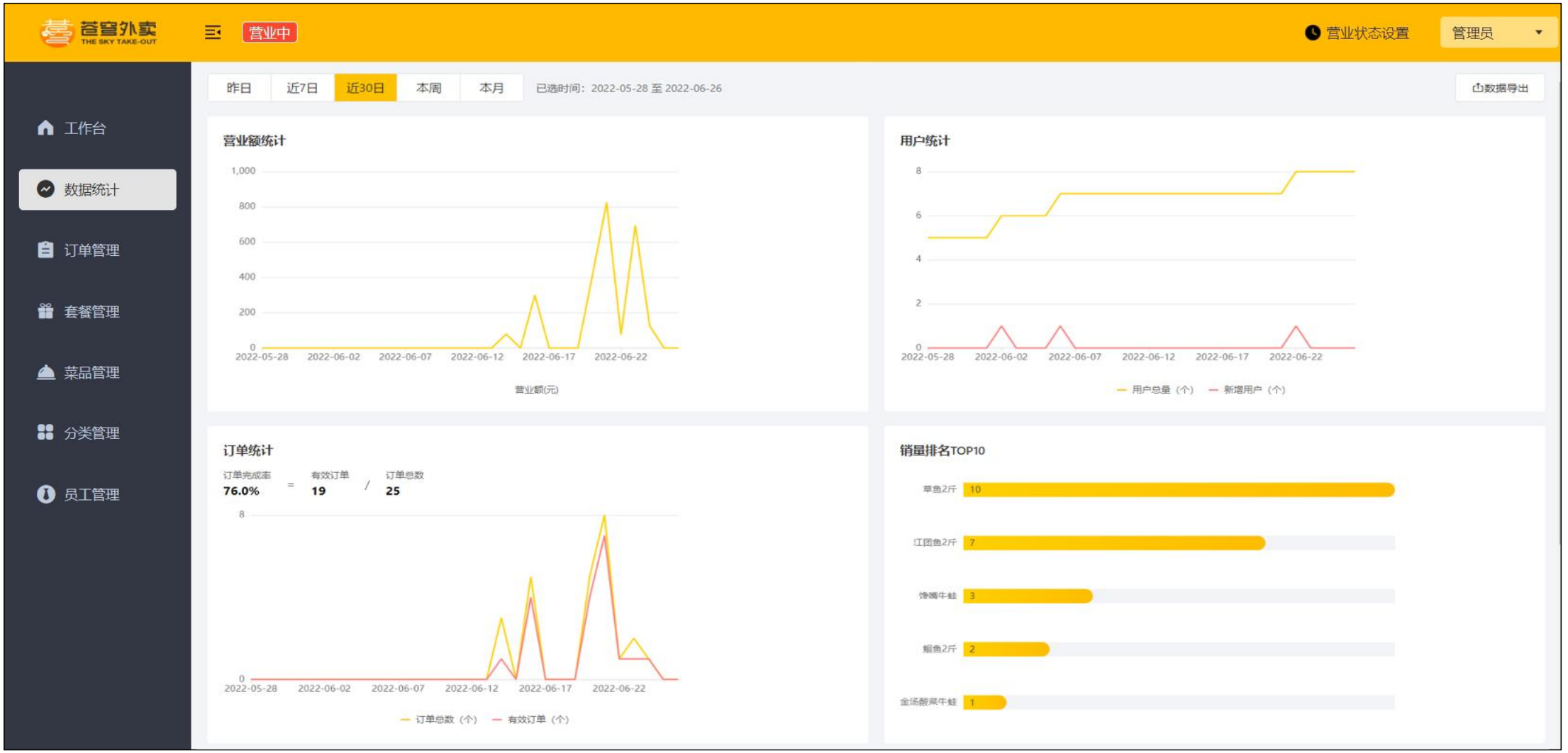
数据统计—图形报表

黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌


苍穹外卖
THE SKY TAKE-OUT
营业中
营业状态设置
管理员
工作台
数据统计
订单管理
套餐管理
菜品管理
分类管理
员工管理
昨日
近7日
近30日
本周
本月
已选时间：2022-05-28 至 2022-06-26
数据导出
营业额统计
1,000
800
600
400
200
0
2022-05-28
2022-06-02
2022-06-07
2022-06-12
2022-06-17
2022-06-22
营业额(元)
用户统计
8
6
4
2
0
用户总量（个）
新增用户（个）
订单统计
订单完成率
76.0%
有效订单
19
订单总数
25
订单总数（个）
有效订单（个）
销量排名TOP10
10
7
3
2
1

