基于Vue技术的新番观测网站开发

★姓名：胡佳俊　　班级：计算机23C2　　学号：2023031480

姓名：张高远　　班级：计算机23C2　　学号：2023031447

姓名：　　班级：　　学号：

（★为本项目的主要负责同学）

大数据与软件学院

二零二三年十二月

一、方案背景介绍

在当今数码时代，动漫产业正蓬勃发展，每周都有大量新的动画作品上线。随着动漫市场的不断扩大，观众对于及时获取最新作品信息、了解未来季度的新番卫星以及分享个人喜好的需求也日益增加。为了满足这一需求，我们提出了一个全新的项目——"新番观测站"，旨在为用户提供一站式的动漫信息服务。

二、方案设计任务分工

胡佳俊：

组件设计： 负责设计和实现 Vue 组件，确保它们可以在整个应用程序中被重复使用，并保持一致性。包括选项式和组合式编程。

爬虫： 项目中需要爬取数据，负责编写和维护爬虫程序，确保从外部数据源获取所需信息。

功能页面设计： 设计和实现应用程序的各个功能页面，确保它们满足用户需求并与其他组件协同工作。

API 接口对接：对接 API 接口，确保前端和后端之间的数据交互顺畅。

张高远：

部分 js 设计

RSS 订阅： 使用 js 获取 rss 订阅，并解析

原型设计： 使用工具创建应用程序的原型，以便在早期阶段获取用户反馈和进行修改。

页面布局和样式： 确保整个应用程序的页面布局和样式一致，符合设计规范。

用户交互设计： 确保用户与应用程序进行交互的方式是直观和用户友好的，提高用户满意度。

三、需求分析

综合性资料库：

动画资料网站致力于建立一个综合性的资料库，涵盖了各类动画作品的相关信息，包括但不限于制作背景、创作者介绍、角色设定、剧情梗概等。

每周推送：

记录每周的动画更新情况，让动画爱好者可以有规划的看动画

首页：

展示最新的动画咨询和推荐内容。

利用 Vue 的动画效果增强用户体验。

搜索功能：

提供全文搜索功能，支持关键词搜索。

技术选型：

使用 Vue.js 框架搭建前端，利用 Vue 的生态系统提高开发效率。

使用爬虫技术获取最近的咨询，并提取重要信息。

使用 flask 与 express 制作代理服务器以解决跨域问题。

使用多家公开平台 API，使得数据可以动态更新。

多媒体播放器集成：

在项目中，我们计划集成两个主要的多媒体播放器，以提供更全面的用户体验：

Bilibili 播放器集成：

目标： 确保用户可以流畅地查看动画和视频，无缝融入 Bilibili 的播放器。

实现： 使用 Bilibili 提供的 API 和嵌入代码，将 Bilibili 播放器嵌入

到项目的相关页面中。确保用户可以直接在我们的应用程序中观看 Bilibili 上的动画和视频内容。

功能： 用户可以通过我们的应用程序浏览和播放 Bilibili 上的热门动画、番剧和其他视频内容，同时享受 Bilibili 播放器的高质量视频体验。

网易云音乐集成：

目标： 为用户提供与动画相关的音乐，增强用户在应用程序中的娱乐体验。

实现： 使用网易云音乐的 API 和嵌入代码，将网易云音乐的播放器嵌入到项目中。确保用户可以直接在我们的应用程序中收听与动画相关的音乐。

功能：用户可以通过我们的应用程序访问并播放网易云音乐上的动画音乐、原声音轨和相关歌曲。这为用户提供了在浏览动画咨询时同时享受动画音乐的机会。

阿里灵积平台集成：

目标：为用户提供强大的自然语言处理能力，提升应用程序中的语言交互和理解水平。

实现： 在阿里灵积平台注册开发者账号，获取 API 密钥和访问凭证。这通常需要在阿里云开发者控制台完成。详细阅读阿里灵积平台的 API 文档，了解接口的调用方式、参数设置以及响应格式。利用选定的编程语言，通过 HTTP 请求将文本数据发送至阿里灵积平台的 API 节点，并处理返回的自然语言处理结果。根据文档中的说明，构建正确格式的请求，以获取模型的语言处理结果。

功能：用户可以通过我们的应用程序进行自然语言交互，向阿里灵积平台提交文本数据，获得自然语言处理的结果，如语义分析、命名实体识别等。

利用阿里灵积平台的能力，实现智能推荐功能，根据用户的语言输入为其推荐相关内容，提升用户体验。

RSS 与 Atom 订阅：

目标:RSS的主要目标是通过一个单一的阅读器简化用户获取新闻、博客、论坛等网站的最新信息，实时提供即时通知和更新，使用户随时了解他们关注的网站的最新动态。

实现:网站通过提供RSS Feed发布其内容，这是一个包含最新文章或更新的XML文件，其中包括标题、链接和摘要等信息；通过XML分析，用户可以获取RSS订阅的信息功能

目标:订阅和管理： 用户可以添加或删除他们关注的网站，并组织这些订阅以便更容易管理。即时通知： RSS阅读器通过定期检查订阅的Feed，及时提供最新内容的通知，减少用户需要主动访问网站的频率。

版本控制：

在项目的开发过程中，我们采用Git作为版本控制系统，以确保代码的协同开发、版本管理和团队协作的有效性。

原型设计

原型设计是产品设计和开发过程中的一项关键活动，它旨在通过创建一个初步的、可交互的模型，以便设计师、开发人员和利益相关者能够更好地理解和验证产品的概念、功能和用户体验。

