尚医通

版本：V1.0

# 排班管理

## 页面效果



排班分成三部分显示：

1、科室信息（大科室与小科室树形展示）

2、排班日期，分页显示，根据上传排班数据聚合统计产生

3、排班日期对应的就诊医生信息

## 接口分析

1. 科室数据使用Element-ui el-tree组件渲染展示，需要将医院上传的科室数据封装成两层父子级数据；
2. 聚合所有排班数据，按日期分页展示，并统计号源数据展示；
3. 根据排班日期获取排班详情数据

## 实现分析

虽然是一个页面展示所有内容，但是页面相对复杂，我们分步骤实现

1. 先实现左侧科室树形展示；
2. 其次排班日期分页展示
3. 最后根据排班日期获取排班详情数据

# 排班管理实现

## 1、科室列表

### 1.1 api接口

#### 1.1.1 添加service接口与实现

在DepartmentService类添加接口

|  |
| --- |
| *//根据医院编号，查询医院所有科室列表*  List<DepartmentVo> findDeptTree(String hoscode); |

在DepartmentServiceImpl类实现接口

|  |
| --- |
| *//根据医院编号，查询医院所有科室列表*  @Override  **public** List<DepartmentVo> findDeptTree(String hoscode) {  *//创建list集合，用于最终数据封装*  List<DepartmentVo> result = **new** ArrayList<>();  *//根据医院编号，查询医院所有科室信息*  Department departmentQuery = **new** Department();  departmentQuery.setHoscode(hoscode);  Example example = Example.*of*(departmentQuery);  *//所有科室列表 departmentList*  List<Department> departmentList = **departmentRepository**.findAll(example);  *//根据大科室编号 bigcode 分组，获取每个大科室里面下级子科室*  Map<String, List<Department>> deparmentMap =  departmentList.stream().collect(Collectors.*groupingBy*(Department::getBigcode));  *//遍历map集合 deparmentMap*  **for**(Map.Entry<String,List<Department>> entry : deparmentMap.entrySet()) {  *//大科室编号*  String bigcode = entry.getKey();  *//大科室编号对应的全局数据*  List<Department> deparment1List = entry.getValue();  *//封装大科室*  DepartmentVo departmentVo1 = **new** DepartmentVo();  departmentVo1.setDepcode(bigcode);  departmentVo1.setDepname(deparment1List.get(0).getBigname());  *//封装小科室*  List<DepartmentVo> children = **new** ArrayList<>();  **for**(Department department: deparment1List) {  DepartmentVo departmentVo2 = **new** DepartmentVo();  departmentVo2.setDepcode(department.getDepcode());  departmentVo2.setDepname(department.getDepname());  *//封装到list集合*  children.add(departmentVo2);  }  *//把小科室list集合放到大科室children里面*  departmentVo1.setChildren(children);  *//放到最终result里面*  result.add(departmentVo1);  }  *//返回*  **return** result;  } |

#### 1.1.2 添加controller接口

|  |
| --- |
| @RestController  @RequestMapping(**"/admin/hosp/department"**)  @CrossOrigin  **public class** DepartmentController {  @Autowired  **private** DepartmentService **departmentService**;  *//根据医院编号，查询医院所有科室列表*  @ApiOperation(value = **"查询医院所有科室列表"**)  @GetMapping(**"getDeptList/{hoscode}"**)  **public** Result getDeptList(@PathVariable String hoscode) {  List<DepartmentVo> list = **departmentService**.findDeptTree(hoscode);  **return** Result.*ok*(list);  }  } |

### 1.2 科室前端

#### 1.2.1 添加路由

在 src/router/index.js 文件添加排班隐藏路由

|  |
| --- |
| **{**  **path: 'hospital/schedule/:hoscode',**  **name: '排班',**  **component: () => import('@/views/hosp/schedule'),**  **meta: { title: '排班', noCache: true },**  **hidden: true**  **}** |

