Математичка гимназија

МАТУРСКИ РАД

- из програмирања и програмских језика -

Прогресивна Веб Апликација

Ученик:
Огњен Маринковић IVц

Ментор:
Петар Радовановић

Београд, јун 2021.

Садржај

1

Увод

Када развијају апликацију за мобилни телефон, софтверски инжењери се често сусрећу са проблемом различитих оперативних система. Главна потешкоћа је што је за сваки оперативни систем било потребно писати посебну апликацију између којих је дељење кода било практично немогуће. Ово је знатно отежавало процес изграђивања апликације појединцима и компанијама који нису имали ресурсе да овакав начин рада приуште.

Предност оваквих апликација је то што су поуздане. Не зависе од других апликација на систему и могу комуницирати са фајл системом и хардвером телефона.

Са друге стране, веб апликације функционишу на свакој платформи која има компатибилни веб-претраживач, што је већина данашњих оперативних система. Изврсне су у достизању нових корисника јер омогућавају приступ свакоме и лако их је проследити путем линка. Међутим, њима недостају све функционалности које поседују системски специфичне апликације.

Као једно од решења настала је Прогресивна Веб Апликација.

2

Прогресивна Веб Апликација

2.1 Појам Прогресивне Веб Апликације

Прогресивне веб апликације су замишљене тако да комбинују функционалност системски специфичних и веб апликација. Циљ прогресивне веб апликације је да буде способна, поуздана и да ју је могуће инсталирати.

- Способност

  Донедавно су само системски специфичне апликације имале могућност коришћења функција самог система као што су, на пример, геолокација и системске нотификације. Данас су ове функционалности омогућене и прогресивним веб апликацијама. Пратећи овај тренд, нови и предстојећи API-еви циљају на то да прошире оквир способности веб-а, додавајући приступ фајл систему, нотификације, контролу медијума и слично.

- Поудзаност

  Прогресивна веб апликација мора бити брза и поуздана како би кориснику остварила осећај да користи „праву", системски специфичну апликацију. Такође мора бити функционална без обзира на брзину интернет конекције.

- Могућност инсталирања

  Инсталиране прогресивне веб апликације раде у засебним прозорима одвојеним од веб-претразивача. Потребно је да „имитирају" системски специфичну апликацију тиме што их корисник може отворити директно из свог система. Опсег могућности веб апликације се знатно повећа након што она бива инсталирана.

2.2 Како функционише инсталација

Да би се неки онлајн сајт признао као Прогресивна Веб Апликација (ПВА), чиме би Google Chrome понудио кориснику да је инсталира, мора испуњавати четири услова, постављених од стране Google-а:

1. Сајт је посећен два пута у року од 5 минута
2. Сајт је хостован на безбедној HTTPS конекцији. Овиме ће се корисници осећати безбедније да апликацији омогуће приступ неким функционалностима њиховог уређаја.
3. Инсталиран је валидан JSON манифест—ово је фајл који говори апликацији како да се понаша након што је инсталирана
4. Инсталиран је валидан Услужни Радник (Service Worker)—најбитнији део било које ПВА. Он је задужен за кешовање свих фајлова, услуживање нотификација, ажурирање садржаја, манипулацију података и др.

   Услужни Радник је скрипта која се налази на уређају корисника и ради независно од било ког постојећег сајта или апликације. Тиме може кориснику показивати садржај сајта и чак манипулисати њиме чак и када није повезан на интернет.

   Када корисник учита сајт први пут, Услужни Радник се инсталира на уређај и ПВА се може инсталирати.

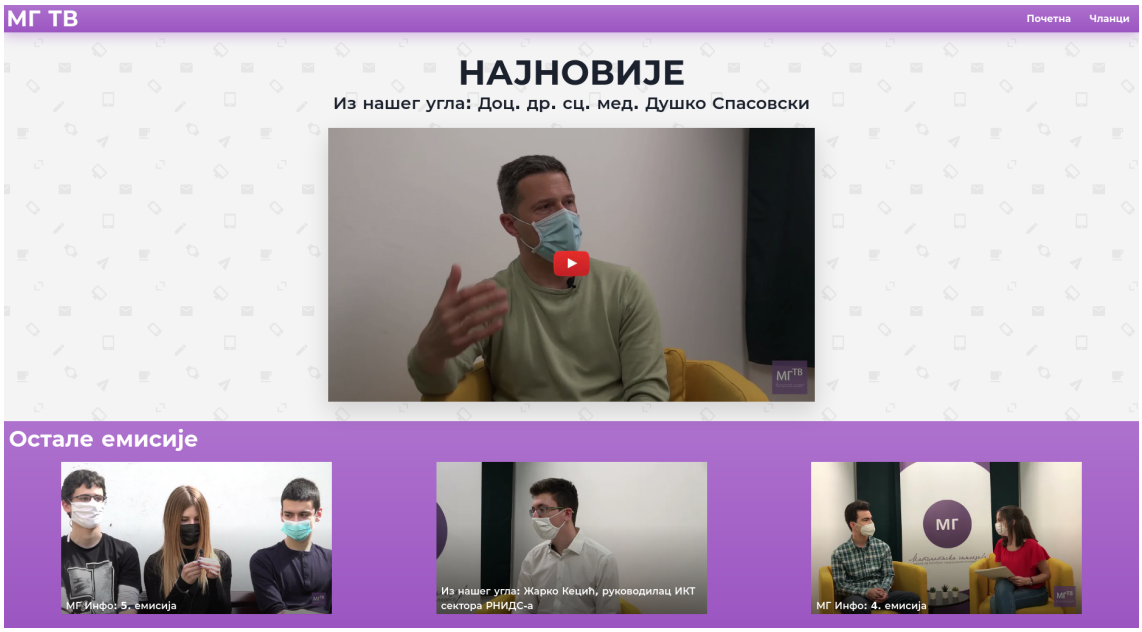
Након што испуни ова четири услова, апликацију корисник може инсталирати када посети сајт, чиме је додаје на свој Почетни Екран. Од тада се она на систему понаша као свака друга апликација.

