e-Škole
RAZVOJ SUSTAVA DIGITALNO ZRELIH ŠKOLA
(II. FAZA)

CARNET
znanje povezuje

Ugovor: Ugovor o nabavi izrade digitalnih obrazovnih sadrzaja za medupredmetne teme hr. 292-100-930/19

Naručitelj: Hrvatska akademska i istraživačka mreža – CARNET, Josipa Marohnića 5, Zagreb
Izvršitelj: Profil Klett, Hektorovićeva 2, Zagreb

TEHNIČKA DOKUMENTACIJA

UGOVOR O NABAVI IZRADE DIGITALNIH OBRAZOVNIH SADRŽAJA ZA MEĐUPREDMETNE TEME BR. 292-100-930/19

Vladimir Obradović

Profil Klett d.o.o. | Ulica Petra Hektorovića 2

Europska unija
Zajedno do fondova EU

EUROPSKI STRUKTURNI I INVESTICIJSKI FONDOVI

Operativni program KONKURENTNOST I KOHEZIJA

ESF UČINKOVITI LJUDSKI POTENCIJALI

Projekt je sufinancirala Europska unija iz europskih strukturnih i investicijskih fondova.

Više informacija o EU fondovima možete naći na web stranicama Ministarstva regionalnoga razvoja i fondova Europske unije: www.strukturnifondovi.hr

Sadržaj

1. Uvod

U ovom dokumentu iznijet ćemo sve informacije koje su potrebne da bi se programski kôd i datoteke komponenti interaktivnog sadržaja dostavljene u sklopu nabave mogli koristiti, razumjeti, održavati i nadograđivati. Za to će biti bitno razumjeti strukturu programskog koda, ali i sadržajnog dijela te dijelove gdje se to dvoje preklapa. Objasniti ćemo tehnologije koje se koriste kao i razloge za njihov odabir.

Europska unija
Zajedno do fondova EU

EUROPSKI STRUKTURNI I INVESTICIJSKI FONDOVI

Operativni program KONKURENTNOST I KOHEZIJA

ESF UČINKOVITI LJUDSKI POTENCIJALI

Projekt je sufinancirala Europska unija iz europskih strukturnih i investicijskih fondova.

Više informacija o EU fondovima možete naći na web stranicama Ministarstva regionalnoga razvoja i fondova Europske unije: www.strukturnifondovi.hr

2. Struktura programskog koda i komponenti interaktivnog sadržaja (arhiva izvornog koda)

A. Osnovne tehnologije na kojima počivaju komponente interaktivnog sadržaja

- HTML - https://www.w3.org/html/
- CSS - https://www.w3.org/Style/CSS/Overview.en.html
- JavaScript - https://www.w3schools.com/js/DEFAULT.asp
- SVG - https://www.w3.org/TR/SVG11/
- VueJS (v2.6.14) - https://vuejs.org/
- SCSS (v10.0.3) - https://sass-lang.com/
- Gulp (v4.0.2) - https://gulpjs.com/
- Webpack - https://webpack.js.org/
- NodeJS (v15.14.0) - https://nodejs.org/en/
- NPM (v7.7.6) - https://www.npmjs.com/

