korisnički(frontend) dio, serverski(backend) dio te bazu podataka. Za izradu korisničkog dijela aplikacije, od bitnijih tehnologija korišteni su VueJs programski okvir(eng. *Framework*) te SASS predprocesor uz ostale manje biblioteke(eng. *libraries*). Serverski dio aplikacije izrađen je u programskom jeziku Python te koristi AIOHTTP za stvaranje HTTP servera na kojemu definiramo REST rute. Također, koristi se PonyORM za spajanje i komunikaciju s PostgreSQL bazom podataka.

Aplikacija ima mnogo potencijala za rast i razvoj, od samog unaprjeđenja trenutnih, do dodavanja posve novih funkcionalnosti. Jedna od većih restrikcija ove aplikacije je što su PDF dokumenti ograničeni na jednu stranicu te bi to dodavanje podrške za više stranica bila idealna nadogradnja projekta.

6.Literatura

1. VueJs, dosutpno na: https://vuejs.org/guide/introduction.html (23.07.2022.)
2. PostgreSQL, dostupno na: https://www.postgresql.org/about (23.07.2022.)
3. SASS, dostupno na: https://sass-lang.com/documentation/ (23.07.2022.)
4. AIOHTTP, dostupno na: https://docs.aiohttp.org/en/stable/web.html (23.07.2022)
5. Alexander Rusev, React vs Vue: The Core Similarities and Differences, dostupno na: https://mentormate.com/blog/react-vs-vue-the-core-differences/ (25.07.2022)
6. Front-end Frameworks, dostupno na: https://2021.stateofjs.com/en-US/libraries/front-end-frameworks/ (25.07.2022)
7. New Benchmarks Show Postgres Dominating MongoDB in Varied Workloads, dostupno na: https://www.enterprisedb.com/news/new-benchmarks-show-postgres-dominating-mongodb-varied-workloads (25.07.2022)
8. Jakub Romanowski, Which Major Companies Use PostgreSQL? What Do They Use It for?, dostupno na: https://learnsql.com/blog/companies-that-use-postgresql-in-business/ (25.07.2022)
9. Prachi, Asynchronous Programming,

7.Popis slika

8.Sažetak

Cilj ovog diplomskog rada je izrada web aplikacije koja predstavlja jedan dio i jedan proces puno većeg sustava. Svrha aplikacije jest stvaranje predloška za PDF dokumente koji se zatim mogu popunjavati s podacima u svrhu generiranja PDF dokumenta.

Aplikacija je rađena sa idejom da je dio puno većeg sustava koji se razvija u FIPU lab-u, taj veći sustav se oslanja na BPMN procesima. Makar je cijeli sustav zamišljen kroz BPMN procese, ova aplikacija nije jedan od procesa nego vanjski proces. To znači da korisnik putem ove aplikacije kreira PDF predložak koji se pohranjuje u bazu. Tek kada BPMN proces dođe do određenog koraka, putem API poziva generiraju se PDF dokumenti po prethodno kreiranom predlošku te se dohvaćaju.

Za izradu aplikacije korišteni su VueJS2 razvojni okvir(eng. *framework*) i SCSS za frontend dio projekta, Python programski jezik i AIOHTTP biblioteka(eng. *library*) za backend dio projekta te PostgreSQL baza podataka za pohranu podataka.

Razvoj aplikacije je bio vođen idejom neovisnosti aplikacije od ostatka sustava u kojemu se nalazi. Ovaj pristup olakšava integraciju aplikacije sa ostatkom sustava te ukoliko se javi potreba, uklanjanje aplikacije iz ostatka sustava.

Ključne riječi: VueJs, SCSS, Python, PostgreSQL, AIOHTTP, PDF, web aplikacije

9.Abstract

The goal of this master's thesis is to create a web application that represents a part of a bigger system. The purpose of the application is to create PDF templates that can be filled with data in order to generate PDF documents.

This application is made with the idea to be a part of a bigger system that is being developed at the FIPU lab. The system is being developed as a BPMN process, but this application is not part of the process, but instead an external process. This means that the user can, through this application create PDF templates that are stored in the database. Once the BPMN process reaches a certain step, through an API call, the PDF documents are generated based on the previously created templates and are retrieved.

The application is developed using the VueJs2 framework and SCSS for the frontend part, Python programming language, AIOHTTP library for the backend part and PostgreSQL as the database.

The application was developed as an independent project not connected to the rest of the system which means that the application can easily be integrated, and if needed, removed from the rest of the system.

Keywords: VueJs, SCSS, Python, PostgreSQL, AIOHTTP, PDF, web applications