Razvoj web aplikacije u okruženju Nuxt.js

Kramar, Petar

Undergraduate thesis / Završni rad

2022

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Organization and Informatics / Sveučilište u Zagrebu, Fakultet organizacije i informatike**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:211:422046

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Petar Kramar

RAZVOJ WEB APLIKACIJE U OKRUŽENJU NUXT.JS

ZAVRŠNI RAD

Varaždin, 2022.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Petar Kramar

Matični broj: 0016145312

Studij: Primjena informacijske tehnologije u poslovanju

RAZVOJ WEB APLIKACIJE U OKRUŽENJU NUXT.JS

ZAVRŠNI RAD

Mentor:

doc. dr. sc. Matija Novak

Varaždin, rujan 2022.

Petar Kramar

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

_____

Sažetak

Tema ovog rada je izrada web aplikacije za vođenje evidencije rada zaposlenika „Karen“ u Nuxt.js okviru. Cilj rada je ukazati na značajke modernog razvoja web aplikacija i softvera te prikazati kako se Vue.js i Nuxt.js okviri nadopunjuju. Osim modernih JavaScript okvira, za izradu rada korišteni su HTML i CSS. HTML kod je pisan u .vue dokumente unutar „<template>“ oznaka te su .vue dokumenti podijeljeni u mnoštvo komponenti i stranica. Osim HTML koda, za izradu moderne web aplikacije, potrebno ju je stilizirati. U ovoj web aplikaciji stilovi su pisani u .scss datoteke. SCSS je preprocesor za CSS koji pojednostavljuje CSS kod i čini ga čitkijim i urednijim. Osim toga, konačna web aplikacija je prikazana i demonstrirana kako bi se vidjelo kako sve ove tehnologije nadopunjuju jedna drugu.

Ključne riječi: web aplikacija, html, css, javascript, vue.js, nuxt.js, web development

Sadržaj

1. Uvod 1
2. Web aplikacije 2
2.1. HyperText Markup Language 2
2.1.1. HTML sintaksa 2
2.2. CSS 3
2.2.1. CSS preprocesor Sass 4
2.3. JavaScript 6
2.3.1. Dinamičnost JavaScripta 6
2.3.2. Korištenje JavaScripta 6
2.3.2.1. Linijski JavaScript 6
2.3.2.2. Interni JavaScript 7
2.3.2.3. Vanjski JavaScript 7
2.3.3. Varijable u JavaScriptu 7
2.3.4. Vrste podataka u JavaScriptu 8
3. Vue.js 10
3.1. Životni ciklus Vue.js aplikacije 10
3.1.1. Vue.js predlošci 13
3.1.2. Vue.js tranzicije i reaktivnost 13
3.1.3. Vue.js usmjeravanje 13
3.1.4. Vue.js mixins 14
3.1.5. Atributi .vue datoteke 14
3.1.5.1. Components 14
3.1.5.2. Data 14
3.1.5.3. Computed 15
3.1.5.4. Methods 16
3.1.5.5. Props 16
3.1.5.6. Watch 17
4. Nuxt.js 18
4.1. Način rada Nuxt.js okvira 19
4.2. Struktura Nuxt.js projekta 20
5. Web aplikacija Karen 23
5.1. Arhitektura aplikacije 23
5.1.1. Dijagram slučaja upotrebe 24
5.1.2. ERA model 25
5.2. Komponente 27

5.2.1. Modal.vue 27
5.2.2. Calendar.vue 29
5.3. Stranice 32
5.3.1. Stranica za prijavu 32
5.3.2. Odgovori 33
5.3.3. Odjeli 34
5.3.4. Admin 35
5.3.5. Pitanja 37
5.4. Responzivni dizajn 38
6. Kritički osvrt 42
7. Zaključak 43
Literatura 44
Popis slika 46

1. Uvod

S obzirom na online svijet u kojem živimo i radimo, praćenje rada zaposlenika i članova tima postaje sve teže i kompleksnije. Voditelji odjela ne mogu detaljnije pratiti rad i projekte ostalih članova tima koji rade od kuće. Stoga je osmišljena aplikacija Karen. Karen je aplikacija koja prati i vodi evidenciju o radu zaposlenika kroz nekoliko kratkih pitanja na koje zaposlenici svojevoljno odgovaraju. Karen je aplikacija izrađena u Vue.js i Nuxt.js okvirima te joj cilj nije kontroliranje zaposlenika, već jednostavno vođenje evidencije kako bi komunikacija unutar tima bila jednostavnija i lakša. Autor ovog završnog rada je zaposlenik tvrtke Shape te će se web aplikacija Karen koristiti kao interni projekt unutar Shapea. Web aplikacija Karen će se nastaviti ažurirati nakon završetka pisanja završnog rada.

Ovaj završni rad podijeljen je na dvije tematski povezane cjeline. Prva cjelina predstavlja teorijski opis web aplikacija te teorijsku razradu teme, a drugi dio rada predstavlja primjer i način funkcioniranja web aplikacije. U prvom dijelu završnog rada navedeni su osnovni alati potrebni za izradu web aplikacije, kao što su HyperText Markup Language (HTML), Cascading Style Sheets (CSS) i JavaScript te je opisan i CSS preprocesor Syntactically Awesome Style Sheets, odnosno Sassy CSS. Osim toga, kako bi se olakšala izrada aplikacije i modernizirale tranzicije i funkcionalnosti aplikacije, korišteni su okviri JavaScripta Vue.js i Nuxt.js, koji su također detaljno teorijski opisani.

U drugom dijelu završnog rada prikazana je arhitektura aplikacije pomoću dva dijagrama koja opisuju funkcionalnosti web aplikaciji te proces izrade web aplikacije Karen. Drugi dio završnog rada detaljnije prikazuje međusobnu povezanost svih prethodno navedenih tehnologija. Svi prethodno navedeni teorijski pojmovi korišteni su za izradu aplikacije te se demonstrira kako aplikacija funkcionira kao cjelina.

2. Web aplikacije

U ovom poglavlju navedene su osnovne tehnologije potrebne za izradu responzivnih web aplikacija. Najvažnije tehnologije potrebne za razvoj web aplikacija su HTML, CSS i JavaScript. Kako bi se olakšao i pojednostavio kod, umjesto čistog CSS-a, korišten je preprocesor Sass. Osim osnovnih jezika, za potrebe izrade ovog završnog rada korišteni su progresivni aplikacijski okviri, točnije Vue.js i Nuxt.js.

2.1. HyperText Markup Language

HyperText Markup Language (HTML) je osnovni jezik koji se upotrebljava za izradu web stranica. HTML je osnovan 1990. godine. Osnovao ga je Timothy Berners-Lee. Kroz godine, HTML je ažuriran i mijenjan. Najnovija verzija HTML-a, odnosno HTML 5.2 objavljen je 14. prosinca 2017. godine. HTML daje preglednicima podatke o strukturi i sadržaju učitane web stranice, a preglednik čita HTML kod i na temelju toga oblikuje web stranicu. HTML je zapravo jezik pomoću kojeg komuniciraju autor koda i preglednik. Važno je napomenuti kako HTML nije programski jezik, s obzirom da HTML ne može izvršiti niti najjednostavniju matematičku operaciju. [1]

2.1.1. HTML sintaksa

Sintaksa HTML-a je relativno jednostavna. Svaki HTML dokument se sastoji od dva glavna elementa, a to su zaglavlje i tijelo. I zaglavlju HTML-a se definiraju elementi koji se ne prikazuju na web stranici, dok se u tijelu HTML-a definiraju svi ostali elementi koji su prikazani na web stranici. Svaki HTML element mora biti unutar oznaka, pa se tako cijeli HTML kod mora nalaziti unutar određenih oznaka. U nastavku se nalazi primjer jednostavnog HTML dokumenta. [2]

```
<!DOCTYPE html>
<html>
     <head>
            <title>Web stranica</title>
     </head>
     <body>
            <h1>Ovo je glavni naslov web stranice!</h1>
            <p>Ovo je paragraf teksta web stranice.</p>
     </body>
</html>
```

2.2. CSS

S obzirom da HTML definira čistu strukturu i sadržaj web stranice, on često ne izgleda privlačno te nije responzivan. Upravo iz tog razloga, definiran je Cascading style sheet (CSS), odnosno stilski jezik za HTML. CSS je globalni standard za stilsko uređivanje HTML dokumenata. CSS se razvijao tijekom godina, ali je uvijek ostao jednostavan za korištenje i čitljiv. CSS se i dan danas koristi kao standardni stilski jezik za izradu web stranica. Svaka CSS oznaka mora imati dva osnovna dijela, a to su selektor i deklaracija. Postoje četiri glavna načina selektiranja HTML elementa, a to su implicitno, eksplicitno, jednoznačno i po potrebi. Deklaracija se dijeli na dva dijela, a to su svojstvo i vrijednost. Svojstvo i vrijednost nalaze se unutar vitičastih zagrada. [3]

Primjer izgleda CSS koda je:

```
selektor {
        svojstvo: vrijednost;
}
```

U nastavku se nalazi primjer CSS naredbi na primjeru prethodnog HTML koda. U primjeru je vidljivo kako se CSS kod može pisati unutar „style“ oznaka u zaglavlju. Također su vidljivi različiti načini odabiranja elemenata koje je potrebno stilizirati. Element „body“ odabran je implicitno, odnosno CSS uputa temelji se na pridruživanju HTML elementu. U ovom primjeru naznačeno je da će boja fonta unutar „body“ elementa biti crvena, veličina fonta 20px i tekst će biti poravnat ulijevo.

Sljedeći način odabiranja elemenata je eksplicitno, odnosno pomoću klase. U HTML naznačeno je kako „h1“ element ima klasu „naslov“. U CSS kodu se pomoću znaka točke odabire eksplicitni način korištenja CSS uputa. Na taj način su klasi „naslov“ dodijeljene gornje i donje margine te pozadinska boja pomoću heksadecimalne vrijednosti. Sljedeći način CSS odabira je uz atribut jednoznačno. U primjeru je vidljivo kako je elementu „p“ dodan id koji ima vrijednost „paragraf“. Pomoću znaka „#“ i odabiremo id „paragraf“ te tako dodajemo stilske deklaracije. Elementu „p“ dodan je i CSS po potrebi, odnosno linijski CSS u kojem je navedeno da se elementu „p“ dodaje crna boja fonta.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Web stranica</title>
    <style type="text/css">
        body {
            color: red;
```

```
                font-size: 20px;
                text-align: left;
        }

        .naslov {
                margin-top: 100px;
                margin-bottom: 250px;
                background-color: #f42fff;
        }

        #paragraf {
                text-align: center;
                padding: 250px;
        }
    </style>
  </head>
 <body>
    <h1 class="naslov">Ovo je glavni naslov web stranice!</h1>
    <p id="paragraf" style="color: black">Ovo je paragraf teksta web
   stranice.</p>
 </body>
</html>
```

2.2.1. CSS preprocesor Sass

Napretkom web tehnologija postalo je teško snalaziti se u ogromni CSS dokumentima. Kada se uzme u obzir nekoliko vrsta selektora te nekoliko stotina deklaracija, CSS kod može postati neuredan i činiti se kompliciranijim nego što zapravo je. Kako bi se rješio ovaj problem, predstavljeno je nekoliko CSS preprocesora. CSS preprocesori ne mijenjaju glavnu sintaksu CSS koda, ali dodaju mnoštvo novih mogućnosti koje čine kod čitljivijim te nude dodatne funkcionalnosti. Jedan od najpopularnijih preprocesora je Sass. Sass uklanja sintaksu koja nije potrebna te pruža mogućnost ugnježđivanja selektora. [4]

Upravo ovo čini CSS kod čišćim i jednostavnijim za snalaženje. Sass dolazi u dvije sintakse, a to su osnovna Sass sintaksa i Sassy CSS, koji je korišten za potrebe ovog rada.

SCSS zadržava glavnu sintaksu CSS-a, naredba se i dalje sastoji od selektora i deklaracije. U nastavku se nalazi primjer SCSS koda na temelju CSS-a prethodnog poglavlja. Kao što je vidljivo, umjesto eksplicitnog i jednoznačnog odabira elemenata, korišteno je ugnježđivanje, a konačni rezultat web stranice izgleda isto. CSS po potrebi ostaje

nepromijenjen. Međutim, ovo nisu sve mogućnosti koje SCSS pruža. Neke od ostalih mogućnosti i prednosti SCSS-a biti će prikazane u praktičnom dijelu projekta.