目录

Contents

Apache ECharts

- 介绍
- 入门案例

Apache ECharts 介绍

Apache ECharts 是一款基于 Javascript 的数据可视化图表库，提供直观，生动，可交互，可个性化定制的数据可视化图表。

官网地址：https://echarts.apache.org/zh/index.html

Apache ECharts 介绍

效果展示：

柱形图
饼形图
折线图

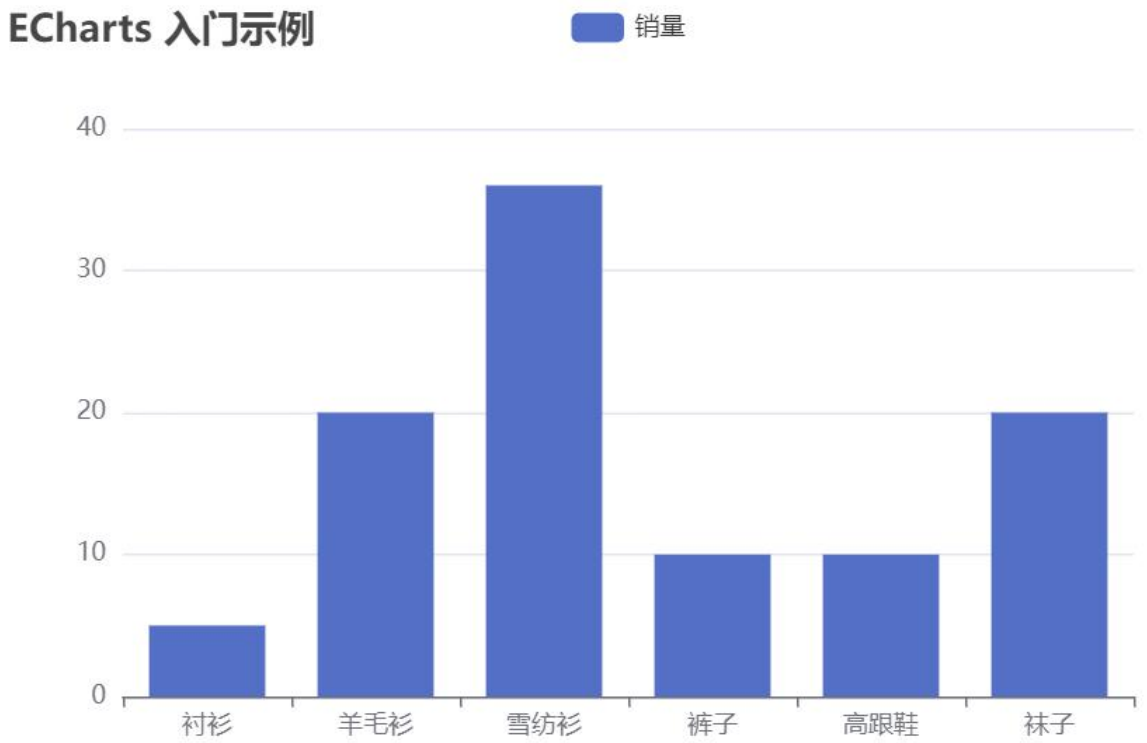
通过直观的图表来展示数据

Apache ECharts

- 介绍
- 入门案例

入门案例

Apache Echarts官方提供的快速入门：https://echarts.apache.org/handbook/zh/get-started/



入门案例

总结：使用Echarts，重点在于研究当前图表所需的数据格式。通常是需要后端提供符合格式要求的动态数据，然后响应给前端来展示图表。

目录

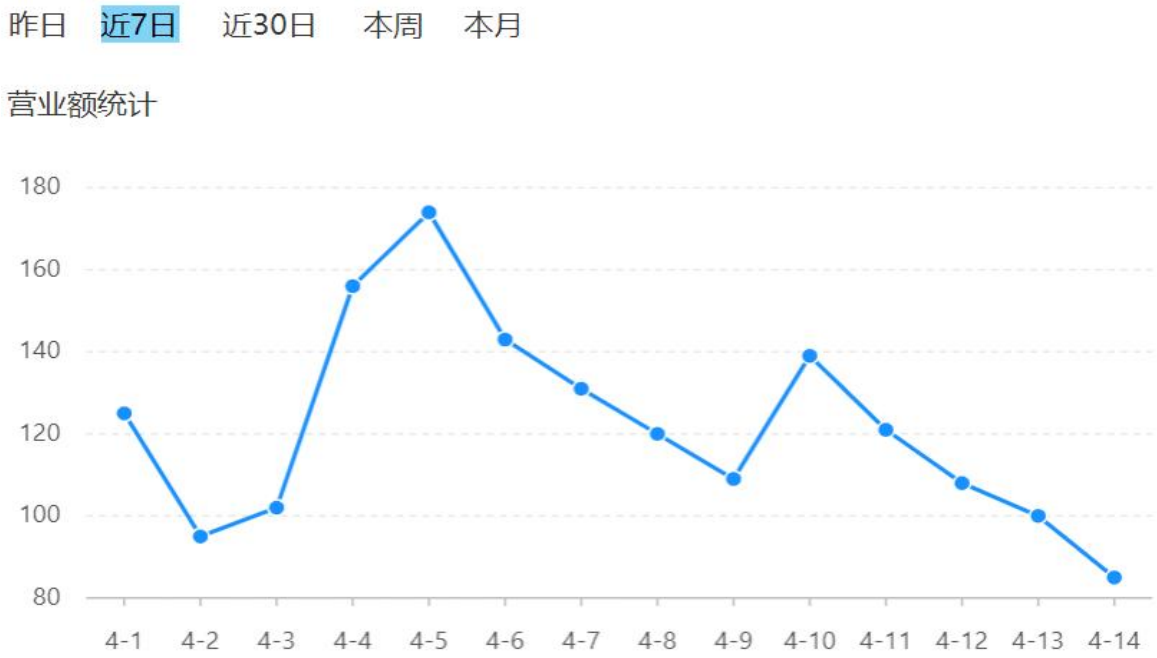
Contents

营业额统计

- 需求分析和设计
- 代码开发
- 功能测试

需求分析和设计

产品原型：



业务规则：

- 营业额指订单状态为已完成的订单金额合计
- 基于可视化报表的折线图展示营业额数据，X轴为日期，Y轴为营业额
- 根据时间选择区间，展示每天的营业额数据

需求分析和设计

接口设计：

基本信息

Path： /admin/report/turnoverStatistics

Method： GET

接口描述：

请求参数

Query

参数名称	是否必须	示例	备注
begin	是	2022-05-01	开始日期
end	是	2022-05-31	结束日期

返回数据

名称	类型	是否必须	默认值	备注	其他信息
code	integer	必须			format: int32
data	object	必须			
├─ dateList	string	必须		日期列表，日期之间以逗号分隔	
├─ turnoverList	string	必须		营业额列表，营业额之间以逗号分隔	
msg	string	非必须			