四、方案概要设计、详细设计

技术栈

原型：figma

前端：HTML5,CSS3, JavaScript,TypeScript,Vue3,Vue_cli,axios

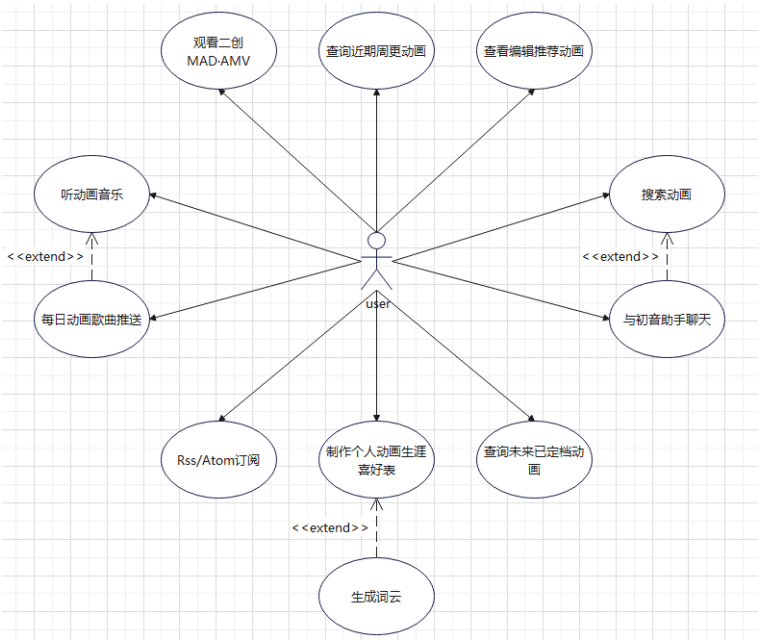
代理服务器：express.js

调用的服务(API):Bilibili,网易云音乐,阿里灵积平台，Bangumi 番组计划

分布式版本控制：Gitee,Github

用例图

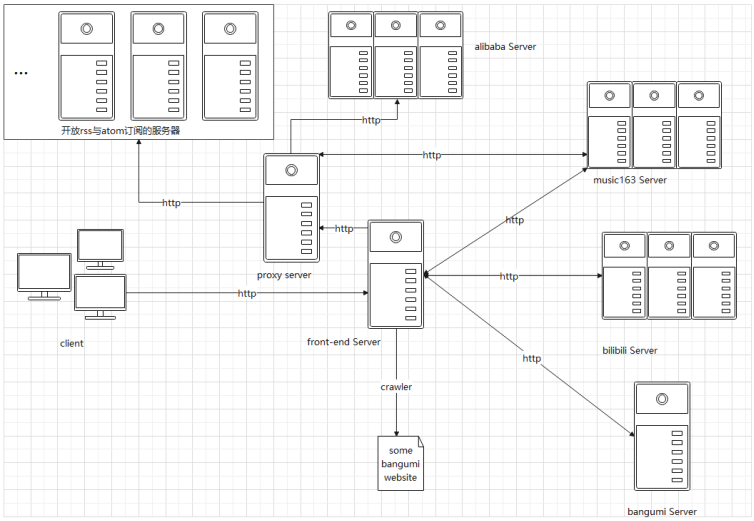
用例图是一种用于描述系统功能的图形化工具，属于统一建模语言（UML）的一部分。用例图主要用于表示系统的功能需求，展示系统与外部实体之间的交互关系。



项目部署图：

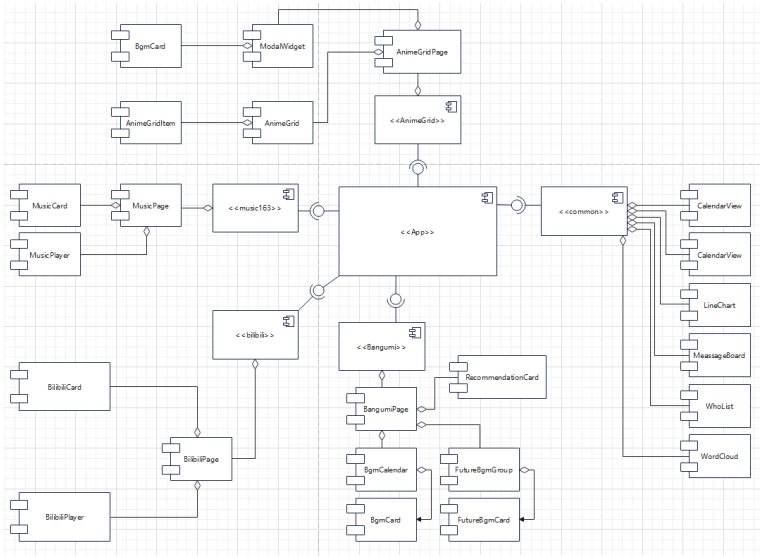
描述前端应用程序的部署结构，包括服务器、CDN、数据库等组件的分布。这幅图显示了一个客户端和服务器之间的网络架构,它展示了数据如何通过HTTP协议在不同服务器和设备之间流动。在此架构中，这个代理服务器作为中间人，

可能对部分内容进行跨域代理，然后将请求转发到前端服务器。前端服务器是处理客户端请求和发送响应的主要接入点。(如下图)



组件图

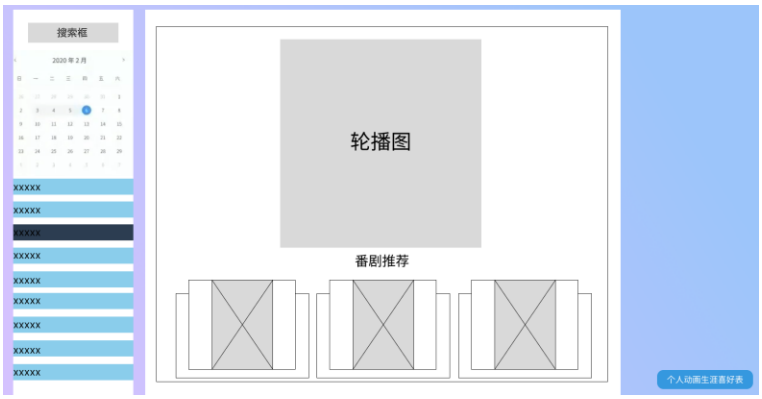
描述系统中各个组件之间的关系和依赖。展示前端应用程序中不同模块或组件之间的依赖关系，以及它们的功能划分。(如下图，组件图)