```
<!DOCTYPE html>
<html>
     <head>
       <title>Web stranica</title>
       <style type="text/scss">
            body {
                    color: red;
                    font-size: 20px;
                    text-align: left;

                    h1 {
                            margin-top: 100px;
                            margin-bottom: 250px;
                            background-color: #f42fff;
                    }

                    p {
                            text-align: center;
                            padding: 250px;
                    }
            }
            </style>
     </head>
     <body>
            <h1>Ovo je glavni naslov web stranice!</h1>
     </body>
</html>
```

2.3. JavaScript

JavaScript je dinamički programski jezik koji se koristi za web razvoj, u web aplikacijama, za razvoj igara i još mnogo toga. Omogućuje implementaciju dinamičkih značajki na web stranicama koje nije moguće učiniti samo s HTML-om i CSS-om. Mnogi preglednici koriste JavaScript kao skriptni jezik za obavljanje dinamičnih stvari na web stranici. Bilo koji padajući izbornik koji je potrebno kliknuti za prikaz ili promjena boja elemenata na stranici je učinak JavaScripta. Bez JavaScripta, sve što bi postojalo na webu bili bi HTML i CSS. Međutim, sami HTML i CSS ograničavaju web stranice. Bez JavaScripta, većina web stranica izgledala bi statično, a dinamičke promjene postojale bi samo kao CSS animacije. [5]

2.3.1. Dinamičnost JavaScripta

HTML definira strukturu web dokumenta i njegov sadržaj. CSS deklarira različite stilove za sadržaj koji se nalazi u web dokumentu. HTML i CSS se često nazivaju označnim jezicima, a ne programskim jezicima, jer oni, u svojoj srži, daju oznake za dokumente s vrlo malo dinamike. JavaScript je, s druge strane, dinamički programski jezik koji podržava matematičke izračune, omogućuje dinamičko dodavanje HTML sadržaja, stvara dinamičke stilske deklaracije, dohvaća sadržaje s druge web stranice i još mnogo toga. [6]

2.3.2. Korištenje JavaScripta

Baš kao i kod CSS-a, JavaScript se može koristiti u HTML-u na različite načine. Načini korištenja JavaScripta u HTML-u su:

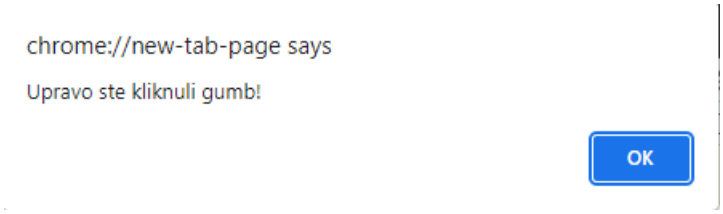
1. Linijski JavaScript
2. Interni JavaScript
3. Vanjski JavaScript

2.3.2.1. Linijski JavaScript

Linijski JavaScript funkcionira slično kao i linijski CSS. Linijski JavaScript se piše kao atribut u HTML oznakama. Na primjer, HTML oznake imaju atribute događaja koji vam omogućuju izvršavanje nekog koda u liniji kada se događaj pokrene. Primjerice:

```
<button onclick="alert('Upravo ste kliknuli gumb!')">Klikni me</button>
```

Ovo je primjer linijskog JavaScripta. HTML kod prikazuje gumb na kojem se nalazi tekst „Klikni me“. Ukoliko korisnik klikne na taj gumb, tada se otvara „alert“, odnosno obavijest web preglednika koja prikazuje tekst koji je naveden u zagradama nakon „alert“ funkcije.



Slika 1: Prikaz alert funkcije

2.3.2.2. Interni JavaScript

Baš kao i *style* oznaka za deklaracije stila unutar HTML stranice, *script* oznaka postoji za JavaScript. Unutar *script* oznake piše se JavaScript kod, a script oznake moguće je otvoriti bilo gdje unutar HTML datoteke. Primjer ovakve oznake je:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Primjeri JavaScripta</title>
</head>
<body>
    <script>
        function upozorenje(){
          alert("Upozorenje unutar script tagova")
          }
    </script>
</body>
</html>
```

2.3.2.3. Vanjski JavaScript

Najčešće korišten način pisanja JavaScripta je vanjski JavaScript, odnosno JavaScript u novoj datoteci. Kada se piše vanjski JavaScript, potrebno je u HTML datoteci upisati script oznake sa poveznicom na vanjsku JavaScript datoteku. Primjer HTML koda je:

```
<script src="/script.js"></script>
```

Nakon toga, u vanjsku JavaScript datoteku može se pisati čisti JavaScript kod te nije potrebno dodavati dodatne script oznake. JavaScript datoteka mora imat .js ekstenziju.

```
alert("I am inside an external file");
```

2.3.3. Varijable u JavaScriptu

Varijable su fundamentalne za sve programske jezike. Varijable se koriste za pohranjivanje podataka, poput niza teksta, brojeva ili ostalih vrsta podataka. Podaci ili vrijednosti pohranjeni u varijablama mogu se deklarirati, ažurirati i dohvatiti kad god je to

potrebno. Općenito, varijable su simbolična imena za vrijednosti. Možete stvoriti varijablu s ključnom riječi „let“ i ključnom riječi „const“, dok se operator dodjele (=) koristi za dodjelu vrijednosti varijabli. Varijable definirane pomoću ključne riječi „const“ se ne mogu ponovno dodijeliti kasnije u kodu. Primjer deklariranja varijable je sljedeći:

```
Let ime = „Petar“;
Let broj = 2;
Let gradovi = [„Zagreb“, „Varaždin“, „Vinkovci“];
```

2.3.4. Vrste podataka u JavaScriptu

Postoji šest osnovnih tipova podataka u JavaScriptu koji se mogu podijeliti u tri glavne kategorije, a to su primitivni, složeni i posebni tipovi podataka. String, broj i boolean su primitivni tipovi podataka. Objekt, niz i funkcije su složeni tipovi podataka. Undefined i null su posebni tipovi podataka. Primitivni tipovi podataka mogu sadržavati samo jednu vrijednost u isto vrijeme, dok složeni tipovi podataka mogu sadržavati zbirke vrijednosti i složenije entitete. [7]

Tipovi podataka u JavaScriptu su:

- String - podatkovni tip string je niz od jednog ili više znakova koji se mogu sastojati od slova, brojeva ili simbola. Stringovi se koriste za predstavljanje tekstualnih podataka, odnosno nizova znakova. Stringovi se pišu između jednostrukih i dvostrukih navodnika.
- Broj - brojčani tip podataka koristi se za predstavljanje pozitivnih ili negativnih brojeva sa ili bez decimalnog mjesta ili brojeva napisanih eksponencijalnom notacijom. Brojevi u JavaScriptu također uključuju neke posebne vrijednosti kao što su Infinity, -Infinity i NaN. Infinity predstavlja matematičku beskonačnost, koja je veća od bilo kojeg broja.
- Boolean – ovaj tip podataka može sadržavati samo dvije vrijednosti, a to su istina i laž, odnosno true ili false. Boolean se najčešće koristi za pohranjivanje vrijednosti kao što su da i ne ili uključeno ili isključeno. Boolean vrijednosti također dolaze kao rezultat usporedbi.
- Polje - Polje je vrsta objekta koji se koristi za pohranjivanje više vrijednosti u jednu varijablu. Svaka vrijednost u polju ima numerički položaj, poznat kao njegov indeks i može sadržavati podatke bilo koje vrste podataka, pa čak i druge nizove. Indeks niza počinje od 0, tako da je prvi element niza 0, a ne 1.

- Objekt - objekt je složena vrsta podataka koja vam omogućuje pohranjivanje zbirki podataka. Objekt sadrži svojstva definirana kao par, odnosno ključ-vrijednost. Ključ je uvijek string, ali vrijednost može biti bilo koja vrsta podataka, čak i funkcije ili drugi objekti.
- Undefined - nedefinirani tip podataka može imati samo jednu vrijednost, a to je posebna vrijednost undefined. Ako je varijabla deklarirana, ali joj nije dodijeljena vrijednost, ima vrijednost undefined.
- Null – ovaj tip podataka može imati samo jednu vrijednost, a to je null vrijednost. Vrijednost null znači da nema vrijednosti. Nije ekvivalent praznom nizu ili 0, nego je tip podataka null jednostavno ništa. Varijabla se može eksplicitno isprazniti od njezinog trenutnog sadržaja dodjeljivanjem null vrijednosti.
- Funkcija - objekt koji se može pozvati i koji izvršava blok koda. Budući da su funkcije objekti, moguće ih je dodijeliti varijablama. Funkcije se mogu koristiti na bilo kojem mjestu gdje se može koristiti bilo koja druga vrijednost. Funkcije se mogu pohraniti u varijablama, objektima i nizovima. Funkcije se mogu proslijediti kao argumenti drugim funkcijama, te se funkcije mogu vratiti iz funkcija.

3. Vue.js

Vue.js je progresivni JavaScript okvir otvorenog koda koji se koristi za razvoj interaktivnih web sučelja. Vue.js je jedan od najpoznatijih okvira koji se koriste za pojednostavljenje web razvoja. Vue.js je kreirao Evan You, bivši zaposlenik Googlea. Prva verzija Vue.js-a objavljena je u veljači 2014. Vue.js je dizajniran na način da se fokusira na laku integraciju s drugim bibliotekama. Osim toga, Vue.js omogućava jednostavan razvoj sofisticiranih jednostraničnih aplikacija. [8]

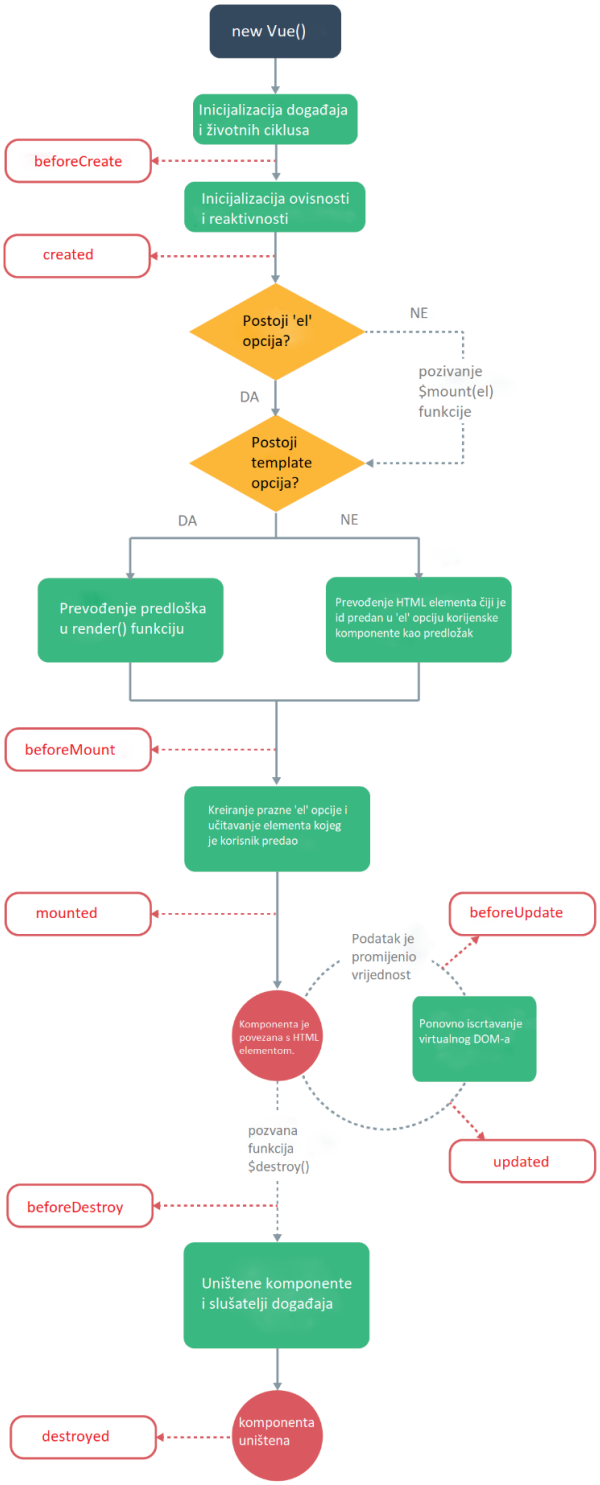
3.1. Životni ciklus Vue.js aplikacije

Sve moderne web aplikacije imaju životni ciklus te je on izuzetno važan za razvoj web aplikacije. Životni ciklus programeru olakšava pronalazak pogrešaka jer mu govori na koji način je došlo do problema, odnosno kada je problem nastao. Metode životnog ciklusa su prozor u to kako biblioteka koja je korištena zapravo funkcionira. Metode životnog ciklusa omogućuju vam da znate kada je vaša komponenta stvorena, dodana u DOM, ažurirana ili uništena. Kreacijske metode su prve koje se pokreću u komponenti. Ove metode omogućuju izvođenje radnji prije nego je komponenta uopće dodana u DOM. Sljedeće metode su „montažne“ metode, odnosno „mounted“ metode. Ove metode su najčešće korištene te one omogućuju pristup komponenti neposredno prije i nakon prvog renderiranja. Koriste se kada je potrebno pristupiti ili modificirati DOM komponente prije ili nakon početnog renderiranja. Nakon metoda za montiranje, dolaze metode za ažuriranje. Metode za ažuriranje se pozivaju svaki put kad se komponenta promijeni ili nešto uzrokuje njezino ponovno prikazivanje. Ove metode koriste se kada je potrebno znati kada se komponenta ponovno prikazuje. Na posljetku, metode za uništavanje aktiviraju se kada se komponenta uništava i uklanja iz DOM-a. [9]

- beforeCreate – ova metoda se poziva odmah kada se instanca inicijalizira, prije obrade drugih opcija kao što su data() ili computed.
- created - kada se ova metoda pozove, postavljeni su reaktivni podaci, izračunata svojstva, metode i promatrači. Međutim, faza *mounted* još nije započela i *$el* metoda još nije dostupna
- beforeMount – kada se ova metoda pozove, komponenta je završila sa postavljanjem svog reaktivnog stanja te će se prvi put izvršiti DOM efekt renderiranja

- mounted – koristi se za izvođenje metoda kojima je potreban pristup renderiranom DOM-u
- beforeUpdate – koristi se za pristup DOM stanju prije nego što Vue ažurira DOM. Osim toga, sigurno je mijenjanje stanja komponenti unutar ove metode
- updated – poziva se nakon bilo kojeg DOM ažuriranja komponente. Može biti uzrokovano različitim promjenama stanja
- beforeDestroy – izvođenje radnji neposredno prije uništenja. Komponenta će i dalje biti potpuno funkcionalna
- destroyed – uništeno je sve što je bilo vezano uz komponentu. Najčešće se koristi za slanje obavijesti udaljenom poslužitelju da je komponenta uništena

Sljedeća slika prikazuje životni ciklus Vue.js aplikacije prema [9]:



Slika 2: Životni ciklus Vue.js aplikacije

3.1.1. Vue.js predlošci

Vue.js koristi sintaksu predloška temeljenu na HTML-u koja omogućuje deklarativno vezanje prikazanog DOM-a na podatke instance osnovne komponente. Svi Vue.js predlošci sintaktički su valjani HTML koji se može analizirati preglednicima usklađenim sa specifikacijama i HTML procesorima. Vue.js kompilira predloške u visoko optimizirani JavaScript kod. U kombinaciji sa sustavom reaktivnosti, Vue.js je u stanju inteligentno odrediti minimalni broj komponenti za ponovno renderiranje i primijeniti minimalnu količinu DOM manipulacija kada se stanje aplikacije promijeni. [10]

3.1.2. Vue.js tranzicije i reaktivnost

Vue.js pruža različite načine za primjenu efekata prijelaza, odnosno tranzicije na HTML elemente kada se dodaju ili ažuriraju u DOM. Vue.js tranzicije uključuju automatsko primjenjivanje klase za CSS prijelaze i animacije, integriranje CSS animacija treće strane, upotreba JavaScripta za direktno manipuliranje DOM-om te integriranje JavaScript biblioteke treće strane. Kada se umetne ili ukloni element umotan u prijelaznu komponentu „<transition>“, Vue.js će automatski odlučiti hoće li na ciljani element primijeniti prijelaze ili animacije. Ukoliko se to dogodi, biti će dodane prijelazne klase CSS-a.

