

TASK MANAGER - VUE 3



Кире Петковски 185045

ФЕВРУАРИ 2023

Вовед	2
Технологии	2
Router	3
Navigation	4
Home page	4
Firebase	5
Firebase - Authentication и Log In Page	6
Task page	6
Task component	6
Task page – Not Taken component	8
Task page – My tasks component	8
Task page – Finished component	9
Task page – In Progress component	10
Modal component - Delete	10
Modal component - Edit	10
Modal component - Comment	11
Користена литература	11

Системот за менаџирање на задачи е процес на организирање и преглед на задачите кои треба да се извршат или веќе се завршени. Добрата организација на задачите придонесува за лесно и ефикасно управување на задачите во системот. Првиот чекор во ефикасното управување со задачи е да се претстави листа од сите задачи кои треба да се завршат. Листата треба да вклучува сè, од мали, дневни задачи до поголеми, долгорочни проекти. Потоа треба да бидат вклучени коментарите или извештај како е завршена задачата. Но и пред се треба да се има контрола над задачите. За секоја задача потребно е да се знае до каде е со извршувањето.

Погледот на Системот за менаџирање на задачи е поделен на администраторски и кориснички поглед:

- Корисникот може ја промени состојбата на таскот и да додаде коментар на таскот.
- Корисникот има опција да земе таск, но може и да му биде доделен од страна на администраторот.
- Администраторот додава задачи во системот и има функции за бришење и ажурирање на тасковите.
- Администраторот може ја промени состојбата на таскот и да додаде коментар на таскот.
- Администраторот има опција да земе таск, но може и сам да си додели таск.
- Администраторот има поглед до сите задачи, додека корисникот има поглед само до тасковите кои му се доделени до него.

ТЕХНОЛОГИИ

Системот за менаџирање на задачи е изграден врз основа на Vue 3. Сите компоненти се креирани и поврзани меѓу себе со идејата за архитектура на Vue. Испраќањето на информации од компонента до нејзината подкомпонента се извршува преку props. А обратниот процес, испраќањето на информации од подкомпонентата на главната компонента е искористен настан кои се праќа преку emit. Системот користи неколку URL патеки. Патеките се дефинирани во посебен JavaScript документ кој е базиран на библиотеката Router.

При изградбата системот додадени се и некои технологии за подобар кориснички графички интерфејс. За дизајнот на табелите и копчињата искористена е дигиталната библиотека Bootstrap 5. Додека за исконите за бришење, додавање, ажурирање, најавен корисник и коментарите искористена е алатката Font Awesome.

Системот е поврзан со Firebase, која е добра платформа за манипулација и презимање на податоци без да се користи некој SQL сервер. Во HTML документот поставени се линкови до библиотеките кои се потребни за негово функционирање. Потребна е и конфигурација со директни линкови до колекциите од каде ги презима информациите.

ROUTER

Поставувањето на рутер е од големо значење. Во завис од патеката на која се наоѓа корисникот, на корисничкиот интерфејс ќе се појави одреден темпјет. Главната карактеристика е во презимањето и генерирањето на податоците кои се потребни за да се прикаже истиот темпјет. Пример, во апликацијата имаме навигација до патеката 'localhost:8080/'. Во позадина е потребно да се генерираат две полиња или низа од насловите на сликите и низа од слики. Потоа во темплејтот се прикажуваа слика со соодветен текст. Додека за патеката 'localhost:8080/task' потребно е да се повикаат одредени библиотеки со цел да се преземат податоци кои се поставени на Firebase. Да се постават во низи и да обработат променливи кои се поврзани со овие податоци, а се неопходни за да се прикажат податоците над корисничкиот интерфејс. Од друга страна ја имаме и патеката за логирање. Која генерира темпјет за форма и притоа чека да се внесат податоци за корисникот.

Податоците се испраќаат како побарување за обработка од страна на Firebase и се чека негов одговор. Односно на апликацијата и се потребни податоци за уникатниот ID за автентикација, корисничкото име и улогата на најавениот корисник.

Корисниците кои не се најавени во системот неможат да пристапат до патеката 'localhost:8080/task'. За таа цел во дефинирањето на патеките, поставен е објект *meta* кој ја содржи променливата *requiresVisitor*. Променливата *requiresVisitor* е *false* за патеката '/task', додека за останатите е *true*. При секоја нова промена на патеката, ќе провери дали корисникот е автентизиран. Доколку не е најавен во системот ќе провери дали пристапува до патека која му е дозволена, инаку ќе го пренасочи кон патеката "localhost:8080/".

