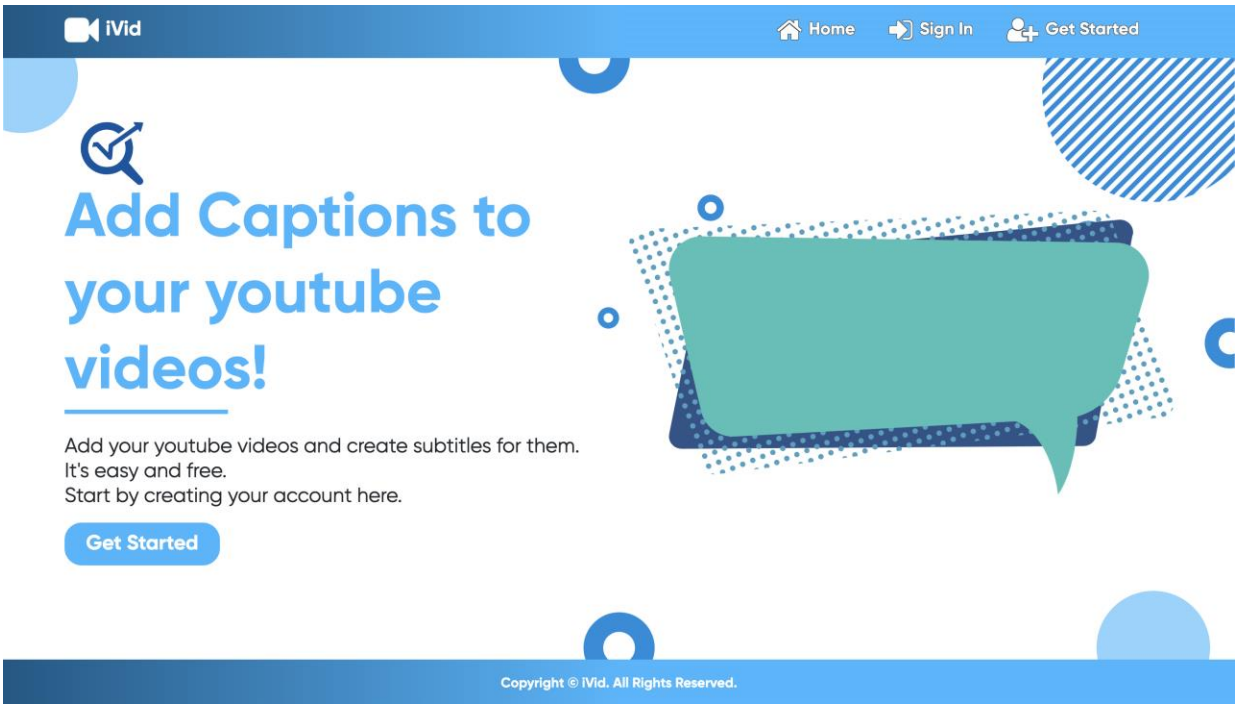
IVid- Проектна задача по предметот Напреден Веб Дизајн



Факултет за информатички науки
и компјутерско инжинерство

Изработиле

Јорданчо Ефтимов (191156)
Јетон Рамадани (191017)

Ментор

д-р Бобан Јоксимоски

Февруари, 2022

Содржина

Вовед

IVid претставува апликација која има за цел да им овозможи на корисниците на истата брзо, лесно и ефикасно да додаваат видео анотации, односно превод на видеата. Изработена е како студентски проект по предметот Напреден Веб Дизајн, од страна на Јорданчо Ефтимов и Јетон Рамадани под менторство на проф. д-р Бобан Јоксимоски.

Во наредните неколку страници, целосно се опишува начинот на развој на апликацијата, како и кратко упатство за користење на истата. Дополнително, наведени се референци кон технологии, рамки и пакети кои помогнаа во нејзиниот развој.

Технологии за развој на iVid

Како што напоменавме и во воведот, целта на овој извештај е да се долови целиот процес на развој на веб апликацијата.

Како главни технологии кои се користат за развојот на iVid се.