营业额统计

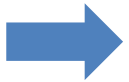
- 需求分析和设计
- 代码开发
- 功能测试

代码开发

根据接口定义设计对应的VO：

返回数据

名称	类型	是否必须	默认值	备注
code	integer	必须		
data	object	必须		
├─ dateList	string	必须		日期列表，日期之间以逗号分隔
├─ turnoverList	string	必须		营业额列表，营业额之间以逗号分隔
msg	string	非必须		



```
public class TurnoverReportVO implements Serializable {

    //日期，以逗号分隔，例如：2022-10-01,2022-10-02,2022-10-03
    private String dateList;

    //营业额，以逗号分隔，例如：406.0,1520.0,75.0
    private String turnoverList;

}
```

代码开发

根据接口定义创建ReportController：

```
@RestController
@RequestMapping("/admin/report")
@Slf4j
@Api(tags = "统计报表相关接口")
public class ReportController {
    @Autowired
    private ReportService reportService;

    /**
     * 营业额数据统计
    * @param begin
     * @param end
     * @return
     */
    @GetMapping("/turnoverStatistics")
    @ApiOperation("营业额数据统计")
    public Result<TurnoverReportVO> turnoverStatistics(
            @DateTimeFormat(pattern = "yyyy-MM-dd") LocalDate begin,
                    @DateTimeFormat(pattern = "yyyy-MM-dd") LocalDate end){

        return Result.success(reportService.getTurnover(begin,end));
    }
}
```

代码开发

创建ReportService接口，声明getTurnover方法：

```
public interface ReportService {

    /**
     * 根据时间区间统计营业额
    * @param begin
     * @param end
     * @return
     */
    TurnoverReportVO getTurnover(LocalDateTime begin, LocalDateTime end);
}
```

代码开发

创建ReportServiceImpl实现类，实现getTurnover方法（第1部分）：

```
@Service
@Slf4j
public class ReportServiceImpl implements ReportService {

    @Autowired
    private OrderMapper orderMapper;

    /**
     * 根据时间区间统计营业额
    * @param begin
     * @param end
     * @return
     */
    public TurnoverReportVO getTurnover(LocalDate begin, LocalDate end) {
        List<LocalDate> dateList = new ArrayList<>();
        dateList.add(begin);

        while (!begin.equals(end)){
            begin = begin.plusDays(1);//日期计算，获得指定日期后1天的日期
          dateList.add(begin);
        }
```

代码开发

创建ReportServiceImpl实现类，实现getTurnover方法（第2部分）：

```
    List<Double> turnoverList = new ArrayList<>();
    for (LocalDate date : dateList) {
        LocalDateTime beginTime = LocalDateTime.of(date, LocalTime.MIN);
        LocalDateTime endTime = LocalDateTime.of(date, LocalTime.MAX);
        //查询营业额
       Map map = new HashMap();
        map.put("status", Orders.COMPLETED);
        map.put("begin",beginTime);
        map.put("end", endTime);
        Double turnover = orderMapper.sumByMap(map);
        turnover = turnover == null ? 0.0 : turnover;
        turnoverList.add(turnover);
    }

    //数据封装
   return TurnoverReportVO.builder()
            .dateList(StringUtils.join(dateList,","))
            .turnoverList(StringUtils.join(turnoverList,","))
            .build();
  }
}
```

代码开发

在OrderMapper接口声明sumByMap方法：

```
/**
 * 根据动态条件统计营业额
 * @param map
 */
Double sumByMap(Map map);
```

代码开发

在OrderMapper.xml文件中编写动态SQL：

```
<select id="sumByMap" resultType="java.lang.Double">
    select sum(amount) from orders
    <where>
        <if test="status != null">
            and status = #{status}
        </if>
        <if test="begin != null">
            and order_time &gt;= #{begin}
        </if>
        <if test="end != null">
            and order_time &lt;= #{end}
        </if>
    </where>
</select>
```

