西安电子科技大学
XIDIAN UNIVERSITY

B 级达标测试实验报告

雾霾探测系统设计实验

20XX 年 XX 月 XX 日

姓名	学号	学院	任务分工	贡献度	签名

指导教师评语：

成　绩

测试教师：

_____年____月____日

实验报告内容基本要求及参考格式

一 问题描述

二 方案设计

三 数据获取

四 结果展示及分析

五 心得与体会

一　问题描述

随着雾霾问题日益严重，出行前获取准确的空气质量信息对健康防护至关重要。本实验旨在开发一款手机端雾霾探测系统，通过整合地理位置、实时天气和空气质量数据，为用户提供可视化的雾霾监测服务。系统需解决以下问题：

- 精准定位并展示城市信息
- 动态显示天气详情及空气质量指数
- 支持温湿度折线图可视化
- 适配不同手机分辨率

二　方案设计

2.1　系统架构 - 前后端分离的微服务架构

- 前端：Vue3 + ElementUI + ECharts
  - 实现响应式布局（HTML5 适配不同分辨率）
  - 动态数据可视化（温湿度折线图）
  - 地理位置自动获取与手动搜索双模式
- 后端：Gin + Gorm + MySQL
  - RESTful API 设计
  - 和风天气 API 数据聚合（由于百度地图 API 部分功能收费，我们采用和风天气 API）
  - 查询记录持久化存储
- 部署：Docker 容器化

2.2　关键技术实现

模块	**技术方案**
定位服务	浏览器 Geolocation API + 和风城市搜索 API
天气模块	集成和风天气实时/预报数据接口
空气质量	调用 AQI 指数接口，实现污染等级颜色映射
数据缓存	MySQL 定时存储查询记录（包含经纬度、城市、天气 JSON 数据）
异常处理	自定义错误码体系（5 类核心错误类型）
数据可视化	ECharts 动态渲染温湿度曲线（支持横向滚动功能）