Jedna od najistaknutijih značajki Vue.js-a je nenametljiv sustav reaktivnosti. Stanje komponenti su reaktivni JavaScript objekti. Kada se JavaScript objekti izmijene, prikaz se ažurira. Reaktivnost čini upravljanje stanjem komponenti jednostavnim i intuitivnim.

3.1.3. Vue.js usmjeravanje

Usmjeravanje je jedna od mnogih značajki koje nudi Vue.js kako bi se korisnicima omogućilo prebacivanje između stranica bez osvježavanja svaki put kada se stranica učita. To rezultira glatkim prijelazima između stranica dajući bolji osjećaj za korisnika. Vue Router službeni je usmjerivač za Vue.js. Duboko se integrira s jezgrom Vue.js-a kako bi izrada jednostranih aplikacija s Vue.js bila izuzetno jednostavna. Vue.js aplikacija zapravo zbirka komponenti, a Vue Router daje sve alate potrebne za navigaciju između komponenti Vue.js-a. Kada se Vue.js aplikacija jednom učita, stranica se više nikada ne osvježava prilikom korištenja navigacije aplikacije. [11]

3.1.4. Vue.js mixins

Mixins su fleksibilan način za distribuciju funkcija za višekratnu upotrebu za Vue komponente. Mixins je objekt i može sadržavati bilo koju opciju komponente. Kada komponenta koristi mixin, sve opcije u mixinu bit će umiješane u vlastite opcije komponente. Kada mixin i sama komponenta sadrže opcije koje se preklapaju, bit će spojene. [12]

3.1.5. Atributi .vue datoteke

Unutar postavljanja .vue datoteke, mogu se grupirati dijelovi koda prema logičkom interesu. Atributi .vue datoteke su dijelovi koda koji se pišu unutar određenog objekta. U nastavku su detaljnije opisani određeni atributi koji su se najčešće koristili prilikom izrade „Karen“ web aplikacije. [10]

3.1.5.1. Components

Komponente omogućuju dijeljenje korisničkog sučelja na neovisne dijelove koji se mogu ponovno koristiti. Uobičajeno je da se aplikacija organizira u stablo ugniježđenih komponenti. Ovo je vrlo slično načinu na koji se ugnježđuju HTML elementi, ali Vue.js implementira vlastiti model komponente koji omogućuje zadržavanje prilagođenog sadržaja i logike u svakoj komponenti. Da bi se koristila podređena komponenta, potrebno ju je uvesti u nadređenu komponentu.
Dodavanje komponenti u .vue datoteku vrši se na sljedeći način:

```
import navigation from './navigation.vue'
export default {
  components: {
    navigation
  }
}
```

3.1.5.2. Data

Data je funkcija koja vraća objekt. Očekuje se da će dana funkcija vratiti običan JavaScript objekt, koji će Vue učiniti reaktivnim. Objekt unutar dana funkcije je sastavljen od svih varijabli koje se planiraju koristiti na stranici. Primjer:

```
data(){
    return{
      width: 100,
      height: 200
    }
  }
```

3.1.5.3. Computed

U Vue.js, computed svojstva, odnosno izračunata svojstva omogućuju stvaranje svojstva koje se može koristiti za modificiranje, manipuliranje i prikaz podataka unutar komponenti na čitljiv i učinkovit način. Vue.js atribut computed može se koristiti za izračun i prikaz vrijednosti na temelju vrijednosti ili skupa vrijednosti u podatkovnom modelu. Također može imati neku prilagođenu logiku koja se predmemorira na temelju computed ovisnosti, što znači da se ne učitava ponovno, ali ima ovisnost koja se mijenja, dopuštajući computedu da donekle sluša promjene i djeluje u skladu s tim. [13]

Computed atribut se može koristiti za rješavanje složenijih problema. Osim toga, computed može filtrirati podatke, rukovanje izračunima, provjeravanje istinosti uvjeta i još mnogo toga. Primjer computed atributa koji ispisuje vrijednost dobivenu funkcijom count je:

```
<template>
  <div class="hello">
    <h1>{{ count }}</h1>
  </div>
</template>


<script>
export default {
  name: "HelloWorld",
  data() {
    return {
      shopNumber: 2
    }
  },
  computed: {
    count: function() {
      return 'The shop number is ' + this.shopNumber
    }
  }
};
</script>
```

3.1.5.4. Methods

Vue metode su funkcije povezana sa svakom Vue instancom i stvorene sa methods svojstvom. Vue metode se koriste za izvođenje određenih radnji kada korisnik komunicira s elementom pomoću v-on direktive, poput klika na gumb ili unosa podataka u tekstualni unos. Primjer metoda u Vue aplikaciji koja prilikom klika na gumb ili unošenja teksta unutar input elementa otvara alert prozor i ispisuje tekst u konzolu:

```
<template>
  <div>
    <button @click="handleClick">Click me</button>
    <input @input="handleInput" />
  </div>
</template>

<script>
export default {
  methods: {
    handleClick() {
      alert('You clicked the button.');
    },
    handleInput() {
      console.log('Text input changed');
    },
  },
};
</script>
```

3.1.5.5. Props

Props je posebna ključna riječ koja označava svojstva. *Props* se mogu registrirati na komponenti za prijenos podataka od nadređene komponente do jedne od njenih podređenih komponenti. Ovo je puno lakše u usporedbi s korištenjem biblioteka za upravljanje stanjem. Podaci u *props* atributu mogu teći samo u jednom smjeru, a to je od vrha ili nadređene komponente prema dnu ili podređenim komponentama. To jednostavno znači da se podaci ne mogu proslijediti od djeteta do roditelja. *Props* služe samo za čitanje te ih dijete ne može mijenjati jer roditelj posjeduje vrijednost *propa*. [14]

Primjer korištenja *props* atributa za dohvaćanje i ispisivanje teksta unutar HTML dokumenta:

```
<template>
  <p>Hi {{ firstName }} {{ lastName }}</p>
</template>
```

```
<script>
export default {
  props: [
    'firstName',
    'lastName'
  ],
}
</script>
```

3.1.5.6. Watch

Vue.js *watch* svojstvo omogućuje slušanje podataka komponente i pokretanje kad god se promijeni određeno svojstvo. *Watch* ili svojstvo promatranja jedinstvena je značajka Vue.js-a koja omogućuje praćenje svojstva stanja komponente i pokretanje funkcije kada se vrijednost tog svojstva promijeni. Primjer *watch* atribtuta za pretvaranje kilometara u metre i obrnuto je:

```
   <template>
      <div>
         Kilometers : <input type = "text" v-model = "kilometers">
         Meters : <input type = "text" v-model = "meters">
      </div>
   </template>

<script>
export default {
  data: {
    kilometers : 0,
    meters:0
    },
  watch : {
    kilometers:function(val) {
    this.kilometers = val;
    this.meters = val * 1000;
  },
    meters : function (val) {
    this.kilometers = val/ 1000;
    this.meters = val;
    }
  }
}
</script>
```

4. Nuxt.js

Nuxt.js je okvir za renderiranje na strani poslužitelja izgrađen na Vue.js. Apstrahira većinu složene konfiguracije uključene u upravljanje asinkronim podacima, međuprogramom i usmjeravanjem. Osim toga, Nuxt.js pomaže u strukturiranju Vue.js aplikacija korištenjem industrijske standardne arhitekture za izgradnju jednostavnih Vue.js aplikacija. [15]

Nuxt.js funkcionira na isti način na koji funkcionira okvir na strani poslužitelja kada korisnik posjeti web stranicu. Ako je omogućeno prikazivanje na strani poslužitelja, zahtjevi se prikazuju na poslužitelju svaki put kada korisnik zatraži stranicu, stoga je potreban poslužitelj kako bi mogao poslužiti stranicu za svaki zahtjev. Također, ako je omogućeno prikazivanje na strani klijenta, ono prikazuje sadržaj stranice u pregledniku pomoću JavaScripta.

Prednosti korištenja Nuxt.js okvira su:

- Jednostavno stvaranje univerzalnih aplikacija - Nuxt.js čini stvaranje aplikacija za renderiranje na strani poslužitelja vrlo jednostavnim. Stvaranje aplikacije na strani poslužitelja sa samo Vue.js okvirom je izuzetno kompliciranо i ima mnogo konfiguracijskih opcija dostupnih. Međutim, značajka za renderiranje na strani poslužitelja već je ugrađena u Nuxt.js i jednostavna je za korištenje. Nuxt.js omogućuje pristup svojstvima *isClient* i *isServer* kako bi Nuxt.js okvir znao prikazuje li nešto na strani klijenta ili na strani poslužitelja. Osim toga, Nuxt.js pruža metodu *asyncData* posvećenu dohvaćanju i prikazivanju podataka na strani poslužitelja.
- Statički prikaz Vue aplikacija s univerzalnim prednostima - Statički generirane web stranice su u porastu u web industriji, a jednom *nuxt generate* naredbom moguće je generirati statičnu verziju web stranice, sa svim HTML-om s odgovarajućim rutama.
- Unapređenje SEO-a – Problem kod JavaScript okvira i SEO-a, poput Vue.js okvira je u tome što stranice izvode dinamičku populaciju sadržaja tijekom učitavanja. To znači da kada Google učita dotičnu web-lokaciju, glavni sadržaj web-lokacije još uvijek nedostaje većinu vremena, odnosno dok se ne otvori stranica gdje se nalazi sadržaj te se on učita. Ovo može biti velika prednost kada je u pitanju brzina, ali i veliki nedostatak kada je u pitanju SEO. Ovdje na

snagu dolazi Nuxt.js. Kako bi poboljšao SEO, Nuxt.js koristi renderiranje na strani poslužitelja. Nuxt.js dohvaća AJAX podatke i renderira Vue.js komponente u HTML nizove na poslužitelju te ih šalje izravno u preglednik kada je sva asinkrona logika gotova, a zatim konačno služi statičku oznaku u potpuno interaktivnu aplikaciju na klijentu. Ova značajka omogućuje sjajno analiziranje DOM elemenata pomoću Google SEO parsera. SEO parser analizira DOM elemente ogromnom brzinom čim se DOM web stranice učita.

- Struktura projekta - prema zadanim postavkama, Nuxt.js pruža urednu početnu strukturu aplikacije koja daje solidnu osnovu za organiziranje aplikacije na razumljiv način. Nuxt.js organizira direktorije koji omogućuju uredno organiziranje strukture vaše aplikacije
- Jednostavni prijelazi između stranica – Nuxt.js stvara jednostavne rute između stranica te dodaje element <*transition*> na svaku stranicu što olakšava stvaranje prijelaza između njih [16]

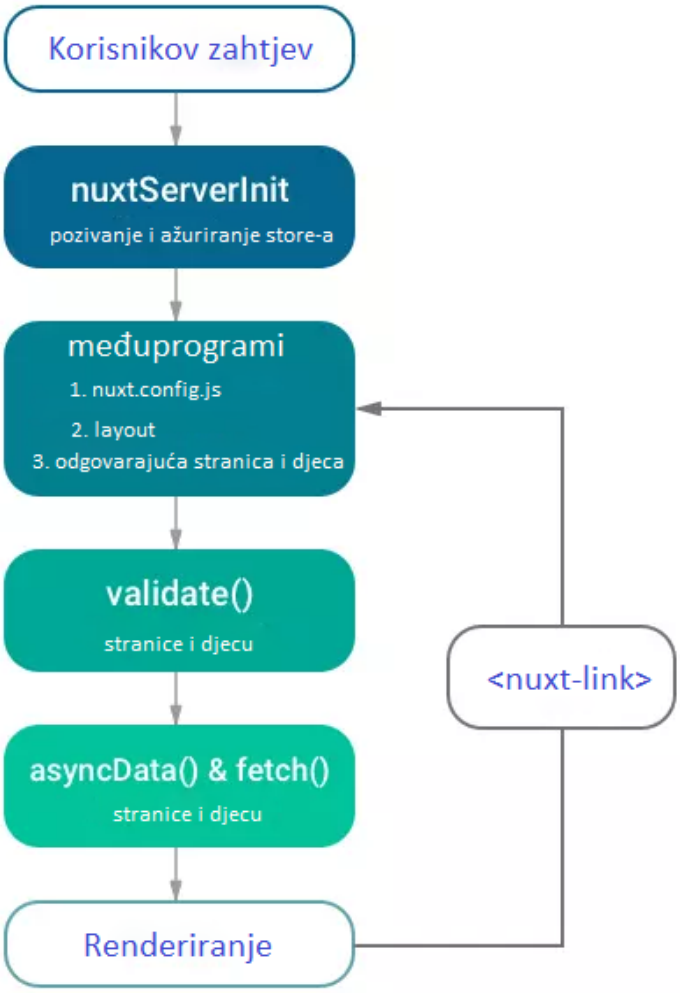
4.1. Način rada Nuxt.js okvira

Kada korisnik posjeti aplikaciju Nuxt.js ili otvori bilo koju od njezinih stranica putem <*nuxt-link*>, događa se sljedeće:

- Kada korisnik prvi put posjeti aplikaciju, ako je *nuxtServerInit* radnja definirana u *storeu*, Nuxt.js će je pozvati i ažurirati *store*.
- Zatim Nuxt.js izvršava sve postojeće međuprograme za stranicu koja je posjećena. Nuxt.js prvo provjerava *nuxt.config.js* datoteku za globalni međuprogram, zatim provjerava odgovarajući *layout* za traženu stranicu i na kraju provjerava stranicu i njihovu djecu.
- Ako je ruta koja se posjećuje dinamička ruka i metoda *validate()* za nju postoji, ruta se provjerava.
- Nakon toga Nuxt.js poziva metode *asyncData()* i *fetch()* za učitavanje podataka prije prikazivanja stranice. Metoda *asyncData()* se koristi za dohvaćanje podataka i njihovo renderiranje na strani poslužitelja, dok se *fetch()* metoda koristi za popunjavanje pohrane prije renderiranja stranice.
- U posljednjem koraku se prikazuje stranica koja sadrži sve odgovarajuće podatke.

