新冠疫情实时追踪系统设计报告

1 作品介绍

这是一个疫情数据可视化系统，名称为新冠疫情实时追踪系统。系统通过五大内容板块，包括地图数据、境外输入、疫情趋势、现有构成、各省详情多方面展示国内疫情数据，供用户了更快捷方便地了解最新的疫情情况。

本系统通过 Python 对已公开数据进行抓取，通过 WebService.cs 文件连接并读取数据库表中数据并封装成 json 数据返回，页面通过 ajax 获取到 json 数据，并基于 Echart 将数据可视化为地图、柱状图、折线图等图表。

2 系统设计

2.1 系统功能设计

新冠疫情实时追踪系统系统主要包括地图数据、境外输入、疫情趋势、现有构成、各省详情、用户中心六大模块，系统功能结构如图 2-1 所示。

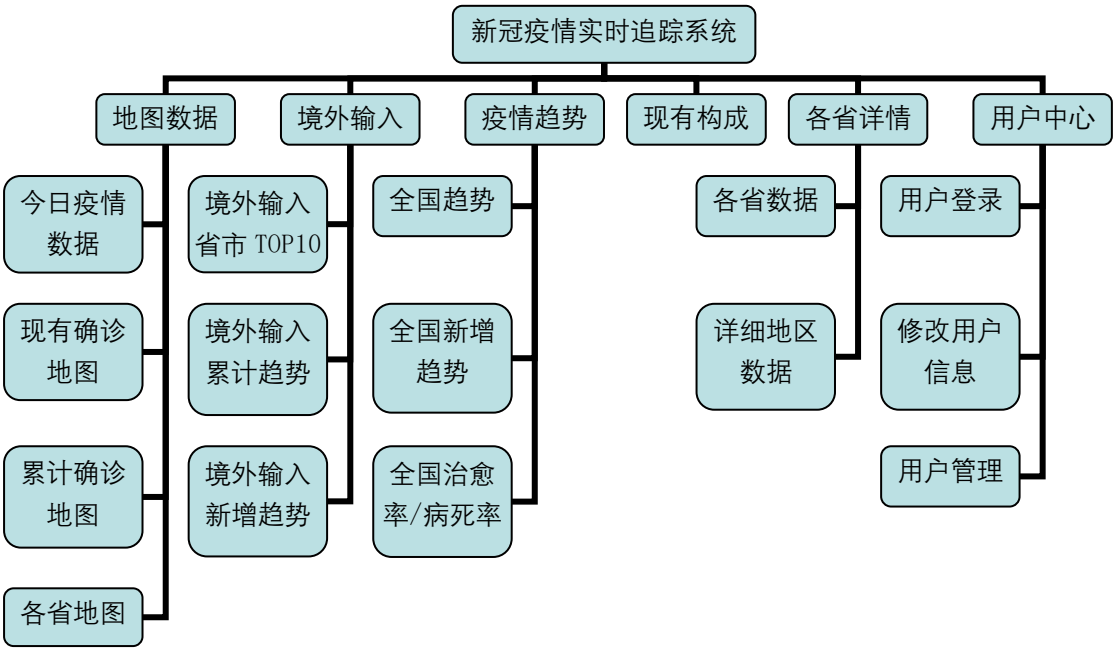
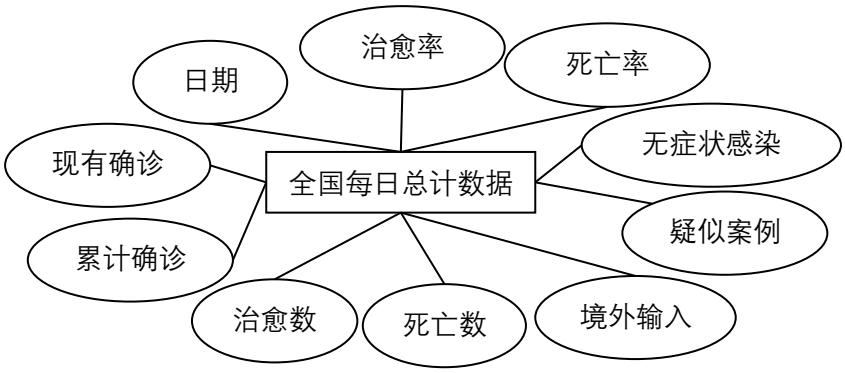