Figma 原型设计

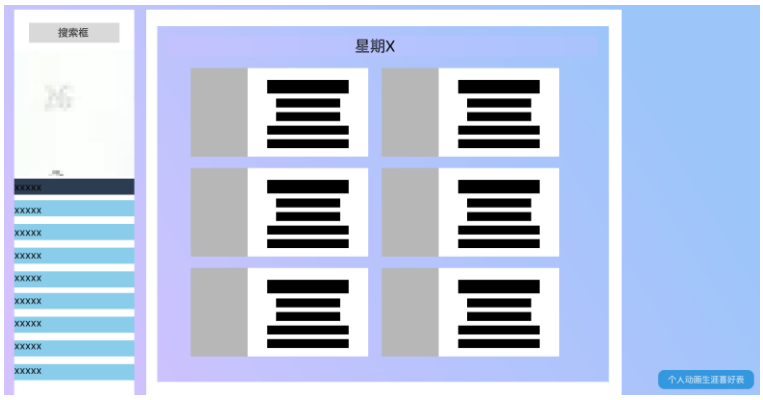
Figma 是一种基于云的协作性设计工具，主要用于用户界面（UI）和用户体验（UX）设计。允许多个设计师在同一项目上实时协作。这使得团队能够同时查看和编辑设计，提高了协作效率。

编剧推荐界面原型设计(如下图)



音乐播放器原型设计(如下图)

番剧日历原型设计(如下图)



配置分离

将配置信息分离到一个独立的配置文件中并使用 YAML 格式,可以确保进行版本控制的时候不把敏感信息上传，保证我的密钥信息安全。(如下图，express 读取 yaml 文件中的应用密钥)

```
const fs = require('fs');
const yaml = require('js-yaml');

function readYamlFile(filePath) {
  try {
    const fileContents = fs.readFileSync(filePath, 'utf8');
    const yamlData = yaml.load(fileContents);
    return yamlData || {};
  } catch (error) {
    console.error(`Error reading YAML file: ${error.message}`);
    return {};
  }
}

module.exports = { readYamlFile };
```

Vue3

项目采用了选项式和组合式俩种 API 风格，选项式： 着重于如何执行某个任务，代码更注重细节和流程控制。通常涉及更多的状态管理和副作用。

组合式： 着重于定义问题的本质，通过声明性的方式表达目标，代码更关注数据的映射和转换。（如下图，组合式的形式）

```
<script lang="ts" setup>

import { useRouter } from 'vue-router'
import AnimeGrid from '@/components/animegrid/AnimeGrid.vue'
import { onMounted } from 'vue';

// import {getAnswer} from '../api/chat/chatAPI.js'
// import HelloWorld from './components/HelloWorld.vue'


let router = null;

const returnMainPage = function () {
    router.push('/bangumi')
}

onMounted(()=>{
    router = useRouter();
});
```

选项式形式（如下图）

```
export default {
    name: 'BangumiPage',
    components: {
        BgmCalendar,
        CalendarView,
        CarouselView,
        RecommendationCard,
        BgmCard,
        LineChart,
        MessageBoard,
        FutureBgmGroup
    },
    data() {
        return {
            router: null,
            year: 2023,
            month: 12,
            day: 10,
            isClassActive: true,
            inputValue: '',
            page: 'BgmCalendar',
            animes: [],
            searchResult: [],
            chartData: [10, 20, 30, 40, 50, 30, 25],
            xAxisLabels: ['2023-11-10', '2023-11-11', '2023-11-12', '2023
        }
    },
    mounted() {
        this.router = useRouter();
        this.getTime();
        this.animes = recommandedMock;
        // console.log(this.animes);
    },
    methods: {
        getTime() {
            const date = new Date()
            this.year = date.getFullYear();
            this.month = date.getMonth() + 1;
            this.day = date.getDate();
        },
        inputUsing() {
```

Http 请求统一出入口

在一个应用中，实现 HTTP 请求的统一出入口有助于提高代码的可维护性、可扩展性和降低重复代码的数量。我们使用 axios 的拦截器来拦截请求的发出，来携带 token。（如下图，关于阿里灵积服务的拦截器）

```
export const chatApi = axios.create({
    baseURL: "https://dashscope.aliyuncs.com",
    timeout: 5000,
});

// 在请求拦截器中添加 Bearer Token
chatApi.interceptors.request.use(
    (config) => {
        const bearerToken = readYamlFile('../config/token.yaml')['token'];

        // 如果存在 Bearer Token，则将其添加到请求头中
        if (bearerToken) {
            config.headers["Authorization"] = `Bearer ${bearerToken}`;
        }

        return config;
    },
    (error) => {
        return Promise.reject(error);
```