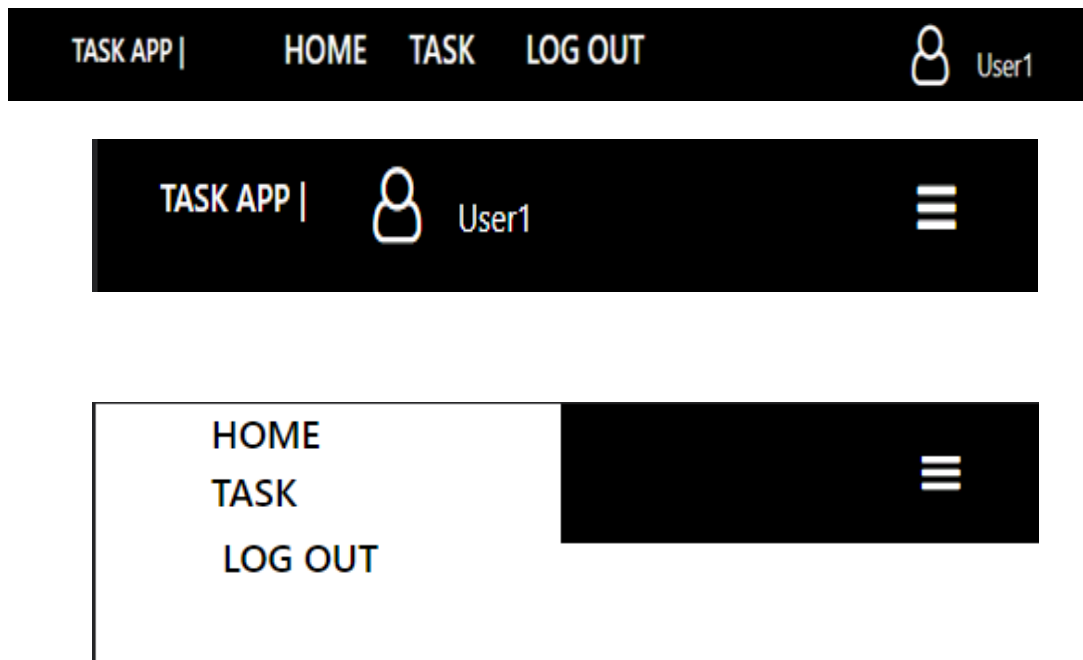
```
const router = createRouter({ options: {
  history: createWebHistory(),
  routes:
    [
      {
        name: 'Home',
        path: '/',
        component: Home,
        meta: {requiresVisitor: true},
      },
      {
        name: 'Task',
        path: '/task',
        component: Task,
        meta: {requiresVisitor: false},
      },
      {
        name: 'LogIn',
        path: '/login',
        component: LogIn,
        meta: {requiresVisitor: true},
      },
    ]
});

router.beforeEach((to, from, next) => {
  const isLoggedIn = window.localStorage.getItem('key: 'username');
  if (isLoggedIn) next()
  else{
    if(to.meta.requiresVisitor) next()
    else next('/')
  }
});
```

слика 1

NAVIGATION

Во компонентата за навигација, поставен е код за проверка на широчината на екран со кој се пристапува до Веб апликацијата. Кодот е поделен во два дела: уреди со широчина на екран помали од 750 и за уреди со ширина на екран поголем од 750. Функцијата за проверката на екранот е поставена во *method* делот. Додека повикувањето за проверка се повикува во *created()* од животниот циклус на компонентата. Односно во *created()* делот поставен *EventListener* со кој се вклучува “resize” (како глобална алатка за пресметување на големината на екранот). А потоа е повикана и самата функција.