#### 1.2.2 添加按钮

在医院列表页面,添加排班按钮

|  |
| --- |
| <router-link :to="'/hospSet/hospital/schedule/'+scope.row.hoscode">      <el-button type="primary" size="mini">排班</el-button>  </router-link> |

#### 1.2.3封装api请求

|  |
| --- |
| {    path: 'hospital/schedule/:hoscode',    name: '排班',    component: () => import('@/views/hosp/schedule'),    meta: { title: '排班', noCache: true },    hidden: true  } |

#### 1.2.4 部门展示

修改/views/hosp/schedule.vue组件

|  |
| --- |
| <template>      <div class="app-container">          <div style="margin-bottom: 10px;font-size: 10px;">选择：</div>              <el-container style="height: 100%">              <el-aside width="200px" style="border: 1px silver solid">                  <!-- 部门 -->                  <el-tree                  :data="data"                  :props="defaultProps"                  :default-expand-all="true"                  @node-click="handleNodeClick">                  </el-tree>              </el-aside>              <el-main style="padding: 0 0 0 20px;">                  <el-row style="width: 100%">                  <!-- 排班日期 分页 -->                  </el-row>                  <el-row style="margin-top: 20px;">                  <!-- 排班日期对应的排班医生 -->                  </el-row>              </el-main>          </el-container>      </div>  </template>  <script>  import hospApi from '@/api/hosp'  export default {      data() {          return {              data: [],              defaultProps: {                  children: 'children',                  label: 'depname'              },              hoscode: null          }      },      created(){          this.hoscode = this.$route.params.hoscode          this.fetchData()      },      methods:{          fetchData() {              hospApi.getDeptByHoscode(this.hoscode)                  .then(response => {                      this.data = response.data                  })          }      }  }  </script>  <style>    .el-tree-node.is-current > .el-tree-node\_\_content {      background-color: #409EFF !important;      color: white;     }    .el-checkbox\_\_input.is-checked+.el-checkbox\_\_label {      color: black;     }  </style> |

说明：底部style标签是为了控制树形展示数据选中效果的

## 2、排班日期分页列表

### 2.1 api接口

#### 2.1.1 添加service接口与实现

在ScheduleService类添加接口

|  |
| --- |
| *//根据医院编号 和 科室编号 ，查询排班规则数据*  Map<String, Object> getRuleSchedule(**long** page, **long** limit, String hoscode, String depcode); |