Datoteka package.json sadrži još precizniji uvid u *libraryje*, *frameworke* i pakete koji se koriste:

```
"devDependencies": {
  "@babel/cli": "^7.10.1",
  "@babel/core": "^7.10.2",
  "@babel/plugin-proposal-optional-chaining": "^7.12.17",
  "@babel/plugin-transform-runtime": "^7.14.5",
  "@babel/preset-env": "^7.4.3",
  "babel-loader": "^8.0.5",
  "babel-plugin-component": "^1.1.1",
  "css-loader": "^2.1.1",
  "eslint": "^7.20.0",
  "eslint-plugin-vue": "^7.6.0",
  "fancy-log": "^1.3.3",
  "fs": "0.0.1-security",
  "gulp-autoprefixer": "^6.0.0",
  "gulp-babel": "^8.0.0",
  "gulp-clean-css": "^3.10.0",
  "gulp-color": "0.0.2",
  "gulp-concat": "^2.6.1",
  "gulp-if": "^2.0.2",
  "gulp-include": "^2.3.1",
  "gulp-merge": "^0.1.1",
  "gulp-newer": "^1.4.0",
  "gulp-plumber": "^1.2.1",
  "gulp-postcss": "^8.0.0",
  "gulp-rename": "^1.4.0",
  "gulp-sass": "^4.1.0",
  "gulp-sourcemaps": "^2.6.5",
  "gulp-strip-css-comments": "^1.2.0",
  "gulp-stylus": "^2.7.0",
  "gulp-svg-sprite": "^1.5.0",
  "gulp-svg-sprites": "^4.1.1",
  "gulp-tap": "^1.0.1",
  "gulp-uglify": "^2.1.2",
```

```
  "gulp-uncss": "^1.0.6",
  "gulp-util": "^3.0.8",
  "mkdirp": "^0.5.1",
  "postcss-reporter": "^6.0.0",
  "postcss-scss": "^2.0.0",
  "sass": "^1.32.12",
  "sass-loader": "^10.0.3",
  "style-loader": "^2.0.0",
  "vue-loader": "^14.2.2",
  "vue-slick": "^1.1.15",
  "vue-style-loader": "^4.1.2",
  "vue-template-compiler": "2.6.14",
  "webpack": "^4.29.6",
  "webpack-cli": "^3.3.0",
  "webpack-dev-server": "^3.3.1",
  "webpack-merge-and-include-globally": "^2.1.16",
  "yargs": "^12.0.5"
},
"dependencies": {
  "@babel/polyfill": "^7.4.3",
  "axios": "^0.20.0",
  "core-js": "^3.0.1",
  "dayjs": "^1.9.4",
  "dom-confetti": "^0.2.2",
  "element-theme-chalk": "^2.13.2",
  "element-ui": "^2.13.2",
  "focus-visible": "^5.1.0",
  "gulp": "^4.0.2",
  "gulp-require-tasks": "^1.2.1",
  "interactjs": "^1.10.0",
  "intersection-observer": "^0.7.0",
  "jquery": "^3.4.0",
  "lodash-es": "^4.17.15",
  "lozad": "^1.9.0",
  "lunr": "^2.3.9",
  "lunr-languages": "^1.4.0",
  "natives": "^1.1.6",
  "patch-package": "^6.4.7",
  "screenfull": "^4.2.0",
  "vue": "2.6.14",
  "vue-axios": "^2.1.5"
}
```


B. Struktura direktorija

Sve datoteke navedene u ovom dijelu sačinjavaju elemente koji su nužni za funkcioniranje komponenti interaktivnog sadržaja, ali ne uključuju nikakve sadržajne datoteke uz iznimku GeoGebra elemenata, a svi ostali elementi sadržaja se nalaze u dostavljenim arhivama komponenti interaktivnog sadržaja

/src – u ovom direktoriju nalaze se sve izvorne datoteke programskog koda. Dva glavna poddirektorija su **carnet** i **common,** oni su komplementarni i ovisni jedan o drugome
/src/common – sadrži općenite dijelove koda, SCSS i JS datoteke koje se koriste u komponentama interaktivnog sadržaja
/src/carnet – sadrži specifične dijelove koda za funkcioniranje komponenti interaktivnog sadržaja, konfiguraciju *build* alata, kao i sve *assete*, a to uključuje

/src/carnet/config – konfiguracija build alata (Webpack)
/src/carnet/fonts – svi fontovi koji se koriste u projektu
/src/carnet/icons – izvorne SVG datoteke koje se *buildaju* u ikone koje koristimo u komponentama interaktivnog sadržaja
/src/carnet/images – grafičke datoteke koje se koriste u komponentama interaktivnog sadržaja, ovdje je riječ o elementima sučelja i slično, ne o sadržajnim datotekama
/src/carnet/scripts – sve JS datoteke
/src/carnet/scss – sve SCSS datoteke
/src/carnet/twig – dijelovi Twig *templatea* koji se koriste za kreiranje konačnog HTML-a komponenti interaktivnog sadržaja
/dist – u ovom direktoriju nalaze se sve kompajlirane datoteke koje se u konačnici koriste pri prikazu komponenti interaktivnog sadržaja kao i neki *3rd party* dodaci koje koristimo za prikaz sadržaja
/dist/GeoGebra – sadrži *3rd party* kod koji služi za prikaz interaktivnih elemenata razvijenih kroz GeoGebra platformu.
/dist/geogebra-src - Izvorne GGB datoteke koje se mogu uređivati kroz GeoGebra aplikaciju
/dist/katex – sadrži *3rd party* kod koji služi za prikaz LaTeX formula
node_modules – sadrži sve pakete iz **package.json** kao i sve njihove međuovisnosti, međutim postaje vidljiv tek nakon inicijalizacije *build* procesa koji je opisan kasnije u tekstu.
package.json – ova datoteka sadrži popis paketa koji su nužni za rad s izvornim kodom
README.md – ova datoteka sadrži kratke upute za programera koji će raditi na kodu