- Prilikom klika na *<nuxt-link>* unutar stranice, Nuxt.js otvara željenu stranicu i ponovno izvršava ove korake, počevši od drugog koraka.

Grafički prikaz načina rada Nuxt.js okvira nalazi se na sljedećoj slici prema [17]:

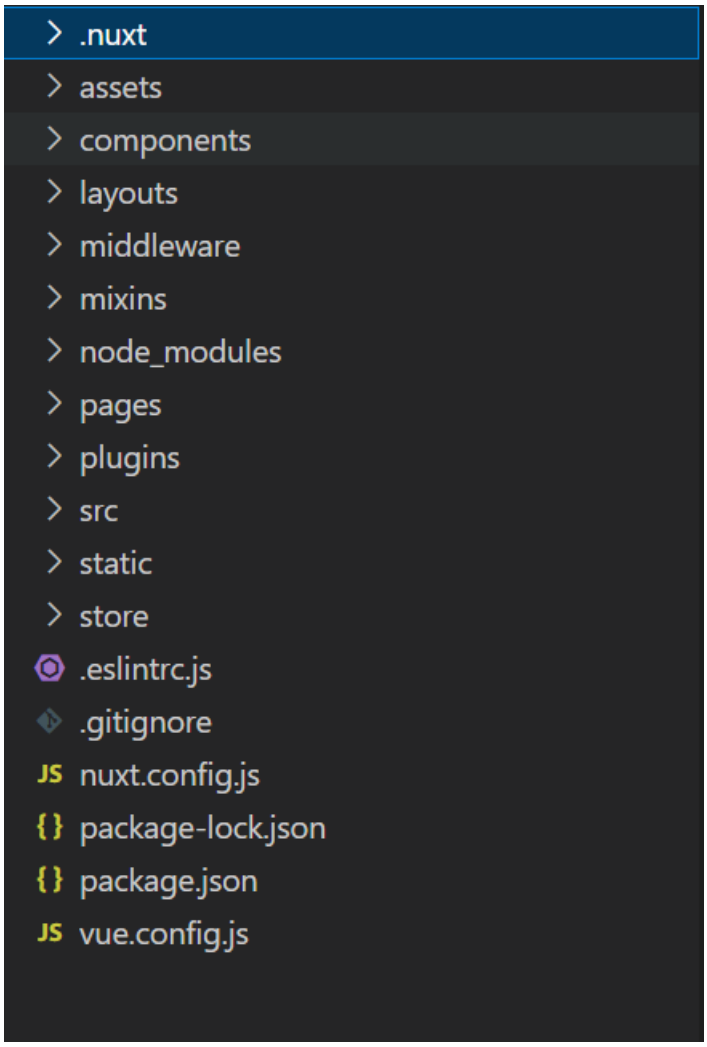


Slika 3: Način rada Nuxt.js okvira

4.2. Struktura Nuxt.js projekta

Prilikom kreiranja Nuxt.js projekta na računalu, postavljaju se različiti direktoriji unutar aplikacije, a svaki od direktorija ima svoju ulogu, odnosno svaki direktorij ima namjenu. Unutar direktorija automatski su smješteni određene datoteke, a prilikom stvaranja novih datoteka potrebno je pratiti već kreirani redoslijed. Ovakva struktura osmišljena je kako bi bolje organizirala aplikaciju te olakšala razvoj iste. Osim što ovakva struktura pojednostavljuje organiziranje i navigaciju unutar Nuxt.js projekta,

ono je također osmišljeno logički kako bi sve datoteke unutar projekta bile povezane. Na sljedećoj slici prikazana je struktura kreiranog Nuxt.js projekta. [18]



Slika 4: Struktura Nuxt.js projekta

Namjena direktorija:

- .nuxt – unutar .nuxt direktorija nalaze se automatski generirane datoteke koje nastaju kada se koristi nuxt dev, odnosno pokretanje localhosta
- assets - direktorij assets se primarno koristi za pohranjivanje datoteka kao što su slike, fontovi, CSS datoteke i slično. Nuxt.js će kompilirati sve datoteke smještene u ovaj direktorij i učiniti ih dostupnima za korištenje u aplikaciji
- static – direktorij static koristi se za pohranjivanje datoteka koje nisu potrebne ili datoteka koje se ne mogu kompilirati. Sve datoteke u direktoriju static mapirane su izravno u korijen poslužitelja te su dostupne pod korijenskim URL-om. Primjerice, kada se slika nalazi u direktoriju images unutar static direktorija, tada je putanja do slike jednaka /images/ime¬-slike.png.

- layouts – ovaj direktorij koristi se za pohranjivanje layouta za web aplikaciju. Nuxt.js generira default.vue stranicu prema zadanim postavkama. Default.vue datoteka prikazuje se na svim stranicama. Prilikom razvoja web aplikacije „Karen“, unutar default.vue datoteke dodana je komponenta Navigation.vue, odnosno navigacija aplikacije. S obzirom da se navigacija aplikacije treba pojaviti na svim stranicama aplikacije, bolje je komponentu dodati na jednu stranicu umjesto dupliciranja koda. S obzirom da se navigacija ne treba prikazivat na stranici login.vue, tada je postavljen uvjet da se komponenta Navigation.vue prikaže ukoliko je putanja datoteke različita od „/login“. Izgled default.vue datoteke:

```
<template>
        <main>
          <Navigation v-if="$route.path !== '/login'"/>
        <nuxt/>
        <Footer/>
        </main>
</template>
```

- middleware – ovaj direktorij koristi se za pohranu datoteka međuprograma koje su JavaScript funkcije koje se pokreću prije iscrtavanja stranice
- components – koristi se za pohranu komponenti. U ovom direktoriju pohranjuju se male komponente koje se koriste više puta u projektu
- mixins – služi za distribuciju funkcija za višekratnu upotrebu komponente
- node_modules – funkcije koje se sekvencijalno pozivaju prilikom pokretanja Nuxt-a
- pages – ovaj direktorij sadržava prikaze i rute aplikacije. Nuxt.js će pretvoriti sve .vue datoteke unutar ovog direktorija i automatski generirati aplikacijski usmjerivač
- plugins – u ovaj direktorij mogu se dodati dodaci koji će se koristiti u aplikaciji. Nakon instaliranja dodatka potrebno je stvoriti novu datoteku za taj dodatak unutar plugins direktorija
- nuxt.config.js i vue.config.js – koriste se za izvođenje bilo koje prilagođene konfiguracije koja se primjenjuje na aplikaciju
- packages.json i packages-lock.json – ova dva direktorija sadržavaju sve ovisnosti aplikacije

5. Web aplikacija Karen

Karen je web aplikacija koja postavlja pitanje zaposlenicima na koja oni proizvoljno odgovaraju te se njihovi odgovori šalju voditelju odjela kojem određeni član tima pripada. Ova aplikacija je osmišljena kako bi pokrila nedostatke već postojećih aplikacija za praćenje rada zaposlenika korištenih u Shapeu. Karen ima tri tipa korisnika, a to su Administrator, voditelj odjela i zaposlenik.

Zaposlenik je tip korisnika koji ima dostupnu samo stranicu „My answers“ gdje mu je prikazan kalendar sa njegovim odgovorima na postavljena pitanja. Korisnik također može odgovoriti na pitanja na koja prethodno nije odgovorio te može ažurirati svoje odgovore.

Voditelj odjela je tip korisnika koji ima sve funkcije kao zaposlenik. Osim toga, voditelj odjela ima stranicu „Channels“ gdje su prikazani svi zaposlenici njegovog odjela. voditelj odjela ima uvid u na koliko je pitanja koji zaposlenik odgovorio u određenom broju dana te može vidjeti sve odgovore zaposlenika njegovog odjela.

Administrator je treći i posljednji tip korisnika aplikacije Karen. Administrator ima sve mogućnosti kao i voditelj odjela. Osim toga, administrator može vidjeti odgovore na postavljena pitanja svih zaposlenika. Administratoru su vidljive još dvije stranice, a to su stranica „Admin“ i stranica „Questions“. Stranica „Admin“ prikazuje sve zaposlenike i odjele, odnosno kanale unutar tvrtke. Administrator može dodavati i brisati odjele, zaposlenike i voditelj odjela. Stranica „Questions“ prikazuje sva pitanja koja Karen postavlja. Administrator ima mogućnost dodavanja novih pitanja ili uklanjanja postojećih pitanja.

5.1. Arhitektura aplikacije

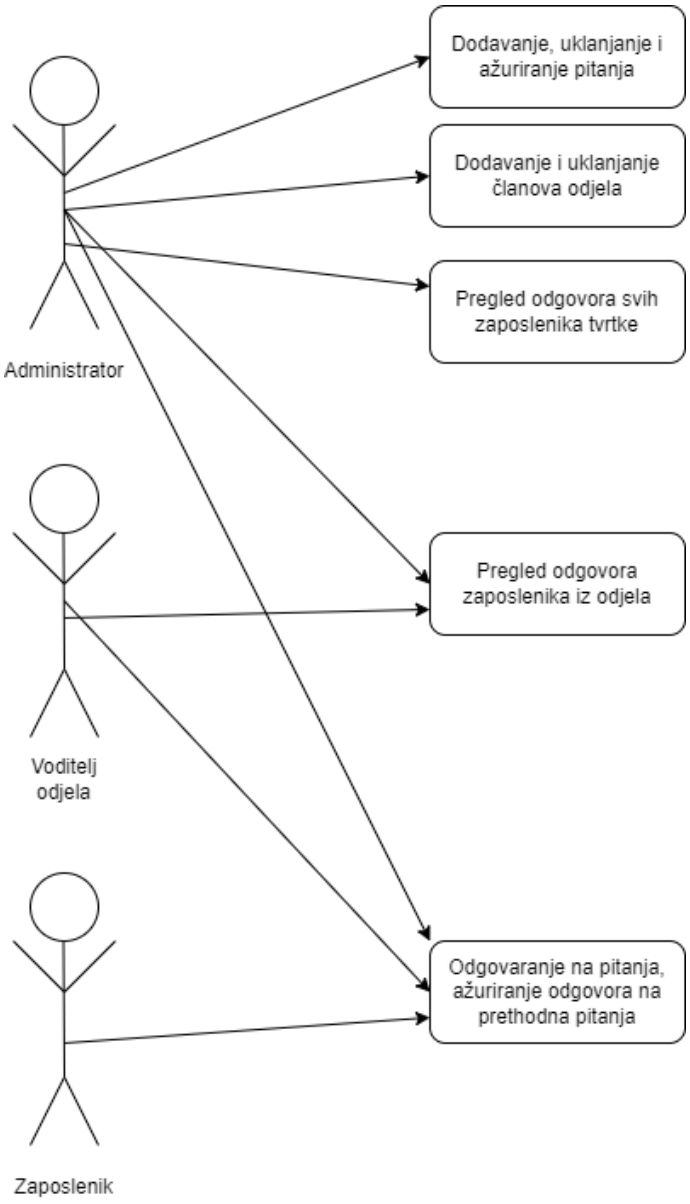
Kako bi se pojednostavio i olakšao prikaz funkcionalnosti Karen web aplikacije, izrađena su dva dijagrama. Dijagram slučaja upotrebe prikazuje kako se aplikacija koristi s obzirom na različite tipove korisnika, a ERA model prikazuje međusobnu povezanost entiteta web aplikacije.

5.1.1. Dijagram slučaja upotrebe

S obzirom da različiti tipovi korisnika imaju različite uloge i mogućnosti unutar aplikacije, ovim dijagramom slučaja uporabe prikazane su mogućnosti svih korisnika aplikacije. Svi korisnici aplikacije najprije se moraju prijaviti u aplikaciju prije nego krenu koristiti bilo kakve mogućnosti aplikacije. Kad se korisnici prijave u aplikaciju, tada se ovlasti mijenjaju. Zaposlenik ima pristup samo pregledavanju svojih odgovora na pitanja, ažuriranja tih odgovora te odgovaranju na nova postavljena pitanja.

Voditelj odjela, kao i zaposlenik, može pregledavati svoje odgovore na pitanja, ažurirati ih i odgovarati na nova pitanja. Osim toga, voditelj odjela ima pristup pregledu odgovora na pitanja svih članova svog odjela te može vidjeti ako zaposlenik nije odgovorio na određena pitanja.