在ScheduleServiceImpl类实现接口

|  |
| --- |
| @Autowired  **private** ScheduleRepository **scheduleRepository**;  @Autowired  **private** MongoTemplate **mongoTemplate**;  @Autowired  **private** HospitalService **hospitalService**; |
| *//根据医院编号 和 科室编号 ，查询排班规则数据*  @Override  **public** Map<String, Object> getRuleSchedule(**long** page, **long** limit, String hoscode, String depcode) {  *//1 根据医院编号 和 科室编号 查询*  Criteria criteria = Criteria.*where*(**"hoscode"**).is(hoscode).and(**"depcode"**).is(depcode);  *//2 根据工作日workDate期进行分组*  Aggregation agg = Aggregation.*newAggregation*(  Aggregation.*match*(criteria),*//匹配条件*  Aggregation.*group*(**"workDate"**)*//分组字段*  .first(**"workDate"**).as(**"workDate"**)  *//3 统计号源数量*  .count().as(**"docCount"**)  .sum(**"reservedNumber"**).as(**"reservedNumber"**)  .sum(**"availableNumber"**).as(**"availableNumber"**),  *//排序*  Aggregation.*sort*(Sort.Direction.***DESC***,**"workDate"**),  *//4 实现分页*  Aggregation.*skip*((page-1)\*limit),  Aggregation.*limit*(limit)  );  *//调用方法，最终执行*  AggregationResults<BookingScheduleRuleVo> aggResults =  **mongoTemplate**.aggregate(agg, Schedule.**class**, BookingScheduleRuleVo.**class**);  List<BookingScheduleRuleVo> bookingScheduleRuleVoList = aggResults.getMappedResults();  *//分组查询的总记录数*  Aggregation totalAgg = Aggregation.*newAggregation*(  Aggregation.*match*(criteria),  Aggregation.*group*(**"workDate"**)  );  AggregationResults<BookingScheduleRuleVo> totalAggResults =  **mongoTemplate**.aggregate(totalAgg,  Schedule.**class**, BookingScheduleRuleVo.**class**);  **int** total = totalAggResults.getMappedResults().size();  *//把日期对应星期获取*  **for**(BookingScheduleRuleVo bookingScheduleRuleVo:bookingScheduleRuleVoList) {  Date workDate = bookingScheduleRuleVo.getWorkDate();  String dayOfWeek = **this**.getDayOfWeek(**new** DateTime(workDate));  bookingScheduleRuleVo.setDayOfWeek(dayOfWeek);  }  *//设置最终数据，进行返回*  Map<String, Object> result = **new** HashMap<>();  result.put(**"bookingScheduleRuleList"**,bookingScheduleRuleVoList);  result.put(**"total"**,total);  *//获取医院名称*  String hosName = **hospitalService**.getHospName(hoscode);  *//其他基础数据*  Map<String, String> baseMap = **new** HashMap<>();  baseMap.put(**"hosname"**,hosName);  result.put(**"baseMap"**,baseMap);  **return** result;  } |
| */\*\**  *\* 根据日期获取周几数据*  *\** ***@param dateTime***  *\** ***@return***  *\*/*  **private** String getDayOfWeek(DateTime dateTime) {  String dayOfWeek = **""**;  **switch** (dateTime.getDayOfWeek()) {  **case** DateTimeConstants.***SUNDAY***:  dayOfWeek = **"周日"**;  **break**;  **case** DateTimeConstants.***MONDAY***:  dayOfWeek = **"周一"**;  **break**;  **case** DateTimeConstants.***TUESDAY***:  dayOfWeek = **"周二"**;  **break**;  **case** DateTimeConstants.***WEDNESDAY***:  dayOfWeek = **"周三"**;  **break**;  **case** DateTimeConstants.***THURSDAY***:  dayOfWeek = **"周四"**;  **break**;  **case** DateTimeConstants.***FRIDAY***:  dayOfWeek = **"周五"**;  **break**;  **case** DateTimeConstants.***SATURDAY***:  dayOfWeek = **"周六"**;  **default**:  **break**;  }  **return** dayOfWeek;  } |

#### 2.1.2 添加根据医院编号获取医院名称接口

在HospitalService类添加接口

|  |
| --- |
| */\*\**  *\* 根据医院编号获取医院名称接口*  *\** ***@param hoscode***  *\** ***@return***  *\*/*  String getName(String hoscode); |

在HospitalServiceImpl类实现接口

|  |
| --- |
| @Override  **public** String getName(String hoscode) {  Hospital hospital = **hospitalRepository**.getHospitalByHoscode(hoscode);  **if**(**null** != hospital) {  **return** hospital.getHosname();  }  **return ""**;  } |

#### 2.1.3 添加controller接口

添加com.atguigu.yygh.hosp.controller.ScheduleController类

|  |
| --- |
| *//根据医院编号 和 科室编号 ，查询排班规则数据*  @ApiOperation(value =**"查询排班规则数据"**)  @GetMapping(**"getScheduleRule/{page}/{limit}/{hoscode}/{depcode}"**)  **public** Result getScheduleRule(@PathVariable **long** page,  @PathVariable **long** limit,  @PathVariable String hoscode,  @PathVariable String depcode) {  Map<String,Object> map  = **scheduleService**.getRuleSchedule(page,limit,hoscode,depcode);  **return** Result.*ok*(map);  } |

### 2.2 排班日期前端

#### 1.2.1封装api请求

创建/api/hosp/schedule.js

|  |
| --- |
| getScheduleRule(page, limit, hoscode, depcode) {      return request({       url: `/admin/hosp/schedule/getScheduleRule/${page}/${limit}/${hoscode}/${depcode}`,       method: 'get'      })    } |