C. *Build* proces

Za početak rada potrebno je pokrenuti naredbu npm install. Ona će instalirati sve potrebne pakete za funkcioniranje *build* procesa, u tom trenutku **node_modules** direktorij postaje dostupan.

Webpack

Glavni dio build procesa kontroliraWebpack i to kompajliranje JS koda, *single file* Vue komponenti te konkatenaciju raznih JS modula u 4 konačna *bundlea:*
/dist/bundle.js – sadrži kod VueJS komponenti
/dist/common.js – sadrži sav kod koji nije u vidu VueJS komponenti
/dist/vendor.js – sadrži sav *3rd party* kod
/dist/vue.js – sadrži VueJS framework

Konfigracija webpacka se nalazi u **/src/carnet/config/webpack.config.carnet.js** (razvojna verzija) i u **/src/carnet/config/webpack.config.carnet.production.js** (produkcijska verzija). Razlika između razvojne i produkcijske konfiguracije Webpacka je što produkcijska kreira datoteke s .min sufiksom u imenu, a JS sadržaj je minificiran. Naredbom npm run carnet pokreće se Webpack u razvojnoj verziji, *watcher* prati promjene nad datotekama i na svaku promjenu pokreće kompajliranje koda.
Naredba npm run carnet_prod pokreće buildanje koda za produkcijsku verziju, ovaj proces može potrajati desetak sekundi.

Struktura JS datoteka

Generalno JS kod je pisan koristeći najmoderniju ES6 sintaksu i mogućnosti, a koristeći Babel *plugin* za Webpack se taj kod konvertira u stariju inaču EcmaScripta koju stariji preglednici mogu interpretirati.

Glavna odlika arhitekture koda jeste hibridna VueJS aplikacija, koja se definira u jednoj datoteci, a koristi primarni *inline template* definiran u HTML-u koji je onda kroz VueJS obogaćen funkcionalnostima koristeći direktive i prilagođene komponente. Dodatno uvelike se oslanja na uključivanje raznih dodatnih VueJS komponenti, koje su većinom pisane u SFC (single file component) formatu sa .vue ekstenzijom, i poneku komponentu koja se opet oslanja na *inline template* i .js ekstenziju.

Sav VueJS kod u konačnici završi u bundle.js datoteci, a za njeno izvršavanje je preduvijet učitavanja vue.js, common.js i vendor.js datoteka. Iz tog razloga se sve skripte poslije vue.js u HTML-u uključuju sa defer atributom.

JS i CSS kod pojedinih interaktivnih komponenti se na stranicu uključuje *inline* i to u procesu *exporta*, odnosno izrade hpub arhive koja je dostavljena naručitelju. Takve komponente su izvedene na način da svoju konfiguraciju spremaju na globalnu varijablu, te se ta varijabla koristi za registraciju i inicijalizaciju globalne VueJS komponente, koja se onda konzumira u *inline templateima* u nastavku HTML-a.

Ovakvom arhitekturom osigurana je relativno malena veličina *bundlea* koji su učitani na svim stranicama s obzirom na funkcionalnosti, a kod za pojedine bogate interaktivne komponente se učitava po potrebi samo na stranicama na kojima se te komponente koriste.

/src/carnet/scripts/main.js – glavna JS datoteka, njen zadatak je inicijalizacija vršne VueJS komponente kao i pozivanje koda koji uključuju Pojmovnik, Tražilicu, renderiranje LaTeX jednadžbi, interaktivnost korisničkog sučelja DOS-a van interaktivnih elemenata, što uključuje razne galerije, *carousele*, *lightboxove*, *scrollanje* na klik, *custom* video player rješenje i slično.