日历

用了 Vue.js 的模板语法，包括动态绑定、事件处理等。通过 v-for 指令循环生成星期的表头和日历的每一天，使用动态绑定来显示当前的月份和年份。在方法部分，prevMonth 和 nextMonth 方法分别用于切换到上一个月和下一个月，并处理年份的变化。isCurrentDay 方法用于判断某一天是否为当前选中的日期，以便在模板中添加相应的样式。(如下图，html 与动态绑定)

```
<div class="calendar-container">
  <h2 class="header">{{ currentMonth }} {{ currentYear }}</h2>
  <div class="month-switch">
    <button @click="prevMonth" class="nav-button">&#60;</button>
    <button @click="nextMonth" class="nav-button">&#62;</button>
  </div>
  <table class="calendar-table">
    <thead>
      <tr>
        <th v-for="day in daysOfWeek" :key="day" class="day-header">{{ day }}</th>
      </tr>
    </thead>
    <tbody>
      <tr v-for="(week, index) in calendar" :key="index">
        <td
          v-for="day in week"
          :key="day.date"
          :class="{ 'prev-month': day.prevMonth, 'current-day': isCurrentDay(day) }"
          class="calendar-cell"
        >
          {{ day.date }}
        </td>
      </tr>
    </tbody>
```

通过计算属性 calendar 动态生成一个表示当前月份的日历数组。使用 JavaScript 的 Date 对象来获取当前月份的第一天和最后一天，然后根据这些信息填充日历数组。每个日期对象包含了日期的数字和一个标志表示是否为上个月的日期（prevMonth）。（如下图，计算属性）

```
computed: {
  calendar() {
    const firstDayOfMonth = new Date(this.currentYear, this.currentMonth - 1, 1);
    const lastDayOfMonth = new Date(this.currentYear, this.currentMonth, 0);
    const firstDayOfWeek = firstDayOfMonth.getDay();
    const lastDateOfMonth = lastDayOfMonth.getDate();
    const calendar = [];
    let currentDate = 1;

    for (let i = 0; i < 6; i++) {
      const week = [];

      for (let j = 0; j < 7; j++) {
        if ((i === 0 && j < firstDayOfWeek) || currentDate > lastDateOfMonth) {
          week.push({ date: "", prevMonth: true });
        } else {
          week.push({ date: currentDate, prevMonth: false });
          currentDate++;
        }
      }

      calendar.push(week);
    }
```

轮播图

使用 Vue.js 的 v-for 指令遍历轮播项数组，渲染每个轮播项的容器。在这些轮播项外有一个父元素包含所有轮播项，通过父元素 CSS 的 transform 属性实现当前轮播项的显示。通过表示当前显示的轮播项的索引。（如下图，html）

```
<div class="auto-carousel">
    <div class="slides" :style="{ transform: `translateX(${-currentIndex * 100}%)` }">
        <div v-for="(slide, index) in slides" :key="index" class="slide">
            <img :src="slide.image" alt="Slide">
        </div>
    </div>
    <button @click="prevSlide" class="nav-button prev">Previous</button>
    <button @click="nextSlide" class="nav-button next">Next</button>
</div>
```

其中两个按钮对应的按钮，分别对应 onclick 函数为 preSlide 和 nextSlide，两者一起控制 currentIndex，令其在 slides 中循环。（如下图，数据与方法）

```
data() {
    return {
        slides: [
            { image: 'https://lain.bgm.tv/pic/cover/c/a6/66/326_D8wjw.jpg' },
            { image: 'https://lain.bgm.tv/pic/cover/c/c2/4c/253_jJJj9.jpg' },
            { image: 'https://lain.bgm.tv/pic/cover/c/67/d1/876_dCfrd.jpg' },
        ],
        currentIndex: 0,
        autoPlayInterval: 5000, // Set the interval for auto play in milliseconds
    };
},
methods: {
    prevSlide() {
        this.currentIndex = (this.currentIndex - 1 + this.slides.length) % this.slides.length;
    },
    nextSlide() {
        this.currentIndex = (this.currentIndex + 1) % this.slides.length;
    },
    startAutoPlay() {
        this.autoPlayTimer = setInterval(this.nextSlide, this.autoPlayInterval);
    },
    stopAutoPlay() {
        clearInterval(this.autoPlayTimer);
    },
},
```

为了让轮播图在规定的时间内运算，防止在其他时候占用性能，我们获取整个组件的生命周期并在对应的钩子函数上创建或销毁资源。mounted()：组件挂载后调用的生命周期钩子，用于在组件加载后开始自动播放。beforeUnmount()：在组件销毁前调用的生命周期钩子，用于在组件销毁前停止自动播放。

```
mounted() {
    this.startAutoPlay();
},
beforeUnmount() {
    this.stopAutoPlay();
},
```

留言板

检查留言内容是否为空，若为空则弹出提示，否则将留言信息添加到 messages 数组中，并清空输入框。而 messages 数组是用于构造 message(html)中的聊天信息栏的，通过 v-for 动态的渲染 message 数量令其与 messages 数组绑定。(如下图，messages 数组及动态绑定)

```
export default {
    data(){
        return {
            messages:[],
            username:'',
            message:'',
        };
    },
    methods:{
        submit(){
            if(this.message == ''){
                alert('请输入留言内容')
                return
            }

            const now = new Date()
            this.messages.push({'username':this.username == ''?'匿名':this.username,'message':this.message,'date':now.getFullYear()+'年'+(now.getMont
            this.message = ''
            this.username = ''
        },
    }
}
</script>
```

```
<div id="messageBoard">
    <div v-for="(item, index) in messages" :key="index" class="message">
        <div class="message-info">
            <div class="info">
                <img src="../../assets/logo.png" alt="" width="50px" height="50px">
                <strong>{{item.username}}</strong>
            </div>
            <span>发布于:{{item.date}}</span>
        </div>
        <div class="content">{{item.message}}</div>
    </div>
</div>
```