1. Laravel (PHP Framework)
2. Vue.js (Progressive Javascript Framework)
3. Inertia.js
4. Bootstrap

Дополнително, во развојот на веб апликацијата, го користевме и Font Awesome v4.7 што претставува библиотека за иконки (SVGs).

Еден од главните пакети кои помогнаа за добивање на информации за видетата со помош на нивното youtube ID е vue-youtube, што претставува компонента која е над Youtube iFrame Player API-то.

Почеток на развој на апликацијата

За иницијализација на проектот најпрво е започнато со иницијализирање на git репозиториум на кој што целиот наш процес на развивање на апликацијата е следен и соодветно push-нат. Начинот на кој е иницијализиран git репозиториумот можете да го погледнете тука.

Откако е завршено со иницијализација на git репозиториумот, креиран е Laravel проект со помош на CLI. Начинот на креирањето на Laravel проект можете да го погледнете тука.

Laravel претставува бесплатен, open source PHP framework кој што нам како развивачи на софтвер ни ја олеснува работата за развивање на веб апликации, и главно во овој проект е користен за backend делот на апликацијата.

За frontend делот на апликацијата се користи Vue.js, меѓутоа, наместо да користиме класичен hard-split на проектите помеѓу frontend(Vue.js) и backend(Laravel), ние ги комбинираме во еден проект со помош на Inertia.js. Inertia.js, всушност ни овозможува развивање на Single Page Applications, меѓутоа немаме рутирање на клиентска страна, ниту пак имаме потреба од API. Начинот на креирање на проект со помош на Inertia.js е објаснет подолу.

При креирање на проект со Inertia.js , најпрво се избираат client-side и server-side frameworks. Inertia.js нуди повеќе можности, меѓутоа за овој проект, како што и веќе неколку пати напоменавме, ги користиме Laravel (server-side) и Vue.js 2 (client-side).

На почеток, прво го иницијализираме server-side делот од апликацијата.

Подесување на серверската страна на Inertia.js

Прво, ги инсталираме серверските адаптери на Inertia.js. Тоа го правиме со помош на следната команда во терминал:

```
composer require inertiajs/inertia-laravel
```

Следно, треба да го подесеме главниот темплејт кој што ќе биде вчитан при првата посета на веб апликацијата. Изгледот на главниот темплејт треба да изгледа вака:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.
    <link href="{{ mix('/css/app.css') }}" rel="stylesheet" />
    <script src="{{ mix('/js/app.js') }}" defer></script>
    @inertiaHead
  </head>
  <body>
    @inertia
  </body>
</html>
```

Откако ќе го подесеме главниот темплејт, следно нешто што треба да направиме е да го подесиме Inertia Middleware-от. Тоа го правиме со следната команда во терминал:

```
php artisan inertia:middleware
```

Штом се генерира HandleInertiaRequests, треба да го регистрираме во App\Http\Kernel како последен елемент во web групата. Со овој чекор завршивме со подесување на серверската страна на Inertia.js.

Подесување на клиентската страна на Inertia.js

За подесување на клиентската страна на Inertia.js, може да користиме или npm или yarn. За потребите на оваа веб апликација ние го користивме npm.

Првиот чекор за подесување на клиенската страна на Inertia.js е инсталирање на клиентските адаптери на Inertia.js. Тоа го правиме со следната команда во терминал:

```
npm install @inertiajs/inertia @inertiajs/inertia-vue
```

Следно, треба да го додадеме следниот код во нашиот главен Javascript фајл (најчесто именуван како app.js) кој што код, има за цел да ја стартува нашата Inertia.js апликјација.

```
import Vue from 'vue'
import { createInertiaApp } from '@inertiajs/inertia-vue'

createInertiaApp({
  resolve: name => require(`./Pages/${name}`),
  setup({ el, App, props, plugin }) {
    Vue.use(plugin)

    new Vue({
      render: h => h(App, props),
    }).$mount(el)
  },
})
```

За крај, како опционален чекор кој што го направивме е тоа што го инсталиравме прогрес индикаторот на Inertia.js. Командата за тоа:

```
npm install @inertiajs/progress
```

Откако се инсталираше, го инизијализиравме во нашиот главен Javascript фајл (најчесто именуван како app.js).