Слика 2

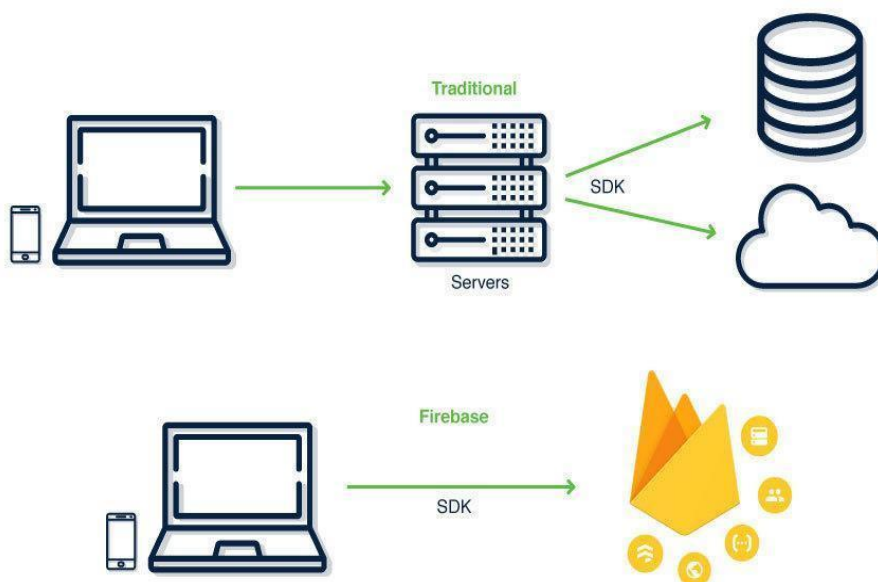
HOME PAGE

Компонентата Home е едноставна компонента која е поделена со *div* таг за слика и *div* таг текст. Промената на сликата и текстот со повикување на *method function* која го зголемува индексо на полињата во кои се додадени сликите и текстот. Функцијата се повикува од *beforeMount()* делот од животниот циклус на компонентата. Целта на оваа компонента е да ја прикаже функционалноста на Router.

Firebase е платформа за развој на мобилни и веб апликации која е креирана од страна на Google во 2014 година. Таа нуди низа на сервиси кои им овозможуваат на програмерите да изградат и да управуваат со апликациите полесно. Firebase вклучува различни карактеристики, како: Real-time Database, Authentication, Cloud Functions, Hosting, Cloud Messaging и Cloud Firestore.

Real-time Database претставува NoSQL база на податоци која е хостирана на облак и им овозможува на програмерите да зачуваат и синхронизираат податоци во реално време на повеќе клиенти. Оваа карактеристика е особено корисна за апликации кои бараат реално временски ажурирања и соработка. Cloud Firestore е најновата база на податоци на Firebase за развој на апликации. Се надоврзува на успехите на базата на податоци во реално време со нов, поинтуитивен модел на податоци. Cloud Firestore, исто така, има побогати, побрзи барања и скали подалеку од Базата на податоци во реално време.

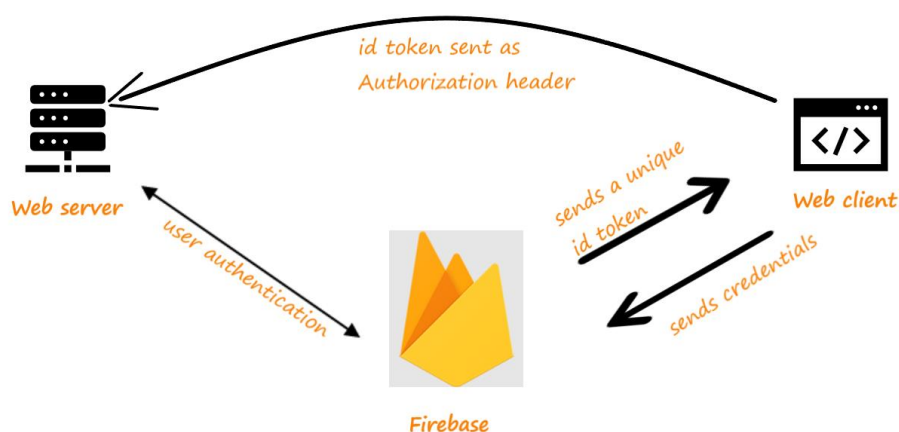
Автентикацискиот сервис на Firebase им овозможува на програмерите лесно да додадат автентикација на корисниците во нивната апликација со неколку методи на најава, како што се е-маил и лозинка, Google, Facebook, и други. Firebase Hosting нуди брз и безбеден веб хостинг сервис за статички и динамички содржини, вклучувајќи HTML, CSS, JavaScript и слики. Исто така, тој вклучува карактеристики како SSL, CDN и поддршка за прилагодено домени.