图 2-1 系统功能结构图

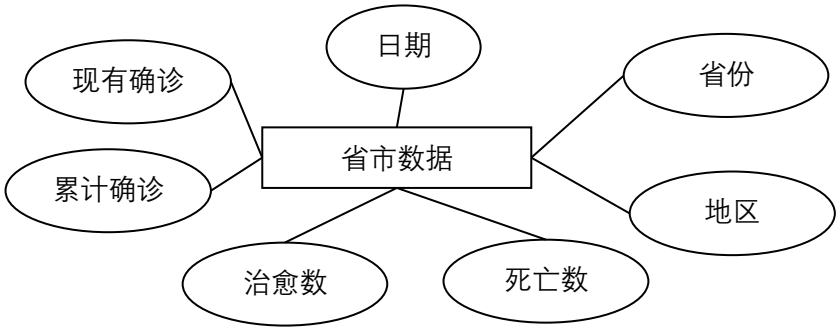
2.2 数据库设计

2.2.1 逻辑设计

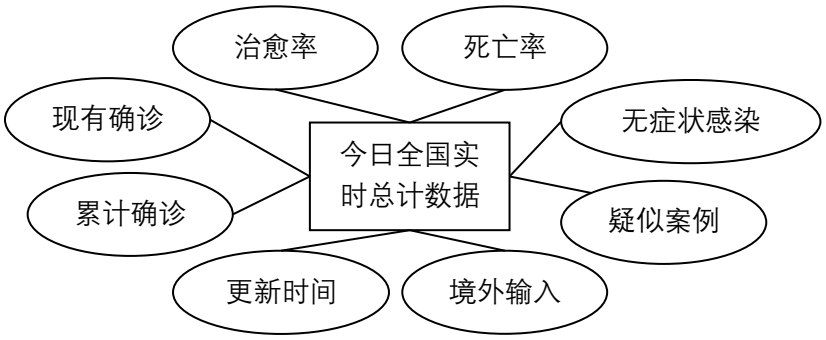
根据系统要求和功能结果，建立疫情数据可视化系统数据库的概念模型，图 2-2 为用 E-R 图表示的疫情数据可视化系统数据库的概念模型。



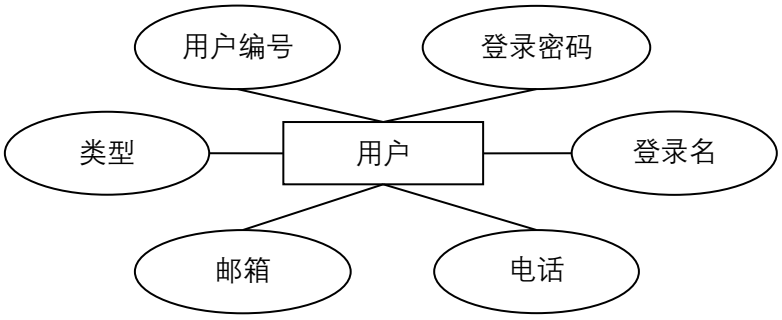
(a)“全国每日总计数据”实体图



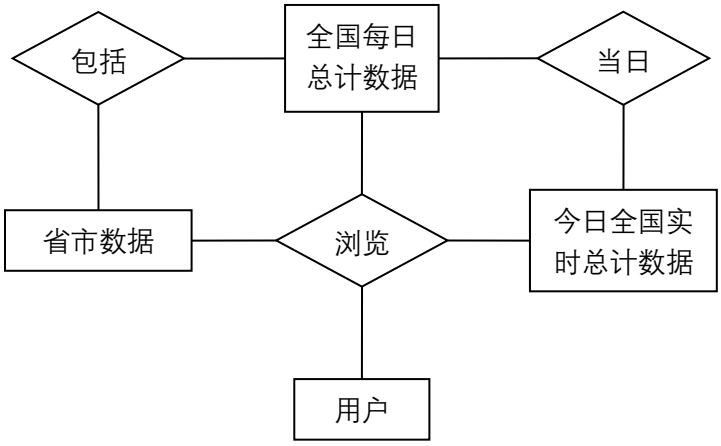
(b)“省市数据”实体图



(c)“今日全国实时总计数据”实体图



(d)“用户”实体图



（e）各实体图之间的关系

图 2-2 疫情数据可视化系统数据库的 E-R 图

2.2.2 表结构设计

疫情数据可视化系统的数据库管理平台采用 SQL Server，在 SQL Server 数据库中创建一个名为 nCoV19 的数据库，在该数据库中创建 4 张表。下面详细介绍这 4 张表的结构及用途。

（1）全国每日总计数据表

全国每日总计数据表用来记录全国每日疫情数据的总体情况，包括日期、感染人数、死亡人数、治愈人数、境外输入等信息。其表结构如表 2-1 所示。

表 2-1 全国每日总计数据表（China_Day_Summary）的表结构

字段名	类型（长度）	是否允许为空	说明
date	nvarchar(255)	否	日期
confirm	float	是	累计感染人数
suspect	float	是	疑似感染人数
dead	float	是	死亡人数
heal	float	是	治愈人数
nowConfirm	float	是	现有感染人数
noInfect	float	是	无症状感染人数
importedCase	float	是	境外输入
deadRate	nvarchar(255)	是	死亡率
healRate	nvarchar(255)	是	治愈率