```
import { InertiaProgress } from '@inertiajs/progress'

InertiaProgress.init()
```

Целосното упатство за иницијализирање на Inertia.js како и сите дополнителни функционалности кои што ги нуди, можете да ги погледнете тука.

Дизајн на веб апликацијата

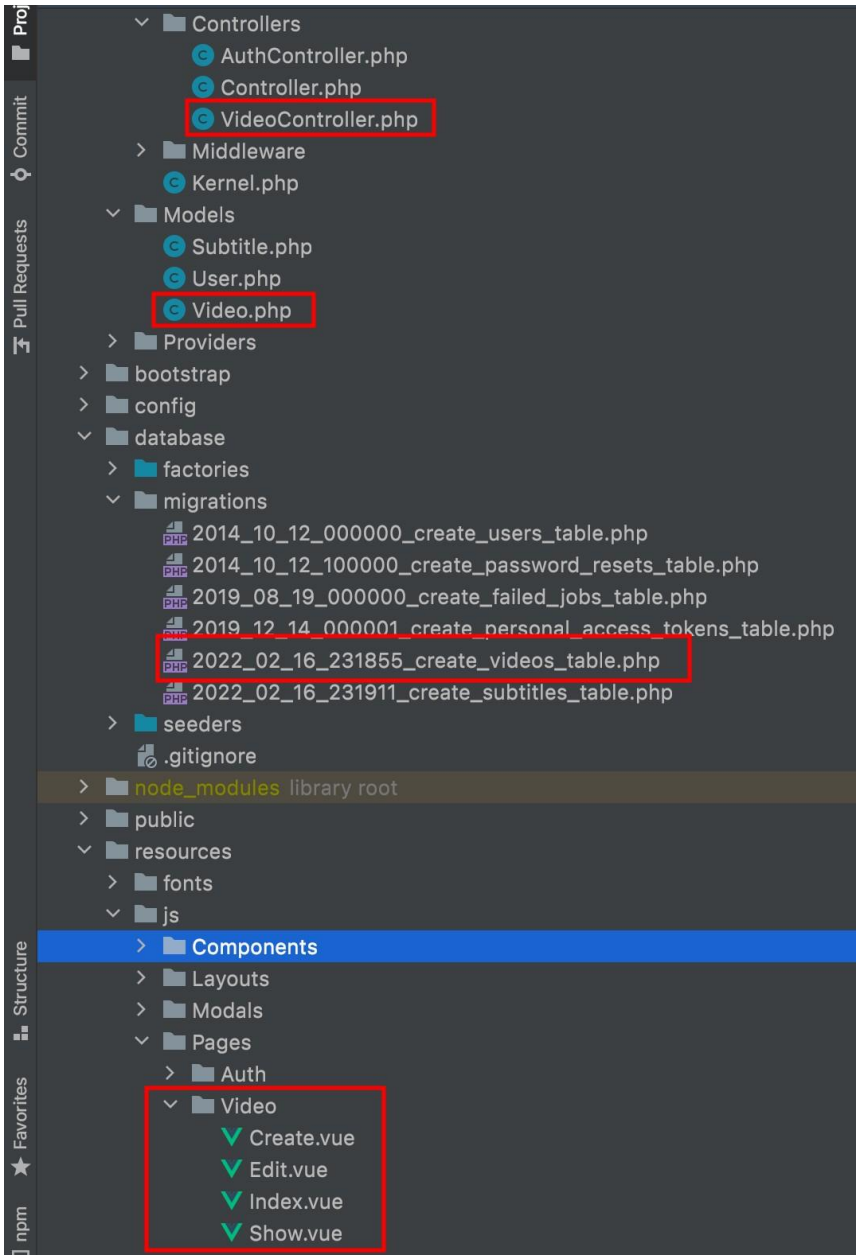
Пред да започнеме со анализа на функционалностите и имплементацијата на истите, најпрво ќе го проанализираме начинот на дизајнирање на самата веб апликација.

Како што напоменавме и погоре, главниот начин со кој што го изработувавме дизајнот на нашата веб апликација е со користење на Bootstrap 5 и дефинирање на наш SCSS. Конкретно, се трудевме да ги користиме класите на Bootstrap 5 за се она каде што ќе можевме, додека за неопходни работи пишувавме наш SCSS. Дополнително, со цел поконзистентен начин на дизајнирање на веб апликацијата, а и подобро дефинирање на темата на нашата апликација, ги менувавме варијаблите(променливите) на Bootstrap 5. Исто така, како технологија која ни помогна во дизајнирање на веб апликацијата го користивме и Font Awesome v4.7 од која што ги користивме иконките кои што се прикажуваат како дел од апликацијата.

Шаблон користен за развивање на апликацијата

Како шаблон користен за развивање на нашата веб апликација е MVC (Model – View – Controller). Inertia.js ни овозможува со помош на контролерот, податоците од базата да ги пратиме на Vue компонентата и да ги прикажеме на крајниот корисник. Начинот на кој што ние ја развиваме апликацијата е така што за секој ентитет кој што треба да се чува во базата, креиравме модел и миграција, а по потреба за дел од моделите креиравме и контролери и погледи односно Vue.js компоненти. Со цел подобро да го разбереме начинот на кој што го имаме градено нашиот софтвер, ќе ја прикажеме конструкцијата на фајловите и именувањето по точно дефинирани правила.

Нашата веб апликација се базира на креирање на преводи за јутјуб видеа, па главните ентитети кои се среќаваат во апликацијата се корисник (User), видео (Video) и превод (Subtitle). Во нашиов случај, а според шаблонот кој што се водиме, ние за овие 3 ентитети треба да имаме модел и миграција, кои ќе бидат именувани соодветно. Моделите би ги имале имињата: User, Video, Subtitle. Миграциите би ги имале имињата: create_users_table, create_videos_table, create_subtitles_table и притоа нивниот редослед на креирање на табели мора да биде во редоследот наведен погоре. Како што кажавме и претходно, по потреба можеме да имаме контролери и погледи (Vue.js компоненти) кои што погледи би се наоѓале во соодветен фолдер именуван според името на моделот, додека самите погледи односно компоненти би биле именувани според името на функцијата која што ги рендерира истите. Конкретно, за моделот Video, ние имаме соодветен контролер кој што се грижи за имлементацијата на основните CRUD методи за видеа и преводи, и тој е именуван соодветно според името на моделот (VideoController). Во прилог имаме слика од тоа како изгледа именувањето и структурата на фајловите и фолдерите кои што се грижат за се она што е потребно за имплементацијата на сите функционалности на ентитетот Video.



Функционален дел на апликацијата

Како што и порано напоменавме, целта на апликацијата е креирање и додавање на превод на јутјуб видеа. За таа цел, за целата логика и начин на додавање на преводи и видеа во базата на податоци се грижи VideoController-от. Во прилог е кодот за основните CRUD операции за видеа и преводи.