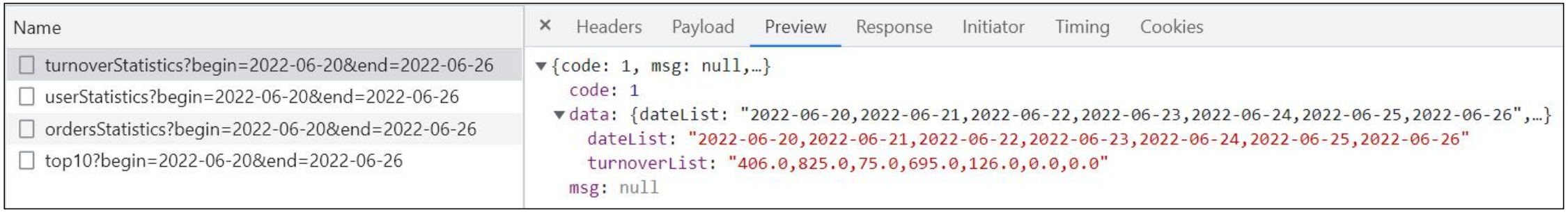
营业额统计

- 需求分析和设计
- 代码开发
- 功能测试

功能测试

可以通过如下方式进行测试：

- 接口文档测试
- 前后端联调测试



目录

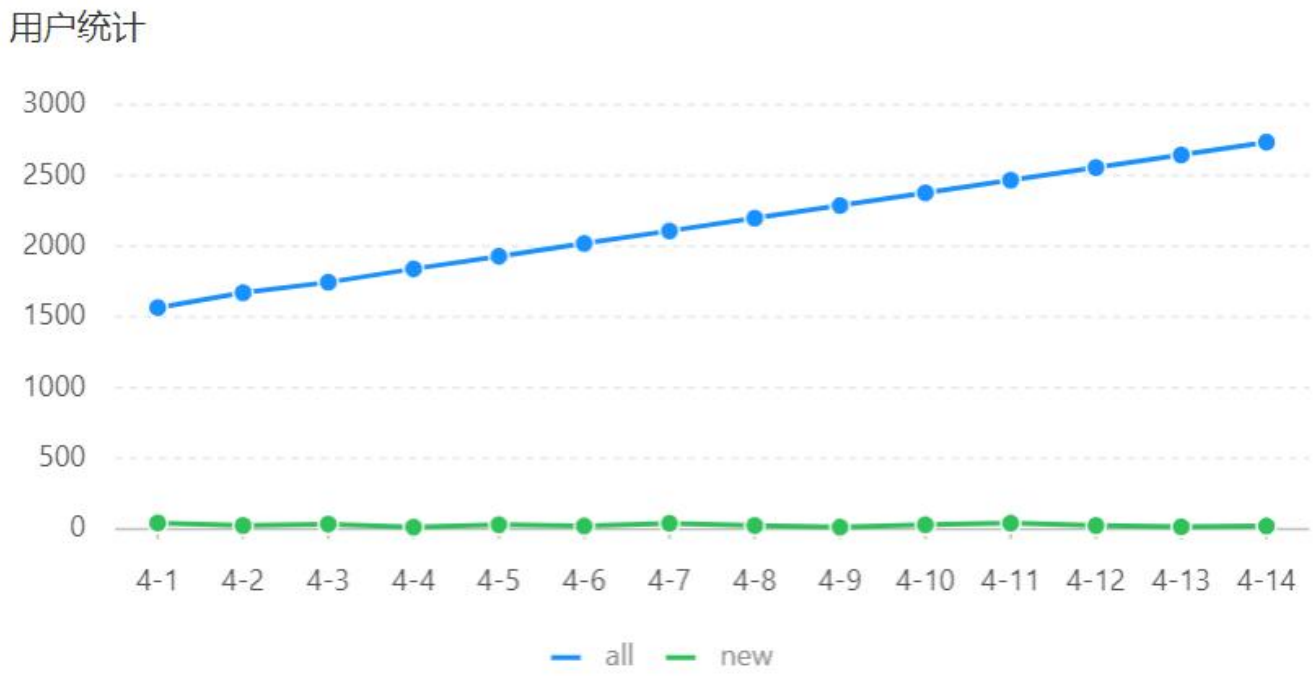
Contents

用户统计

- 需求分析和设计
- 代码开发
- 功能测试

需求分析和设计

产品原型：



业务规则：

- 基于可视化报表的折线图展示用户数据，X轴为日期，Y轴为用户数
- 根据时间选择区间，展示每天的用户总量和新增用户量数据

需求分析和设计

接口设计：

基本信息

Path： /admin/report/userStatistics

Method： GET

接口描述：

请求参数

Query

参数名称	是否必须	示例	备注
begin	是	2022-05-01	开始日期
end	是	2022-05-31	结束日期

返回数据

名称	类型	是否必须	默认值	备注	其他信息
code	integer	必须			format: int32
data	object	必须			
├─ dateList	string	必须		日期列表，以逗号分隔	
├─ newUserList	string	必须		新增用户数列表，以逗号分隔	
├─ totalUserList	string	必须		总用户量列表，以逗号分隔	
msg	string	非必须			

用户统计

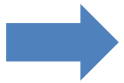
- 需求分析和设计
- 代码开发
- 功能测试

代码开发

根据用户统计接口的返回结果设计VO：

返回数据

名称	类型	是否必须	默认值	备注
code	integer	必须		
data	object	必须		
├─ dateList	string	必须		日期列表，以逗号分隔
├─ newUserList	string	必须		新增用户数列表，以逗号分隔
├─ totalUserList	string	必须		总用户量列表，以逗号分隔
msg	string	非必须		



```java
public class UserReportVO implements Serializable {

    //日期，以逗号分隔，例如：2022-10-01,2022-10-02,2022-10-03
    private String dateList;

    //用户总量，以逗号分隔，例如：200,210,220
    private String totalUserList;

    //新增用户，以逗号分隔，例如：20,21,10
    private String newUserList;

}
```