（2）省市数据表

省市数据表用来记录各省份地区每日的疫情数据的情况，包括省份地区名称、日期、感染人数、死亡人数、治愈人数等信息。其表结构如表 2-2 所示。

表 2-2 省市数据表（China_Data）的表结构

字段名	类型（长度）	是否允许为空	说明
province	nvarchar(255)	是	省份名称
city	nvarchar(255)	否	城市/地区名称
confirm_num	float	是	累计感染人数
nowconfirm_num	float	是	现有感染人数
heal_num	float	是	治愈人数
dead_num	float	是	死亡人数
Date	datetime	否	日期

（3）今日全国实时总计数据表

今日全国实时总计数据表用来记录当日全国疫情数据的总体情况，包括更新时的状态（新增数据/实时数据）、感染人数、死亡人数、治愈人数等信息。其表结构如表 2-3 所示。

表 2-3 今日全国实时总计数据表（China_Today_Add）的表结构

字段名	类型（长度）	是否允许为空	说明
state	nvarchar(255)	否	更新状态[实时/新增]
confirm	float	是	累计感染人数
suspect	float	是	疑似感染人数
dead	float	是	死亡人数
heal	float	是	治愈人数
nowConfirm	float	是	现有感染人数
noInfect	float	是	无症状感染人数
importedCase	float	是	境外输入

（4）用户信息表

用户信息表记录用户的登录名称和密码等信息。其表结构如表 2-4 所示。

表 2-4 用户信息表（User_Info）的表结构

字段名	类型（长度）	是否允许为空	说明
userId	int	否	用户编号
userName	varchar(50)	是	用户名称
userPwd	varchar(50)	是	登录密码
isAdm	bit	是	是否为管理员
tel	varchar(50)	是	电话
email	varchar(50)	是	邮箱

3 框架结构设计

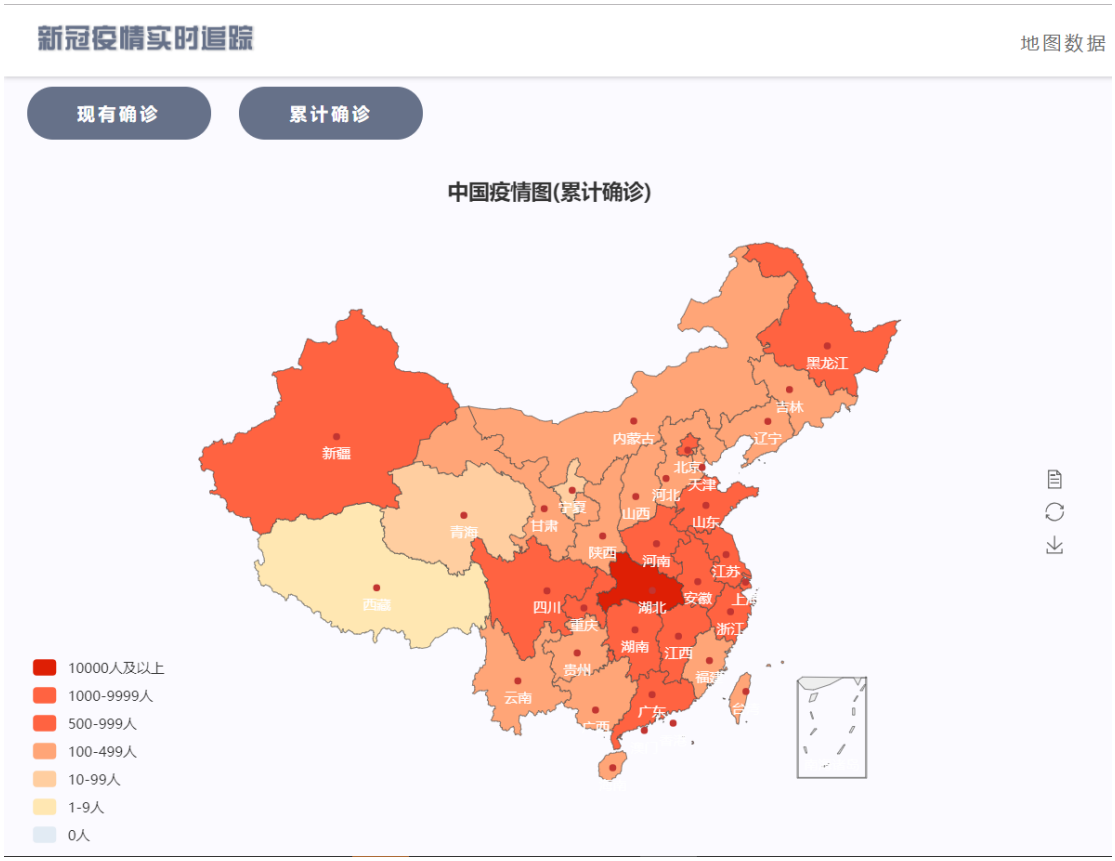
疫情数据可视化系统中使用了 MVC 框架，有利于各层级之间的功能分离。Model 层建立了名为 Series 的类，用于存放 echart 渲染时图表时所需要的参数，例如配置项图表类型、图表数据、横纵轴等。