2.3 核心接口设计

2.3.1 根据前端定位查询天气

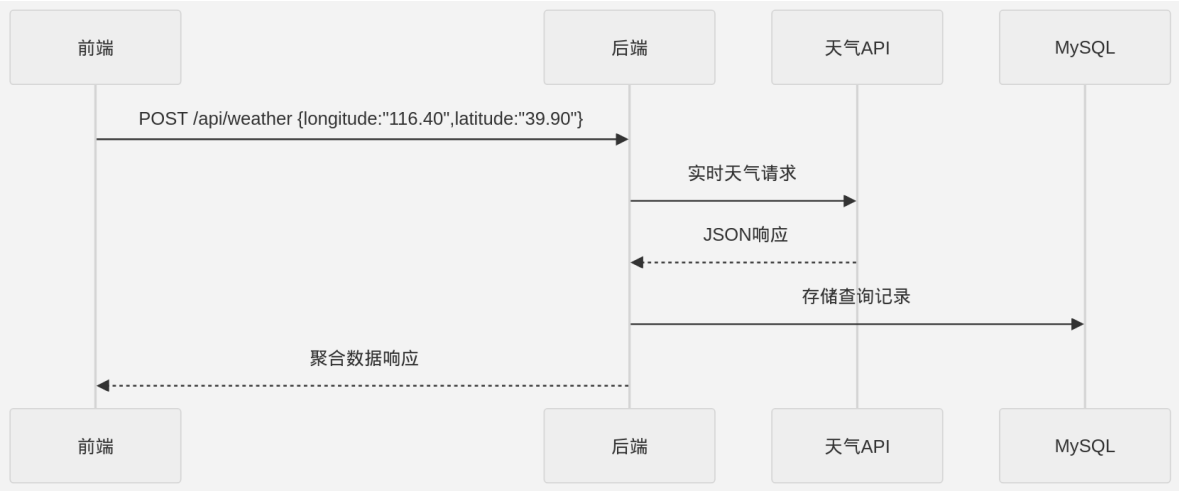

图 1: 根据前端定位查询天气

2.3.2 根据地名搜索查询天气

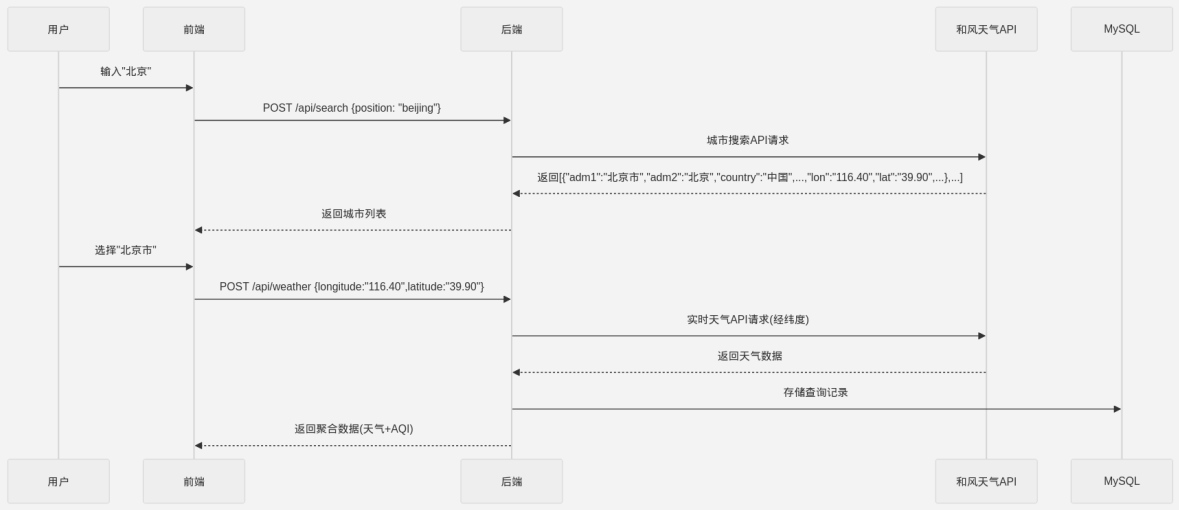

图 2: 根据地名搜索查询天气

三 数据获取

3.1 数据源架构 - 和风天气 API

- 城市定位 API（经纬度 → 行政区域）
- 实时天气 API（温度/湿度/天气现象）
- 空气质量 API（AQI/健康建议）

- 天气预报 API（3 日预测）
- 逐小时预报 API（24 小时数据）

3.2 关键数据处理

3.2.1 天气数据获取

```
func QueryWeatherByLonLat(c *gin.Context) {
  // 解析请求坐标
  var position *request.QueryWeatherLonLatRequest
  if err := c.ShouldBind(&position); err != nil {
    ResponseFail(c, http.StatusBadRequest, constant.DataParseError, err.Error())
    return
  }

  // 将 Longitude 和 Latitude 转换为 float64 类型
  longitude, err := strconv.ParseFloat(position.Longitude, 64)
  if err != nil {
    ResponseFail(c, http.StatusBadRequest, constant.InvalidParameter, "经度参数格式错误")
    return
  }
  latitude, err := strconv.ParseFloat(position.Latitude, 64)
  if err != nil {
    ResponseFail(c, http.StatusBadRequest, constant.InvalidParameter, "纬度参数格式错误")
    return
  }

  // 对经纬度进行截断并格式化为两位小数
  position.Longitude = strconv.FormatFloat(math.Trunc(longitude*100)/100, 'f', 2, 64)
  position.Latitude = strconv.FormatFloat(math.Trunc(latitude*100)/100, 'f', 2, 64)
  // 向和风天气发起调用
  requestHandler := NewAPIHandler()
  // 位置信息
  pos, err := requestHandler.QueryForPositionWithLonLat(position)
  if err != nil {
    ResponseFail(c, http.StatusInternalServerError, constant.CannotQueryPositionInfo, err.Error())
    return
  }
  // 天气信息
  weather, err := requestHandler.QueryForNowWeather(position)
  if err != nil {
    ResponseFail(c, http.StatusInternalServerError, constant.CannotQueryWeatherInfo, err.Error())
    return
  }
  // 空气质量
  airQuality, err := requestHandler.QueryAirQuality(position)
  if err != nil {
    ResponseFail(c, http.StatusInternalServerError, constant.CannotQueryAirQualityInfo, err.Error
    ())
    return
  }
  // 未来天气
  nextWeather, err := requestHandler.QueryNextWeather(position)
  if err != nil {
    ResponseFail(c, http.StatusInternalServerError, constant.CannotQueryNextWeatherInfo, err.Error
    ())
```

```
    return
  }
  // 逐小时天气
  hourlyWeather, err := requestHandler.QueryHourlyWeather(position)
  if err != nil {
    ResponseFail(c, http.StatusInternalServerError, constant.CannotQueryHourlyWeatherInfo, err.
    Error())
    return
  }
  city, ok := pos["name"].(string)
  if !ok {
    ResponseFail(c, http.StatusInternalServerError, constant.CannotQueryPositionInfo, "无法查询到
    城市名称")
    return
  }
  // 将 weather 转换为 JSON 字符串
  weatherJSON, err := json.Marshal(weather)
  if err != nil {
    ResponseFail(c, http.StatusInternalServerError, constant.CannotQueryWeatherInfo, "无法转换天气
    信息为 JSON 字符串")