#### 1.2.2 页面展示

修改/views/hosp/schedule.vue组件

|  |
| --- |
| <template>      <div class="app-container">  **<div style="margin-bottom: 10px;font-size: 10px;">选择：{{ baseMap.hosname }}  /  {{ depname }}  /  {{ workDate }}</div>**              <el-container style="height: 100%">              <el-aside width="200px" style="border: 1px silver solid">                  <!-- 部门 -->                  <el-tree                  :data="data"                  :props="defaultProps"                  :default-expand-all="true"                  @node-click="handleNodeClick">                  </el-tree>              </el-aside>              <el-main style="padding: 0 0 0 20px;">                  <el-row style="width: 100%">                      <!-- 排班日期 分页 -->  **<el-tag v-for="(item,index) in bookingScheduleList" :key="item.id" @click="selectDate(item.workDate, index)" :type="index == activeIndex ? '' : 'info'" style="height: 60px;margin-right: 5px;margin-right:15px;cursor:pointer;">**  **{{ item.workDate }} {{ item.dayOfWeek }}<br/>**  **{{ item.availableNumber }} / {{ item.reservedNumber }}**  **</el-tag>**  **<!-- 分页 -->**  **<el-pagination**  **:current-page="page"**  **:total="total"**  **:page-size="limit"**  **class="pagination"**  **layout="prev, pager, next"**  **@current-change="getPage">**  **</el-pagination>**                  </el-row>                  <el-row style="margin-top: 20px;">                  <!-- 排班日期对应的排班医生 -->                  </el-row>              </el-main>          </el-container>      </div>  </template>  <script>  import hospApi from '@/api/hosp'  export default {      data() {          return {              data: [],              defaultProps: {                  children: 'children',                  label: 'depname'              },  **hoscode: null,**  **activeIndex: 0,**  **depcode: null,**  **depname: null,**  **workDate: null,**  **bookingScheduleList: [],**  **baseMap: {},**  **page: 1, // 当前页**  **limit: 7, // 每页个数**  **total: 0 // 总页码**          }      },      created(){          this.hoscode = this.$route.params.hoscode          this.workDate = this.getCurDate()          this.fetchData()      },      methods:{          fetchData() {              hospApi.getDeptByHoscode(this.hoscode)                  .then(response => {                      this.data = response.data  **// 默认选中第一个**  **if (this.data.length > 0) {**  **this.depcode = this.data[0].children[0].depcode**  **this.depname = this.data[0].children[0].depname**  **this.getPage()**  **}**              })          },  **getPage(page = 1) {**  **this.page = page**  **this.workDate = null**  **this.activeIndex = 0**  **this.getScheduleRule()**  **},**  **getScheduleRule() {**  **hospApi.getScheduleRule(this.page, this.limit, this.hoscode, this.depcode).then(response => {**  **this.bookingScheduleList = response.data.bookingScheduleRuleList**  **this.total = response.data.total**  **this.scheduleList = response.data.scheduleList**  **this.baseMap = response.data.baseMap**  **// 分页后workDate=null，默认选中第一个**  **if (this.workDate == null) {**    **this.workDate = this.bookingScheduleList[0].workDate**  **}**  **})**  **},**  **handleNodeClick(data) {**  **// 科室大类直接返回**  **if (data.children != null) return**  **this.depcode = data.depcode**  **this.depname = data.depname**  **this.getPage(1)**  **},**  **selectDate(workDate, index) {**  **this.workDate = workDate**  **this.activeIndex = index**  **},**  **getCurDate() {**  **var datetime = new Date()**  **var year = datetime.getFullYear()**  **var month = datetime.getMonth() + 1 < 10 ? '0' + (datetime.getMonth() + 1) : datetime.getMonth() + 1**  **var date = datetime.getDate() < 10 ? '0' + datetime.getDate() : datetime.getDate()**  **return year + '-' + month + '-' + date**  **}**      }  }  </script>  <style>    .el-tree-node.is-current > .el-tree-node\_\_content {      background-color: #409EFF !important;      color: white;     }    .el-checkbox\_\_input.is-checked+.el-checkbox\_\_label {      color: black;     }  </style> |