3

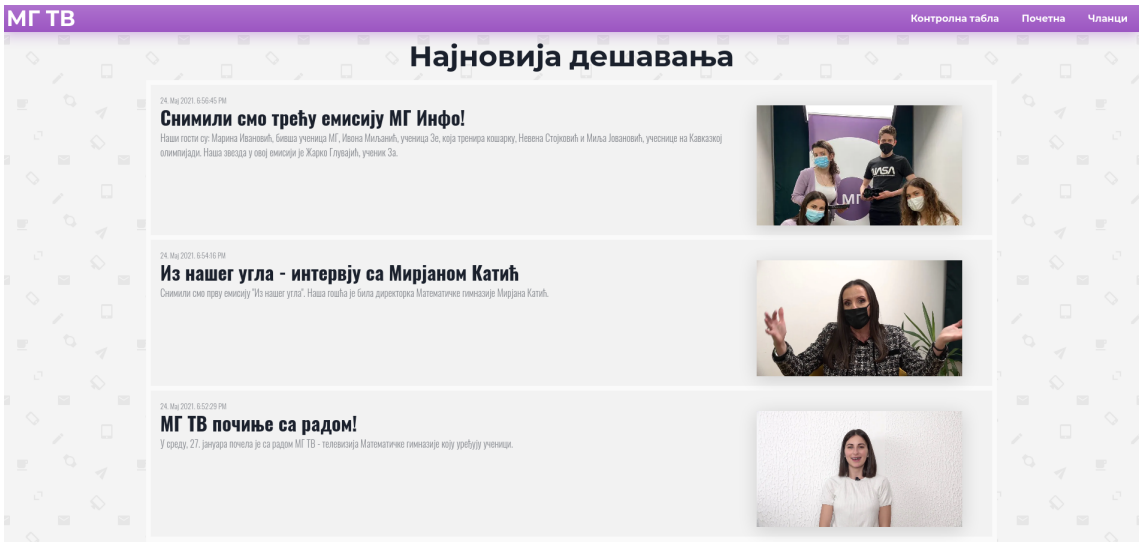
МГ ТВ Сајт, Прогресвина Веб Апликација

3.1 Опис пројекта

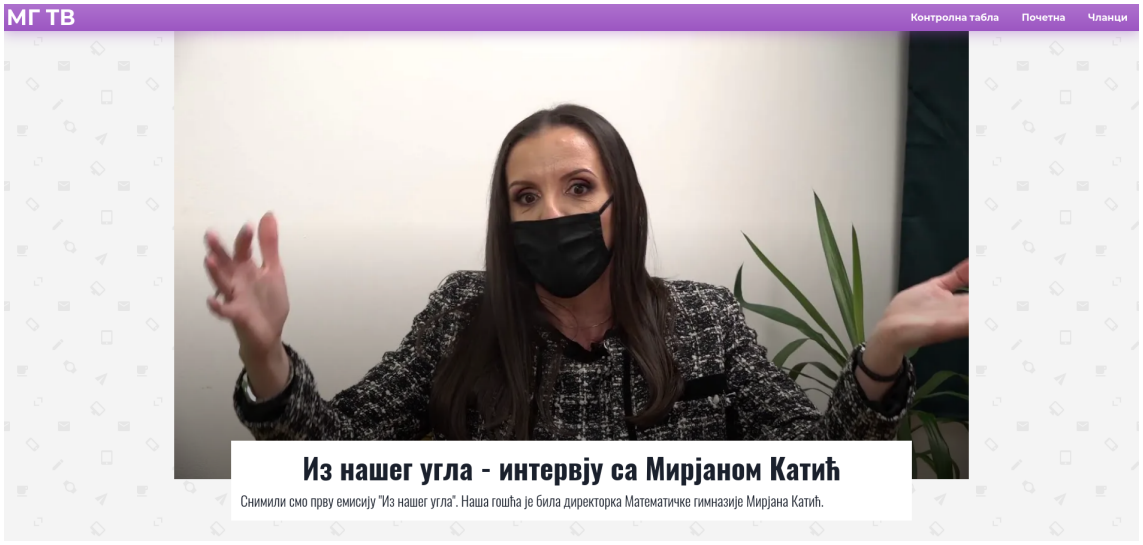
Сајт телевизије Математичке гимназије написан уз овај пројекат је пример Прогресивне веб апликације. Састоји се од почетне стране која приказује најновије снимке МГ ТВ-а на Јутјубу и странице која приказује најновије написане чланке.