Prvi dio main.js datoteke sadrži razne *polyfille* koji omogućavaju korištenje modernih JS mogućnosti u starijim preglednicima, konkretno IE.

U nastavku vidimo pozivanje raznih VueJS mixina koji se koriste u vršnoj komponenti:

initMediaHub – kreira instancu VueJS komponente koja je zadužena za to da na stranici na kojoj se nalaze komponente interaktivnog sadržaja samo jedan video/audio player može biti aktivan u svakom trenutku. Funkcionira na način da pri početnom učitavanju stranice na svaki audio i video element dodaje *event listenere* koji brinu o tome da ako se jedan element pokrene, pauzira sve ostale.

initLatex – definira konfiguraciju za Katex (https://katex.org/) koji služi za prikaz LaTeX formula u komponentama interaktivnog sadržaja. Koristi se autorender plugin https://katex.org/docs/autorender.html, definiraju *delimiteri* unutar kojih će se nalaziti LaTeX izrazi, te posebne klase unutar kojih će se ignorirati latex izrazi. Naime Katex zna ulaziti u konflikte sa VueJS template *renderingom* ako se ne konfigurira da ignorira neke dijelove DOM-a.

initSearch – dodaje u DOM skriptu za pretraživanje sadržaja komponenti interaktivnog sadržaja. Budući da se indeks po kojem se vrši pretraživanje generira tek pri *exportu*, odnosno izradi arhive koja se dostavlja naručitelju, nužno je da se tražilica ova uvjetovano uključuje na stranicu.

scrollToBlock – metoda koju vrša komponenta koristi da bi korisniku pomaknula scroll poziciju preglednika do točno određenog bloka, ovisno o zapisu u URL-u

initSideControls – koristi IntersectionObserver kako bi na blokovima koji se nalaze u *viewportu* prikazao kontrole za inkluzivni prikaz bloka i alternativni audio sadržaj

U nastavku datoteke definirane su globalne VueJS komponente koje se koriste. Nećemo ići u raspisivanje detalja svake pojedinačno, ali ćemo istaknuti neke zanimljivije:

video-player – budući da se HTML datoteke komponenti interaktivnog sadržaja mogu pokretati u raznim okruženjima, uključujući direktno s diska, a bez poslužitelja, web preglednici nameću određena ograničenja na korisnika primarno radi sigurnosti. Jedno od tih ograničenja jeste blokiranje XHR mrežnih upita. Budući da za učitavanje titlova u HTML video se koristi XHR, koristimo rješenje koje se oslanja na to da je sadržaj titlova ispisan u sami HTML. Pri učitavanju videa, SRT titlova se učitava iz DOM-a i koristeći WebVTTParser se kreira traka s titlovima koja se onda učitava na video element.

block-image – komponenta zadužena za rudimentarni prikaz slika na stranicama, kao i za napredniji prikaz gdje se slika može otvarati u *fullscreen* prikazu, imati naslov, opis i dodatne metapodatke. Konfiguracija se radi u HTML-u koristeći VueJS props.

izzi-icon – komponenta za prikaz SVG ikone. Koristi VueJS props kako bi se definiralo koju ikonu i u kojoj dimenziji želimo prikazati

izzi-block – bazna generička komponenta koja služi kao wrapper za svaki blok koji se vidi na stranici, od najjednostavnijih, do najkompleksnijih. Ova komponenta brine o prikaz blokova, inkluzivnom sadržaju blokova, alternativnom audio sadržaju, omogućava da se korisniku namjesti scroll tako da vidi točno određeni blok i slično.

glossary – komponenta koja brine o prikazu Pojmovnika, povezivanju pojedinih pojmova sa slovom, *scrollanjem* do pozicije za određeno slovo i sve ostalo vezano za Pojmovnik

analytics – *renderless* komponenta koja na Google Analytics šalje podatke o korištenju komponenti interaktivnog sadržaja, pratimo pokretanje/zaustavljanje video *playera*, pregledavanje jedinica, otvaranje/zatvaranje galerija slika i pokušaje rješavanja kvizova.