4 系统实现

4.1 疫情地图模块

疫情地图页面如图 4-1 所示。主要通过地图用于展示全国各省市累计/现有确诊数状况。地图以七种不同颜色划分数据范围，用户可以点击数据范围选中或取消数据在地图上的显示。用户将鼠标放在地图的不同区域上，可以显示当前省市的确诊数情况。用户也点击工具栏查看地图上的所有数据情况以及刷新或下载该地图图片。

（a）“现已确诊”疫情地图



（b）“现已确诊”疫情地图

图 4-1 疫情地图页面

页面加载时，后台通过 WebService.cs 中的方法连接到本地数据库，并对数据库进行查询等相关操作，代码如下所示。

```
    [WebMethod]
    public string getdata()
                                            {
     List<string> categoryList = new List<string>();
     List<double> confirmList = new List<double>();
     List<double> nowconfirmList = new List<double>();
     List<string> legendList = new List<string>();
     //考虑到图表的series数据为一个对象数组 这里额外定义一个series的类
     List<Series> seriesList = new List<Series>();
     //设置legend数组
     legendList.Add("确诊人数");
        //填写第一个Series，定义一个Series对象
        Series seriesObj = new Series();
        seriesObj.name = "确诊人数";
        seriesObj.data = new List<double>();
        SqlConnection con = connect();
        SqlCommand cmd = new SqlCommand();
        cmd.Connection = con;
        cmd.CommandText = "select province,sum(confirm_num),sum(nowconfirm_num) from China_Data
where Date=(select MAX(Date) from China_Data) group by province";
        SqlDataReader sdr = null;
        try
        {
            con.Open();
            sdr = cmd.ExecuteReader();
            while (sdr.Read())
            {
                categoryList.Add(sdr.GetValue(0).ToString());
                confirmList.Add(Convert.ToDouble(sdr.GetValue(1)));
                nowconfirmList.Add(Convert.ToDouble(sdr.GetValue(2)));
                seriesObj.data.Add(Convert.ToDouble(sdr.GetValue(1)));
            }
            seriesList.Add(seriesObj);
        }
        catch (Exception ex)
        {errorMsg = ex.Message; }
        finally
        {con.Close();}
        //最后调用相关函数将List转换为Json
        var newObj = new
        {
            category = categoryList,
            confirm = confirmList,
            nowconfirm = nowconfirmList,
            series = seriesList,
            legend = legendList
        };
        JavaScriptSerializer serializer = new JavaScriptSerializer();
        string result = serializer.Serialize(newObj);
        return result;
       }
```

页面中 ajax 通过脚本调用 web 服务获得后台数据，并绑定到 echart 图表配置项 option 的属性 series 中，代码如下所示。

```
//通过Ajax获取数据
                $.ajax({
                    type: "POST",
                    async: false,
                    //contentType: 'application/json; charset=utf-8',
                    //url: "../WebService.asmx/getdataechart",
                    //dataType: "json", //返回数据形式为json
                    contentType: 'application/x-www-form-urlencoded',
                    url: "/WebService.asmx/getdata",
                    dataType: "xml",
                    success: function (result) {
                        console.log(result);
                        var res = result.getElementsByTagName("string")[0].innerHTML;
                        var obj = JSON.parse(res);
                        var newArr = [];
                        if (res) {
                            var province = obj.category;
                            var data = obj.nowconfirm;
                            for (var i = 0; i < province.length; i++) {
                                var json = {
                                    name: province[i],
                                    value: data[i]
                                }
                                newArr.push(json)
                            }
                            myChart.setOption({
                                series: [
                                    {
                                        name: '确诊数',
                                        type: 'map',
                                        mapType: 'china',
                                        selectedMode: 'single',
                                        roam: false,
                                        label: {
                                            show: true,
                                            color: 'rgb(0, 0, 0)'
                                        },
                                        data: newArr
                                    }
                                ]
                            });

                        }
                    },
                    error: function (XMLHttpRequest, textStatus, errorThrown) {
                        alert(XMLHttpRequest.responseText);
                        alert(XMLHttpRequest.status);
                        alert(XMLHttpRequest.readyState);
                        alert(textStatus);
                    }
                });
```

