

# 设计与实现报告

## 目录

1. 简介
2. 系统架构
3. 关键组件
4. 实现细节
5. 外部信息
6. 挑战与解决方案

## 简介

## 背景

智能旅行伴侣是一个为了帮助旅行者规划和导航他们的旅行而开发的个人项目。由于需要大量的信息来源，为一次旅行收集所需的所有信息可能会很具挑战性。此应用整合了多种API，提供了一个集中的解决方案。

## 目标

1. **简化旅行计划**：通过整合多种API，为旅行者提供一站式服务，使他们能够方便地规划和导航他们的旅行。
2. **集中化信息源**：鉴于现有的大量信息来源，您的应用旨在集中这些信息，以便用户不必在多个平台上查找旅行相关信息。
3. **提供直观的交互方式**：应用中的自然语言交互功能使用户能够以自然和直观的方式查询信息。
4. **助力决策**：通过功能如路线规划、实时天气查询、火车信息查询和景点查询，帮助用户做出更明智的旅行决策。
5. **提高旅行体验**：总体目标是通过上述所有功能，增强和简化用户的旅行体验，从而使旅行变得更为轻松和愉快。

## 开发人员信息

版本: [1.0]

开发人员: [2051828 莫益萌]

# 系统架构

本应用为Web端应用程序，未设置后端。

## 前端:

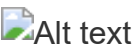
- **技术栈**: 使用React或Vue.js等现代JavaScript框架来构建。
- **功能**: 负责应用的用户界面和与用户的交互。
- **通讯**: 通过获取外部的API进行数据交互。

# 关键组件

## 聊天组件

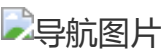
聊天组件是整个应用程序的骨干框架。用户可以在底下的输入框输入自然语言，随后应用程序会根据用户语言中所含的关键词进行API调用的判断。

- **正常聊天**: 若不含任何关键词，则用户可以与讯飞星火API正常聊天。
- **API调用**: 若用户的输入中含有关键词，则调用相应的API函数，进行下一步数据的生成与处理。



## 导航组件

导航组件主要负责导航功能。当用户输入所在城市、起点、终点，并且包含'导航'关键字时，讯飞星火API会提取出用户所输入的城市、起点、终点，并进行公共交通模式的导航。




- **重新导航**: 若用户对自然语言定位的地点不太满意，用户在地图内单击自己想要的地点，点击选为起点或选为终点，进行重新导航。
- **生成建议**: 用户可以选择不同的方案，并点击“生成文本建议”按钮以生成GPT文本。GPT文本会根据用户选择的导航方案的明细、距离和时间，以及现在的实时天气，结合GPT的文本生成能力，生成出行建议的文本。



# 火车票组件

火车票组件主要负责火车票的查询功能。当用户输入日期、起点城市、终点城市，并且包含‘火车’、‘高铁’等关键字时，讯飞星火API会提取出用户所输入的日期、起点、终点，并进行12306网站的调用。

火车票查询图片

- **显示时刻表和余票信息**：系统会生成该日期的所有班次，并显示时刻表和余票信息。
- **直观显示距离**：此外，底下的地图将起点与终点连成线，更加方便用户直观地了解其距离。


# 天气组件

天气组件支持用户查询某城市的实时天气和7日天气。当用户的输入内容包含一个城市以及‘天气’这一关键字时，讯飞星火API会提取出用户所指明的城市。接下来，它会调用高德地图的天气API以获取该城市的实时天气信息，并同时调用API网站上的天气API以获取7日的天气预报。这些信息将会被展示在主界面中。

天气预报图片

# 景区组件

景区组件支持用户对某一城市或某一景点进行搜索。当用户输入地点名称及‘景区’、‘旅游’等关键字时，讯飞星火API会提取出用户所指的景点或城市。接着，该系统会调用Wikipedia以及API网站上的景区介绍API进行搜索。搜索的结果将分别展示在页面上，用户可以根据自己的需求来进行浏览和阅读。