折线图

通过画笔在画布上进行绘制，然后通过 const ctx = canvas.getContext("2d");获取画布的 2D 上下文，以便后续的绘制操作。接着，使用 ctx.clearRect(0, 0, this.width, this.height);清空画布，确保每次重新绘制时不会叠加之前的内容。随后，计算数据点之间的横向间隔，以确定数据点在横轴上的位置。通过 const maxY = Math.max(...this.data);和 const minY = Math.min(...this.data);计算数据的最大值和最小值，以确定纵向的数值范围。再进行计算纵向的缩放比例，使数据点在纵轴上能够合适地显示在画布上。在计算坐标轴、刻度线的位置和数值时，包括横轴和纵轴的刻度线。

在绘制坐标刻度和数值阶段，通过循环遍历数据点，在横轴和纵轴上绘制刻度线，并在刻度线旁边显示对应的数值。接下来，进行绘制折线，使用 ctx.beginPath();开始绘制路径，然后使用 ctx.lineTo()连接数据点，形成折

线。在设置线的样式时，使用 ctx.strokeStyle 设置折线的颜色，使用 ctx.lineWidth 设置线的宽度。最后，通过使用 ctx.stroke();完成折线的绘制。（如下图，绘制折线图部分）

```
drawChart() {
    const canvas = this.$refs.chartCanvas;
    const ctx = canvas.getContext("2d");

    // 清空画布
    ctx.clearRect(0, 0, this.width, this.height);

    // 计算数据点之间的间隔
    const dataPointsCount = this.data.length;
    const intervalX = this.width / (dataPointsCount - 1);

    // 计算数据的最大值和最小值
    const maxY = Math.max(...this.data);
    const minY = Math.min(...this.data);

    // 计算数据点的纵向缩放比例
    const scaleY = (this.height-50) / (maxY - minY);

    // 计算竖轴刻度的值和间隔
    const yTickCount = 5; // 可以根据需要调整刻度数量
    const yTickInterval = (maxY - minY) / (yTickCount - 1);

    // 开始绘制坐标系
    ctx.beginPath();
    ctx.moveTo(40, 10);
    ctx.lineTo(40, this.height - 20);
    ctx.lineTo(this.width - 10, this.height - 20);
    ctx.strokeStyle = "black";
    ctx.stroke();

    // 绘制坐标刻度和数值
    for (let i = 0; i < dataPointsCount; i++) {
        const x = i * intervalX + 40;
        const y = this.height - 20 - (this.data[i] - minY) * scaleY;

        // 绘制竖直刻度线
        ctx.beginPath();
        ctx.moveTo(x, this.height - 20);
        ctx.lineTo(x, this.height - 15);
        ctx.strokeStyle = "black";
```

获取网易云二次元音乐日推

由于大部分的 http 都是很像的所以这里就简单的描述一下获取网易云音乐的二次元音乐日推。两个方法都是异步函数，使用了 await 来等待异步操作的完成。在成功时，会更新组件的状态，而在失败时，会打印错误信息到控制台。这种异步处理方式通常用于处理异步请求，保持代码的清晰性和可读性。这里运行的耗时任务是向网易云音乐服务器发送 http 请求获取最新的音乐数据。(如下图)

```
    async detail() {
        try {
            const response = await detail({ 'id': this.musicGroupId, 's': this.s })
            // console.log(response);
            this.tracks = response.data.playlist.tracks
            // console.log('tracks:'+this.tracks);
        } catch (error) {
            console.error('Error fetching data:', error);
        }
    },
    async getUrl(id) {
        try {
            const response = await songUrl({ 'id': id })
            console.log(response);
            this.url = response.data.data[0].url

        } catch (error) {
            console.error('Error fetching data:', error);
        }
    },
    changeMusic(index) {
        this.name = this.tracks[index].name
        this.pic = this.tracks[index].al.picUrl
        this.getUrl(this.tracks[index].id)
    }
},
mounted() {
    this.router = useRouter();
    this.detail()
```

大语言模型的接入

将大语言模型整合到网站中，可以为用户提供更智能和自然的在线体验。这包括实现智能客服和交互，使用户能够通过与模型对话获取信息、解决问题。此外，模型还可用于生成网站内容，如文章、产品描述和评论，从而提高内容的更新频率和质量。搜索引擎的性能也可通过模型进行优化，提供更准确和相关的搜索结果，从而改善用户体验。自动化任务，如邮件回复、报告生成和表单处理，也可以借助模型实现。个性化推荐、语言翻译、情感分析以及虚拟助手等功能，都可以通过整合大语言模型来丰富网站的交互和服务，提升用户满意度。

其中聊天记录为这一次打开的文本，在这个页面临时存储并没有加入到数据库中。主要的获取回答模块交由反向代理服务器进行处理。实现方法与上面的请求类似，这里不再阐述。（代码如下图）

```
async sayWithMiku() {
    this.history.push({ 'sayer': 'me', 'text': this.inputValue })
    // const response = await getAnswer({"model": "qwen-v1",   "input": {"prompt": this.inputValue},"parameters": {}})
    const response = await getAnswer({ 'question': this.inputValue })
    this.history.push({ 'sayer': 'miku', 'text': response.data.data.text })
    this.inputValue = ''
}
```

RSS 订阅与 Atom 订阅：

首先，我们开始通过获取 XML 文件的方式，然后运用 RSS 2.0 格式对 XML 进行解析，以便有效地提取必要的数据。这关键数据通常包括标题、链接、摘要等信息，从而形成一份清晰的 RSS 订阅。