    return
  }
  var queryRecord *model.QueryRecord
  queryRecord = &model.QueryRecord{
    City:        city,
    Longitude:   position.Longitude,
    Latitude:    position.Latitude,
    WeatherInfo: string(weatherJSON),
    CreateTime:  time.Now(),
    Type:        1,
    Deleted:     0,
  }
  if err := config.DataBase.Create(queryRecord).Error; err != nil {
    ResponseFail(c, http.StatusInternalServerError, constant.DataBaseSaveError, err.Error())
    return
  }
  data := map[string]interface{}{
    "pos":            pos,
    "weather":        weather,
    "air_quality":    airQuality,
    "next_weather":   nextWeather,
    "hourly_weather": hourlyWeather,
  }
  ResponseSuccessWithData(c, data)
}
```

1: 天气数据获取

3.2.2 位置数据获取

```
func SearchPositionWithName(c *gin.Context) {
  //  拿到模糊查询的地名
  var posName *request.QueryPositionRequest
  if err := c.ShouldBind(&posName); err != nil {
    ResponseFail(c, http.StatusBadRequest, constant.DataParseError, err.Error())
    return
  }
```

```
  if posName.Position == "" {
    ResponseFail(c, http.StatusBadRequest, constant.DataParseError, "请输入查询地名")
    return
  }
  // 调用api
  requestHandler := NewAPIHandler()
  posList, err := requestHandler.SearchPosition(posName)
  if err != nil {
    ResponseFail(c, http.StatusInternalServerError, constant.CannotQueryPositionInfo, err.Error())
    return
  }
  data := map[string]interface{}{
    "pos_list": posList,
  }
  ResponseSuccessWithData(c, data)
}
```

2: 位置数据获取

3.3 数据库设计

```
CREATE TABLE query_records (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    city VARCHAR(64) NOT NULL COMMENT '城市名称',
    longitude VARCHAR(32) NOT NULL COMMENT '经度',
    latitude VARCHAR(32) NOT NULL COMMENT '纬度',
    weather_info VARCHAR(1024) NOT NULL COMMENT '天气JSON',
    type TINYINT NOT NULL COMMENT '查询类型',
    deleted TINYINT DEFAULT 0 COMMENT '逻辑删除'
);
```

3: 数据库设计

四 结果展示及分析

4.1 系统界面

- Header 区
  - 城市定位（自动获取 + 手动搜索）
- Body 区
  - 实时天气卡片（温度/天气图标）
  - AQI 健康提示（颜色预警）
  - 3 日预报（温度区间条）
  - 温湿度曲线（可横向滚动）

4.2 典型数据展示

4.2.1 城市查询返回结果


```
{
    "data": {
        "pos_list": [
            {
                "adm1": "北京市",
                "adm2": "北京",
                "country": "中国",
                "fxLink": "https://www.qweather.com/weather/beijing-101010100.html",
                "id": "101010100",
                "isDst": "0",
                "lat": "39.90499",
                "lon": "116.40529",
                "name": "北京",
                "rank": "10",
                "type": "city",
                "tz": "Asia/Shanghai",
                "utcOffset": "+08:00"
            },
            ...
        ]
    },
    "massage": "success"
}
```