代码开发

根据接口定义，在ReportController中创建userStatistics方法：

```
/**
 * 用户数据统计
 * @param begin
 * @param end
 * @return
 */
@GetMapping("/userStatistics")
@ApiOperation("用户数据统计")
public Result<UserReportVO> userStatistics(
        @DateTimeFormat(pattern = "yyyy-MM-dd") LocalDate begin,
        @DateTimeFormat(pattern = "yyyy-MM-dd") LocalDate end){

    return Result.success(reportService.getUserStatistics(begin,end));
}
```

代码开发

在ReportService接口中声明getUserStatistics方法：

```
/**
 * 根据时间区间统计用户数量
 * @param begin
 * @param end
 * @return
 */
UserReportVO getUserStatistics(LocalDate begin, LocalDate end);
```

代码开发

在ReportServiceImpl实现类中实现getUserStatistics方法（第1部分）：

```
/**
 * 根据时间区间统计用户数量
 * @param begin
 * @param end
 * @return
 */
public UserReportVO getUserStatistics(LocalDate begin, LocalDate end) {
    List<LocalDate> dateList = new ArrayList<>();
    dateList.add(begin);

    while (!begin.equals(end)){
        begin = begin.plusDays(1);
        dateList.add(begin);
    }

    List<Integer> newUserList = new ArrayList<>(); //新增用户数
   List<Integer> totalUserList = new ArrayList<>(); //总用户数
```

代码开发

在ReportServiceImpl实现类中实现getUserStatistics方法（第2部分）：

```
for (LocalDate date : dateList) {
        LocalDateTime beginTime = LocalDateTime.of(date, LocalTime.MIN);
        LocalDateTime endTime = LocalDateTime.of(date, LocalTime.MAX);
        //新增用户数量 select count(id) from user where create_time > ? and create_time < ?
        Integer newUser = getUserCount(beginTime, endTime);
        //总用户数量 select count(id) from user where  create_time < ?
        Integer totalUser = getUserCount(null, endTime);

        newUserList.add(newUser);
        totalUserList.add(totalUser);
    }

    return UserReportVO.builder()
            .dateList(StringUtils.join(dateList,","))
            .newUserList(StringUtils.join(newUserList,","))
            .totalUserList(StringUtils.join(totalUserList,","))
            .build();
}
```

代码开发

在ReportServiceImpl实现类中创建私有方法getUserCount：

```
/**
 * 根据时间区间统计用户数量
 * @param beginTime
 * @param endTime
 * @return
 */
private Integer getUserCount(LocalDateTime beginTime, LocalDateTime endTime) {
    Map map = new HashMap();
    map.put("begin",beginTime);
    map.put("end", endTime);
    return userMapper.countByMap(map);
}
```

代码开发

在UserMapper接口中声明countByMap方法：

```
/**
 * 根据动态条件统计用户数量
 * @param map
 * @return
 */
Integer countByMap(Map map);
```

代码开发

在UserMapper.xml文件中编写动态SQL：

```
<select id="countByMap" resultType="java.lang.Integer">
    select count(id) from user
    <where>
        <if test="begin != null">
            and create_time &gt;= #{begin}
        </if>
        <if test="end != null">
            and create_time &lt;= #{end}
        </if>
    </where>
</select>
```

用户统计

- 需求分析和设计
- 代码开发
- 功能测试

功能测试

可以通过如下方式进行测试：

- 接口文档测试
- 前后端联调测试

目录

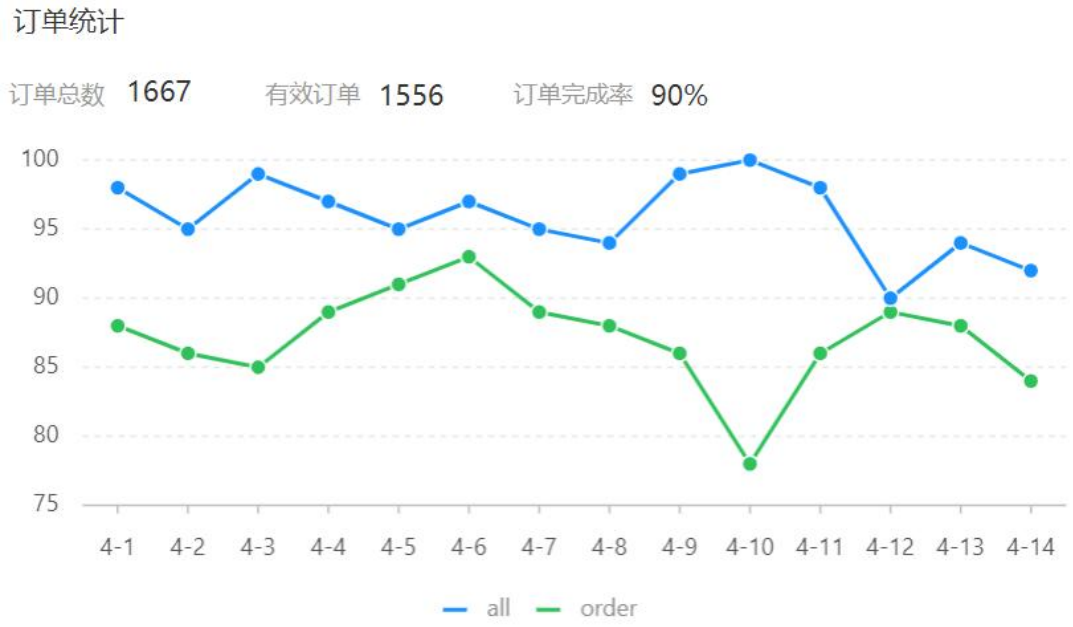
Contents

订单统计

- 需求分析和设计
- 代码开发
- 功能测试

需求分析和设计

产品原型：



业务规则：

- 有效订单指状态为“已完成”的订单
- 基于可视化报表的折线图展示订单数据，X轴为日期，Y轴为订单数量
- 根据时间选择区间，展示每天的订单总数和有效订单数
- 展示所选时间区间内的有效订单数、总订单数、订单完成率，订单完成率 = 有效订单数 / 总订单数 * 100%