同样的方法也适用于 Atmo，其数据的获取和解析过程类似。通过这一步骤，我们能够在 XML 中识别和提取关键的信息，为用户提供整合、可读的订阅内容。这种处理方式不仅确保数据的准确性，同时也为用户提供了更为便捷的信息获取途径。（如下图,rss 订阅）

```
async function fetchRSS(url) {
    try {
        const response = await axios.get(url);
        return response.data;
    } catch (error) {
        console.error("Error fetching RSS feed:", error);
        throw error;
    }
}

// 解析RSS feed的函数
async function parseRSS(xmlData) {
    return new Promise((resolve, reject) => {
        parseStringAsync(xmlData, { explicitArray: false }, (err, result) => {
            if (err) {
                console.error("Error parsing RSS feed:", err);
                reject(err);
                return;
            }

            // 提取所需的信息
            const items = result.rss.channel.item;
            const res = [];
            for (let i = 0; i < items.length; i++) {
                const title = items[i].title;
                const link = ite const items: any
                const pubDate = items[i].pubDate;
                res.push({ title: title, link: link, pubDate: pubDate });
            }

            resolve(res);
        });
    });
}
```

动态个人动画生涯爱好表

制作个人动画生涯爱好表可以帮助你明确动画领域中自己的兴趣和目标，从而更有针对性地发展相关技能。此外，制作爱好表可以成为学习动画的指南，规划学习路径，避免零散学习。通过建立技能清单，你能够有力地展示自己在动画领域的专业素养，为未来的发展提供支持。更新爱好表有助于监控个人在动画领域的进展，保持动力。分享表格则可以获得有益的反馈和建议，同时也为构建网络与合作机会提供契机，让你找到更多志同道合的伙伴。我们制作的动态个人动画生涯喜好表可以更好的写出自己想写的 tag（如下图）

编程实现（如下图）

```
data() {
    return {
        tags: [],
        inputValue: '',
        isModal: false,
        curIndex: 0,
        words:[]
    }
},
methods: {
    addTag() {
        if (this.inputValue == '') {
            console.log(this.tags);
            return
        }
        this.tags.push({ tagName: this.inputValue, imgUrl: '' })
        this.inputValue = ''
    },
    openModal(index) {
        this.$refs.modalRef.openModal();
        this.isModal = true
        this.curIndex = index
        // console.log('click');
    },
    activate() {
        this.isModal = false
    },
    handleDataFromChild(img_src,title) {
        // console.log(data);
        // Vue.set(this.tags, this.curIndex, { ...this.tags[this.curIndex], imgUrl: data
        // this.$set(this.tags, this.curIndex, { ...this.tags[this.curIndex], imgUrl: da
        // this.tags[this.curIndex].imgUrl = data;
        // this.$forceUpdate();
        // triggerRef(this.tags)
        this.words.push({"text":title})
        this.tags[this.curIndex].imgUrl = img_src;
    },
    capture() {
        const captureDiv = this.$refs.captureDiv;
        if (window.ClipboardItem) {
            html2canvas(captureDiv, {
                // backgroundColor: null,//画出来的图片有白色的边框,不要可设置背景为透明色 (
```

词云生成

词云以直观的方式呈现文本数据中的关键词，使得信息更易于理解。通过字体大小、颜色的变化，用户可以迅速识别出数据中的重要概念或关键词。通过词云，可以在文本数据中发现潜在的主题或趋势。重复出现的词汇可能反映了文本的主题，有助于用户发现数据中的模式和关联。在动画圈中通过词云可以更好的挖掘有相同爱好的人（如下图，词云）

编程实现(如下图)：

```
        currentLineLength += wordLength;
      } else {
        this.wordLines.push([...currentLine]);
        currentLine = [word];
        currentLineLength = wordLength;
      }
    });

    if (currentLine.length > 0) {
      this.wordLines.push([...currentLine]);
    }
  },
  calculateWordLength(word) {

    return word.text.length;
  },
  getWordStyle() {
    const randomColor = this.getRandomColor();
    const fontSize = Math.floor(Math.random() * (2.5 - 1 + 1)) + 1; // Random font size between 1 and 2.5
    return {
      fontSize: `${fontSize}em`,
      color: randomColor,
      cursor: 'pointer',
      margin: '5px',
      display: 'inline-block',
    };
  },
  getRandomColor() {
    const letters = '0123456789ABCDEF';
    let color = '#';
    for (let i = 0; i < 6; i++) {
      color += letters[Math.floor(Math.random() * 16)];
    }
    return color;
  },
  handleWordClick(word) {

    console.log(`Clicked on word: ${word.text}`);
  },
 },
};
```

函数 getRandomColor()，其目的是生成一个随机的十六进制颜色代码。函数内部首先定义了一个包含十六进制颜色字符的字符串 letters，然后初始化一个变量 color 为 '#'，表示颜色代码的起始部分。接下来，通过一个循环，随机选择 letters 字符串中的字符，构建一个六位的颜色代码。最后，将生成的颜色代码返回。整个过程利用了 Math.random() 函数生成随机数，以及 Math.floor() 函数将随机数映射到整数，从而得到随机的颜色代码。

```
  getRandomColor() {
    const letters = '0123456789ABCDEF';
    let color = '#';
    for (let i = 0; i < 6; i++) {
      color += letters[Math.floor(Math.random() * 16)];
    }
    return color;
  },
```

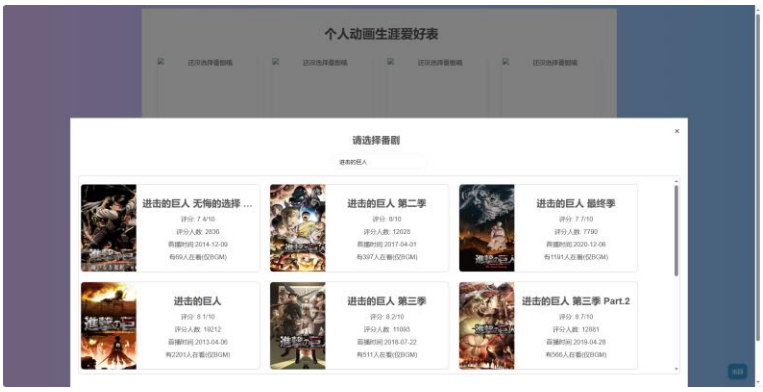
函数 getWordStyle()，其目的是生成一个包含随机字体大小、随机颜色以及其他样式属性的对象。首先，通过调用先前定义的 getRandomColor() 函数获取一个随机颜色。然后，利用 Math.random() 生成一个介于 1 到 2.5 之间的随机字体大小，并通过对象字面量构建一个包含字体大小、颜色、光标样式、外边距和行内块显示属性的对象。最后，将这个样式对象作为函数的返回值。因此，