Слика 3.1: Почетна страница



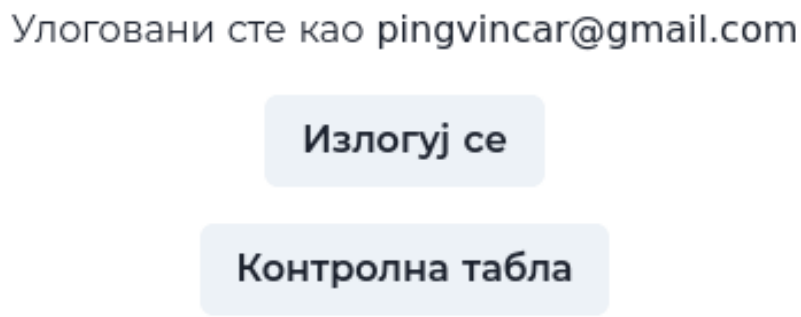
Слика 3.2: Страница Чланци



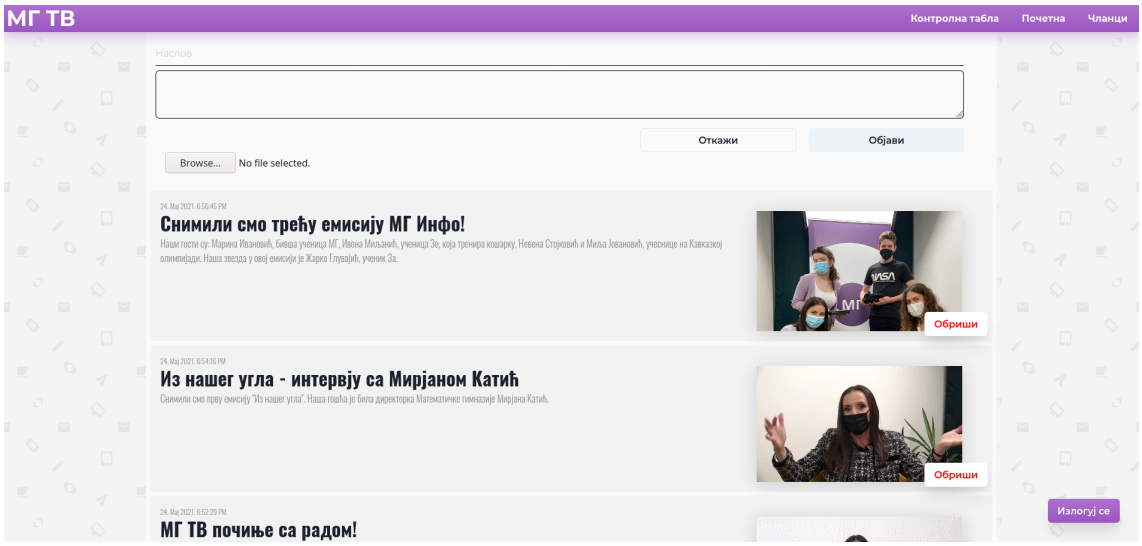
Слика 3.3: Страница чланка

На сајту такође постоје странице које нису доступне за јавност. Како би се њима приступило потребно је улоговати се као администратор на /login страници. Након тога се може приступити контролној табли на којој је могуће написати нови чланак и обрисати старе.

Слика 3.4: Страница за логовање



Слика 3.5: Страница која потврђује аутентикацију



Слика 3.6: Страница контролне табле

3.2 Коришћене технологије

3.2.1 React и Next.js

React је JavaScript библиотека за грађење коринсичких интерфејса. Главне особине ове библиотеке су:

- Декларативност

  Довољно је да програмер дизајнира изглед апликације за свако њено стање и React ће ефикасно ажурирати и рендеровати потребне компоненте када се подаци измене.

- Заснованост на компонентама

  Целокупан кориснички интерфејс се састоји од засебних компонената које управљају сопственим стањем. Овиме се код не ремети мењањем једне компоненте и омугућава се да се исте компоненте пишу само једном, а појављују на више места у апликацији.

Next.js је радни оквир (framework) за грађење статичких и апликација рендерованих на серверу у React-у. Неке од главних функционалности које пружа Next.js су:

- Рендеровање статичких страница и страница на серверу

  Next.js може рендеровати странице на два начина. Статичке странице, чији се подаци не мењају често, или уопште, ће рендеровати само једном, у време изградње апликације. Странице рендероване на серверу ће Next.js генерисати када корисник серверу упути захтев да му се прикаже страница. Ово омогућава интернет претраживачима и ботовима да боље индексују странице и повећава ефикасност сајта.

- Оптимизација слика

  Једна од ставки која највише успорава сајтове данас јесу лоше оптимизоване слике за веб. Next.js аутоматски оптимизује величину слике коју шаље кориснику како би се сајт учитао најбрже могуће.