```php
class VideoController extends Controller
{
    // функција која ги враќа сите видеа на одреден корисник, по 12 на страница
    public function index(Request $request): Response
    {
        $videos = $request->user()->videos()->paginate(12);
        return Inertia::render('Video/Index', compact('videos'));
    }

    // функција која го враќа погледот за креирање на видео заедно со преводи
    public function create(): Response
    {
        return Inertia::render('Video/Create');
    }

    // функција која се грижи за валидација и соодветно креирање на видео и преводи за тоа видео
    // во база
    public function store(Request $request): RedirectResponse
    {
        // валидација на барањето
        $validated = $request->validate([
            'url' => 'required',
            'title' => 'required|min:3|max:100',
            'subtitles' => 'required|array',
            'subtitles.*.from' => 'required|numeric|min:0',
            'subtitles.*.to' => 'required|numeric|min:0',
            'subtitles.*.description' => 'required'
        ]);

        DB::transaction(function () use ($validated) {
            $video = new Video($validated);
            $video->user()->associate(auth()->user());
            $video->save();
            $video->subtitles()->createMany($validated['subtitles']);
        });

        return redirect()->route('videos.index');
    }

    // функција која што го враќа погледот за промена на преводите на соодветно видео, заедно со
    // претходно зачуваните преводи за тоа видео
    public function edit(Video $video): Response
```

```
    {
        $video->loadMissing('subtitles');
        return Inertia::render('Video/Edit', compact('video'));
    }