## 3、根据排班日期获取排班详情列表

### 3.1 api接口

#### 3.1.1 添加repository接口

在ScheduleRepository类添加接口

|  |
| --- |
| *//根据医院编号 、科室编号和工作日期，查询排班详细信息*  List<Schedule> findScheduleByHoscodeAndDepcodeAndWorkDate(String hoscode, String depcode, Date toDate); |

#### 3.1.2 添加service接口与实现

在ScheduleService类添加接口

|  |
| --- |
| *//根据医院编号 、科室编号和工作日期，查询排班详细信息*  List<Schedule> getDetailSchedule(String hoscode, String depcode, String workDate); |

在ScheduleServiceImpl类实现接口

|  |
| --- |
| *//根据医院编号 、科室编号和工作日期，查询排班详细信息*  @Override  **public** List<Schedule> getDetailSchedule(String hoscode, String depcode, String workDate) {  *//根据参数查询mongodb*  List<Schedule> scheduleList =  **scheduleRepository**.findScheduleByHoscodeAndDepcodeAndWorkDate(hoscode,depcode,**new** DateTime(workDate).toDate());  *//把得到list集合遍历，向设置其他值：医院名称、科室名称、日期对应星期*  scheduleList.stream().forEach(item->{  **this**.packageSchedule(item);  });  **return** scheduleList;  }  *//封装排班详情其他值 医院名称、科室名称、日期对应星期*  **private void** packageSchedule(Schedule schedule) {  *//设置医院名称*  schedule.getParam().put(**"hosname"**,**hospitalService**.getHospName(schedule.getHoscode()));  *//设置科室名称*  schedule.getParam().put(**"depname"**,**departmentService**.getDepName(schedule.getHoscode(),schedule.getDepcode()));  *//设置日期对应星期*  schedule.getParam().put(**"dayOfWeek"**,**this**.getDayOfWeek(**new** DateTime(schedule.getWorkDate())));  } |

#### 3.1.3 添加根据部门编码获取部门名称

1，在DepartmentService类添加接口

|  |
| --- |
| *//根据科室编号，和医院编号，查询科室名称*  String getDepName(String hoscode, String depcode); |

2，在DepartmentService类添加接口实现

|  |
| --- |
| *//根据科室编号，和医院编号，查询科室名称*  @Override  **public** String getDepName(String hoscode, String depcode) {  Department department = **departmentRepository**.getDepartmentByHoscodeAndDepcode(hoscode, depcode);  **if**(department != **null**) {  **return** department.getDepname();  }  **return null**;  } |

#### 3.1.4 添加controller

|  |
| --- |
| *//根据医院编号 、科室编号和工作日期，查询排班详细信息*  @ApiOperation(value = **"查询排班详细信息"**)  @GetMapping(**"getScheduleDetail/{hoscode}/{depcode}/{workDate}"**)  **public** Result getScheduleDetail( @PathVariable String hoscode,  @PathVariable String depcode,  @PathVariable String workDate) {  List<Schedule> list = **scheduleService**.getDetailSchedule(hoscode,depcode,workDate);  **return** Result.*ok*(list);  } |

### 3.2 排班日期前端

#### 3.2.1封装api请求

在/api/hosp/schedule.js文件添加方法

|  |
| --- |
| //查询排班详情    getScheduleDetail(hoscode,depcode,workDate) {      return request ({        url: `/admin/hosp/schedule/getScheduleDetail/${hoscode}/${depcode}/${workDate}`,        method: 'get'      })    } |