4: 城市查询返回结果

4.2.2 天气查询返回结果


```
{
    "data": {
        "air_quality": {
            "aqi": 35,
            "aqiDisplay": "35",
            "category": "优",
            "code": "cn-mee",
            "color": {
                "alpha": 1,
                "blue": 0,
                "green": 228,
                "red": 0
            },
            "health": {
                "advice": {
                    "generalPopulation": "各类人群可正常活动。",
                    "sensitivePopulation": "各类人群可正常活动。"
                },
                "effect": "空气质量令人满意，基本无空气污染。"
            },
            "level": "1",
            "name": "AQI (CN)",
            "primaryPollutant": null
        },
        "hourly_weather": [
```

```
            {
                "cloud": "94",
                "dew": "-2",
                "fxTime": "2025-05-01T13:00+00:00",
                "humidity": "23",
                "icon": "152",
                "precip": "0.0",
                "pressure": "997",
                "temp": "20",
                "text": "少云",
                "wind360": "87",
                "windDir": "东风",
                "windScale": "1",
                "windSpeed": "5"
            },
            ...
        ]
        "next_weather": [
            {
                "cloud": "14",
                "fxDate": "2025-05-01",
                "humidity": "22",
                "iconDay": "100",
                "iconNight": "151",
                "moonPhase": "蛾眉月",
                "moonPhaseIcon": "801",
                "moonrise": "07:23",
                "moonset": "23:43",
                "precip": "0.0",
                "pressure": "1000",
                "sunrise": "05:16",
                "sunset": "19:09",
                "tempMax": "26",
                "tempMin": "13",
                "textDay": "晴",
                "textNight": "多云",
                "uvIndex": "9",
                "vis": "25",
                "wind360Day": "315",
                "wind360Night": "90",
                "windDirDay": "西北风",
                "windDirNight": "东风",
                "windScaleDay": "1-3",
                "windScaleNight": "1-3",
                "windSpeedDay": "3",
                "windSpeedNight": "3"
            },
            ...
        ]
        "pos": {
            "adm1": "北京市",
            "adm2": "北京",
            "country": "中国",
            "fxLink": "https://www.qweather.com/weather/haidian-101010200.html",
            "id": "101010200",
            "isDst": "0",
            "lat": "39.95607",
            "lon": "116.31032",
```

```
            "name": "海淀",
            "rank": "15",
            "type": "city",
            "tz": "Asia/Shanghai",
            "utcOffset": "+08:00"
        },
        "weather": {
            "cloud": "100",
            "dew": "-5",
            "feelsLike": "18",
            "humidity": "31",
            "icon": "104",
            "obsTime": "2025-05-01T20:48+08:00",
            "precip": "0.0",
            "pressure": "1003",
            "temp": "19",
            "text": "阴",
            "vis": "30",
            "wind360": "0",
            "windDir": "北风",
            "windScale": "0",
            "windSpeed": "0"
        }
    },
    "massage": "success"
}
```