    // функција која што го ажурира соодветното видео со вредностите кои што се во испратеното
    // барање
    public function update(Video $video, Request $request): RedirectResponse
    {
        $validated = $request->validate([
            'title' => 'required|min:3|max:100',
            'subtitles' => 'required|array',
            'subtitles.*.from' => 'required|numeric|min:0',
            'subtitles.*.to' => 'required|numeric|min:0',
            'subtitles.*.description' => 'required'
        ]);
        $video->fill($validated);
        $video->subtitles()->delete();
        $video->subtitles()->createMany($validated['subtitles']);
        $video->save();
        return redirect()->route('videos.index');
    }

    // функција која што го враќа соодветнодо видео заедно со сите негови преводи
    public function show(Video $video): Response
    {
        $video->loadMissing('subtitles');
        return Inertia::render('Video/Show', compact('video'));
    }

    // функција која што го брише соодветното видео заедно со сите негови преводи
    public function destroy(Video $video): RedirectResponse
    {
        $video->delete();
        return redirect()->back();
    }
}
```

Од друга страна, целата манипулација со јутјуб видеата (моментално време во кое што се наоѓа видеото додека е пуштено, вкупно време на видеото, thumbnail-от на видеоти и слично) се случува на frontend апликацијата. Како што и погоре напоменавме, за делот со

манипулација со јутјуб видеото користиме соодветен пакет, (компонента) која што ни овозможува работа со сите потребни методи кои ни ги враќаат овие информации. Пакетот е достапен на овој [линк.]

Кодот кој што се грижи за логиката за додавање на видео и преводи на соодветното видео се наоѓа во прилог:

```
export default {
  name: "Create",
  layout: DefaultLayout,
  data() {
    return {
      time: 0,
      subtitleToShow: null,
      form: this.$inertia.form({
        url: '',
        title: '',
        subtitles: []
      }),
      from: {
        value: '',
        error: null
      },
      to: {
        value: '',
        error: null
      },
      description: {
        value: '',
        error: null
      }
    }
  },
  methods: {

    // функција која што додава превод во низата од преводи само доколку преводот има
    // валидно време од и до
    addNewSubtitle() {
      this.from.error = null
      this.to.error = null
      this.description.error = null
      if (!this.validateSubtitle(this.from.value, this.to.value)) {
        this.to.error = "You can't add subtitles that are overlapping."
        this.from.error = "You can't add subtitles that are overlapping."
```

```
            return
        }
        if (parseInt(this.from.value) >= parseInt(this.to.value)) {
            this.from.error = "The from value cannot be bigger than or equal to the 'to' value."
            return;
        }
        if (this.from.value === '') this.from.error = 'Please insert a value for this field.'
        if (this.to.value === '') this.to.error = 'Please insert a value for this field.'
        if (this.description.value === '') this.description.error = 'Please insert a value for this field.'
        if (parseInt(this.to.value) < 0) this.to.error = 'Please insert a value greater than 0.'
        if (parseInt(this.from.value) < 0) this.from.error = 'Please insert a value greater than 0.'
        if (this.to.error !== null || this.from.error !== null || this.description.error !== null) return
        this.form.subtitles.push({
            from: this.from.value,
            to: this.to.value,
            description: this.description.value
        })
        this.from.value = ''
        this.to.value = ''
        this.description.value = ''
    },

    // функција која што го валидира преводот да не се преклопува со веќе постоечки превод
    validateSubtitle(from, to) {
        for (const subtitle of this.form.subtitles) {
            if ((parseInt(from) > parseInt(subtitle.from) && parseInt(from) < parseInt(subtitle.to))
                || (parseInt(to) > parseInt(subtitle.from) && parseInt(to) < parseInt(subtitle.to))
                || (parseInt(from) <= parseInt(subtitle.from) && parseInt(to) >= parseInt(subtitle.to))) {
                return false;
            }
        }
        return true;
    },

    // функција која што го брише преводот од низата на преводи
    removeSubtitle(id) {
        this.form.subtitles.splice(id, 1);
    },