#### 1.2.2 页面展示

修改/views/hosp/schedule.vue组件

|  |
| --- |
| <template>      <div class="app-container">          <div style="margin-bottom: 10px;font-size: 10px;">               选择  ：{{ baseMap.hosname }}  /  {{ depname }}  /  {{ workDate }}</div>              <el-container style="height: 100%">              <el-aside width="200px" style="border: 1px silver solid">                  <!-- 部门 -->                  <el-tree                  :data="data"                  :props="defaultProps"                  :default-expand-all="true"                  @node-click="handleNodeClick">                  </el-tree>              </el-aside>              <el-main style="padding: 0 0 0 20px;">                  <el-row style="width: 100%">                      <!-- 排班日期 分页 -->                      <el-tag v-for="(item,index) in bookingScheduleList" :key="item.id" @click="selectDate(item.workDate, index)" :type="index == activeIndex ? '' : 'info'" style="height: 60px;margin-right: 5px;margin-right:15px;cursor:pointer;">                          {{ item.workDate }} {{ item.dayOfWeek }}<br/>                          {{ item.availableNumber }} / {{ item.reservedNumber }}                      </el-tag>                      <!-- 分页 -->                      <el-pagination                          :current-page="page"                          :total="total"                          :page-size="limit"                          class="pagination"                          layout="prev, pager, next"                          @current-change="getPage">                      </el-pagination>                  </el-row>                  <el-row style="margin-top: 20px;">                  <!-- 排班日期对应的排班医生 -->                      <el-table                          v-loading="listLoading"                          :data="scheduleList"                          border                          fit                          highlight-current-row>                          <el-table-column                                      label="序号"                                      width="60"                                      align="center">                              <template slot-scope="scope">                                              {{ scope.$index + 1 }}                              </template>                          </el-table-column>                          <el-table-column label="职称" width="150">                              <template slot-scope="scope">                                              {{ scope.row.title }} | {{ scope.row.docname }}                              </template>                          </el-table-column>                          <el-table-column label="号源时间" width="80">                              <template slot-scope="scope">                                              {{ scope.row.workTime == 0 ? "上午" : "下午" }}                              </template>                          </el-table-column>                          <el-table-column prop="reservedNumber" label="可预约数" width="80"/>                          <el-table-column prop="availableNumber" label="剩余预约数" width="100"/>                          <el-table-column prop="amount" label="挂号费(元)" width="90"/>                          <el-table-column prop="skill" label="擅长技能"/>                      </el-table>                  </el-row>              </el-main>          </el-container>      </div>  </template>  <script>  import hospApi from '@/api/hosp'  export default {      data() {          return {              data: [],              defaultProps: {                  children: 'children',                  label: 'depname'              },              hoscode: null,              activeIndex: 0,              depcode: null,              depname: null,              workDate: null,              bookingScheduleList: [],              baseMap: {},              page: 1, // 当前页              limit:7, // 每页个数              total: 0, // 总页码              scheduleList:[] //排班详情          }      },      created(){          this.hoscode = this.$route.params.hoscode          this.workDate = this.getCurDate()          this.fetchData()      },      methods:{          //查询排班详情          getDetailSchedule() {              hospApi.getScheduleDetail(this.hoscode,this.depcode,this.workDate)                  .then(response => {                      this.scheduleList = response.data                  })          },          fetchData() {              hospApi.getDeptByHoscode(this.hoscode)                  .then(response => {                      this.data = response.data                      // 默认选中第一个                      if (this.data.length > 0) {                          this.depcode = this.data[0].children[0].depcode                          this.depname = this.data[0].children[0].depname                          this.getPage()                      }                })          },          getPage(page = 1) {              this.page = page              this.workDate = null              this.activeIndex = 0              this.getScheduleRule()          },          getScheduleRule() {              hospApi.getScheduleRule(this.page, this.limit, this.hoscode, this.depcode).then(response => {                  this.bookingScheduleList = response.data.bookingScheduleRuleList                  this.total = response.data.total                  this.scheduleList = response.data.scheduleList                  this.baseMap = response.data.baseMap                  // 分页后workDate=null，默认选中第一个                  if (this.workDate == null) {                        this.workDate = this.bookingScheduleList[0].workDate                  }                  //调用查询排班详情                  this.getDetailSchedule()              })          },          handleNodeClick(data) {              // 科室大类直接返回              if (data.children != null) return               this.depcode = data.depcode               this.depname = data.depname               this.getPage(1)          },          selectDate(workDate, index) {              this.workDate = workDate              this.activeIndex = index              //调用查询排班详情              this.getDetailSchedule()          },          getCurDate() {              var datetime = new Date()              var year = datetime.getFullYear()              var month = datetime.getMonth() + 1 < 10 ? '0' + (datetime.getMonth() + 1) : datetime.getMonth() + 1              var date = datetime.getDate() < 10 ? '0' + datetime.getDate() : datetime.getDate()              return year + '-' + month + '-' + date          }      }  }  </script>  <style>    .el-tree-node.is-current > .el-tree-node\_\_content {      background-color: #409EFF !important;      color: white;     }    .el-checkbox\_\_input.is-checked+.el-checkbox\_\_label {      color: black;     }  </style> |