5: 天气查询返回结果

4.3 界面展示

4.3.1 电脑端

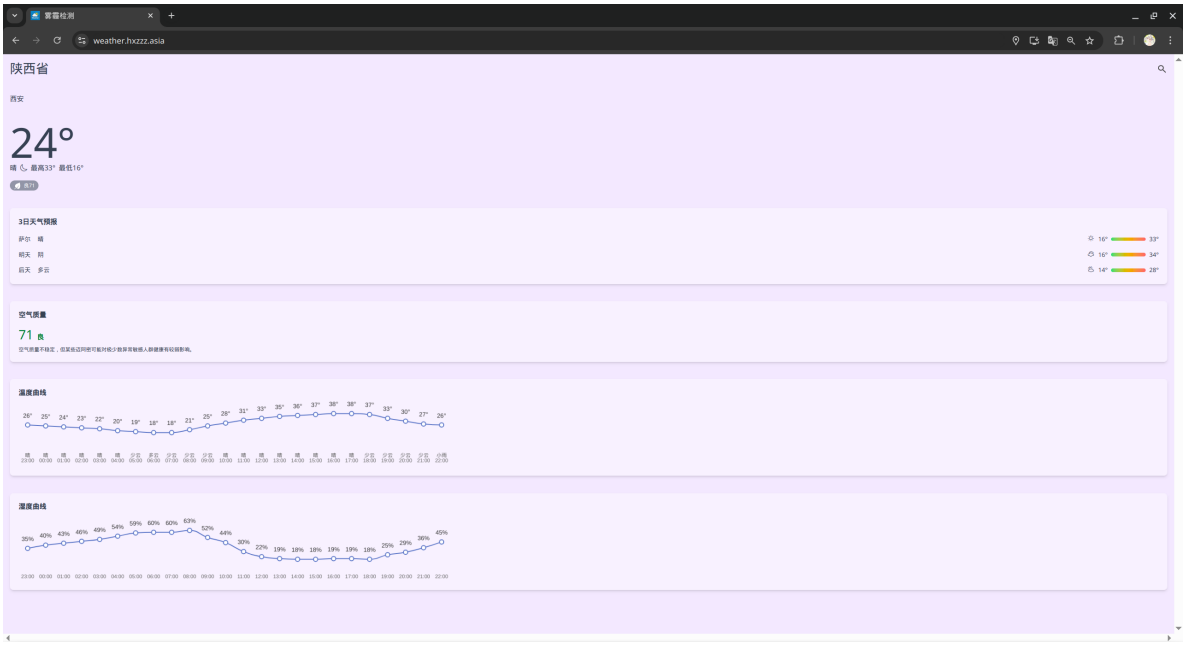

图 3: 电脑端-浏览器全屏


霧霾检测
weather.hxzzz.asia
陕西省
西安
24°
晴 最高33° 最低16°
良71
3日天气预报
萨尔 晴 16° 33°
明天 阴 16° 34°
后天 多云 14° 28°
空气质量
71 良
空气质量不稳定，但某些迈阿密可能对极少数异常敏感人群健康有较弱影响。
温度曲线
26° 25° 24° 23° 22° 20° 19° 18° 18° 21° 25° 28° 31° 33° 35° 36° 37°
晴 23:00 晴 00:00 晴 01:00 晴 02:00 晴 03:00 晴 04:00 少云 05:00 多云 06:00 少云 07:00 少云 08:00 少云 09:00 晴 10:00 晴 11:00 晴 12:00 晴 13:00 晴 14:00 晴 15:00
湿度曲线
35% 40% 43% 46% 49% 54% 59% 60% 60% 63% 52% 44% 30% 22% 19% 18% 18%
23:00 00:00 01:00 02:00 03:00 04:00 05:00 06:00 07:00 08:00 09:00 10:00 11:00 12:00 13:00 14:00 15:00

图 4: 电脑端-浏览器拉伸

4.3.2 手机端

图 5: 手机端-截长屏

五　心得与体会

5.1　技术收获

- 掌握了前后端分离开发模式的实际应用
- 深入理解了 API 对接和数据处理方法
- 深入理解 RESTful API 设计规范与错误处理机制

5.2　难点突破

- 数据一致性：采用最终一致性策略处理第三方 API 延迟
- 移动端适配：使用 rem 布局 +CSS 媒体查询实现响应式
- 中文城市名搜索兼容性：集成拼音转换库

5.3　实验总结

本次实验基于 Vue+Gin 全栈技术实现了一个雾霾探测系统，通过精准的地理位置服务和和风天气 API 接口，实现了城市定位、天气信息查询、空气质量监测及数据可视化等功能。该系统采用前后端分离架构，后端对接第三方 API 进行数据聚合与存储，前端通过响应式设计适配移动端，并运用 ECharts 实现温湿度数据的动态图表展示。实验过程中深入实践了 RESTful API 设计、异常处理优化和数据缓存策略，最终构建了一个具有实用价值的天气服务应用，既满足了用户获取实时天气与空气质量的需求，也为后续扩展预警通知、历史数据分析等功能奠定了良好基础。