S obzirom da je administrator isto zaposlenik tvrtke, on također mora odgovarati na postavljena pitanja. Stoga administrator ima pristup stranici za pregled svojih odgovora, ažuriranje odgovora te odgovaranju na nova pitanja. Administrator također može pregledati odgovore svih korisnika aplikacije, neovisno o odjelu u kojem se nalazi. Osim toga, administrator ima mogućnost dodavanja i uklanjanja novih članova pojedinih odjela te administrator može dodati, ukloniti i ažurirati pitanja. Ova struktura prikazana je na sljedećem dijagramu slučaja uporabe:



Slika 5: Dijagram slučaja uporabe web aplikacije "Karen"

5.1.2. ERA model

ERA model, odnosno ERA dijagram je dijagram odnosa entiteta. Ovaj dijagram vizualno prikazuje različite entitete unutar sustava i njihov međusobni odnos. Za izradu web aplikacije Karen, dijagram odnosa entiteta korišten je tijekom faze planiranja projekta. ERA dijagram pomaže identificirati različite elemente sustava i međusobne odnose. U ERA modelu postoje tri osnovna elementa, a to su entitet, atribut i odnos. [19] Entitet može biti osoba, mjesto, događaj ili objekt koji je relevantan za dani sustav. Primjerice, entiteti web aplikacije Karen su korisnik, tip korisnika, odjel, pitanje i odgovor. Entiteti su predstavljeni kao naslov tablice. Atributi unutar ERA modela su

svojstva, osobine ili karakteristike entiteta. Treći element ERA modela je odnos. Odnosi ERA modela opisuju međusobnu povezanost i djelovanje različitih entiteta.

Kada je riječ o ERA modelu web aplikacije Karen, ono nije kompliciranо. Postoji 5 entiteta. Entitet korisnik označava sve korisnike web aplikacije Karen. Entitet korisnik sadrži 6 atributa, a to su *korisnik_id*, *datum_pridruzivanja*, email, ime, prezime i aktivan. *Korisnik_id* je atribut tipa broj koji označava pojedinog korisnika, *datum_pridruzivanja* je atribut tipa datum/vrijeme koji označava kada je korisnik dodan u aplikaciju. Ime, prezime i email označavaju podatke o korisniku, a atribut aktivan označava je li korisnik i dalje član pojedinog odjela.

Entitet odjel ima atribut *odjel_id* koji je tipa broj i označava pojedini odjel te atribut *naziv_odjela* koji označava imena odjela, primjerice *backend*, *frontend*, *design* ili *mobile*.

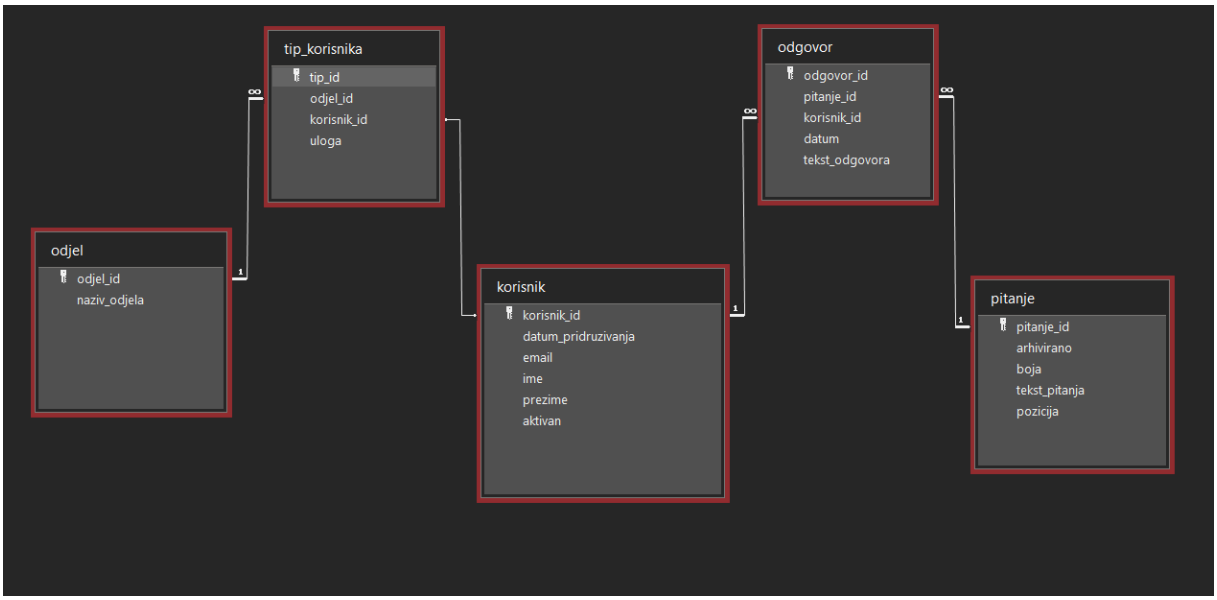
Entitet *tip_korisnika* ima 4 atributa. *Tip_id* označava tip korisnika, a on može biti 0, 1 ili 2. 0 označava zaposlenika, 1 označava voditelja odjela a 2 označava administratora. *Odjel_id* je vanjski ključ koji se veže na entitet odjel, a *korisnik_id* je vanjski ključ povezan s entitetom korisnik. Atribut uloga označava riječ koja opisuje tipa korisnika, a ona može biti administrator, voditelj ili zaposlenik.

Entitet pitanje ima 5 atributa. Atribut *pitanje_id* označava pojedinačno pitanje, atribut arhivirano je tipa da/ne i označava je li pitanje arhivirano, atribut boja označava pojedinu boju pitanja, *tekst_pitanja* označava samo pitanje, a pozicija označava redoslijed pitanja u kalendaru.

Entitet odgovor ima 5 atributa. Atribut *odgovor_id* označava pojedinačni odgovor, *pitanje_id* je vanjski ključ na entitet pitanje a *korisnik_id* je vanjski ključ na entitet korisnik. Atribut datum označava datum pojedinog odgovora na pitanje, a *tekst_odgovora* označava korisnikov odgovor na pitanje.

Entiteti unutar ERA modela su povezani vezama. Korisnik je putem primarnog ključa *korisnik_id* povezan na vanjski ključ *korisnik_id* iz entiteta *tip_korisnika* putem veze 1 na 1, što znači da jedan korisnik može biti samo jedan tip korisnika. Korisnik ne može ujedno biti i administrator i voditelj odjela. Iako administrator ima sve mogućnosti kao i ostali tipovi, on je i dalje jedan tip korisnika. Osim toga, entitet korisnik je putem veze jedan-na-više povezan sa entitetom odgovor. Ova veza označava da svaki jedan korisnik ima više odgovora. Entitet *odjel* je povezan sa entitetom *tip_korisnika* putem veze jedan-na-više. Ovo označava da jedan tip korisnika pripada jednom odjelu, ali jedan odjel može imati više tipova korisnika. Entitet *pitanje* je putem veze jedan na više

povezan sa entitetom *odgovor.* Ova veza označava da jedno pitanje ima više odgovora. Na sljedećoj slici nalazi se grafički prikaz ERA modela web aplikacije „Karen“.



Slika 6: ERA model

5.2. Komponente

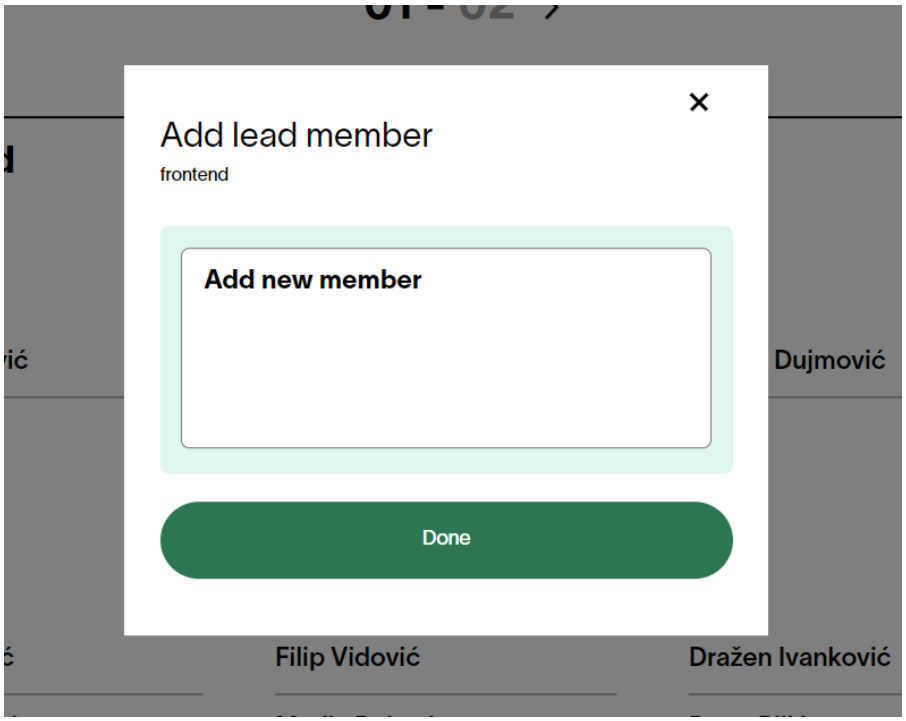
Glavna ideja prilikom izrade Vue.js i Nuxt.js aplikacije je korištenje brojnih dijelova koda zvanih komponente. Komponente su ono što čini različite dijelove web aplikacije i mogu se ponovno koristiti i uvesti u stranice kako bi kod web aplikacije bio čitljiv i uredan. Za web aplikaciju Karen korištene su tri glavne komponente, a to su Navigaion.vue, Modal.vue i Calendar.vue.

5.2.1. Modal.vue

Modal.vue je komponenta koja se koristi za sve iskočne prozore unutar Karen aplikacije. Unutar komponente nalaze se Vue.js atribute props, kao što su value, title i subtitle. Ovo su prazna svojstva koja se mijenjaju ovisno o komponenti koja se otvara. Unutar ostalih stranica u kodu postoji element *<Modal>.* Unutar tog elementa upisuju se sve vrijednosti koje komponenta modal treba sadržavati. Kod komponente modal izgleda ovako:

```
<template>
  <div class="editingModalWrapper" v-if="isOpen" @click="isOpen = false">
    <div class="editingModal" @click.stop>
      <span @click="isOpen = false"></span>
      <h3 v-text="title"/>
      <p v-if="subtitle">{{subtitle}}</p>
      <slot/>
      </div>
    </div>
</template>

<script>
  export default {
    props: {
      value: {},
      title: {},
      subtitle: {}
    },

  computed: {
    isOpen: {
      get(){
        return this.value
      },

      set(value){
        this.$emit('input', value)
      }
    }
  },
};
</script>
```

Kao što je vidljivo u kodu, ne postoje „hard-coded“ vrijednost, već se sve ispunjava dinamički. Primjerice, kada administrator želi dodati novog člana tima u frontend odjel, tada će prop title postati „Add lead member“, a subtitle će postati „Frontend“. Ovisno o tome što komponenta *modal* treba sadržavati, sve vrijednosti se ispunjavaju unutar elementa <*Modal*>. Dizajn modala vidljiv je na sljedećoj slici:



Slika 7: Izgled modala za dodavanje novih članova tima

5.2.2. Calendar.vue

S obzirom da se kalendar koristi na dvije stranice web aplikacije Karen, kako bi se izbjeglo dupliciranje koda, kreirana je komponenta „Calendar.vue“. Ova komponenta sadržava kalendar s prethodnim odgovorima na pitanja korisnika. Navigacija kalendara je vrlo jednostavna te prikazuje ime mjeseca za koji je kalendar otvoren te datume od ponedjeljka do petka. Pritiskom na strelice u navigaciji kalendara, kalendar se prebacuje iz trenutnog tjedna u sljedeći ili prethodni tjedan, ovisi koja je strjelica pritisnuta. Tijelo kalendara napravljeno je pomoću *v-for* petlje i *display: grid;* CSS naredbe. Unutar sljedećeg koda nalaze se dvije *v-for* petlje. Prva petlja izlistava dane u tjednu i broj dana u mjesecu. Prvo što je potrebno je definiranje varijable u *dana* vue atributu. Tamo je definirano da je:

```
today: new Date(),
year: new Date().getFullYear()
```

Nakon toga, potrebno je definirati današnji dan pomoću *mounted* vue atributa:

```
mounted() {
 this.today.setDate(this.today.getDate() - this.today.getDay() + 1);
  this.today = new Date(this.today);
```

```
}
```

Nakon toga, potrebno je pomoću *computed* atributa definirati funkciju koja će definirati 5 radnih dana u tjednu:

```
computed: {
  days() {
    const days = [];
    for (let i = 0; i < 5; i++) {
      const date = new Date(this.today);
      date.setDate(date.getDate() + i + this.offset);
      days.push(date);
    }

    return days;
  },
}
```

Posljednje što je potrebno učiniti prije pisanja *v-for* petlje u kodu je metoda koja formatira datume. Ova funkcija radi na način da ukoliko datum ima 1 znamenku, doda 0 prije broja, a ukoliko datum ima više znamenki, ostavi cijeli datum. Primjerice, ako je datum 3.6., biti će zapisan kao 03.06. Primjer metode za formatiranje datuma:

```
formatDay(currentDate) {
  return currentDate.getDate().toString().padStart(2, "0");
},
```

Kada je definirano sve što je potrebno za kalendar, može se koristiti *v-for* petlja. U ovom slučaju korištene su dvije *v-for* petlje. Prva petlja korištena je za raspis dana u tjednu i raspis broja dana u mjesecu. Druga petlja korištena je za ispisivanje odgovora na pitanja. Osim toga, u kodu je korišten *v-if* uvjet. Ukoliko postoji odgovor na pitanje za određeni dan, tada je odgovor upisan. Ukoliko pitanje ne postoji, izvršava se linija koda koja ima *v-else* uvjet te on postavlja *input* element za unos odgovora na pitanje. Prilikom korištenja *v-if* i *v-else* atributa, obavezno je da elementi koji sadrže navedene atribute budu susjedni elementi. Primjer *v-for* petlje i *v-if* uvjeta za kalendar:

```
<div class="calendarWrapper">
      <div class="calendarBody" v-for="day in days" :key="day.id">
        <h4>{{ day.toLocaleDateString("en-EN", { weekday: "short" })
}}</h4>
        <div class="calendarDayNumber">
          <p>{{ formatDay(day) }}</p>
          <button class="elementEditWrapper"
@click="openModal('calendar')">
```

```
            <img inline src="@/assets/svgs/editPen.svg" alt="Edit
button" />
          </button>
        </div>
        <ul>
          <li
            class="question"
            v-for="question in questionList"
            :key="question.id"
          >
            <p>
              <span></span>{{ question.question }}
              <span class="questionAnswer" v-if="question.answer">
                {{ question.answer }}
              </span>
              <input type="text" v-else />
            </p>
            <br />
          </li>
        </ul>
      </div>
    </div>
```

Prvi dio ovog SCSS koda sadržava naredbe za ekrane čija je širina veća od 768 piksela. Tu je definirana naredba *display: grid;* i naredba *grid-template-columns: repeat(5, 1fr);*. Ovim naredbama definira se da je kalendar postavljen u jednom redu gdje se ispisuju svih 5 radnih dana tjedna. Ukoliko se širina ekrana smanji ispod 768 piksela, tada će se prikazati stilovi za veličinu ekrana „tablet“. Ovim stilovima definira se da postoji *scroll* po Y osi kojim se pregledavaju dani u tjednu. Primjer SCSS koda:

```
.calendarWrapper {
  display: grid;
  grid-template-columns: repeat(5, 1fr);
  grid-gap: s(30);
  @include media('<=tablet'){
    display: flex;
    overflow-x: scroll;
    max-width: 100%;
  }
}
```

5.3. Stranice

Stranice Nuxt.js projekta su sve što se nalazi u ranije spomenutom *pages* direktoriju. Ovdje se nalaze sve stranice koje su dostupne unutar web aplikacije „Karen“.