调用 getWordStyle() 将返回一个随机定义的文本样式对象，用于在前端应用中设置文本的外观。

```
getWordStyle() {
  const randomColor = this.getRandomColor();
  const fontSize = Math.floor(Math.random() * (2.5 - 1 + 1)) + 1; // Random font size between 1 and 2.5
  return {
    fontSize: `${fontSize}em`,
    color: randomColor,
    cursor: 'pointer',
    margin: '5px',
    display: 'inline-block',
  };
},
```

拟态框

拟态框是在前端网页中经常出现的一种展现信息的方式，拟态框通过在屏幕上居中显示并覆盖其他内容，有效地将用户的注意力集中在弹出框中的信息上。这有助于确保用户看到并理解重要的信息。（如下图）



编程实现（如下图）

```
export default {
  components: {
    BgmCard
  },
  data() {
    return {
      isVisible: false,
      inputValue: '',
      searchResult: []
    };
  },
  methods: {
    openModal() {
      this.isVisible = true;
      // console.log('visible')
    },
    closeModal() {
      this.searchResult = [];
      this.inputValue = '';
      this.isVisible = false;
    },
    async searchAnime() {
      const response = await search(this.inputValue, { 'type': 2, 'responseGroup': 'large' })
      console.log(response.data.list)
      this.searchResult = response.data.list
    },
    sendDataToParent(imgSrc,title) {
      this.closeModal()
      // 传递数据给父组件
      // console.log("son:"+imgSrc);
      this.$emit('sendData', imgSrc, title);
    }
  },
};
</script>
```

反向代理服务器

在前端使用 JavaScript 在浏览器中访问某个 API 时，由于跨域限制而无法直接访问，而通过创建一个后端应用来代理这个请求，那么这个后端应用可以被称为代理服务器。代理服务器的作用是在客户端和目标服务器之间充当中间人，转发请求并返回响应，从而绕过浏览器的同源策略限制。这种做法常被称为“跨域代理”或“反向代理”。通过这种方式，前端应用可以通过与同一域的后端通信，而后端应用则负责与目标 API 通信，使得前端能够绕过浏览器的跨域限制而获得所需数据。（如下图，express 服务器对请求的代理）

```
app.post("/call", async (req, res) => {
    try {
        const question = req.body.question;
        const apiUrl =
            "https://dashscope.aliyuncs.com/api/v1/services/aigc/text-generation/ge
        const requestData = {
            model: "qwen-v1",
            input: {
                prompt: question,
            },
            parameters: {},
        };

        const response = await axios.post(apiUrl, requestData, {
            headers: {
                "Content-Type": "application/json",
                Authorization: `Bearer ${bearerToken}`,

            },
        });

        // 将 API 的响应转发给前端
        res.status(response.status).json({'code':200,'data':response.data.output});
```

爬虫

爬虫技术在网络信息抓取和数据挖掘领域中发挥着重要作用。通过爬虫，我们可以定期从特定网站上抓取最新的即将更新的动画信息，然后将这些数据整合展示在我们的应用程序中。这种方法使得我们能够及时获取最新的动画更新情况，为用户提供高质量的服务。（如下图，部分爬虫代码）

```python
def scrape_website(url):
    headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36'}

    # 发送网络请求获取HTML内容
    req = urllib.request.Request(url, headers=headers)
    response = urllib.request.urlopen(req)
    soup = BeautifulSoup(response.read(), 'html.parser')

    posts = soup.find_all(class_='post-body')
    result = []

    for post in posts:
        future_intro_list = post.find_all(class_='future_intro')

        for index, intro in enumerate(future_intro_list):
            title = intro.text.strip()
            items = []

            # Find the next sibling of the current future_intro
            next_element = intro.find_next_sibling()
            # Loop until the next future_intro or the end of the post-body
            while next_element and next_element.name != 'p' and 'future_intro' not in next_element.get('class', []):

                if next_element.name == 'div'and next_element.find('td',class_='future_title_') is not None:
                    future_title = next_element.find('td',class_='future_title_').text.strip()
                    future_date = next_element.find('p',class_=re.compile('^future_date')).text.strip()
                    future_type = next_element.find('p',class_=re.compile('^future_type_')).text.strip()
                    img_url = next_element.find('img')['src']

                    item = {
                        'future_title': future_title,
                        'future_date': future_date,
                        'future_type': future_type,
                        'img_url': img_url
                    }

                    items.append(item)

                next_element = next_element.find_next_sibling()

            result.append({'title': title, 'items': items})
```