Слика 3

FIREBASE - AUTHENTICATION И LOG IN PAGE

Firebase има посебна алатка за автентикација, содржи табела од корисници. За потребите на веб апликацијата, автентикацискиот протокол се одвива преку email и password. За секој корисник автоматски е генериран уникатен ID. Креирана е и колекција која ни служи за одредување на корисничкото име и улогата на корисникот. Документите од колекцијата Users имаат ID кое е исто со UID од табелата на автентикација. Со најава на корисникот се презима UID од автентикациската табела и потоа се проверува корисничко име и улога од колекцијата Users.



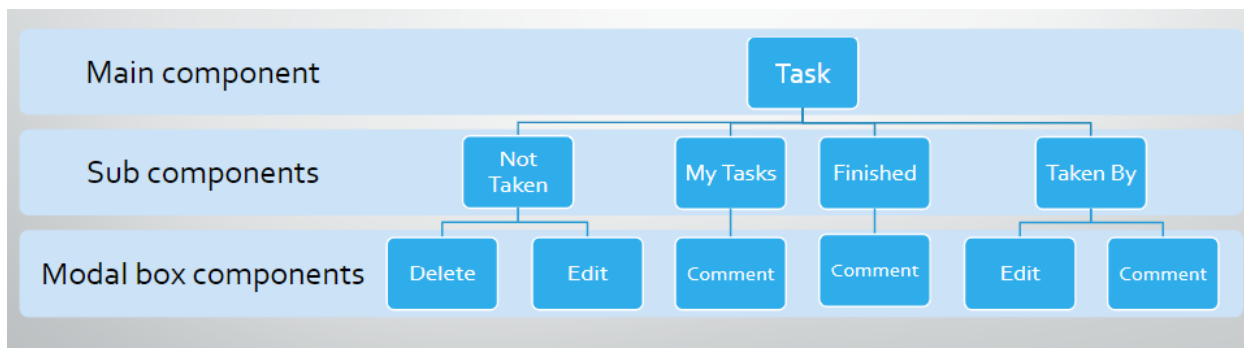
слика 4

TASK PAGE

TASK COMPONENT

Компонентата Task е поставена на патека `/task`. Со пристапување на патеката се презимаат податоците од *Firebase*. Потоа преку *props* ги предава податоците на останатите компоненти. Доколку има некоја промена на таскот, компонентата ќе ги ажурира промените со добиените податоци од *\$emit* настанот.

За пристапувањето на под компонентите креирано е навигација со помош на табови. Табовите овозможуваат полесно контролирање на активната под компонента. И избегнување од дополнително презимање на податоците од *Firebase*. Додека Модалните компоненти се вклучуваат кога ќе биде кликнато на иконите за бришење, ажурирање или коментирање.



слика 5

Во create() е поставен делот со кој компонентата ги презима податоците од Firebase (слика 6). Со креирање на компонентата од Firebase се побаруваат податоците од сите документи од колекциите Tasks и Users. При презимање на податоците се креира објект во кој се пополнува со преземените податоци. Објектот се додава во предходно дефинирана низа.

Add() е метода која од формата го презима објаснувањето за таскот и креира објект кој е потребен за да се прати до Firebase за да се сочува таскот. Функцијата ќе се повика кога администраторот ќе креира таск. Праќањето на податоци до Firebase се одвива со праќање на објект. Firebase ќе го преземе објектот и ќе го зачува како документ во колекцијата на Tasks.

Објектот task содржи низа од коментари, опис на таскот, ID број за идентификација на таск (различен од ID на документот), корисник на кој му припаѓа таскот и статус на таскот.

```

created() {
  db.collection( collectionPath: "Users").get().then((querySnapshot) => {
    querySnapshot.forEach( callback: (doc) => {
      var getUser = {
        'username': doc.data().Username,
      }
      this.Users.push(getUser);
    });
  });

  db.collection( collectionPath: "Tasks").onSnapshot( onNext: (querySnapshot) => {
    querySnapshot.forEach( callback: (doc) => {
      if (!this.tasks.find(({id}) => id === doc.id)) {
        const task = {
          'id': doc.id,
          'task_id': doc.data().ID,
          'description': doc.data().Description,
          'status': doc.data().Status,
          'user': doc.data().User,
          'comments': doc.data().Comment,
        }
        console.log("CREATE for: " + this.user.username + " " + this.user.role);
        this.tasks.push(task);
      }
      if (this.NextIDTask <= doc.data().ID) {
        this.NextIDTask = doc.data().ID;
      }
    });
  });
}