- Фајл-систем усмеравање

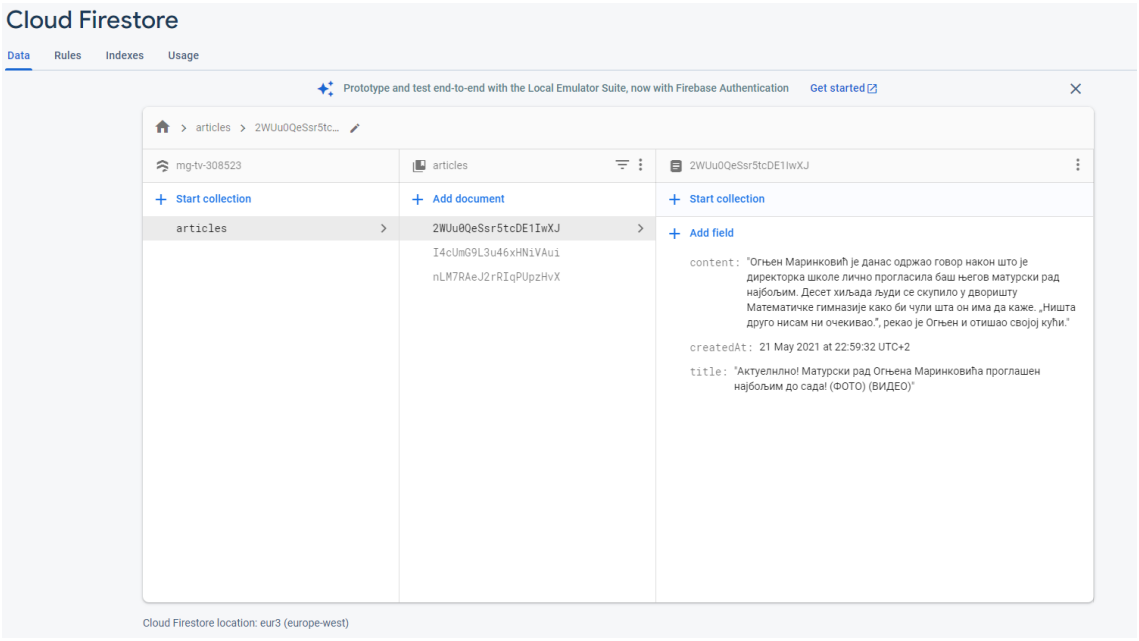
  Next.js ће аутомацки направити линкове за сваку страницу на основу њеног имена у фајл систему.

3.2.2 Chakra UI

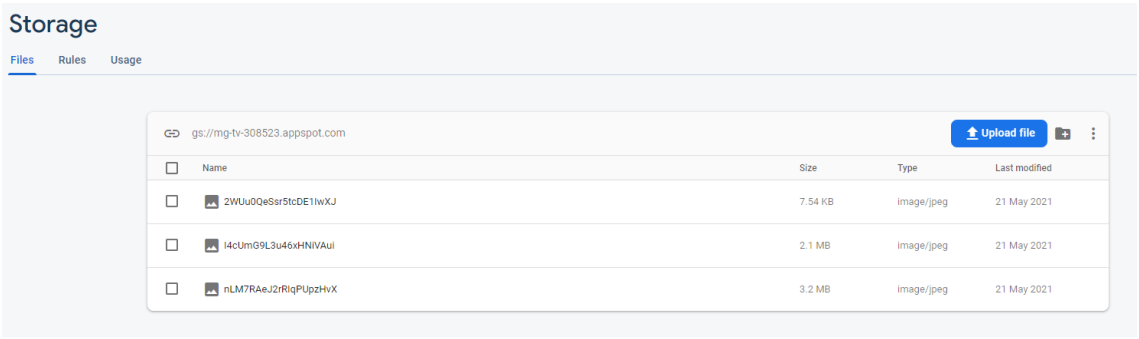
Chakra UI је једноставна, модуларна и приступачна библиотека компонената која олакшава и убрзава процес прављења корисничког интефејса у React апликацијама.

3.2.3 Firebase

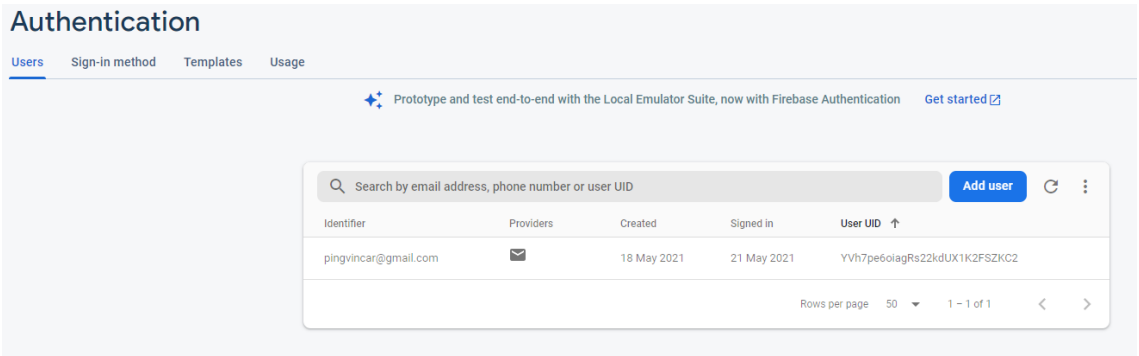
Firebase је Google-ова платформа за брзу изградњу апликација високог квлаитета. У оквиру овог пројекта платформа је коришћена као back-end за аутентикацију корисника и базу података. Помоћу Firebase-а је избегнута одговорност управљања серверима и скалирања апликације.



Слика 3.7: База података чланака (Firebase Firestore)



Слика 3.8: Складиште слика (Firebase Storage)



Слика 3.9: База података корисника (Firebase Authentication)

4

Делови изворног кода

4.1 Почетна страна

```
export async function getStaticProps() {
  const res = await fetch(`${YOUTUBE_API}?part=snippet%2CcontentDetails&maxResults=4&playlistId=${PLAYLIST_ID}&
key=${process.env.YOUTUBE_API_KEY}`)
  const data = await res.json()
  const videos = await data.items

  return {
    props: {
      videos,
    },
    revalidate: 60
  }
}

export default function Home({ videos }) {

  const newestVideo = videos[0]
  const otherVideos = videos.slice(1)

  return (
    <>
      <Head>
          <title>{'МГ ТВ: Насловна'}</title>
      </Head>
      <TopBar />
      <Box display="flex" flexDirection="column" alignItems="center">
        <NewestVideo video={newestVideo} />
        <OtherVideos videos={otherVideos} />
      </Box>
    </>
  )
}
```

Слика 4.1: Изворни код почетне странице

Функција `getStaticProps()` је део Next.js оквира и позива се када се апликација изграђује. У конкретном случају почетне стране, та функција шаље захтев

Youtube-овом API-ју за најновијим снимцима на МГ ТВ каналу. Затим се најновији видео прослеђује компоненти NewestVideo, а остали компоненти OtherVideos.