景区搜索图片

# 提示组件

提示组件可以对用户的输出进行提示。点击相关关键词和相关句子，可以直接将信息发送给系统。

提示图片

# 实现细节

## 3. 聊天组件的深入细节

- **关键词识别**：系统采用关键词识别的方式进行API选用的判断。

```

async APIChoose(){
  const apiConfigurations = [    // 配置API的关键词和相应信息
    /*此处省略 */
  ];
  // 创建一个关键词列表, 其中包含配置索引和关键词
  let keywordsList = apiConfigurations.map((config, index) => {
    return {
      index: index,
      keywords: config.keywords
    };
  });
  console.log(JSON.stringify(keywordsList))
  let matchedConfig = null;
  for (let config of apiConfigurations) {
    // 循环遍历每个配置的关键词
    for (let keyword of config.keywords) {
      if (this.userInput.includes(keyword)) {
        // 设置API类型
        this.ApiType = config.apiType;
        if(this.ApiType !== 6){
          await this.getsparkResponse(`这是输入文本: “${this.userInput}”${config.extracti
        }
        // 设置匹配的配置为当前配置
        matchedConfig = config;
        break;
      }
    }
    // 如果找到了匹配的配置, 跳出循环
    if (matchedConfig) break;
  }

  // 如果没有找到匹配的配置
  if(!matchedConfig) {
    this.ApiType = 0;
    await this.getsparkResponse(this.userInput);
    this.chatHistory.push({ role: 'gpt', content: this.sparkHistory, type: 'text'});
    this.gptResponse = '';
    this.sparkHistory = '';
  }
},

```

- 如何与讯飞星火API结合：通过设置讯飞星火的输出格式，使其输出特定的格式，之后便可用正则表达式进行提取。

// 正则表达式匹配

```
responseText += '。';
const matchDate = responseText.match(/日期: (.?)/);
const matchCity = responseText.match(/城市: (.?)/);
const matchStart = responseText.match(/起点: (.?)/);
const matchEnd = responseText.match(/终点: (.?)。/);
const matchLocation = responseText.match(/地点: (.?)。/);
const matchWeatherCity = responseText.match(/城市: (.?)。/);

let city, date, startPoint, endPoint, location;
let today, yyyy, mm, dd, defaultDate, missingValues = [];
// 根据ApiType值提取相应的数据
switch (this.ApiType) {
  case 1: // 城市、起点、终点
    city = matchCity ? matchCity[1] : null;
    startPoint = matchStart ? matchStart[1] : null;
    endPoint = matchEnd ? matchEnd[1] : null;
    console.log(city, startPoint, endPoint, location);
    /*此处省略 */

    break;

  case 2: // 飞机
    /*此处省略 */

    break;

  case 3: // 火车
    today = new Date();
    yyyy = today.getFullYear();
    mm = String(today.getMonth() + 1).padStart(2, '0');
    dd = String(today.getDate()).padStart(2, '0');
    defaultDate = `${yyyy}-${mm}-${dd}`;
    if (!matchDate) {
      date = defaultDate;
    }
    else {
      date = this.formatDate(matchDate[1]) ? this.formatDate(matchDate[1]) : defaultDate;
    }
    console.log(defaultDate, matchDate, date);
    startPoint = matchStart ? matchStart[1] : null;
    endPoint = matchEnd ? matchEnd[1] : null;

    if (!startPoint || startPoint == 'AAA') {
      missingValues.push("起点");
    }
  }
}
```

```

    }
    if (!endPoint || endPoint == 'BBB') {
        missingValues.push("终点");
    }

    if(this.handleMissingValue(missingValues))
    {
        this.chatHistory.push({ role: 'gpt', content: '', type: 'train', date: date })
    }
    this.sparkHistory = [];

    break;
case 4: // 景点
    location = matchLocation ? matchLocation[1] : null;
    console.log(location,matchLocation);
    if (!location || location == 'BBB') {
        missingValues.push("景点");
    }
    if(this.handleMissingValue(missingValues))
    {

        this.chatHistory.push({ role: 'gpt', content: '', type: 'scenary', location: location })
    }

    this.sparkHistory = [];
    break;
case 5: //天气
    city = matchWeatherCity ? matchWeatherCity[1] : null;

    if (!city || city == 'CC') {
        missingValues.push("城市");
    }
    if(this.handleMissingValue(missingValues)){
        this.chatHistory.push({ role: 'gpt', content: '', type: 'weather', city: city })
    }

    this.sparkHistory = [];
    break;
case 6: //GPT
    this.gptInput = this.userInput;
    this.getGPTResponse();
    break;