```

слика 6

```

addTask() {
  if (this.newTaskDescription.length > 0) {
    if (this.givenUser !== "") {
      this.status = "In Progress";
    } else {
      this.status = "Not Taken";
    }
  }
  const task = {
    'Comment': [],
    'Description': this.newTaskDescription,
    'ID': this.NextIDTask + 1,
    'User': this.givenUser,
    'Status': this.status,
  }
  db.collection( collectionPath: "Tasks").add(task)
    .then(
      this.tasks = []
    );
  this.newTaskDescription = "";
  this.givenUser = "";
}

```

слика 7

TASK PAGE – NOT TAKEN COMPONENT

Администраторски поглед

[New tasks](#) [My tasks](#) [Finished tasks](#) [Task in Progress](#)

ID	Task	Status	Take task	Delete	Edit
5	Fifth for testing task app	Not Taken	+	🗑	✎

Кориснички поглед

[New tasks](#) [My tasks](#) [Finished tasks](#)

ID	Task	Status	Take task
5	Fifth for testing task app	Not Taken	+


слика 8

Првото отварање на страната Task ќе го активира Not taken табот. Во табеларниот приказ ќе се појават сите задачи со за кои статусот е со вредност Not Taken или Release. Статус Not Taken означува дека задачот е креиран без да му се додели на некој корисник. Додека, статус Release означува дека задачот бил доделен на некој корисник, но корисникот го ослободил задачот за да може друг корисник да го преземе.

Погледот на не земени задачи можат да пристапат сите корисници, но има разлика од администраторски и кориснички поглед. Администраторот може да го земе задачот, да го избрише или да го ажурира описот на задачот. Корисникот има опција на презимање на задача.

TASK PAGE – MY TASKS COMPONENT

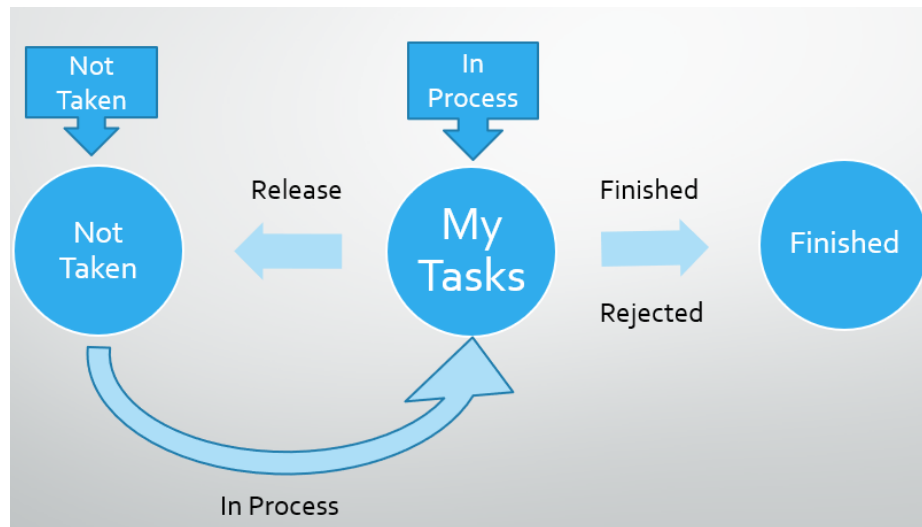
[New tasks](#) [My tasks](#) [Finished tasks](#) [Task in Progress](#)

ID	Task	Status	Comment	Close
4	Nov task	In Progress 	💬	✓
2	Task for Admin	In Progress	💬	✓

Слика 9

Вториот таб My Tasks е наменет за корисниците да ги гледаат само задачите кои им се доделени. Во овој дел корисникот е должен да го промени статусот, да додаде коментар и да го затвори задачот. Статусот може да се промени со кликање врз текстот. Предходно ја објаснивме разликата на два статуси Not taken и Release. Статусот Not taken се генерира со креирањето на новиот задача. Статусот Release е статус кои корисникот го променува при затворање на задачот. Постојат и други статуси кои може да се забележат во овој таб. При клик можат да се одберат статусите:

- In Process: Таскот не е завршен. Не може да се затвори таскот без предходно да се промени статусот.
- Release: Таскот е ослободен за да го преземе некој друг корисник. Таскот се враќа во табот Not Taken.
- Rejected: Таскот е затворен, но не може да се заврши поради одредена пречка.
- Finished: Таскот е затворен и успешно е завршена задачата зададена во описот.