    // функција која што го валидира јутјуб линкот од видеото
    validateYoutubeURL(url) {
        const p =
/^(?:https?:\/\/)?(?:m\.|www\.)?(?:youtu\.be\/|youtube\.com\/(?:embed\/|v\/|watch\?v=|watch\?.+
&v=))((\w|-){11})(?:\S+)?$/;
        if (url.match(p)) {
            return url.match(p)[1];
```

```
        }
        return false;
    },
    async playing() {
        this.processId = setInterval(() => {
            this.player.getCurrentTime().then((time) => {
                this.time = time;
            });
        }, 100);
    },
    pauseVideo() {
        this.player.pauseVideo();
    },
    paused() {
        clearInterval(this.processId);
    },
    submit() {
        this.form.post(this.$route('videos.store'));
    },
},
computed: {
    videoId() {
        if (!this.form.url) return null;
        if (!this.validateYoutubeURL(this.form.url) || this.form.url.indexOf('v=') === -1) return null;
        let video_id = this.form.url.split('v=')[1];
        let ampersandPosition = video_id.indexOf('&');
        if (ampersandPosition !== -1) {
            video_id = video_id.substring(0, ampersandPosition);
        }
        return video_id
    },
    player() {
        return this.$refs.youtube.player;
    }
},
watch: {

    // функција која што постојано го гледа времето на видеото и ја повикува функцијата секогаш
    // кога ќе се смени времето на видеото
    time: function () {
        for (let i = 0; i < this.form.subtitles.length; i++) {
            if (this.form.subtitles[i].from < this.time && this.form.subtitles[i].to > this.time) {
                this.subtitleToShow = this.form.subtitles[i].description;
                break;
            } else {
```

```
            this.subtitleToShow = null
        }
    }
  }
 }
}
```

Начин на користење на апликацијата

За крај да го објасниме целиот процес на користење на веб апликацијата.

Веб апликацијата iVid е достапна на следниов линк. За да може одреден корисник да ги користи сите функционалности на нашата апликација, мора да е соодветно регистриран како корисник на истата. За регистрација потребни се следниве податоци: име и презиме, емаил и пасворд. Откако корисникот ќе се регистрира на нашата веб апликација, може слободно и бесплатно да ги користи услугите кои што ги нудиме. Откако регистрацијата ќе биде успешна, корисникот може да го користи својот профил да се најавува кога сака со тоа што има пристап само до своите видеа кои ги има креирано. По најавата на корисникот, тој може да одбере да креира видео со видео анотациите кои што сака да ги додаде (преводи) или да избере да ги листа своите видеа. Доколку одбере да креира видео, потребните информации за креирање се јутјуб линкот до видеото, и соодветно да се внесе наслов на видеото. Откако корисникот ќе внесе валиден јутјуб линк, му се овозможува да внесува видео анотации (преводи) со тоа што тие ќе бидат валидни видео анотиации (преводи). Под валидни видео анотации, се мисли на преводи кои што немаат негативно време на почеток, и немаат време кое што се преклопува едно со друго. Откако ќе ги внесе сите преводи за соодветното видео, корисникот може да избере да го зачува видеото. Доколку сите валидации поминат, и на frontend и на backend, видеото соодветно ќе се креира заедно со сите видео анотации. По креирање на видеото, корисникот може да избере да ги листа видеата. При листање на видеата, за секое свео видео, корисникот има можност да го едитира, гледа и избрише видеото. При едитирање на видеото, корисникот може да го менува насловот и преводите на видеото, додека јутјуб линкот на видеото нема пермисија да го менува. Ако избере да го гледа видеото, корисникот го гледа заедно со сите преводи кои што ги има додадено за него. На крај, доколку избере да го избрише видеото, при избор на опцијата се отвора поп ап прозорец кој што бара од корисникот конфирмација за тоа дали навистина е сигурен за бришењето на видеото. Доколку го потврди изборот, видеото се брише заедно со сите негови креирани преводи.

Референци

1. https://laravel.com/
2. https://vuejs.org/
3. https://getbootstrap.com/
4. https://fontawesome.com/v4.7/icons/
5. https://inertiajs.com/
6. https://github.com/anteriovieira/vue-youtube
7. https://dev.to/rabeeaali/how-to-deploy-inertia-js-to-heroku-2cdn