4.1.1 NewestVideo Komponenta

```
import { Box, Heading,Button, AspectRatio } from '@chakra-ui/react'
import { useState } from 'react'

import Image from 'next/image'

import YoutubeEmbed from '../youtube-embed/youtube-embed'


export default function NewestVideo ({ video }) {

    const [showEmbed, setShowEmbed] = useState(0)

    return (
        <Box backgroundImage='url(email-pattern.png)' w="100%" display="flex" flexDir="column" alignItems="center"
p="32px">
            <div dangerouslySetInnerHTML={{__html:"<!-- Background pattern from Toptal Subtle Patterns -->"}} />
            <Box display="flex" flexDir="column" alignItems="center" mb="24px">
                <Heading as="h1" textTransform="uppercase" fontSize={['32px','40px','56px']}>Најновије</Heading>
                <Heading as="h2" align='center' fontSize={['16px','20px','28px']} fontWeight="600">
{video.snippet.title}</Heading>
            </Box>
            <Box filter="drop-shadow(0px 15px 20px rgba(0, 0, 0, 0.15));">
                {showEmbed ? <YoutubeEmbed key={video.snippet.resourceId.videoId} embedId=
{video.snippet.resourceId.videoId} /> :
                <Box>
                    <AspectRatio w={[320,400,560,800,800]} ratio={16 / 9} position='relative'>
                        <Button aria-label='youtube video' borderRadius='0' onClick={() => {
                            setShowEmbed(1)
                        }}>
                            <Image
                                src={`https://i.ytimg.com/vi/${video.snippet.resourceId.videoId}/maxresdefault.jpg`}
                                layout='fill'
                                objectFit='contain'
                                alt='Newest Video Thumbnail'
                            />
                            <Box position='absolute' zIndex='1'>
                                <Image
                                    src={'/youtube.png'}
                                    height={64}
                                    width={64}
                                    priority={true}
                                    alt='Youtube Logo'
                                />
                            </Box>
                            <Box w='100%' h='100%' position='absolute' bg='rgba(0, 0, 0, 0.3);'></Box>
                        </Button>
                    </AspectRatio>
                </Box>
                }
            </Box>
        </Box>
    )
```

Слика 4.2: NewestVideo компонента

Компонента NewestVideo приказује најновији видео који може да се гледа и на самом сајту (Youtube-ов видео плејер је убачен у сајт). Међутим, учита-

вање Youtube-овог видео плејера је поприлично споро, па све док корисник не кликне да пусти видео, сајт заправо неће приказивати плејер. За то је задужен уокривени део кода на слици. Променљива `showEmbed` чува стање плејера (да ли га приказати или не). Када је та променљива једнака нули, кориснику се приказују само главна слика видео снимка и Youtube лого који имитирају плејер. У моменту када корисник кликне на слику, `showEmbed` постаје једнака јединици и видео се пушта.

4.1.2 OtherVideos Komponenta

```
import { ListItem, UnorderedList, Box, Heading } from "@chakra-ui/layout";

import VideoCard from '../video-card/video-card'

export default function OtherVideos ({ videos }) {
    return (
    <Box as="section" w="100%" bg="linear-gradient(#AE72CE, #9C56C2)" p="8px" pb="16px">
        <Heading as="h3" color="white" mb="16px">Остале емисије</Heading>
        <Box display="flex" alignItems="center" h="90%">
            <UnorderedList ml='0' display="flex" justifyContent="space-around" w="100%" listStyleType="none"
flexWrap="wrap">
                {videos.map((video) => (
                    <ListItem key={video.snippet.resourceId.videoId}>
                        <VideoCard key={video.snippet.resourceId.videoId} videoId={video.snippet.resourceId.videoId}
title={video.snippet.title}/>
                    </ListItem>
                ))}
            </UnorderedList>
        </Box>
    </Box>
    )
}
```

Слика 4.3: OtherVideos компонента

Компонента OtherVideos приказује следећа три видео снимка. Као аргумент прихвата листу `videos` и затим методом `map` пролази кроз листу и за сваки елемент приказује једну компоненту VideoCard.

```
import { Box, Text, Link } from '@chakra-ui/react'
import Image from 'next/image'

export default function VideoCard ({ videoId, title }) {

    const thumbnailUrl = `https://i.ytimg.com/vi/${videoId}/maxresdefault.jpg`
    const videoUrl = `https://www.youtube.com/watch?v=${videoId}`

    return (
        <Link href={videoUrl}>
            <Box position="relative">
                <Image
                    src={thumbnailUrl}
                    width={445}
                    height={250}
                    priority={true}
                    alt='Youtube Video Thumbnail'
                />
                <Box position="absolute" bottom='5.5px' w='100%' h='100%' bg="linear-gradient(180deg, rgba(196, 196,
196, 0) 42.4%, rgba(0, 0, 0, 0.5) 99.99%, rgba(196, 196, 196, 0) 100%, rgba(0, 0, 0, 0) 100%)" />
                <Text position="absolute" bottom="8px" left="8px" fontWeight='600' color='white'>{title}</Text>

            </Box>
        </Link>
    )
}
```

Слика 4.4: VideoCard компонента

Компонента VideoCard прихвата као аргументе id видео снимка и његов наслов. Помоћу id-ја налази слику снимка и преко ње исписује наслов.