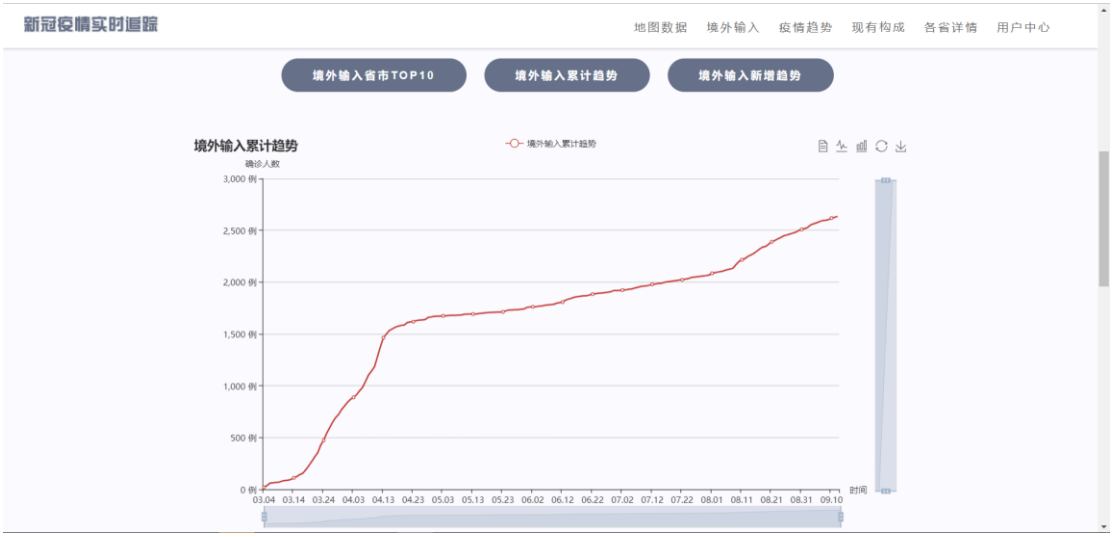
同时设置好图表中 option 的其他属性，通过 echarts.init 渲染图表和 setOption() 使用指定的配置项和数据显示图表，代码如下所示。

```
updatePanel_now();
function updatePanel_now() {
    var myChart = echarts.init(document.getElementById('main_now'));
    myChart.on('click', function (params) {
        var city = params.name;
        if (city == '北京') {
            window.location.replace("map_Beijing.aspx");
        }
        else if (city == '广东') {
            window.location.replace("map_Guangdong.aspx");
        }
    });
    option = {
        title: {
            text: '中国疫情图(现有确诊)',
            left: 'center'
        },tooltip: {
            trigger: 'item'
        },legend: {
            orient: 'vertical',
            left: 'left',
            data: ['中国疫情图']
        },visualMap: {
            type: 'piecewise',
            pieces: [
                { min: 10000, max: 10000000, label: '10000人及以上', color: '#de1f05' },
                { min: 1000, max: 9999, label: '1000-9999人', color: '#ff6341' },
                { min: 500, max: 999, label: '500-999人', color: '#ff6341' },
                { min: 100, max: 499, label: '100-499人', color: '#ffa577' },
                { min: 10, max: 99, label: '10-99人', color: '#ffcea0' },
                { min: 1, max: 9, label: '1-9人', color: '#ffe7b2' },
                { min: 0, max: 0, label: '0人', color: '#e2ebf4' },
            ],
            color: ['#E0022B', '#E09107', '#A3E00B'],
            x: "left",
            y: "bottom",
            splitList: [
                { start: 0, end: 49 },
                { start: 50, end: 99 },
                { start: 100, end: 149 },
                { start: 150, end: 199 },
                { start: 200 }
            ],
        }, toolbox: {
            show: true,
            orient: 'vertical',
            left: 'right',
            top: 'center',
            feature: {
                mark: { show: true },
                dataView: { show: true, readOnly: false },
                restore: { show: true },
                saveAsImage: { show: true }
            }
        },roamController: {
            show: true,
            left: 'right',
            mapTypeControl: {
                'china': true
            }
        },series: [
            {
                name: '确诊数',
                type: 'map',
                mapType: 'china',
                roam: false,
                label: {
                    show: true,
                    color: 'rgb(0, 0, 0)'
                },
                data: []
            }
        ]
    };
    //使用指定的配置项和数据显示图表
    myChart.setOption(option);
```

4.2 境外输入模块

境外输入模块如图 4-2 所示。以柱状图或折线图的形式展示了境外输入数最高的十个省市的情况、累计趋势和新增趋势。用户使用鼠标滚轮或拉动 xy 轴上的区域范围可以查看感兴趣的数据范围。