需求分析和设计

接口设计：

基本信息

Path： /admin/report/ordersStatistics

Method： GET

接口描述：

请求参数

Query

参数名称	是否必须	示例	备注
begin	是	2022-05-01	开始日期
end	是	2022-05-31	结束日期

返回数据

名称	类型	是否必须	默认值	备注	其他信息
code	integer	必须			format: int32
data	object	必须			
├─ dateList	string	必须		日期列表，以逗号分隔	
├─ orderCompletionRate	number	必须		订单完成率	format: double
├─ orderCountList	string	必须		订单数列表，以逗号分隔	
├─ totalOrderCount	integer	必须		订单总数	format: int32
├─ validOrderCount	integer	必须		有效订单数	format: int32
├─ validOrderCountList	string	必须		有效订单数列表，以逗号分隔	
msg	string	非必须			

订单统计

- 需求分析和设计
- 代码开发
- 功能测试

代码开发

根据订单统计接口的返回结果设计VO：

返回数据

名称	类型	是否必须	默认值	备注
code	integer	必须		
data	object	必须		
├─ dateList	string	必须		日期列表，以逗号分隔
├─ orderCompletionRate	number	必须		订单完成率
├─ orderCountList	string	必须		订单数列表，以逗号分隔
├─ totalOrderCount	integer	必须		订单总数
├─ validOrderCount	integer	必须		有效订单数
├─ validOrderCountList	string	必须		有效订单数列表，以逗号分隔
msg	string	非必须		

```java
public class OrderReportVO implements Serializable {

    //日期，以逗号分隔，例如：2022-10-01,2022-10-02,2022-10-03
    private String dateList;

    //每日订单数，以逗号分隔，例如：260,210,215
    private String orderCountList;

    //每日有效订单数，以逗号分隔，例如：20,21,10
    private String validOrderCountList;

    //订单总数
    private Integer totalOrderCount;

    //有效订单数
    private Integer validOrderCount;

    //订单完成率
    private Double orderCompletionRate;

}
```

代码开发

在ReportController中根据订单统计接口创建orderStatistics方法：

```
/**
 * 订单数据统计
 * @param begin
 * @param end
 * @return
 */
@GetMapping("/ordersStatistics")
@ApiOperation("订单数据统计")
public Result<OrderReportVO> orderStatistics(
        @DateTimeFormat(pattern = "yyyy-MM-dd") LocalDate begin,
        @DateTimeFormat(pattern = "yyyy-MM-dd") LocalDate end){

    return Result.success(reportService.getOrderStatistics(begin,end));
}
```

代码开发

在ReportService接口中声明getOrderStatistics方法：

```
/**
 * 根据时间区间统计订单数量
 * @param begin
 * @param end
 * @return
 */
OrderReportVO getOrderStatistics(LocalDate begin, LocalDate end);
```

代码开发

在ReportServiceImpl实现类中实现getOrderStatistics方法（第1部分）：

```
/**
 * 根据时间区间统计订单数量
 * @param begin
 * @param end
 * @return
 */
public OrderReportVO getOrderStatistics(LocalDate begin, LocalDate end) {
    List<LocalDate> dateList = new ArrayList<>();
    dateList.add(begin);

    while (!begin.equals(end)){
        begin = begin.plusDays(1);
        dateList.add(begin);
    }

    //每天订单总数集合
   List<Integer> orderCountList = new ArrayList<>();
    //每天有效订单数集合
   List<Integer> validOrderCountList = new ArrayList<>();
```

代码开发

在ReportServiceImpl实现类中实现getOrderStatistics方法（第2部分）：

```
for (LocalDate date : dateList) {
    LocalDateTime beginTime = LocalDateTime.of(date, LocalTime.MIN);
    LocalDateTime endTime = LocalDateTime.of(date, LocalTime.MAX);
    //查询每天的总订单数 select count(id) from orders where order_time > ? and order_time < ?
    Integer orderCount = getOrderCount(beginTime, endTime, null);

    //查询每天的有效订单数 select count(id) from orders where order_time > ? and order_time < ? and status = ?
    Integer validOrderCount = getOrderCount(beginTime, endTime, Orders.COMPLETED);

    orderCountList.add(orderCount);
    validOrderCountList.add(validOrderCount);
}

//时间区间内的总订单数
Integer totalOrderCount = orderCountList.stream().reduce(Integer::sum).get();
//时间区间内的总有效订单数
Integer validOrderCount = validOrderCountList.stream().reduce(Integer::sum).get();
```