4.2 Страна Чланци (articles)

```
import { Box, Heading } from '@chakra-ui/react'

import TopBar from '../components/topbar/topbar'

import { getArticlesData } from '../lib/articles'
import { getImageUrls } from '../lib/adminImages'

import ArticlesList from '../components/articles-list/articles-list'

import Head from 'next/head'

export async function getStaticProps() {

    const articlesWithoutUrl = await getArticlesData()

    const articles = await getImageUrls(articlesWithoutUrl)

    return {
        props: {
            articles
        },
        revalidate: 60
    }
}

export default function Articles({ articles }) {
    return (
        <>
            <Head>
                <title>{'МГ ТВ: Чланци'}</title>
            </Head>
            <Box backgroundImage='url(email-pattern.png)'>
                <TopBar />
                <Heading textAlign='center' m='16px' size='2xl'>Најновија дешавања</Heading>
                <Box h='100vh' w={['100%', '100%', '90%', '75%']} m='auto' bg='#FAFAFA'>
                    <ArticlesList articles={articles} deleteButtons={false}></ArticlesList>
                </Box>
            </Box>
        </>
    )
}
```

Слика 4.5: Изворни код странце Чланци

Страна Чланци у функцији `getStaticProps()` позива функције `getArticlesData()` и `getImgUrls()`.

```
export async function getArticlesData() {
    const res = firebase.firestore().collection('articles').orderBy('createdAt', 'desc')
    const data = await res.get()
    const articles = data.docs.map(doc => ({
        id: doc.id,
        ...doc.data(),
        createdAt: srbenda(new Date(doc.data().createdAt.toDate()).toString()),
    }))

    return articles
}
```

Слика 4.6: Функција `getArticlesData()`

`getArticlesData()` шаље захтев Firebase Firestore-у (бази података) за инфомације о чланцима поређаним по датуму објављивања.

```
export async function getImageUrls(articlesWithoutUrl) {

    return (articlesWithoutUrl.map((article) => {
        let imgUrl = `https://firebasestorage.googleapis.com/v0/b/mg-tv-308523.appspot.com
/o/${article.id}?alt=media`

        return {
            imgUrl,
            ...article
        }
    }))
}
```

Слика 4.7: Функција `getImageUrls(articlesWithoutUrl)`

`getImageUrls(articlesWithoutUrl)` из id-ја сваког чланка враћа линк ка његовој слици. Ово је оволико једноставно јер свака слика има исти назив као id њеног чланка. То је обезбеђено у функцији која је задужена за качење слика на сервер са којом ћемо се сусрести касније.

4.3 Login и Authenticated странице

```
import { Box, Input, FormControl, FormLabel, Button } from '@chakra-ui/react';
import { useForm } from 'react-hook-form'
import firebase from '../firebase'
import firebaseAdmin from '../firebaseAdmin'
import nookies from 'nookies'
import Head from 'next/head'

export async function getServerSideProps(ctx) {
    try {
        const cookies = nookies.get(ctx);
        const token = await firebaseAdmin.auth().verifyIdToken(cookies.token);
        return {
            redirect: {
                permanent: false,
                destination: '/authenticated',
            },
            props: {}
        }
    }
    catch(err) {
        return {
            props: {},
        };
    }
}

export default function Login() {
    const { register, handleSubmit } = useForm()

    async function onSubmit(data) {
        await firebase.auth().signInWithEmailAndPassword(data.email, data.password);
        window.location.href='/authenticated';
    }

    return (
        <>
            <Head>
                <title>{'МГ ТВ: Улоруј се'}</title>
            </Head>
            <Box h='100vh' bg="linear-gradient(#AE72CE, #9C56C2);" display='grid' placeItems='center'>

                <Box w='40%' minW='300px'>
                    <form onSubmit={handleSubmit(onSubmit)}>
                        <FormControl mb='16px'>
                            <FormLabel htmlFor='email' color='white'>e-mail</FormLabel>
                            <Input type='email' {...register("email")}></Input>
                        </FormControl>
                        <FormControl mb='16px'>
                            <FormLabel htmlFor='password' color='white'>password</FormLabel>
                            <Input type='password' {...register("password")}></Input>
                        </FormControl>
                        <Button type="submit">Log in</Button>
                    </form>
                </Box>

            </Box>
        </>
    );
}
```

Слика 4.8: Изворни код странице за логовање

За разлику од `getStaticProps()` функције, `getServerSideProps()` се позива сваки пут када корисник затражи да види страницу. На Login страници се тада проверавају колачићи у којима су информације о кориснику (да ли је улогован). У случају да јесте, преусмеравамо га на страницу Authenticated. У супротном, корисник мора прво да се улогује.

```
import React from 'react';
import nookies from 'nookies';
import { useRouter } from 'next/router'

import { Box, Text, Button } from '@chakra-ui/react'


import firebase from '../firebase'
import Link from 'next/link';
import { checkToken } from '../auth/checkToken';

import Head from 'next/head'

export async function getServerSideProps(ctx) {

    const cookies = nookies.get(ctx);
    return await checkToken(cookies.token, '/login')

}

export default function Authenticated({ message }) {

    const router = useRouter();

    return (
        <>
            <Head>
                <title>{'МГ ТВ: ' + message}</title>
            </Head>
            <Box h='100vh' display='grid' placeItems='center'>
                <Box display='flex' alignItems='center' flexDirection='column'>
                    <Text mb='16px'>{ message }</Text>
                    <Button
                        onClick={async () => {
                        await firebase
                            .auth()
                            .signOut()
                            .then(() => {
                            router.push("/");
                            });
                        }}
                        mb='16px'
                    >
                        Излогуј се
                    </Button>
                    <Link href='/control-panel'>
                        <Button>
                            Контролна табла
                        </Button>
                    </Link>
                </Box>
            </Box>
        </>
    );
}
```