Gulp

Dodatno uz korištenje Webpacka za JS, koristi se i Gulp, primarno za kompajliranje SCSS datoteka te za izradu ikona od SVG-ova. Pokretanjem naredbe gulp u direktoriju **/src/carnet** pokreće se *watcher* koji prati promjene nad SCSS datotekama i na svaku promjenu pokreće kompajliranje koda. Za Gulp nemamo odvojene razvojne i produkcijske verzije, već se producirani CSS automatski minificira, a *source-maps* su dostupne kako bi se olakšao razvoj.

SCSS datoteke su logički grupirane na više razina i ima ih jako velik broj pa nećemo prolaziti svaku pojedinačno, no objasnit ćemo osnovne principe i najvažnije pojedine datoteke. Svaka SCSS datoteka treba imati prefix _ prije nego li se uveze u **style.scss** kako bi se ispravno kompajlirala kao *partial*, a ne samostalna CSS datoteka. SCSS je pisan imajući na umu BEM metodologiju (https://en.bem.info/methodology/), nužno je napomenuti da ima i komada *3rd party* koda koji je pisan drugačijom metodologijom, u takvim slučajevima smo pokušali

poštovati stil autora koji je taj kôd izvorno pisao, ali opet ostati dosljedni vlastitoj metodologiji koja je ipak varijacija na BEM, gdje koristimo __ kao separator u imenima blokova i elemenata, a -- kao *modifier* za varijacije boje, veličine i tomu sličnog.

Struktura SCSS datoteka

Kao osnovica za stilove korišten je Bootstrap, odnosno pojedine njegove komponente, kako bi se ubržao razvoj i postigao ujednačeni prikaz u različitim web preglednicima.

```
// Mixins
@import "../../common/scss/mixins/general-mixins";
@import "../../common/scss/mixins/breakpoints";

// Utilities
@import "../../common/scss/utilities/functions";
@import "../../common/scss/utilities/variables";
@import "../../common/scss/utilities/reboot";
@import "../../common/scss/utilities/utilities";
@import "../../common/scss/utilities/flex";
@import "../../common/scss/utilities/spacing";
@import "../../common/scss/utilities/embed";
@import "../../common/scss/utilities/debug";

// Layout mixins
@import "../../common/scss/mixins/grid-framework";
@import "../../common/scss/mixins/grid";
```

Dodatne bitnije SCSS datoteke su:

/src/carnet/scss/style.scss – glavna SCSS datoteka koja poziva sve ostale
/src/carnet/scss/_general.scss – općeniti stilovi
/src/carnet/scss/theme/_palettes.scss – varijacije boja za razne komponente interaktivnog sadržaja
/src/carnet/scss/theme/_variables.scss – razne varijable pomoću kojih se može značajno prilagodit izgled komponenti interaktivnog sadržaja
/src/carnet/scss/type/_font-face.scss - @font-face definicije web fontova koje se koriste u komponentama interaktivnog sadržaja
/src/carnet/scss/settings/_dark-mode.scss – razne prilagode za prikaz u noćnom prikazu
/src/carnet/scss/settings/_dyslexic.scss – razne prilagodbe za prikaz uz font za disleksičare
/src/carnet/scss/components/* - razni SCSS-ovi za prikaz komponenti sučelja
/src/carnet/scss/templates/* - razni SCSS-ovi za prikaz pojednih *templatea* komponenti interaktivnog sadržaja

Ikone

Generiranje ikona koristeći gulp ne događa se automatizmom kao koajliranje SCSS-ova, već se pokreće naredbom gulp svgicons, kada se pokaže potreba za doradom seta ikona. Preduvijet za funkcioniranje ove naredbe je da se ikone kao individualne SVG datoteke postave u direktorij **/src/carnet/icons,** a direktorij **/src/carnet/icons/_build** isprazni. Nakon

pokretanja gulp svgicons, u **/src/carnet/icons/_build/symbol/svg** direktoriju možemo pronaći SVG datoteku čiji sadržaj treba kopirati u **/src/carnet/twig/functions/icons.twig.** Također **/src/carnet/icons/_build/symbol/ sprite.symbol.html** sadrži primjer korištenja ikona kao i CSS potreban za njihovo funkcioniranje. Upravo taj generirani inline CSS treba iskopirati u datoteku **/src/carnet/scss/icons/_icons.scss** kako bi se ikone prikazivale u točnoj dimenziji i kako bi naslijeđivale boje postavljene u CSS-u.
Kod pripreme ikona bitno je znati da dimenzija treba biti 24x24px, a ne bi trebalo koristiti stroke ni fill atribute, naime njihova vrijednost se postavlja CSS-om, a ne *inline*. Ovakav pristup nam omogućava da imamo samo jednu od svake pojedine ikone, i CSS-om reguliramo prikaz, a ne moramo proizvesti svaku varijaciju posebno. Naziv SVG datoteke u izvornom direktoriju će biti naziv po kojemu će se ikona kasnije pozivati u HTML-u.