代码开发

在ReportServiceImpl实现类中实现getOrderStatistics方法（第3部分）：

```
    //订单完成率
    Double orderCompletionRate = 0.0;
    if(totalOrderCount != 0){
        orderCompletionRate = validOrderCount.doubleValue() / totalOrderCount;
    }

    return OrderReportVO.builder()
            .dateList(StringUtils.join(dateList, ","))
            .orderCountList(StringUtils.join(orderCountList, ","))
            .validOrderCountList(StringUtils.join(validOrderCountList, ","))
            .totalOrderCount(totalOrderCount)
            .validOrderCount(validOrderCount)
            .orderCompletionRate(orderCompletionRate)
            .build();
}
```

代码开发

在ReportServiceImpl实现类中提供私有方法getOrderCount：

```
/**
 * 根据时间区间统计指定状态的订单数量
 * @param beginTime
 * @param endTime
 * @param status
 * @return
 */
private Integer getOrderCount(LocalDateTime beginTime, LocalDateTime endTime, Integer status) {
    Map map = new HashMap();
    map.put("status", status);
    map.put("begin",beginTime);
    map.put("end", endTime);
    return orderMapper.countByMap(map);
}
```

代码开发

在OrderMapper接口中声明countByMap方法：

```
/**
 * 根据动态条件统计订单数量
 * @param map
 */
Integer countByMap(Map map);
```

代码开发

在OrderMapper.xml文件中编写动态SQL：

```
<select id="countByMap" resultType="java.lang.Integer">
    select count(id) from orders
    <where>
        <if test="status != null">
            and status = #{status}
        </if>
        <if test="begin != null">
            and order_time &gt;= #{begin}
        </if>
        <if test="end != null">
            and order_time &lt;= #{end}
        </if>
    </where>
</select>
```

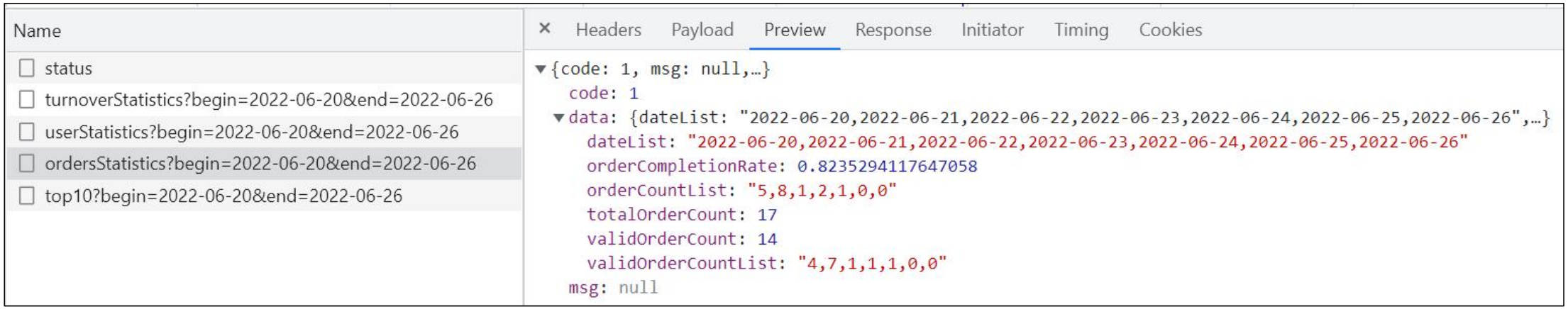
订单统计

- 需求分析和设计
- 代码开发
- 功能测试

功能测试

可以通过如下方式进行测试：

- 接口文档测试
- 前后端联调



目录

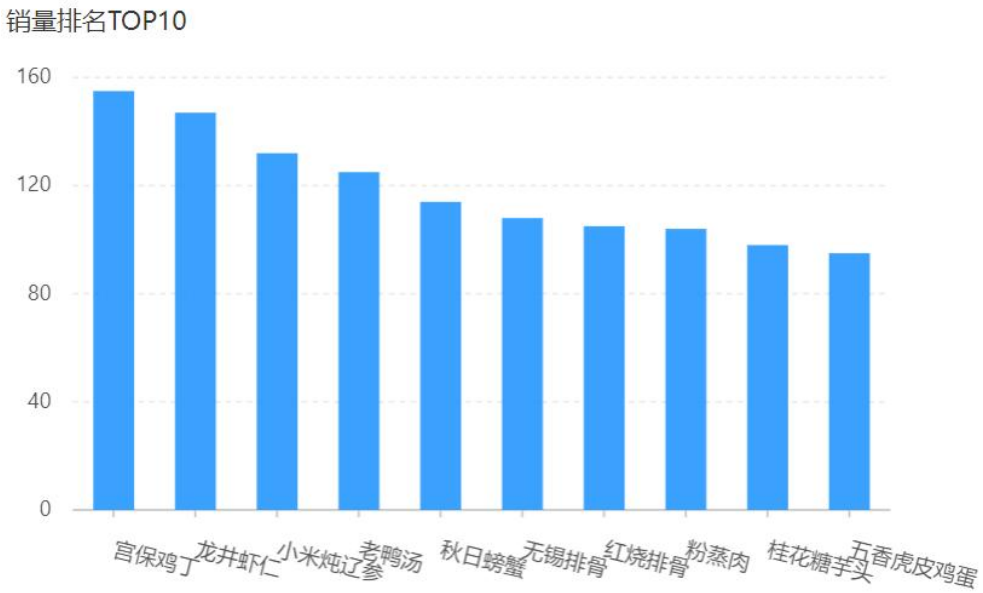
Contents

销量排名Top10

- 需求分析和设计
- 代码开发
- 功能测试

需求分析和设计

产品原型：



业务规则：

- 根据时间选择区间，展示销量前10的商品（包括菜品和套餐）
- 基于可视化报表的柱状图降序展示商品销量
- 此处的销量为商品销售的份数

需求分析和设计

接口设计：

基本信息

Path： /admin/report/top10

Method： GET

接口描述：

请求参数

Query

参数名称	是否必须	示例	备注
begin	是	2022-05-01	开始日期
end	是	2022-05-31	结束日期

返回数据

名称	类型	是否必须	默认值	备注	其他信息
code	integer	必须			format: int32
data	object	必须			
├─ nameList	string	必须		商品名称列表，以逗号分隔	
├─ numberList	string	必须		销量列表，以逗号分隔	
msg	string	非必须			