Слика 4.9: Изворни код странице која потврђује аутентикацију

Authenticated страница је веома слична. У случају да је корисник улогован, добија приступ контролној табли, а у супротном бива преусмерен на Login страницу.

4.4 Страница Контролне табле

```
...

export async function getServerSideProps(ctx) {

    const cookies = nookies.get(ctx);
    let res = await checkToken(cookies.token, '/login')

    const articlesWithoutUrl = await getArticlesData()

    const articles = await getImageUrls(articlesWithoutUrl)

    return {
        ...res,
        props: {
            articles
        }
    }
}

...
```

Слика 4.10: Функција `getServerSideProps()` контролне табле

`getServerSideProps()` овде проверава да ли је корисник улогован и прибавља податке о чланцима.

```
...

async function onSubmit (data) {

    const res = await addArticle(data.title, data.content, imgSize)
    if(image) {
        const res1 = await uploadImage(image, res.id)
    }
    router.replace(router.asPath)
}

async function onImageChange (e) {
    const reader = new FileReader();
    let file = e.target.files[0];

    const u = URL.createObjectURL(file)
    const img = new Image();

    img.onload = () => {
        setImgSize({w: img.width, h: img.height})
    }

    img.src = u

    if(file) {
        reader.onload = () => {
            if (reader.readyState === 2) {
                setImage(file);
            }
        };
        reader.readAsDataURL(e.target.files[0]);
        }
    else {
        setImage(null);
    }
}

...

<form style={{width: '100%'}} onSubmit={handleSubmit(onSubmit)}>
  <Box w='95%' m='16px' display='flex' flexDir='column' alignItems='end'>
      <Input borderColor='black' focusBorderColor='#9C56C2' placeholder='Наслов' variant='flushed' mb='8px'
{...register('title')}></Input>
      <Textarea borderColor='black' focusBorderColor='#9C56C2' mb='16px' {...register('content')}></Textarea>
      <Box display='flex' w='40%' justifyContent='space-between'>
          <Button w='48%' variant='outline' onClick={()=>{setShowCreatePost(0)}}>Откажи</Button>
          <Button w='48%' type='submit'>Објави</Button>
      </Box>
      <Input type='file' border='none' _focus="{{outline:none}}" accept="image/x-png,image/jpeg" onChange={(e) =>
{onImageChange(e);}}></Input>
  </Box>
</form>

...
```

Слика 4.11: Део кода контролне табле задужен за додавање нових чланака

Овај део странице Контролне табле је задужен за додавање нових чланака. Функција `onImageChange(e` се позива сваки пут када корисник окачи слику. Слика се затим чита, памте јој се димензије и референца за касније.

Када корисник притисне дугме **Објави**, позива се функција `onSubmit(data)`. Из ње се позивају функције `addArticle(data.title, data.content, imgSize)` и `uploadImage(image, res.id)` које су задужене за убацивање података у базу.

```
export async function addArticle(title, content, imgSize) {
    let id

    try {
        const res = await firebase.firestore().collection('articles').add({
            title: title,
            content: content,
            createdAt: firebase.firestore.FieldValue.serverTimestamp(),
            imgHeight: imgSize.h,
            imgWidth: imgSize.w
        }).then((docRef) => {
            id = docRef.id
        })
        return {
            code: 'SUCCESS',
            id: id
        }
    }
    catch(err) {
        return {
            code: 'ERROR'
        }
    }

}
```

Слика 4.12: Функција која додаје информације о новом чланку у базу

```
export async function uploadImage(image, id) {

    if (image) {
        const storageRef = firebase.storage().ref();
        const imageRef = storageRef.child(id);
        await imageRef.put(image)
        .then(() => {
            return {message: "Image uploaded successfully to Firebase."};
        });
    } else {
        return {message: "Please upload an image first."};
    }
}
```

Слика 4.13: Функција која качи слику чланка у складиште слика.