D. Komponente

Postoje dvije vrste komponenti koje su integrirane u sustav.

Prva su VueJS komponente koje su integrirane unutar vršne VueJS komponente. One se nalaze u **/src/common/scripts/components** direktoriju odakle su import-ane u main.js datoteku. Neke od njih su već opisane u prethodnom poglavlju “Struktura JS datoteka”.

Druga su eksterne VueJS komponente koje počivaju na zasebnim repozitorijima te u procesu exporta svoju konfiguraciju spremaju na globalnu varijablu odakle registraciju i inicijalizaciju preuzima globalna VueJS komponenta. Njihova struktura je sljedeća:

/dist – u ovom direktoriju nalaze se sve kompajlirane datoteke koje se u konačnici koriste pri prikazu digitalnih obrazovnih sadržaja. Sve datoteke dolaze u dvije verzije: standalone i app. Standalone verzija za korištenje tijekom razvoja i build-a ukoliko će komponenta na stranici stajati samostalno. App verzija za integrirano korištenje unutar vršne VueJS komponente.
/node_modules – sadrži sve pakete iz **package.json** kao i sve njihove međuovisnosti, međutim postaje vidljiv tek nakon inicijalizacije *build* procesa kasnije opisanog u tekstu.
/public – sadrži index.html file u kojem se obavlja inicijalizacija komponente za korištenje u developmentu. File sadrži linkove na css stylesheet i VueJS skripte. Neke komponente sadrže config prop koji sadrži podatke koji se koriste prilikom prikaza komponente u developmentu.
/src – u ovom direktoriju nalaze se sve izvorne datoteke programskog koda. Sadrži poddirektorije **assets**, **components**, **translations** te file-ove **App.vue**, **main.js**, **izzi.js** i **messages.js**
/src/App.vue – glavni file komponente koji u sebi sadrži html template unutar VueJS komponente koja uz mixine te metode definirane unutar komponente zajedno sa css-om specifičnim za pojedinu komponentu rezultira kodom izvršavanjem kojega se komponenta prikazuje na stranici
/src/izzi.js – file koji se koristi za kreiranje builda prilikom integracije u vršnu komponentu.
/src/main.js – file koji se koristi za kreiranje builda prilikom developmenta
/src/messages.js – file koji se koristi za kompilaciju prijevoda
***/src*/assets** – direktorij koji sadrži slike i audio zapise koji se koriste u komponenti
/src/components – direktorij koji sadrži komponente koje imaju neku specifičnu logiku i koje se zatim import-aju u glavnu komponentu
/src/translations – direktorij koji sadrži JSON file-ove s prijevodima
babel.config.js – ova datoteka sadrži konfiguraciju babel dodatka
vue.config.js – ova datoteka sadrži webpack konfiguraciju VueJS komponente
package.json – *ova datoteka sadrži popis paketa koji su nužni za rad s izvornim kodom*

README.md *– ova datoteka sadrži kratke upute za programera koji će raditi na kodu*

Za početak rada potrebno je pokrenuti naredbu npm install. Ona će instalirati sve potrebne pakete za funkcioniranje komponente, u tom trenutku **node_modules** direktorij postaje dostupan. Nakon toga naredba npm run dev pokreće komponentu za rad u razvojnoj verziji, dok naredba npm run build pokreće buildanje koda za produkcijsku verziju što može potrajati desetak sekundi.