销量排名Top10

- 需求分析和设计
- 代码开发
- 功能测试

代码开发

根据销量排名接口的返回结果设计VO：

返回数据

名称	类型	是否必须	默认值	备注
code	integer	必须		
data	object	必须		
├─ nameList	string	必须		商品名称列表，以逗号分隔
├─ numberList	string	必须		销量列表，以逗号分隔
msg	string	非必须		

```java
public class SalesTop10ReportVO implements Serializable {

    //商品名称列表，以逗号分隔，例如：鱼香肉丝,宫保鸡丁,水煮鱼
    private String nameList;

    //销量列表，以逗号分隔，例如：260,215,200
    private String numberList;

}
```

代码开发

在ReportController中根据销量排名接口创建top10方法：

```
/**
 * 销量排名统计
 * @param begin
 * @param end
 * @return
 */
@GetMapping("/top10")
@ApiOperation("销量排名统计")
public Result<SalesTop10ReportVO> top10(
            @DateTimeFormat(pattern = "yyyy-MM-dd") LocalDate begin,
            @DateTimeFormat(pattern = "yyyy-MM-dd") LocalDate end){

    return Result.success(reportService.getSalesTop10(begin,end));
}
```

代码开发

在ReportService接口中声明getSalesTop10方法：

```
/**
 * 查询指定时间区间内的销量排名top10
 * @param begin
 * @param end
 * @return
 */
SalesTop10ReportVO getSalesTop10(LocalDate begin, LocalDate end);
```

代码开发

在ReportServiceImpl实现类中实现getSalesTop10方法：

```
/**
 * 查询指定时间区间内的销量排名top10
 * @param begin
 * @param end
 * @return
 */
public SalesTop10ReportVO getSalesTop10(LocalDate begin, LocalDate end) {
    LocalDateTime beginTime = LocalDateTime.of(begin, LocalTime.MIN);
    LocalDateTime endTime = LocalDateTime.of(end, LocalTime.MAX);
    List<GoodsSalesDTO> goodsSalesDTOList = orderMapper.getSalesTop10(beginTime, endTime);

    String nameList = StringUtils.join(goodsSalesDTOList.stream().map(GoodsSalesDTO::getName).collect(Collectors.toList()),",");
    String numberList = StringUtils.join(goodsSalesDTOList.stream().map(GoodsSalesDTO::getNumber).collect(Collectors.toList()),",");

    return SalesTop10ReportVO.builder()
            .nameList(nameList)
            .numberList(numberList)
            .build();
}
```

代码开发

在OrderMapper接口中声明getSalesTop10方法：

```
/**
 * 查询商品销量排名
 * @param begin
 * @param end
 */
List<GoodsSalesDTO> getSalesTop10(LocalDateTime begin, LocalDateTime end);
```

代码开发

在OrderMapper.xml文件中编写动态SQL：

```
<select id="getSalesTop10" resultType="com.sky.dto.GoodsSalesDTO">
    select od.name name,sum(od.number) number from order_detail od ,orders o
    where od.order_id = o.id
        and o.status = 5
        <if test="begin != null">
            and order_time &gt;= #{begin}
        </if>
        <if test="end != null">
            and order_time &lt;= #{end}
        </if>
    group by name
    order by number desc
    limit 0, 10
</select>
```

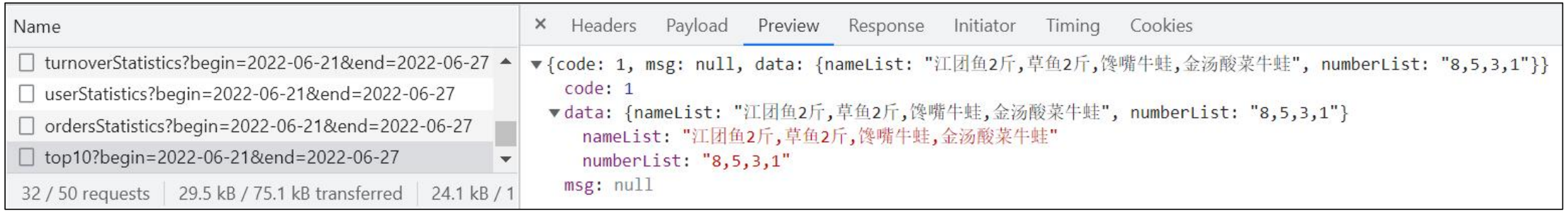
销量排名Top10

- 需求分析和设计
- 代码开发
- 功能测试

功能测试

可以通过如下方式进行测试：

- 接口文档测试
- 前后端联调



传智教育旗下高端IT教育品牌