（a）“境外输入省市 TOP10”页面



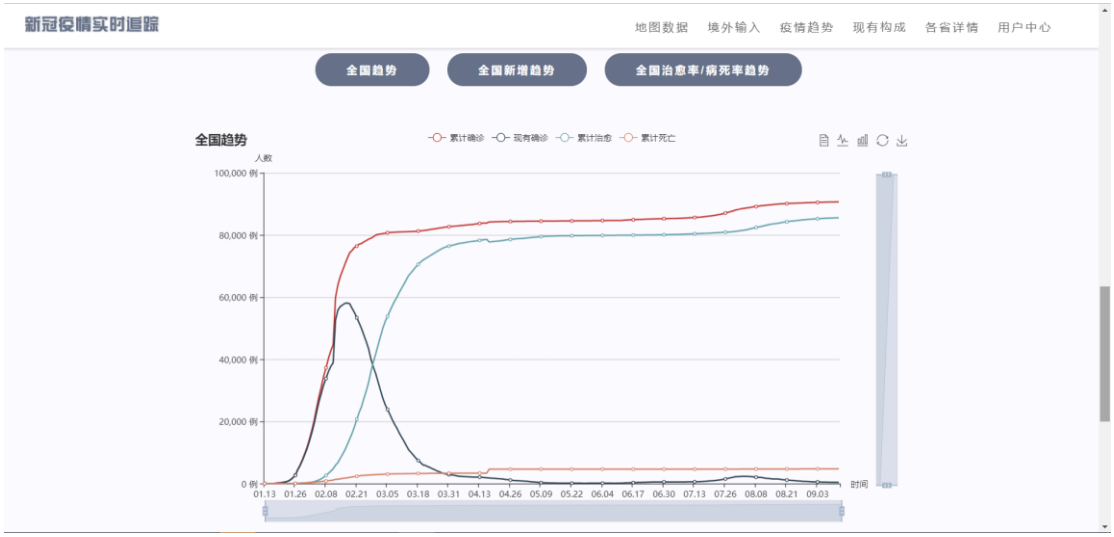
（b）“境外输入累计趋势”页面

（c）“境外输入新增趋势”页面

图 4-2 境外输入页面

4.3 疫情趋势模块

疫情趋势模块如图 4-3 所示。以折线图的形式展示了全国总体数据的趋势情况，如累计确诊趋势、累计治愈趋势、新增趋势等。用户使用鼠标滚轮或拉动 xy 轴上的区域范围可以查看感兴趣的数据范围。



（a）“全国趋势”页面

（b）“全国新增趋势”页面

（c）“全国治愈率/病死率趋势”页面

图 4-3 全国趋势页面

4.4 现有构成模块

现有构成模块如图 4-4 所示。以饼状图的形式展示了全国最新的病例构成情况，由港澳台、本土病例、境外输入病例构成。

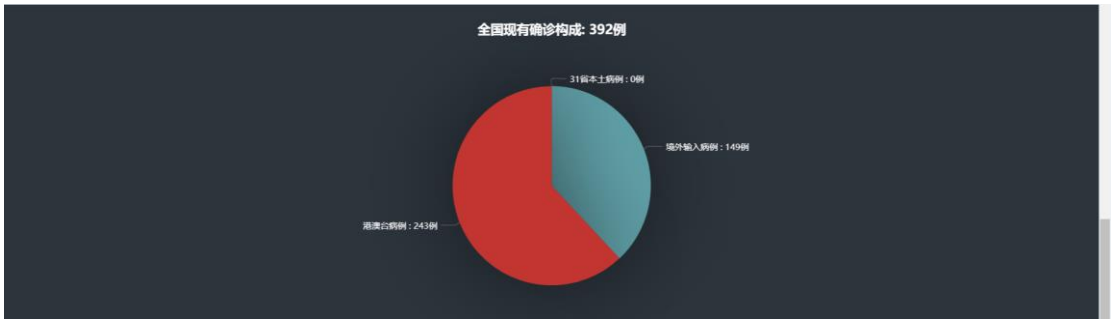


图 4-4 现有构成页面

境外输入、疫情趋势、现有构成等通过 echart 图表显示的模块的实现与上述疫情地图模块的实现方式类似，主要是使用指定图表的配置项的不同。

4.5 实时疫情状况模块

实时疫情状况模块如图 4-5 所示。页面显示系统最近的更新时间以及更新后的数据，包括累计确诊、累计死亡、累计治愈、现有确诊等数据，同时显示当前数据与前一日数据的比较情况。

图 4-5 实时疫情状况页面

页面加载时，后台通过连接数据库读取数据并将数据绑定到页面中。绑定更新时间的代码如下所示。

```
protected void bindUpdateTime()
{
    SqlConnection con = connect();
    SqlCommand cmd = new SqlCommand();
    cmd.Connection = con;
    cmd.CommandText = "select top 1 Date from China_Data order by Date desc ";
    try
    {
        con.Open();
        SqlDataReader sdr = cmd.ExecuteReader();
        while (sdr.Read())
        {
            update_time_txt.InnerText = update_time_txt.InnerText + sdr["Date"].ToString();
        }
        sdr.Close();
    }
    catch (Exception ex)
    {
        Response.Write("错误原因是：" + ex.Message);
    }
    finally
    {
        if (con.State == ConnectionState.Open) con.Close();
    }
}
```