5.3.1. Stranica za prijavu

Svi zaposlenici tvrtke Shape imaju službeni mail koji završava s „@shape404.agency“. Kako bi se olakšao ulazak u aplikaciju Karen, umjesto odabiranja novog korisničkog i lozinke, jedina mogućnost je korištenjem službenog Shape maila. Dizajn stranice za prijavupodijeljen je u dva dijela. Lijevi dio stranice sadrži gumb za prijavljivanje u aplikaciju putem Google računa. Na dnu lijevog dijela stranice nalazi se logo od aplikacije Karen te link na profil tvrtke Shape. Desna strana stranice za prijavu sadržava ilustraciju Karen te pejorativno značenje termina Karen koje se pokazuje prilikom prelaska mišem preko slike.
Za prijavu u aplikaciju putem Googlea potrebno je koristiti Nuxt.js auth paket. Paket je definiran u nuxt.config.js datoteci. Kod unutar Nuxt.js auth objekta izgleda ovako:

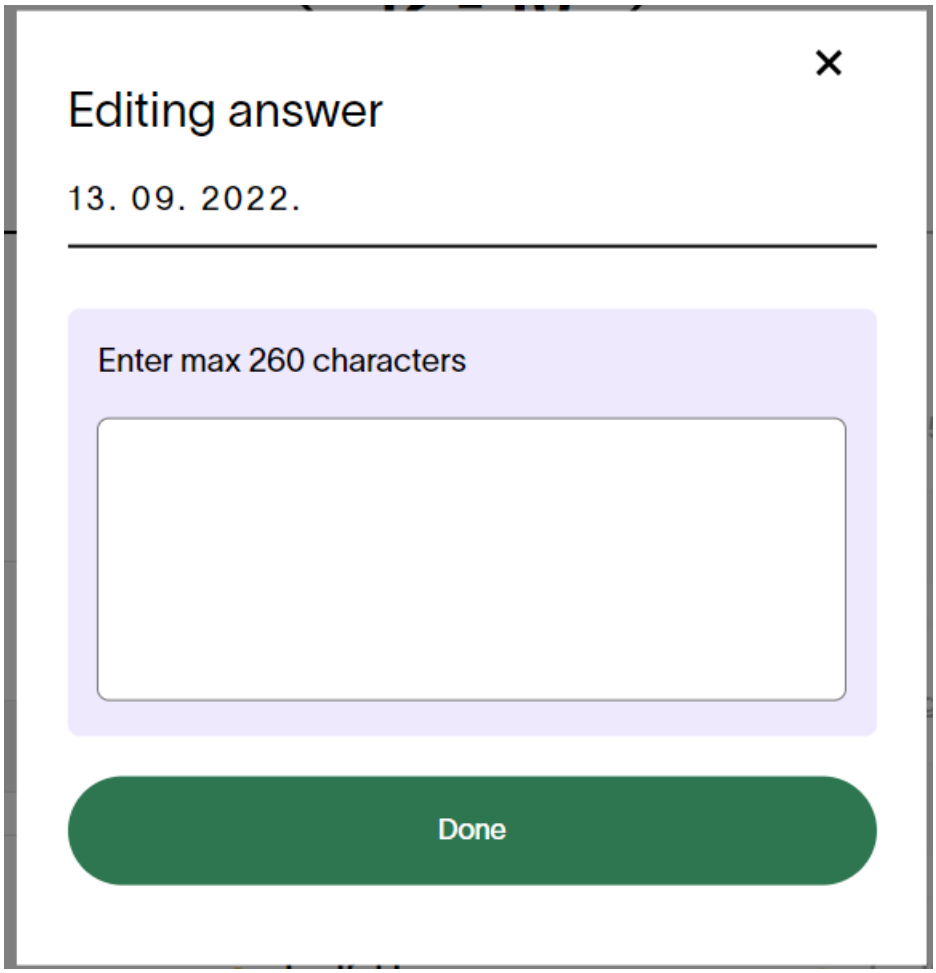
```
auth: {
    strategies: {
      google: {
        client_id: process.env.GOOGLE_CLIENT_ID,
        userinfo_endpoint: undefined,
        redirect_uri: 'http://404wikitest.7of9.agency/',
        response_type: 'id_token',
        token_key: 'id_token',
        access_type: 'offline'
      },
    },

    redirect: {
      login: '/login/',
      logout: '/login/',
      home: '/'
    }
  },
```

Prilikom klika na gumb za prijavu koristi se metoda *login()*:

```
  methods: {
    login(){
      this.$auth.loginWith('google');
    }
  }
```

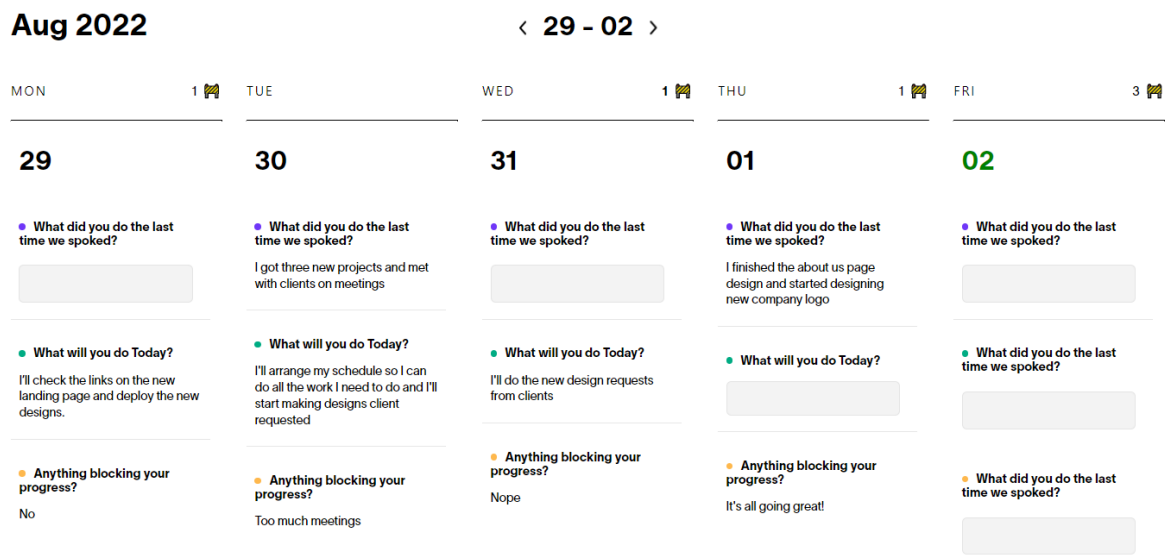
5.3.2. Odgovori

Naslovna stranica Karen web aplikacije je stranica „My answers“, odnosno „Moji odgovori“. Ovo je naslovna stranica svim korisnicima web aplikacije Karen jer su svi korisnici ujedno i zaposlenici koji odgovaraju na postavljena pitanja unutar aplikacije. Naslovna stranica prikazuje glavnu navigaciju aplikacije i kalendar sa prethodnim odgovorima na pitanja. Naslovna stranica sadrži komponentu *Calendar.vue*. Osim toga, ova stranica sadržava <*Modal*> oznake kao HTML elemente. Klikom na gumb za uređivanje prethodnih odgovora na pitanja otvara se modal koji prikazuje sve odgovore za navedeni dan.



Slika 8: Izgled modala za uređenje odgovora na pitanje

Dizajn „My answers“ stranice nalazi se na sljedećoj slici:

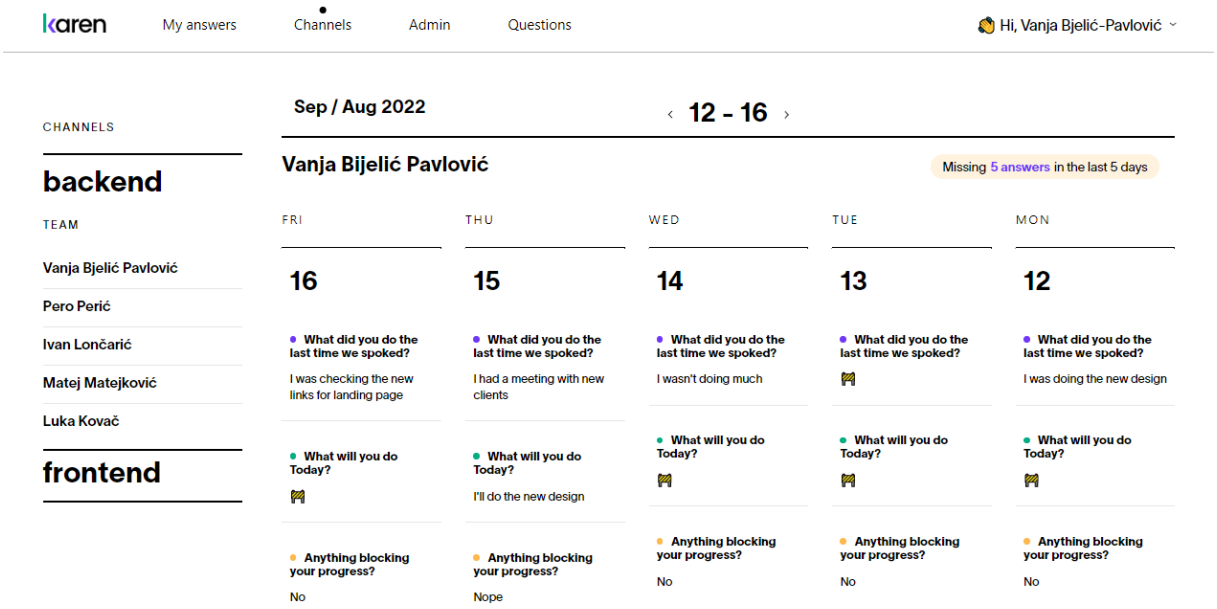


Slika 9: Dizajn "My answers" stranice

5.3.3. Odjeli

Channels je stranica koja je možda najvažnija za voditelje odjela. Ova stranica je također podijeljena na dva dijela. S lijeve stranice nalazi se popis odjela i članova tima po odjelima, a s desne strane nalazi se kalendar s odgovorima pojedinog člana tima. Voditelj pojedinog odjela ima uvid samo u odgovore članova tima koji pripadaju njegovom odjelu, a administrator ima pristup svim odjelima i članovima odjela. Pored imena člana tima za koga je otvoren kalendar zapisano je na koliko on odgovora nije odgovorio u posljednjih nekoliko dana, ovisno o tome koliko je dana odabrano u padajućem izborniku. Pored imena svih članova tima označeno je na koliko pitanja nisu odgovorili u zadnjih 10 dana. Članovi tima također su izlistani pomoću *v-for* petlje koja iterira po određenom odjelu.

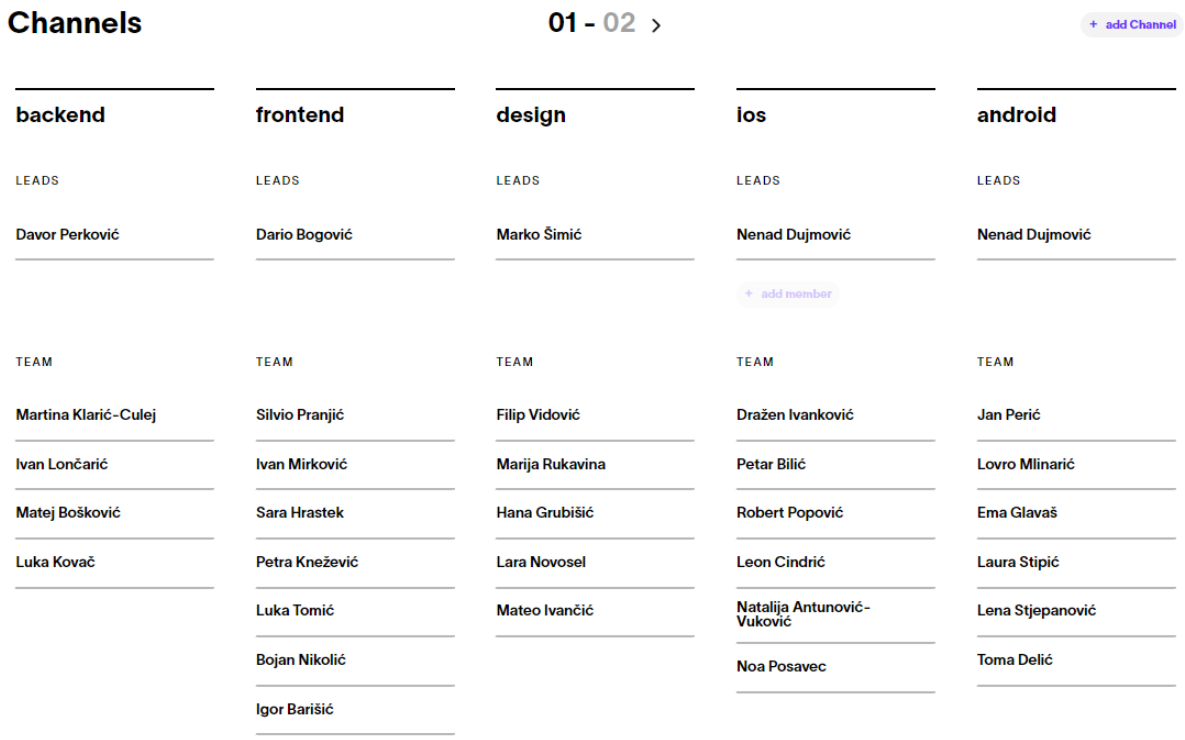
Channels stranica koristi istu komponentu kalendara kao i stranica „My answers“, ali u ovoj stranici, kalendar ide od novijeg prema starijem datumu zbog praktičnosti, kako bi voditelj odjela prvo mogao vidjeti posljednje odgovore članova svog odjela. To je učinjeno pomoću ugrađene metode u JavaScript koja se naziva *reverse()*. Metoda *reverse()* preokreće polje i vraća referencu na isto polje, pri čemu prvi element polja sada postaje posljednji, a posljednji element polja postaje prvi. Drugim riječima, redoslijed elemenata u polju biti će okrenut u smjeru suprotnom od uobičajenog polja.