слика 10

TASK PAGE – FINISHED COMPONENT

Администраторски поглед:

[New tasks](#) [My tasks](#) [Finished tasks](#) [Task in Progress](#)

ID	Task	Status	User	Comment
7	Task for User1	Finished	User1	
6	Task for User2	Finished	User2	
2	Task for Admin	Rejected	Admin	

Кориснички поглед:







[New tasks](#) [My tasks](#) [Finished tasks](#)

ID	Task	Status	Comment
7	Task for User1	Finished	

слика 11

Едноставен таб со кој се прикажуваат завршените таскови. Овој дел е поделен во два погледа: администраторски и кориснички поглед. Администраторскиот поглед има преглед кон сите затворени таскови, без разлика за кој корисник станува збор. Корисничкиот поглед има преглед само кон неговите затворени таскови.

TASK PAGE – IN PROGRESS COMPONENT

New tasks My tasks Finished tasks Task in Progress					
ID	Task	Status	User	Comment	Edit
4	Nov task	In Progress	Admin		
3	New task	In Progress	User1		
1	First task	In Progress	User1		

слика 12

Последниот таб *In Progress* може да пристапи само администраторот. На овој таб се прикажуваат сите задачи кои се доделени на некој корисник и сèуште не се затворени. Администраторот може да додаде коментар на задачот и може да го ажурира описот на задачот. Во принцип администраторот има контрола на сите задачи. Во секое време може да го промени описот на задачот или да даде одговор на некој task како коментар. Додавањето на нов коментар на одреден task може да го види и корисникот на кој припаѓа овој task. Бришењето на задачите е овозможено само кога корисникот ќе го ослободи задачот (статус Release) или има статус Not Taken.

MODAL COMPONENT - DELETE

Modal Delete е под компонента за која е дозволен пристап само за администраторот и се наоѓа само во Not Taken табот. Преку props се испраќаат задачите и индекс за точно кој task е побарано да се избрише. Како одговор чека дали е потврдено или не е потврдено бришењето на дадениот task. Доколку е потврдено Not taken компонентата преку emit ќе ја извести главната компонента Task за направена промена на одреден task. Исто така, компонентата ќе испрати барање до Firebase за бришење на документ со ID на задачот од колекцијата Tasks.

MODAL COMPONENT - EDIT

Modal Edit е под компонента за која е дозволен пристап само за администраторот и се наоѓа во Not Taken и In Process табот. Преку props се испраќаат задачите и индекс за точно кој task е побарано да се ажурира. Како одговор чека дали е направена измена или не е направена измената на дадениот task. Доколку е потврдено Not taken (или ако е повикан од In Process) компонентата преку emit ќе ја извести главната компонента Task за направена промена на одреден task. Исто така, компонентата ќе испрати барање до Firebase за ажурирање на документ со ID на задачот од колекцијата Tasks.

MODAL COMPONENT - COMMENT

Modal Comment е под компонентата за која е дозволен пристап за сите корисници и се наоѓа во My tasks, Finished и In Process табот. Преку props се испраќаат тасковите и индекс за точно кој таск е потребно да се прочитаат и да се додаде нов коментар. Како одговор чека дали е притиснато копчето за затварање на модалот. Коментарите за тасковите се презимаат заедно со останатите податоци за тасковите при пристапот на Task компонентата. Во created() се презимаат коментарите кои биле испратени преку props и се додаваат во погледот. Компонентата емитува настан само за затварање на модалот. Доколку се додаде нов коментар се испраќа барање до Firebase за да се додаде коментарот во листата за коментари.

КОРИСТЕНА ЛИТЕРАТУРА

- https://www.youtube.com/playlist?list=PL4cUxeGkcC9itfjle0ji1xOZ2cjRGY_WB
- <https://www.npmjs.com/package/vue3-tabs-component>
- <https://www.youtube.com/watch?v=u2AwJAFeaKc&list=WL&index=2&t=1178s>
- <https://firebase.google.com/docs?authuser=0&hl=en>
- <https://vuejs.org/guide/introduction.html#api-styles>
- <https://fontawesome.com/search?m=free&o=r>
- <https://getbootstrap.com/docs/5.0/getting-started/introduction/>