绑定更新数据的代码如下所示。

```
    protected void bindTodayInfo()
    {
        SqlConnection con = connect();
        SqlCommand cmd = new SqlCommand();
        cmd.Connection = con;
        cmd.CommandText = "select confirm,dead,heal,nowConfirm,noInfect,importedCase from
China_Today_Add where state='add'";

        Label[] add_txt = new Label[] { today_confirm_add, today_dead_add, today_heal_add,
today_nowConfirm_add,today_noInfect_add,today_importedCase_add };
        string sym = "+ ";
        try
        {
            con.Open();
            SqlDataReader sdr = cmd.ExecuteReader();
            while (sdr.Read())
            {
                for(int i = 0; i < 6; i++)
                {
                    if (Convert.ToInt32(sdr.GetValue(i)) < 0) sym = "";
                    else sym = "+ ";
                    add_txt[i].Text= add_txt[i].Text+sym+ sdr.GetValue(i).ToString();
                }
            }
            sdr.Close();

            cmd.CommandText = "select confirm,dead,heal,nowConfirm,noInfect,importedCase from
China_Today_Add where state='now'";
            SqlDataReader sdr1 = cmd.ExecuteReader();
            while (sdr1.Read())          //循环读取结果集
            {
                today_confirm.InnerText = sdr1["confirm"].ToString().Trim();
                today_dead.InnerText = sdr1["dead"].ToString().Trim();
                today_heal.InnerText = sdr1["heal"].ToString().Trim();
                today_nowConfirm.InnerText = sdr1["nowConfirm"].ToString().Trim();
                today_noInfect.InnerText = sdr1["noInfect"].ToString().Trim();
                today_importedCase.InnerText = sdr1["importedCase"].ToString().Trim();
            }
            sdr1.Close();
        }
        catch (Exception ex)
        {
            Response.Write("错误原因是：" + ex.Message);
        }
        finally
        {
            if (con.State == ConnectionState.Open) con.Close();
        }
}
```

4.6 各省详情模块

各省详情模块如图 4-6 所示。页面以表格的形式显示全国各省的总体数据情况，点击具体省市的查看详情按钮，可得到该省市的具体地区的详细数据。

图 4-6 各省详情页面

点击“查看详情”事件代码如下所示。

```
    protected void GridView1_SelectedIndexChanged(object sender, EventArgs e)
    {
        string province_name = GridView1.Rows[GridView1.SelectedIndex].Cells[0].Text;
        Detail_DataBind(province_name);
}

    public void Detail_DataBind(string province_name)
    {
        div_total.Attributes.Add("class", "left");
        div_total.Attributes.Add("style", "margin-left:100px");
        div_details.Attributes.Add("class", "left");
        SqlConnection con = connect();
        string sql;

sql = "select city,nowconfirm_num,confirm_num,heal_num,dead_num from China_Data where Date=(select
Max(Date) from China_Data) and province='"+province_name+"'";
        SqlDataAdapter sda = new SqlDataAdapter(sql, con);
        DataTable dt = new DataTable();

        Try
        {
            sda.Fill(dt);
            foreach (DataRow dr in dt.Rows)
            {
                for (int i = 1; i < dt.Columns.Count; i++)
                {
                    if (Convert.ToInt32(dr[i].ToString())<0)
                    {
                        dr[i] = 0;
                    }

                }
            }

            GridView2.DataSource = dt;
            GridView2.DataBind();
        }
        catch (Exception ex)
        {
            Response.Write("错误原因是：" + ex.Message);
        }
        finally
        {
            if (con.State == ConnectionState.Open) con.Close();
        }
    }
```

4.7 登录模块

登录模块如下图 4-7 所示。用户登录模块根据用户类型，验证用户名和密码是否正确，若不正确则提示登录失败，退出系统；若正确将根据用户类型导航到相应页面。

图 4-7 登录模块

“确定”按钮的事件代码如下：

```
protected void Button1_Click(object sender, EventArgs e)
    {
        if (IsValid)
        {
            string userName = user.Text.Trim();    //用户名
            string password = pwd.Text.Trim();     //密码
            Session["entryType"] = RadioButtonList1.SelectedValue;    //记录登录用户的类型
            string cs = Session["entryType"].ToString();
            string sql;
            sql = "select count(*) from User_Info where userName=@name  and  userPwd=@pass";
            //}
            //调用用户名和密码验证函数
            if (checkPwd(sql, userName, password))     //验证成功
            {
                Session["userName"] = userName;  //记录登录用户名
                    Response.Redirect("user.aspx");
            }
            else    //验证失败
            {
                Response.Write("<script>alert('登录失败')</script>");
            }
      }
```

其中，checkPwd 方法为自定义的用户名和密码验证函数，其第一个参数为需要执行的 SQL 语句，第二个参数为用户名，第三个参数为密码。若用户名和密码验证成功则返回布尔值 True，否则返回布尔值 False。checkPwd 方法的实现代码如下。

```
public static bool checkPwd(string sql, string name, string pass)
    {
        string conStr = ConfigurationManager.ConnectionStrings["CnnString"].ConnectionString;
        SqlConnection con = new SqlConnection(conStr);          //创建数据库连接
        con.Open();          //打开数据库连接
        SqlCommand com = new SqlCommand(sql, con);          //创建SqlCommand对象
        com.Parameters.Add(new SqlParameter("name", SqlDbType.VarChar, 50));      //设置参数类型
        com.Parameters["name"].Value = name;          //设置参数值
        com.Parameters.Add(new SqlParameter("pass", SqlDbType.VarChar, 50));    //设置参数类型
        com.Parameters["pass"].Value = pass;          //设置参数值
        //判断验证用户名和密码是否正确，并返回布尔值
        if (Convert.ToInt32(com.ExecuteScalar()) > 0)
//返回指定用户名和密码的记录数大于0，此用户名和密码正确。
        {
            con.Close();
            return true;
        }
        else
        {
            con.Close();
            return false;
        }
    }
```