E. Dodatne specifičnosti

KaTeX

Rješenje odabrano za prikaz LaTeX formula jeste https://katex.org/, odabrano je spram "klasičnog" MathJax rješenja zbog drastično povećanih performansi učitavanja, što se naročito osjeti na mobilnim i tablet uređajima. Vokabular podržanih LaTeX izraza je vidljiv na poveznici https://katex.org/docs/supported.html i dovoljno je širok da pokrije svrhu i namjenu izrađenih natječaja. Više detalja o konkretnoj implementaciji je rečeno pri opisu **initLatex** mixina, najvažnije je imati na umu da ako se koristi u VueJS komponentama, treba paziti da se dijelove koje će VueJS u *templateima* populirati s nekakvim vrijednostima s modela izolira od autorender Katex plugina, na način da im se postavi klasa **js-katex-ignore**.

3. Struktura direktorija sadržajnog dijela (HPub arhiva)

Kako bi struktura direktorija i datoteka bila razumljiva i programskim parserima, a u svrhu distribucije sadržaja koristimo hpub standard https://github.com/bakerframework/baker/wiki/hpub-specification.

U *root* direktoriju se nalazi datoteka **index.html** i **Ovdje pokrenite.html** koje su sadržajno istovjetne, a obje se mogu koristiti za pokretanje komponenti interaktivnog sadržaja.

Ukoliko spomenute .html datoteke sadrže neku od eksternih VueJS komponenti, tada će kompajlirani JavaScript kod pripadajuće komponente biti inline ubačen unutar script taga, dok će kompjlirani css kod iste komponente biti inline ubačen unutar style taga.

/datastore – sadrži sve multimedijalne datoteke koje se koriste u sadržajima. Datoteke su podijeljene u direktorije prvo po jedinstvenom identifikatoru sadržaja u kojem se nalaze, zatim po tipu sadržaja, i zatim po datumu, npr
/datastore/18/publication/10964/pictures/2020/01/22

Ukoliko se radi o eksternim VueJS komponentama one će unutar **/datastore** direktorija nalaziti u components direktoriju, te zatim u direktoriju nazvanom po imenu komponente.

/profil/dist – sadrži sve kompajlirane JS i CSS datoteke, kao i sve potrebne grafičke *assete*, fontove itd. Ukoliko dolazi do kakvih funkcionalnih dorada isporučenih arhiva sadržaja, upravo ovaj direktorij bi se trebao zamijeniti s **dist** direktorijem iz arhive koda.

Ostatak sadržaja direktorija je popisan u datoteci **book.json.**

4. Dizajn i vizualni standard

A. Boje

Dizajn komponenti interaktivnog sadržaja izveden je u 4 primarne palete boja, kako bi se postigla distinkcija između komponenti interaktivnog sadržaja različitih predmeta. Svaka paleta definirana je s dvije primarne boje koje su međusobno komplementarne i koriste se za naglašavanje pojedinih dijelova sučelja poput *call-to-actiona*, klikabilnih elemenata, *hero* sekcija itd.

Boje koje se koriste u paletama:
Paleta 1: Primarna boja: #f43d98; Sekundarna boja: #0f80c1;
Paleta 2: Primarna boja: #ffd452; Sekundarna boja: #61b021;
Paleta 3: Primarna boja: #3de9f4; Sekundarna boja: #b04921;
Paleta 4: Primarna boja: #ffbd00; Sekundarna boja: #0fc1c1;

Palete boja su definirane u mixinu u datoteci **/src/carnet/scss/theme/_palettes.scss** koji nakon kompajliranja rezultira primjenom boja na različite elemente.

Boje koje se korsite za razne elemente u interaktivnim komponentama:

```
// Color Variables
$colorBlack: #000;
$colorWhite: #FFF;
$colorDirtyWhite: #e5e5e5;
$colorSilverDark: #596071;
$colorSilverLight: #838BA0;
$colorBluePrimary: #1763a6;
$colorRedPrimary: #d0021b;
$colorGrayPrimary: #4a4a4a;
$colorGreenLight: #a3d76e;
$color-base-gray: #3d3d3d;
$dm-blue: #308fe4;

$color-win: #a5da6f
$color-fail: #cd001c;
$color-focus-ring: #c99300;
$color-border-gray: #e0e0e0;
$color-border-gray-rgba: rgba(#000, 0.12);
$color-bg-off-white: #fafafa;

// Dark mode
$black: #1a1a1a;
$off-black: #2b2b2b;
$border-black: #000;
$border-light: #8a8a8a;
```

Boje koje se koriste za razne elemente u interaktivnim komponentama su navedene u datoteci **/src/carnet/scss/theme/_variables.scss**

B. Tipografija

Kod odabira tipografije vodilo se računa o sljedećim aspektima:
- mora podržavati veliki broj slovnih znakova i tipografskih elemenata
- mora podržavati znakove iz različitih jezika i pisama
- mora biti jednostavna i čitka
- mora biti lako i uvijek dostupna
- obitelj mora sadržavati što veći broj rezova zbog različitih primjena.