# 服务网关

## 1、网关介绍

API网关出现的原因是微服务架构的出现，不同的微服务一般会有不同的网络地址，而外部客户端可能需要调用多个服务的接口才能完成一个业务需求，如果让客户端直接与各个微服务通信，会有以下的问题：

（1）客户端会多次请求不同的微服务，增加了客户端的复杂性。

（2）存在跨域请求，在一定场景下处理相对复杂。

（3）认证复杂，每个服务都需要独立认证。

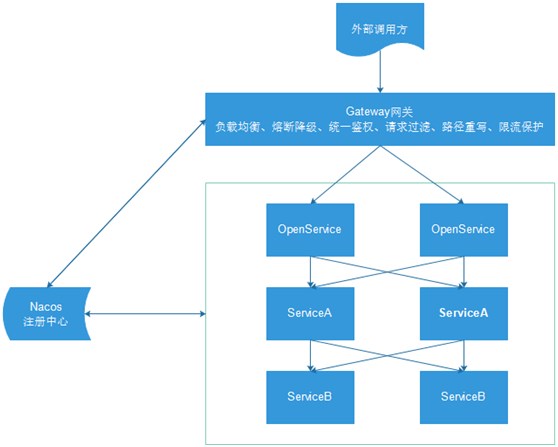
（4）难以重构，随着项目的迭代，可能需要重新划分微服务。例如，可能将多个服务合并成一个或者将一个服务拆分成多个。如果客户端直接与微服务通信，那么重构将会很难实施。

（5）某些微服务可能使用了防火墙 / 浏览器不友好的协议，直接访问会有一定的困难。

以上这些问题可以借助 API 网关解决。API 网关是介于客户端和服务器端之间的中间层，所有的外部请求都会先经过API 网关这一层。也就是说，API 的实现方面更多的考虑业务逻辑，而安全、性能、监控可以交由 API 网关来做，这样既提高业务灵活性又不缺安全性

## 2、Spring Cloud Gateway介绍

Spring cloud gateway是spring官方基于Spring 5.0、Spring Boot2.0和Project Reactor等技术开发的网关，Spring Cloud Gateway旨在为微服务架构提供简单、有效和统一的API路由管理方式，Spring Cloud Gateway作为Spring Cloud生态系统中的网关，目标是替代Netflix Zuul，其不仅提供统一的路由方式，并且还基于Filer链的方式提供了网关基本的功能，例如：安全、监控/埋点、限流等



## 3、搭建server-gateway模块

服务网关

### 3.1 搭建server-gateway

搭建过程如common模块

### 3.2 修改配置pom.xml

修改pom.xml

|  |
| --- |
| <**dependencies**>  <**dependency**>  <**groupId**>com.atguigu.yygh</**groupId**>  <**artifactId**>common-util</**artifactId**>  <**version**>1.0</**version**>  </**dependency**>  <**dependency**>  <**groupId**>org.springframework.cloud</**groupId**>  <**artifactId**>spring-cloud-starter-gateway</**artifactId**>  </**dependency**>  *<!-- 服务注册 -->*  <**dependency**>  <**groupId**>com.alibaba.cloud</**groupId**>  <**artifactId**>spring-cloud-starter-alibaba-nacos-discovery</**artifactId**>  </**dependency**>  </**dependencies**> |