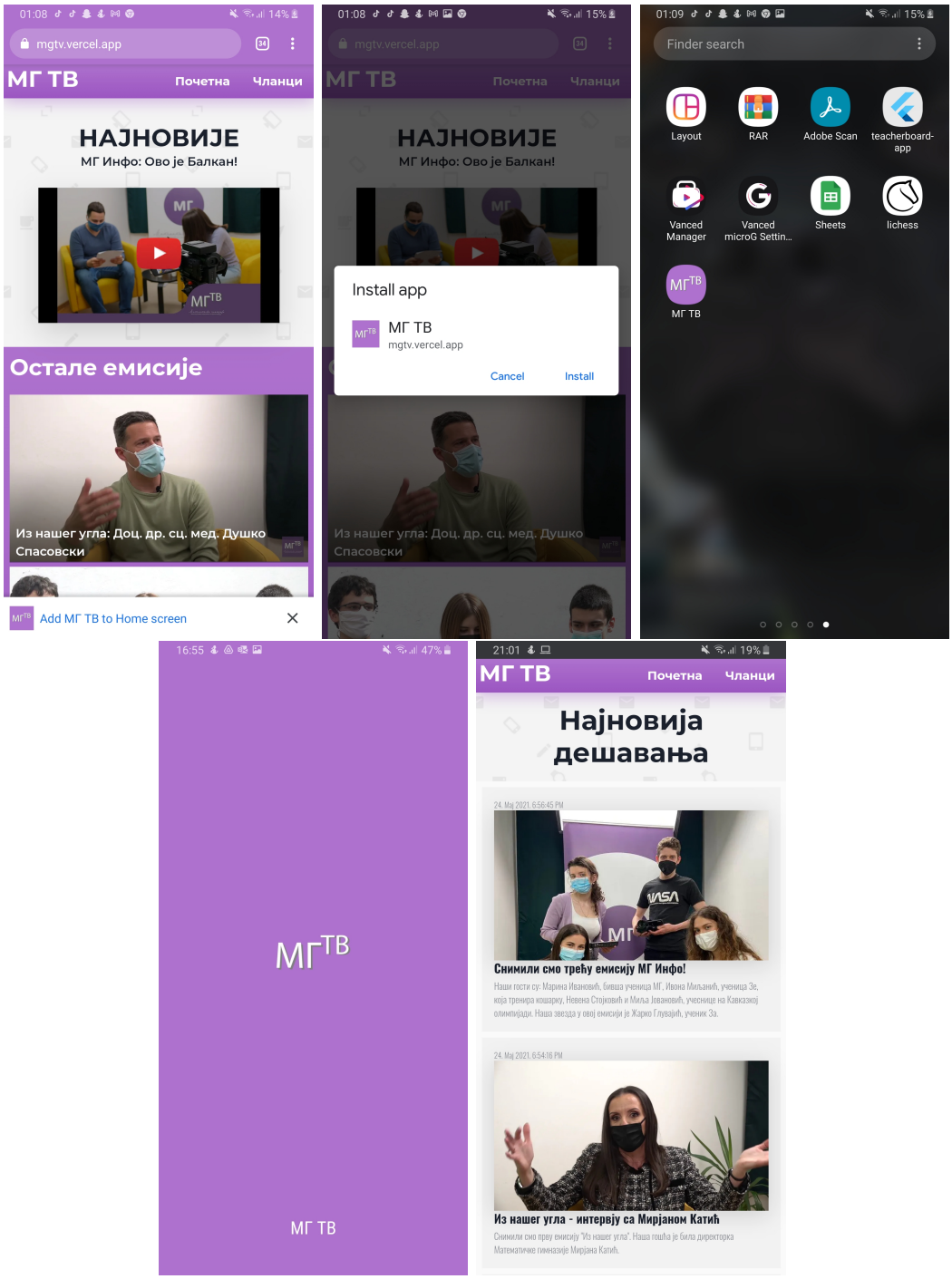
4.5 manifest.json

Да бисмо могли да инсталирамо апликацију из Google Chrome веб претразивача потребан нам је `manifest.json`.

```
{
    "short_name": "МГ ТВ",
    "name": "МГ ТВ",
    "description": "Сајт телевизије Математичке гимназије",
    "icons": [
      ...
      {
        "src": "images/icons/icon-144x144.png",
        "sizes": "144x144",
        "type": "image/png"
      },
      {
        "src": "images/icons/icon-152x152.png",
        "sizes": "152x152",
        "type": "image/png"
      },
      {
        "src": "images/icons/icon-192x192.png",
        "sizes": "192x192",
        "type": "image/png"
      },
      ...
    ],
    "dir": "ltr",
    "lang": "sr",
    "start_url": "../articles",
    "orientation": "portrait",
    "display": "standalone",
    "theme_color": "#AE72CE",
    "background_color": "#AE72CE"
  }
```

Слика 4.14: `manifest.json`

4.6 Инсталација



Слика 4.15: Процес инсталације

5

Закључак

Циљ овог рада био је да покаже како у данашње време, са разним доступним технологијама, постаје све једноставније направити комплексну веб апликацију. Уз све предности које поседује прогресивна веб апликација, она у блиској будућности може постати озбиљан ривал традиционалним апликацијама. Ова технологија је релативно нова, али се веома брзо развија и у њој се види потенцијал.

Апликација приказана у раду је отвореног кода (open-source) и код се може погледати на GitHub-у[1]. Свако жељан да унапреди сајт је добродошао да допринесе пројекту.

Хтео бих да се захвалим свом ментору, Петру Радовановићу, што ми је помгао да напишем овај матурски рад, као и што ме је мотивисао да проширујем своје знање из програмирања током ранијих година када ми је предавао.

[1] https://github.com/ognjenMarinkovic27/mgtv

Литература

[1] https://web.dev/progressive-web-apps/

[2] Andreas Biørn-Hansen, Tim A. Majchrzakand, Tor-Morten Grønli *Progressive Web Apps: The Possible Web-native Unifier for Mobile Development* https://www.scitepress.org/Papers/2017/63537/63537.pdf

[3] Mark Pedersen *Progressive Web Apps: Bridging the Gap Between Web and Mobile* https://www.sitepoint.com/progressive-web-apps-bridging-the-gap-between-web-and-mobile/

[4] https://reactjs.org/

[5] https://chakra-ui.com/

[6] https://firebase.google.com/docs