Slika 10: Dizajn "Channels" stranice

5.3.4. Admin

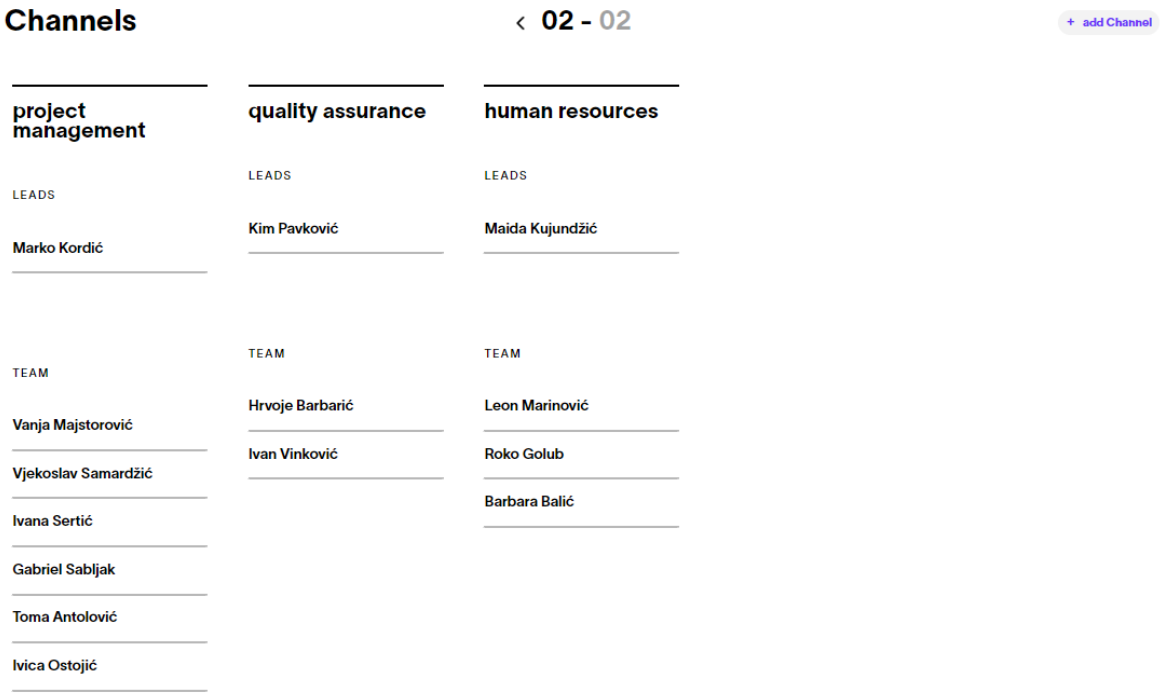
Pristup ovoj stranici dopušten je samo administratoru, stoga i naziv „Admin“. Administrator ovdje vidi popis svih timova, voditelja tima i ostalih zaposlenika u timu. Administrator ima opciju dodavanja i uklanjanja novih odjela, članova tima i voditelja. Dizajn stranice „Admin“ napravljen je tako da administratoru budu vidljivi svi odjeli, voditelji odjela te ostali članovi odjela.



Slika 11: Prikaz "Admin" stranice

S obzirom da postoji više od 5 odjela unutar tvrtke, potrebno je napraviti paginaciju. Unutar data atributa dodana je vrijednost 1 varijabli „page“, tako da uvijek bude prikazana stranica 1 prilikom otvaranja stranice. Zatim je u HTML kodu dodan v-if uvjet. Ukoliko je varijabli page pridružena vrijednost koja je veća od 1, tada će se prikazati gumb za prelazak na prethodnu stranicu, a ukoliko nije, gumb će biti skriven. Isto tako, ukoliko je varijabli page dodana vrijednost manja od ukupnog broja stranica, tada će gumb za prelazak na sljedeću stranicu biti prikazan, a u protivnom skriven.

Prelazak na sljedeću stranicu napisan je u *computed* Vue.js atributu. Prvo je izračunat ukupan broj potrebnih stranica, tako da se vidi ukupni broj odjela, podijeli sa 5 zbog prikaza na jednoj stranici i zaokruži na višu decimalu. Kada je izračunat broj stranica, tada je potrebno napisati funkciju za prelazak s jedne stranice na drugu. To je učinjeno pomoću funkcije *channels*. Ova funkcija vraća kopiju polja liste odjela od prvog do petog elementa u polju te se ti odjeli prikazuju na stranici. Pritiskom na gumb za prikaz sljedećih 5 odjela, prikazuju se sljedeći odjeli.



Slika 12: Prikaz funkcionalnosti paginacije na "Admin" stranici

HTML kod:

```
<div>
  <img src="/resources/images/arrowLeft.png" alt="" v-if="page>1"
@click="page--"/>
  <p>{{ (page).toString().padStart(2, 0) }} - <span>{{
(numberOfPages).toString().padStart(2, 0) }}</span></p>
  <img src="/resources/images/arrowLeft.png" class="buttonRight"
alt="" v-if="page<numberOfPages" @click="page++"/>
</div>
```

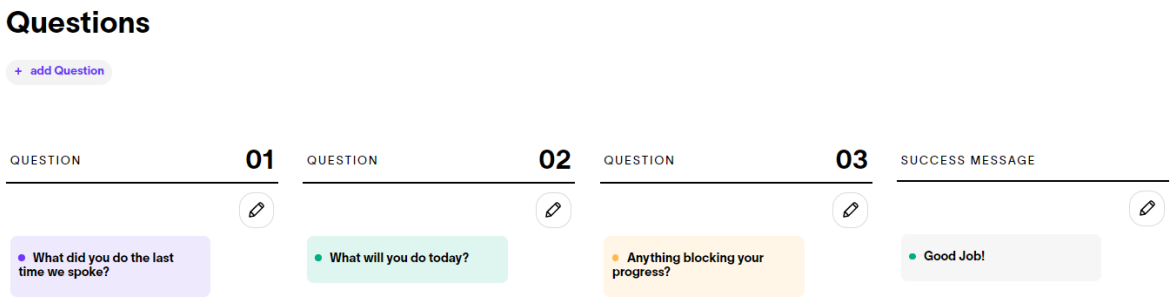
Vue.js kod:

```
computed: {
  channels(){
    return this.channelList.slice(5*(this.page-1), 5*this.page)
  },
  numberOfPages(){
    return Math.ceil(this.channelList.length / 5);
  }
}
```

5.3.5. Pitanja

Stranica Questions vidljiva je samo administratoru i služi za pregled, ažuriranje i dodavanje pitanja. Na stranici Questions vidljiva su sva pitanja koja se postavljaju korisnicima aplikacije Karen. Pitanja su izlistana pomoću *v-for* petlje, a stilizirana pomoću *display: grid* stilske naredbe. Administrator može dodavati i ažurirati postojeća

pitanja, a klikom na gumb za ažuriranje ili dodavanje, otvara se skočni prozor u kojoj administrator može unijeti svoje promjene.



Slika 13: Dizajn "Questions" stranice

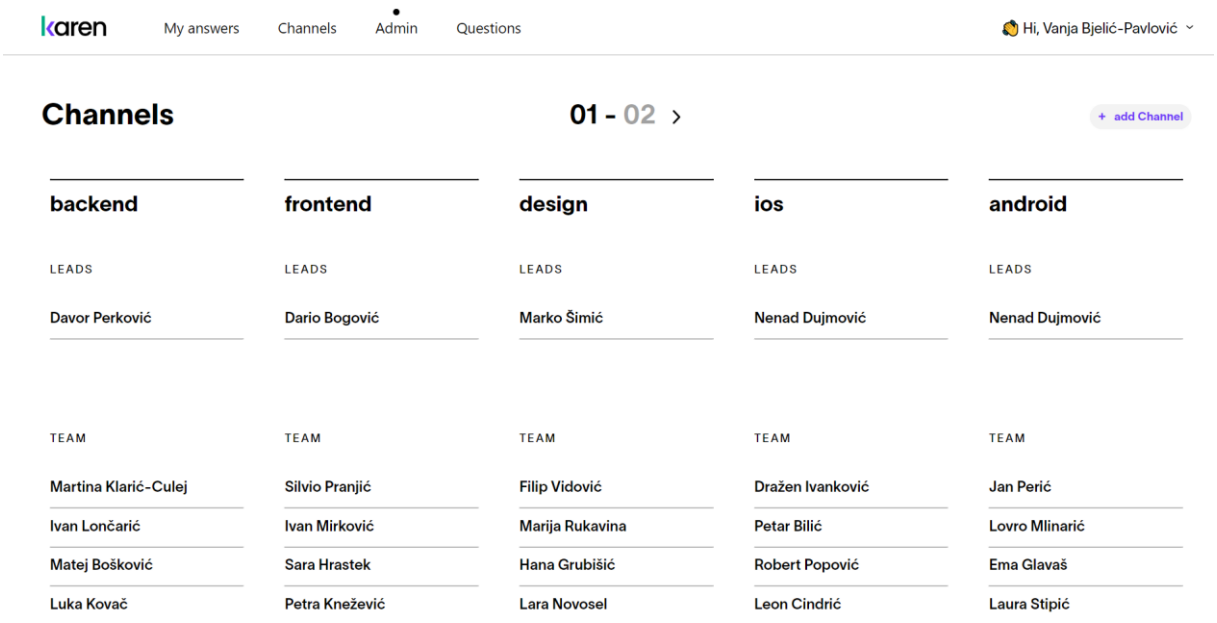
5.4. Responzivni dizajn

Kako je tehnologija napredovala, tako je napredovao i responzivni dizajn web stranica. Razlog tome je korištenje web stranica na velikom broju uređaja. Kako bi stranica izgledala pristupačno i bila korisna, potrebno je prilagoditi dizajn stranice svim veličinama ekrana. Web aplikacija Karen čini upravo to. Sve stranice ove web aplikacije mogu se koristiti na ekranima bilo koje širine ili visine, bez da se izgubi glavna funkcionalnost aplikacije. Ovo je moguće uz pomoć REM-ova (Root EM) i media upita unutar SCSS-a. Root EM označava mjernu jedinicu koja predstavlja veličinu fonta korjenskog elementa. To znači da je 1 rem jednak veličini fonta HTML elementa, koji za većinu preglednika ima vrijednost 16px. Međutim, radi lakšeg i čitljivijeg pisanja koda, korištena je funkcija koja vrijednost unutar zagrade pretvara u REM. Funkcija izgleda ovako:

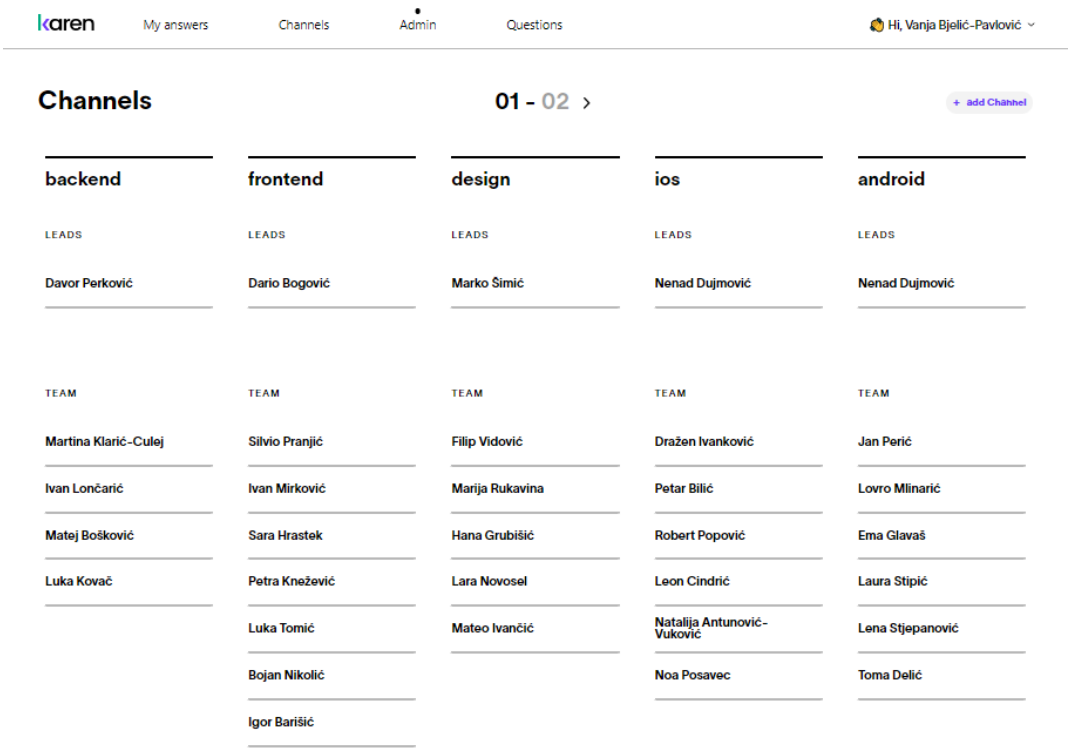
```
@function s($px) {
  @return #{$px}rem;
}
```

Iako je dizajn web aplikacije izračunat u pikselima, izrada web aplikacije koristi funkciju koja pretvara piksele u REM. Razlog ovome je jednostavniji i čišći izgled web aplikacije prilikom promjene širine ekrana. U nastavku su dvije slike koje prikazuju izgled stranice na širini ekrana od 910px i 1920px. Kao što je vidljivo na slikama, što je širina ekrana veća, to su veće veličine svih elemenata gdje im je veličina određena

u REM-ovima. Kako se veličina smanjuje, tako se svi elementi smanjuju kako bi aplikacija izgledala praktično na svim veličinama ekrana. Primjerice, veličina fonta imena odjela na ekranu širine 1920px je *26rem*, a na širini ekrana 910px je *15px.*



Slika 14: Dizajn stranice "Admin" na ekranu širine 1920px



Slika 15: Dizajn stranice "Admin" na ekranu širine 910px

Međutim, samo izražavanje veličine elemenata u REM-ovim nije dovoljno za prikaz stranice na mobilnim uređajima ili tabletima. Stoga je potrebno dodati *break point*, odnosno širinu ekrana gdje se SCSS dizajn mijenja, odnosno poprima nove vrijednosti. Prije svega, potrebno je definirati širine ekrana *break pointa.* To je učinjeno u *_media.scss* datoteci.