4.8 管理用户信息模块

管理用户信息页面如 4-8 所示。管理员登录后，在后台可以查看所有已注册的用户的信息，包括用户 ID、用户名、密码、电话、邮箱等。管理员可以修改用户数据或删除用户信息。

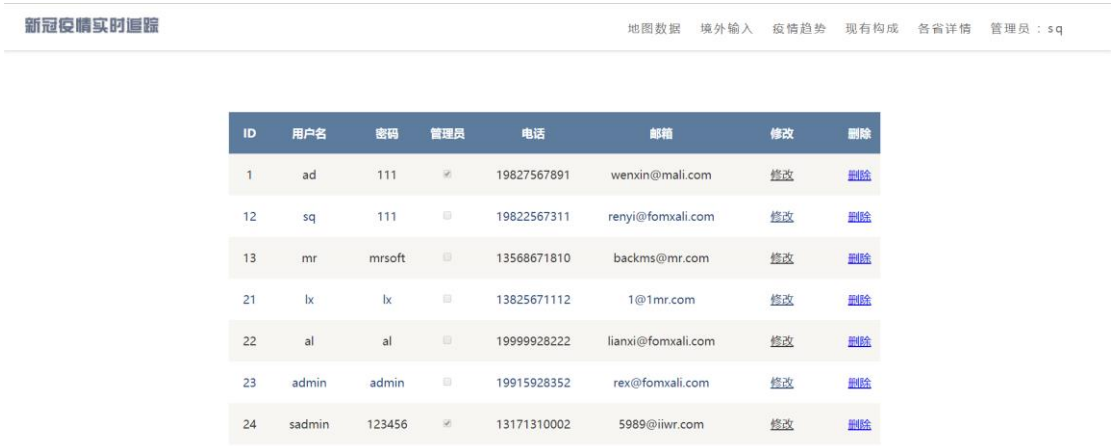


图 4-8 管理用户信息模块

修改用户信息功能的代码如下：

```
// 修改
    protected void GridView1_RowEditing(object sender, GridViewEditEventArgs e)
    {
        GridView1.EditIndex = e.NewEditIndex;
        GridView1_DataBind();
    }
    // 更新
    protected void GridView1_RowUpdating(object sender, GridViewUpdateEventArgs e)
    {
        string id = GridView1.Rows[e.RowIndex].Cells[0].Text.Trim();
        string name =
((TextBox)(GridView1.Rows[e.RowIndex].Cells[1].Controls[0])).Text.ToString().Trim();
        string pwd =
((TextBox)(GridView1.Rows[e.RowIndex].Cells[2].Controls[0])).Text.ToString().Trim();
        string tel =
((TextBox)(GridView1.Rows[e.RowIndex].Cells[4].Controls[0])).Text.ToString().Trim();
        string email =
((TextBox)(GridView1.Rows[e.RowIndex].Cells[5].Controls[0])).Text.ToString().Trim();

        string conStr = ConfigurationManager.ConnectionStrings["CnnString"].ConnectionString;
        SqlConnection con = new SqlConnection(conStr);          //创建数据库连接
        SqlCommand cmd = new SqlCommand();
        cmd.Connection = con;

        cmd.CommandText = "update User_Info set userName =
'"+name+"',userPwd='"+pwd+"',tel='"+tel+"',email='"+email+"' where userId="+id+"";
        try
        {
            con.Open();
            cmd.ExecuteNonQuery();
        }
        catch (Exception ex) { Response.Write("数据更新失败，原因是：" + ex.Message); }
        finally
        {
            con.Close();
            GridView1.EditIndex = -1;
            GridView1_DataBind();
        }
    }
    // 取消
    protected void GridView1_RowCancelingEdit(object sender, GridViewCancelEditEventArgs e)
    {
        GridView1.EditIndex = -1;
        GridView1_DataBind();
}
```

删除读者信息的功能代码如下：

```
    // 删除
    protected void GridView1_RowDeleting(object sender, GridViewDeleteEventArgs e)
    {

        string id = GridView1.Rows[e.RowIndex].Cells[0].Text;
        string sqlDel = "delete from User_Info where userId='" + id + "'";
        string conStr = ConfigurationManager.ConnectionStrings["CnnString"].ConnectionString;
        SqlConnection con = new SqlConnection(conStr);

        try
        {
            con.Open();          //打开数据库连接
            SqlCommand com = new SqlCommand(sqlDel, con);        //创建SQLCommand对象
            com.ExecuteNonQuery();        //执行SQL语句
        }
        catch (Exception ex)
        {
            Response.Write("数据更新失败，原因是：" + ex.Message);
        }
        finally
        {
            con.Close();       //关闭数据库连接
        }
        GridView1_DataBind();
       }
```