Odabrana tipografija koja zadovoljava navedeno je IBM Plex Sans.

Dostupna je besplatno na https://fonts.google.com/specimen/IBM+Plex+Sans

"Fall back" tipografija koja se koristi ako postoji problem s učitavanjem fontova s interneta je bilo koja san serifna tipografija koja je postavljena na korisnikovom računalu kao osnovna (npr. Arial, Verdana, Helvetica...).

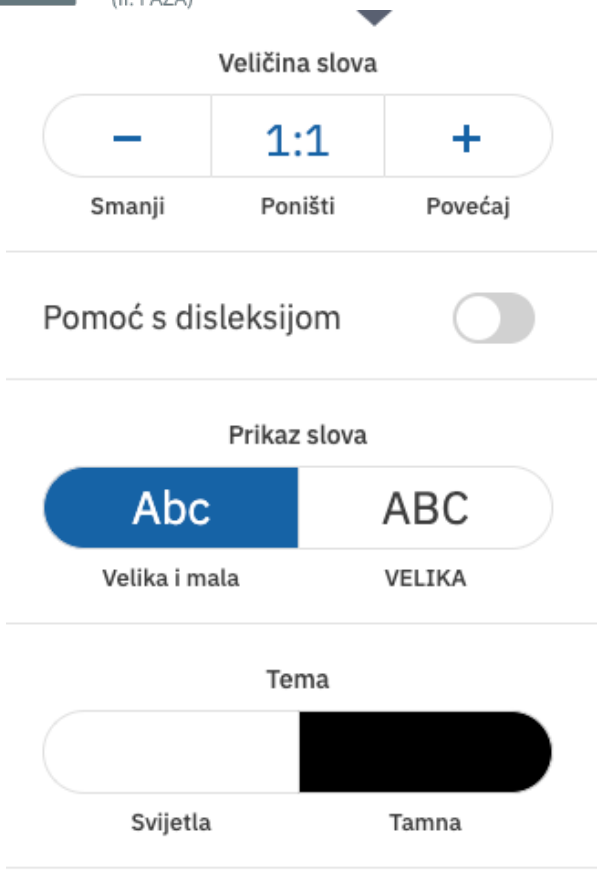
C. Ikone i piktogrami

U sučelju dodatno značenje i/ili uočljivost nekim elementima dana je koristeći ikone uz tekstualne oznake. Ikone koje se koriste su primarno inspirirane Material Icons setom od Googlea, primarno zbog svoje univerzalne prepoznatljivosti, neutralnog estetskog stila i otvorenog izvornog koda, što nam je omogućilo da ih koristimo kao SVG-ove kroz ranije opisani *build* proces radi postizanja boljih performansi učitavanja i bolje pristupačnosti.

Dostupne su besplatno na https://material.io/resources/icons/?style=baseline

D. Pristupačnost

Pristupačnost je aspekt dizajna koji smo imali na umu od početka i prožet je svugdje, to uključuje brigu o dovoljno jakim kontrastima boja, čitkoj tipografiji, razmaku između elemenata i položaju na stranici, vidljivom označavanju *hover i focus* stanja kako bi korisniku bilo jasno kakvu interakciju može očekivati na stranici prije nego li se ona dogodi. Dodatno uz to korisniku dozvoljavamo da sam po želi podešava pojedine aspekte sučelja.



Korisnik sam može:

- povećati sve tekstualne elemente (to uključuje sve veličine naslova, podnaslova, tekstualnih blokova, potpisa pod slikama, videima, galerijama tid.)
- zamijeniti tipografiju za onu prilagođenu za osobe s disleksijom
- može omogućiti prikaz isključivo u velikim slovima radi lakše čitljivosti
- uključiti noćni prikaz u kojemu je povećan kontrast između elemenata, a dodatno se smanjuje opterećenje na oči u večernjim satima kada je osvjetljenje slabije