### 3.3 在resources下添加配置文件

1、application.properties

|  |
| --- |
| *# 服务端口*  **server.port**=**80**  *# 服务名*  **spring.application.name**=**service-gateway**  *# nacos服务地址*  **spring.cloud.nacos.discovery.server-addr**=**127.0.0.1:8848**  *#使用服务发现路由*  **spring.cloud.gateway.discovery.locator.enabled**=**true**  *#设置路由id*  **spring.cloud.gateway.routes[0].id**=**service-hosp**  *#设置路由的uri*  **spring.cloud.gateway.routes[0].uri**=**lb://service-hosp**  *#设置路由断言,代理servicerId为auth-service的/auth/路径*  **spring.cloud.gateway.routes[0].predicates**= **Path=/\*/hosp/\*\***  *#设置路由id*  **spring.cloud.gateway.routes[1].id**=**service-cmn**  *#设置路由的uri*  **spring.cloud.gateway.routes[1].uri**=**lb://service-cmn**  *#设置路由断言,代理servicerId为auth-service的/auth/路径*  **spring.cloud.gateway.routes[1].predicates**= **Path=/\*/cmn/\*\*** |

### 3.4添加启动类

|  |
| --- |
| **package** com.atguigu.yygh;  **import** org.springframework.boot.SpringApplication;  **import** org.springframework.boot.autoconfigure.SpringBootApplication;  @SpringBootApplication  **public class** ServerGatewayApplication {  **public static void** main(String[] args) {  SpringApplication.*run*(ServerGatewayApplication.**class**, args);  }  } |

### 3.5 跨域处理

跨域：浏览器对于javascript的同源策略的限制 。

以下情况都属于跨域：

| 跨域原因说明 | 示例 |
| --- | --- |
| 域名不同 | www.jd.com 与 www.taobao.com |
| 域名相同，端口不同 | www.jd.com:8080 与 www.jd.com:8081 |
| 二级域名不同 | item.jd.com 与 miaosha.jd.com |

如果域名和端口都相同，但是请求路径不同，不属于跨域，如：

www.jd.com/item

www.jd.com/goods

http和https也属于跨域

而我们刚才是从localhost:1000去访问localhost:8888，这属于端口不同，跨域了。

**3.5.1 为什么有跨域问题？**

跨域不一定都会有跨域问题。

因为跨域问题是浏览器对于ajax请求的一种安全限制：一个页面发起的ajax请求，只能是与当前页域名相同的路径，这能有效的阻止跨站攻击。

因此：跨域问题 是针对ajax的一种限制。

但是这却给我们的开发带来了不便，而且在实际生产环境中，肯定会有很多台服务器之间交互，地址和端口都可能不同，怎么办？

**3.5.2解决跨域问题**

全局配置类实现

CorsConfig类

|  |
| --- |
| @Configuration  **public class** CorsConfig {  @Bean  **public** CorsWebFilter corsFilter() {  CorsConfiguration config = **new** CorsConfiguration();  config.addAllowedMethod(**"\*"**);  config.addAllowedOrigin(**"\*"**);  config.addAllowedHeader(**"\*"**);  UrlBasedCorsConfigurationSource source = **new** UrlBasedCorsConfigurationSource(**new** PathPatternParser());  source.registerCorsConfiguration(**"/\*\*"**, config);  **return new** CorsWebFilter(source);  }  } |

### 3.6服务调整

目前我们已经在网关做了跨域处理，那么service服务就不需要再做跨域处理了，将之前在controller类上添加过@CrossOrigin标签的去掉，防止程序异常

### 3.7测试

通过平台与管理平台前端测试