爬虫可以模拟用户在目标网站上的浏览行为，自动化地访问页面并提取我们需要的信息。这可能包括动画的更新时间、名称、封面图片、简介等。

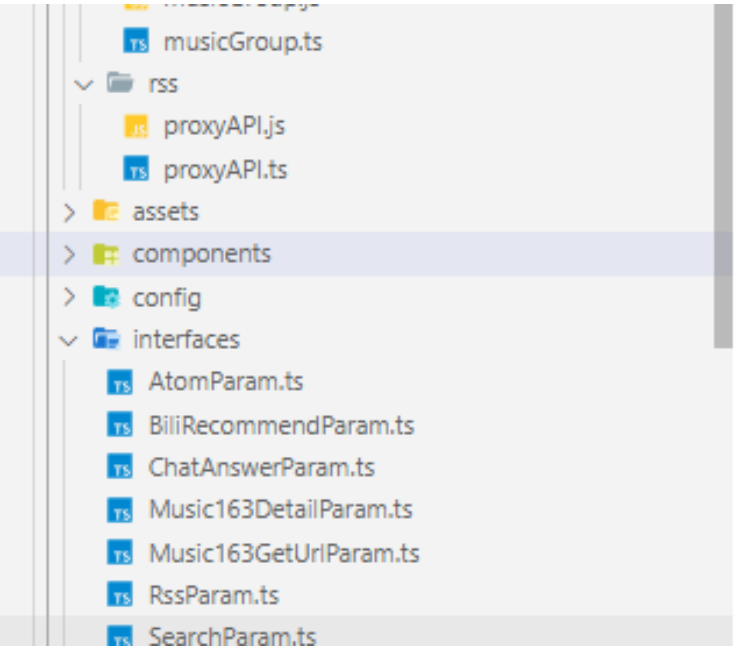
最后输出这些整理好的数据为一个 json 文件（如下图，输出的 json 文件格式）


```json
[
  {
    "title": "即将着陆[冬季档] (2024年1月±)",
    "items": [
      {
        "future_title": "憧憬成为魔法少女",
        "future_date": "2024冬",
        "future_type": "漫改",
        "img_url": "https://i0.hdslb.com/bfs/new_dyn/2d3bf78274c03508b5f99608c00cafe5512995925.jpg"
      },
      {
        "future_title": "愚蠢天使与恶魔共舞",
        "future_date": "2024冬",
        "future_type": "漫改",
        "img_url": "https://i0.hdslb.com/bfs/new_dyn/2a147422afead02f1f937ac1b7eb689a512995925.jpg"
      },
      {
        "future_title": "勇气爆发BangBravern",
        "future_date": "2024冬",
        "future_type": "原创",
        "img_url": "https://i0.hdslb.com/bfs/new_dyn/6379cf6c6171b71d63486bf4c2278793512995925.jpg"
      },
      {
        "future_title": "事与愿违的不死冒险者",
        "future_date": "2024",
        "future_type
        "img_url": "https://i0.hdslb.com/bfs/new_dyn/1888b46b5384abfe02cf5c8cfd33b31d512995925.jpg"
      },
      {
        "future_title": "狩火之王第2期",
        "future_date": "2024冬",
        "future_type": "小说改",
        "img_url": "https://i0.hdslb.com/bfs/new_dyn/8247e0aace6b3165258820a8cbb1f622512995925.jpg"
      },
      {
        "future_title": "奇异贤伴 黑色天使Part.2",
        "future_date": "2024冬",
        "future_type": "游戏改",
        "img_url": "https://i0.hdslb.com/bfs/new_dyn/89f3d8ad0b68c387d5adc7bda1b84d4d512995925.jpg"
      },
      {
        "future_title": "到了30岁还是处男似乎会变成魔法师",
```


Follow link (ctrl + click)

引入 TypeScript 进行工程化

TypeScript 相对于纯粹的 JavaScript 具有许多优势，这些优势可以提高代码的可维护性、可读性和稳定性。在保留 js 的同时引入 TypeScript（如下图）



静态类型检查： TypeScript 提供了强大的静态类型检查，能在编译时捕获许多潜在的错误，例如类型不匹配、未定义的变量等。这有助于减少在运行时出现的错误，提高代码的质量。

面向对象编程支持： TypeScript 提供了接口、类、模块等面向对象编程的特性，使得代码的组织和结构更具可扩展性。这对于大型项目和团队协作非常有益。

```
1  export interface Music163DetailParam {
2      id: string;
3      s: string;
4  }
5
```

丰富的类型系统： TypeScript 的类型系统非常丰富，包括基本类型、联合类型、交叉类型、泛型等。这使得开发者可以更精确地表达代码的意图，并更好地使用类型系统来辅助开发。

五、总结

新番观测站的开发旨在为用户提供高质量的动画信息和咨询服务。通过用户友好的界面设计，实现了直观、轻松的动画浏览和搜索体验。推荐系统和社交互动功能的引入提高了用户留存和参与度。在项目中取得的成就主要体现在用户体验的提升、社交互动的增强以及网站性能和稳定性的保障。用户能够轻松地找到感兴趣的动画，并通过评论和分享功能进行社交互动。同时，项目在前端和后端的性能优化方面取得了显著的进展，保障了网站的稳定运行。

然而，还存在一些需要改进的方面。首先，可以进一步优化性能，确保网站的加载速度更快。其次，需要加强用户信息和数据的安全性，防范潜在的安全威胁。最重要的是引入 gRPC 作为通信协议，以提高与跨域代理服务间通信的效率。gRPC 的优势在于性能提升、多语言支持、自动化代码生成和双向流式通信，有助于项目的可扩展性和灵活性。使用 gRPC 就不用自己写关于各类编程语言的 libraries，极大的提高了开发效率。

改进计划包括引入 gRPC 技术，并根据其支持的微服务架构对核心功能进行服务拆分，提高系统的可维护性和扩展性。此外，计划实施 gRPC 缓存，减少对底层数据存储的频繁访问，以及引入监控和日志功能，更好地了解系统的运行状况。此外，我还想引入 Protocol Buffers 技术来取代部分通信所使用的 json，Protocol Buffers 使用二进制编码，相较于基于文本的格式（如 XML 和 JSON），它更紧凑，占用更少的网络带宽和存储空间。

通过这些改进，我们期望进一步提升项目的性能、灵活性和可维护性，为用户提供更出色的动画咨询体验。