```

```
        default:
            break;
    // ... 更多的case
}
```

## 4. 导航组件的深入细节

- 地图技术的应用:本应用使用了百度地图的官方API，并实现了导航，地址编码转换等多个功能。



```

async function TransNavigation(startPoint, endPoint, city, mapId, callback) {
  if (typeof BMapGL === 'undefined') {
    console.error('百度地图API未加载');
    return;
  }

  let map;
  try {
    map = new BMapGL.Map(mapId); //创建地图实例
    map.centerAndZoom(city, 12);
  } catch (error) {
    console.error("Error with BMapGL:", error.message);
    return;
  }

  const StartGeo = new BMapGL.Geocoder(); //创建地址编码
  const EndGeo = new BMapGL.Geocoder();

  try {
    const sPoint = await getPoint(StartGeo, startPoint, city);
    const ePoint = await getPoint(EndGeo, endPoint, city);

    console.log(sPoint, ePoint);

    const transit = new BMapGL.TransitRoute(map, {

      onSearchComplete: function(results){ //搜索完成后进行赋值
        if (transit.getStatus() === BMAP_STATUS_SUCCESS) {
          let plans = [];
          let starts = [];
          let ends = [];
          for(let i = 0; i < results.getNumPlans(); i++) {
            plans.push(results.getPlan(i));
            starts.push(results.getStart());
            ends.push(results.getEnd());
          }
          if(plans.length > 0)
          {
            showPlanOnMap(plans[0],map,starts[0],ends[0]);
          }
          callback && callback(plans, map ,starts ,ends); // 将方案返回给回调函数
        }
      },
    },

```

```

});
transit.setPageCapacity(5); // 设置每页返回方案个数为5

transit.search(sPoint, ePoint);

map.addEventListener('click', function (e) {

    getAddressFromPoint(e.latlng, (adr)=>{
        console.log(adr)
        AddressEmitter.emit('address-found', adr);
    })
});

} catch (error) {
    alert(error.message);
}
}

```

该段代码实现了导航部分的基本逻辑。

## 5. 火车票查询功能细节

- 如何获取12306网站的数据：12306网站提供一些官方接口，能够获得其信息。其中，`station_names`数组存储了全国所有的火车站。12306网站不提供官方文档，因此需要自行探究数据格式问题。

```

function fetchTrainData(date, start, end) {
  return new Promise((resolve, reject) => {
    let data = {
      train_date: date,
      from_station: resultMap[start],
      to_station: resultMap[end],
      purpose_codes: 'ADULT',
      data: []
    };

    const vor_url = `/otn/leftTicketPrice/query?leftTicketDTO.train_date=${data.train_date}`;
    axios.get(vor_url);

    axios.get('/otn/leftTicket/query', {
      params: {
        'leftTicketDTO.train_date': data.train_date,
        'leftTicketDTO.from_station': data.from_station,
        'leftTicketDTO.to_station': data.to_station,
        'purpose_codes': 'ADULT'
      }
    })
    .then(response => {
      data = response.data;
      console.log(data);
      resolve(data);
    })
    .catch(error => {
      console.error("请求出错:", error);
      reject(error);
    });

  });
}

```

这是实现12306数据获取的简单逻辑。

## 6. 天气组件的工作机制

本组件旨在提供天气信息，以下是主要的实现细节和使用的技术。

## 6.1 API调用方法

天气信息来自外部的API服务。为了进行HTTP请求，我们使用了 `axios` 库。

## 6.2 查询实时天气

- **API URL:** <https://restapi.amap.com/v3/weather/weatherInfo>
- **实现细节:**
  - i. 从外部JSON文件加载城市数据。
  - ii. 根据提供的城市名查找相应的 `adcode`。
  - iii. 使用找到的 `adcode` 和API key向天气API发出请求。
  - iv. 解析API的响应并提取所需的天气数据。

```
import axios from 'axios';

const WEATHER_API_URL = 'https://restapi.amap.com/v3/weather/weatherInfo';

const getWeather = async (cityName) => {
  const citiesData = require('../assets/cityData.json');
  const cities = citiesData.sheet1;
  // ...(其余代码略)
}

export default getWeather;
```