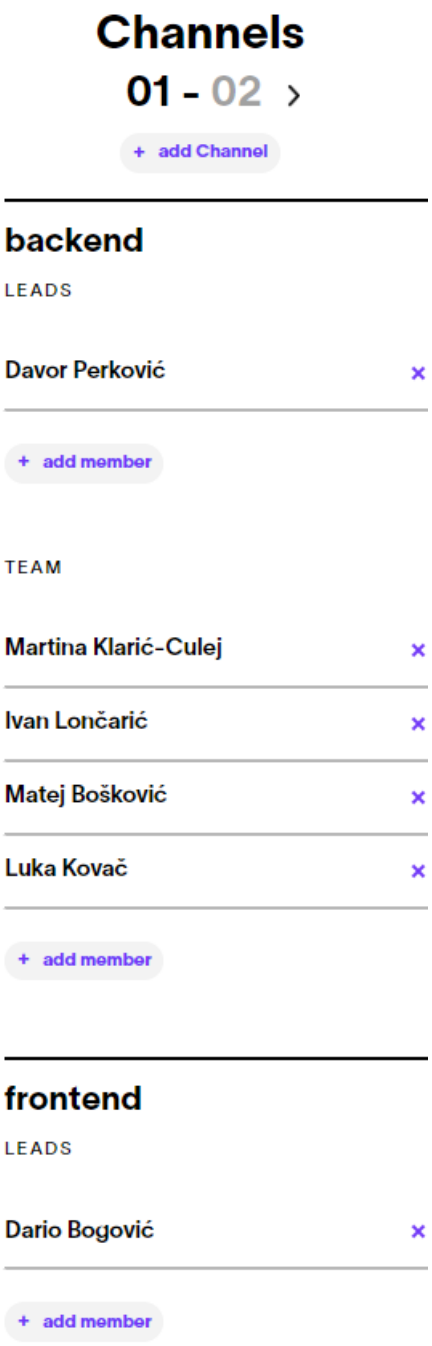
```
$breakpoints: (
  "desktop-large": 1800px,
  "desktop": 1200px,
  "tablet": 900px,
  "mobile": 600px,
);
```

Također, kako bi se olakšalo pisanje CSS-a, korištena je direktiva *@include*. Ova direktiva označava korištenje *mixina* unutar CSS-a kako bi se mogao ponovno koristiti.

```
@include media(">tablet") {
   font-size: #{math.div(100, map.get($resolutions, "desktop"))}vw;
 }
```

Primjer ponovnog korištenja *@include* direktive je prilikom korištenja responzivnog dizajna za tablet i mobilne uređaje. Kada je širina ekrana veća od 900 piksela, tada element sa klasom *channel* treba biti vodoravno poravnat, stoga *display: flex* naredba. Međutim, kada se širina ekrana smanji ispod 900 piksela, potrebno je element s klasom *channel* okomito poravnati pomoću naredbe *flex-direction: column*. Primjer mobilnog dizajna admin stranice nalazi se na sljedećoj slici. S obzirom da je na mobilnim uređajima nemoguće prijeći kursorom preko elementa, uklonjeni su sve *:hover* pseudo-klase, odnosno elementi koji nisu vidljivi dok korisnik ne prijeđe mišem preko elementa.

```
li span, .buttonAdd{
  visibility: hidden;
  opacity: 0;
  transition: color 300ms, opacity 300ms, visibility 300ms, background-
color 300ms;
  @include media('<=tablet'){
    visibility: visible;
    opacity: 1;
  }
}
```



Slika 16: Mobilni dizajn "Admin" stranice

6. Kritički osvrt

Kao i svi drugi JavaScript okviri, Vue.js i Nuxt.js imaju svoje prednosti i nedostatke. Osim što je Vue.js jedan od najpopularnijih JavaScript okvira, on je ujedno i najfleksibilniji. Vue.js će raditi jednako dobro u svakom okruženju. Međutim, uz Nuxt.js, razvoj web aplikacija podiže se razinu iznad. Osim jednostavnosti izrade web aplikacije, Nuxt.js pruža razne funkcionalnosti aplikacije. Jedna od glavnih prednosti Nuxt.js okvira je njegova veličina. S obzirom na malu veličinu Nuxt.js-a, učitavanje i instalacija biblioteka su brzi, što vrlo pozitivno utječe na SEO i dizajn. Osim toga, nije potrebno trošiti dodatno vrijeme na optimizaciju.

Nuxt.js je također vrlo jednostavan za korištenje. S obzirom na korištenje komponenti i podijeljenih blokova koda, Nuxt.js kod uvijek izgleda jednostavno i čisto. HTML predložak i sve ostale metode su odvojene jedna od druge, što omogućuje kontrolu procesa i razumijevanje značenja dijelova koda. Upravo zato je pronalazak i rješavanje grešaka jednostavnije. Za postavljanje Nuxt.js aplikacije potreban je samo naredbeni redak.

Najveći nedostatak Nuxt.js-a je manjak dokumentacije. S obzirom da je Nuxt.js relativno nov JavaScript okvir, ne postoji brojna službena i neslužbena dokumentacija, za razliku od ostalih JavaScript okvira poput React-a i Angulara-a. Osim toga, Nuxt.js je ovisan o Vue.js-u, te je Nuxt.js zapravo okvir od Vue-a. Stoga, kada se izrađuje aplikacija u Nuxt.js-u, Vue.js je neophodan dio aplikacije.

7. Zaključak

Cilj ovog završnog rada bio je izrada web aplikacije za praćenje projekata i rada zaposlenika, s naglaskom na isplativost projekata i poboljšanje komunikacije unutar tvrtke. Ova web aplikacija prikazuje kako se većina projekata tvrtke može automatizirati uz pomoć raznih aplikacija i sustava. Razvoj aplikacija i sustava često može biti kompliciran i zahtjevan posao, ali određeni programski okviri olakšavaju taj posao. Primjer takvih programskih okvira su Vue.js i Nuxt.js. Vue.js izrađuje brzu i kvalitetnu web aplikaciju, ali u kombinaciji s Nuxt.js-om, ova aplikacija dobiva dodatne mogućnosti i čini dopunjava nedostatke Vue.js-a.

U ovom završnom radu opisan je proces razvoja web aplikacije zajedno sa svim alatima potrebnim za razvoj web aplikacije. Osim osnovnih alata za definiranje strukture i stila web aplikacije, naglasak je na Vue.js i Nuxt.js programskim okvirima. Nakon što je izrađen i opisan kompletan teorijski dio, definirane su glavne značajke web aplikacije „Karen“. Prikazan je dijagram slučaja uporabe i ERA model koji grafički prikazuju način korištenja web aplikacije te samu unutarnju strukturu web aplikacije. Kada su postavljene sve osnove izrade aplikacije, uslijedio je sam razvoj web aplikacije.

Web aplikacija „Karen“ sastoji se od pet stranica. Stranica za prijavu služi za prijavu u sustav putem Google računa. Naslovna stranica web aplikacije prikazuje kalendar s pitanjima i odgovorima korisnika. Stranica „Channels“ služi kako bi voditelji odjela mogli vidjeti članove svog odjela i njihove odgovore na ponuđena pitanja, a administrator može vidjeti odgovore svih korisnika aplikacije, neovisno o odjelu u kojima se nalaze. Stranica „Admin“ služi kako bi administrator mogao brisati i dodavati korisnike u aplikaciju. Stranica „Questions“ omogućava administratoru dodavanje i ažuriranje pitanja koji se postavljaju zaposlenicima.

S obzirom da je web aplikacija „Karen“ interni projekt tvrtke „Shape“, ovo nije kraj izrade web aplikacije. Web aplikacija „Karen“ će se konstantno ažurirati i razvijati kako bi se dodavale dodatne potrebne funkcionalnosti.

Literatura

[1] B. Lawson and R. Sharp, Introducing HTML5. New Riders, 2011.

[2] “HTML: HyperText Markup Language | MDN.” https://developer.mozilla.org/en-US/docs/Web/HTML (pristupljeno 22.08.2022.).

[3] “Cascading Style Sheets, level 1,” Cascading Style Sheets, p. 69.

[4] “Sass: Syntactically Awesome Style Sheets.” https://sass-lang.com/ (pristupljeno 23.08.2022).

[5] “JavaScript | MDN.” https://developer.mozilla.org/en-US/docs/Web/JavaScript (pristupljeno 23.08.2022).

[6] JavaScript: The Good Parts. Pristupljeno: 23.08.2022. [Online]. Available: https://books.google.com/books/about/JavaScript_The_Good_Parts.html?id=PXa2bby0oQ0C

[7] S. H. Jensen, A. Møller, and P. Thiemann, “Type Analysis for JavaScript,” in Static Analysis, vol. 5673, J. Palsberg and Z. Su, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 238–255. doi: 10.1007/978-3-642-03237-0_17.

[8] A. Kyriakidis, K. Maniatis, and E. You, “The Majesty of Vue.js 2,” p. 30.

[9] “The Vue Instance — Vue.js.” https://v2.vuejs.org/v2/guide/instance.html#Lifecycle-Diagram (pristupljeno 09.09.2022).

[10] “Template Syntax | Vue.js.” https://vuejs.org/guide/essentials/template-syntax.html (pristupljeno 09.09.2022).

[11] “Vue Router.” https://router.vuejs.org/ (pristupljeno 09.09.2022).

[12] “Using mixins and custom functions in Vue - LogRocket Blog.” https://blog.logrocket.com/mixins-custom-functions-vue/ (pristupljeno 23.08.2022).

[13] “Computed Properties | Vue.js.” https://vuejs.org/guide/essentials/computed.html (pristupljeno 09-09.2022).

[14] C. Macrae, Vue.js: Up and Running: Building Accessible and Performant Web Apps. O’Reilly Media, Inc., 2018.

[15] L. T. Kok, Hands-on Nuxt.js Web Development: Build universal and static-generated Vue.js applications using Nuxt.js. Packt Publishing Ltd, 2020.

[16] M. Kiš, “Web aplikacija za upravljanje projektima unutar tvrtke,” info:eu-repo/semantics/masterThesis, Josip Juraj Strossmayer University of Osijek. Faculty of Electrical Engineering, Computer Science and Information Technology Osijek. Department of Software Engineering. Chair of Programming Languages and Systems, 2021. Pristupljeno: 09.09.2022. [Online]. Available: https://urn.nsk.hr/urn:nbn:hr:200:766091

[17] O. D. Tech, “The Complete Nuxt Guide,” Medium, Dec. 16, 2021. https://levelup.gitconnected.com/the-complete-nuxt-guide-940751e1a6a5 (pristupljeno 09.09.2022.).

[18] K. Patel, “Universal application code structure in Nuxt.js,” We’ve moved to freeCodeCamp.org/news, Oct. 08, 2018. https://medium.com/free-code-camp/universal-application-code-structure-in-nuxt-js-4cd014cc0baa (pristupljeno 09.09.2022).

[19] “Entity Relationship Diagram (ERD) | ER Diagram Tutorial,” Creately Blog, Mar. 07, 2017. https://creately.com/blog/diagrams/er-diagrams-tutorial/ (pristupljeno 09.09.2022).

Popis slika

Slika 1: Prikaz alert funkcije 7
Slika 2: Životni ciklus Vue.js aplikacije 12
Slika 3: Način rada Nuxt.js okvira 20
Slika 4: Struktura Nuxt.js projekta 21
Slika 5: Dijagram slučaja uporabe web aplikacije "Karen" 25
Slika 6: ERA model 27
Slika 7: Izgled modala za dodavanje novih članova tima 29
Slika 10: Izgled modala za odgovaranje na pitanja cijelog dana 33
Slika 11: Dizajn "My answers" stranice 34
Slika 12: Dizajn "Channels" stranice 35
Slika 13: Prikaz "Admin" stranice 36
Slika 14: Prikaz funkcionalnosti paginacije na "Admin" stranici 37
Slika 15: Dizajn "Questions" stranice 38
Slika 16: Dizajn stranice "Admin" na ekranu širine 1920px 39
Slika 17: Dizajn stranice "Admin" na ekranu širine 910px 39
Slika 18: Mobilni dizajn "Admin" stranice 41