## 6.3 查询每周天气

- **API URL:** <https://apis.tianapi.com/tianqi/index>
- **实现细节:**
  1. 使用`axios`库向API发出GET请求。
  2. 将API的响应保存到`weeklyWeather`属性中。

```
getWeeklyWeather(){
  axios.get('https://apis.tianapi.com/tianqi/index', {
    params: {
      key: 'api-key',
      city: this.cityname,
      type: 7,
    }
  })
  .then(response => {
    this.weeklyWeather = response.data.result.list;
  })
  .catch(err => {
    console.error(err);
  });
}
```

## 7. 景区组件的实现策略

本组件旨在结合维基百科和特定的景点搜索API为用户提供相关搜索结果。以下是主要的实现细节：

### 7.1 结构概述

- 组件包含两个主要部分：维基百科搜索结果和景点搜索结果。
- 搜索结果以列表形式展示，其中每个结果包含链接、简短摘要和缩略图（如果可用）。

### 7.2 API调用方法

使用 `axios` 库进行HTTP请求来获取所需数据。

#### 7.2.1 维基百科API

- **API URL**： `https://zh.wikipedia.org/w/api.php`
- **主要功能**：根据用户的查询字符串返回维基百科的搜索结果。
- **参数**：包括搜索的字符串、请求的格式（JSON）等。

#### 7.2.2 景点搜索API

- **API URL**： `https://apis.tianapi.com/scenic/index`
- **主要功能**：返回与查询字符串相关的景点信息。
- **参数**：包括API密钥、查询字符串以及请求的结果数量。

## 7.3 代码实现

```
async search() {
  alert('速速科学上网!');
  if (this.query) {
    try {
      const response = await axios.get('https://zh.wikipedia.org/w/api.php', {
        params: {
          format: 'json',
          action: 'query',
          generator: 'search',
          gsrnamespace: 0,
          gsrlimit: 10,
          prop: 'pageimages|extracts',
          pilimit: 'max',
          exintro: true,
          explaintext: true,
          exsentences: 1,
          exlimit: 'max',
          origin: '*',
          gsrsearch: this.query
        }
      });
      this.pages = Object.values(response.data.query.pages);
    } catch (error) {
      console.error('Error:', error);
    }
  }
},
async TouristSearch() {
  axios.get('https://apis.tianapi.com/scenic/index', {
    params: {
      key: 'api-key',
      word: this.query,
      num: 5,
    }
  })
  .then(response => {
    console.log(response.data);
    this.SearchData = response.data.result.list;
  })
  .catch(err => {
    console.error(err);
  })
}
```

```
});  
},
```

## 外部信息

本项目调用的API及参考资料如下：

1. 百度地图API: <https://lbsyun.baidu.com/index.php?title=jspopularGL>
2. 讯飞星火API: <https://www.xfyun.cn/doc/spark/Web.html>
3. openAI API: <https://platform.openai.com/docs/libraries>
4. 高德地图天气API: <https://lbs.amap.com/api/webservice/guide/api/weatherinfo>
5. 维基百科API: [https://www.mediawiki.org/wiki/API:Main\\_page/zh](https://www.mediawiki.org/wiki/API:Main_page/zh)
6. 12306网站: <https://www.12306.cn/index/>
7. 七日天气预报API: <https://www.tianapi.com/apiview/72>
8. 景点查询API: <https://www.tianapi.com/apiview/93>

## 挑战与解决方案

1. **如何进行API选择**：讯飞星火的API识别能力有限，容易出错且不好处理，所以目前暂时采用关键词识别的模式。
2. **如何实现信息的传入？**：解决方案：使用正则表达式提取信息，随后传入信息。
3. **如何实现API之间的数据交互？**：解决方案：可以使用多种方式，如用户决定交互时机，发送信号等等。