# 实验三 智能制造系统库存管理模块设计与实现

# 1 EV-MES库存模块功能分析

## 1.1功能模块划分

**1.1.1 库存基础管理模块**

（1）物料信息管理：支持新建物料编码、名称、规格型号、存放位置等基本信息管理。

（2）库存数量管理：实现库存数量的增加、减少、更新等操作。

（3）物料查询功能：提供物料列表展示，支持分页显示和物料名称/编码模糊搜索。

（4）物料编辑功能：支持编辑现有物料的所有字段信息。

（5）物料删除功能：提供物料删除操作，支持数据完整性保护。

**1.1.2二维码模块**

（1）二维码生成功能：自动为每个物料生成包含物料信息的二维码。

（2）二维码识别功能：支持查看和扫描物料的二维码图片。

**1.1.3库存可视化模块**

（1）实时库存统计：提供总物料种类、总库存量等关键指标统计。

（2）统计图表展示：提供库存位置分布、数量分布、类别分布等图表。

（3）数据汇总展示：以卡片形式展示关键库存指标。

## 1.2系统功能图

如图1-1所示，整个库存管理划分为基础管理、二维码管理与可视化三大模块：基础模块集中维护物料信息及库存数量，支持完整CRUD；二维码模块为每件物料赋予唯一数字身份，实现生成与识别；可视化模块则以卡片和图表形式实时呈现库存总量、分布等，确保库存管理完善。



图1-1 系统功能图

# 2 EV-MES库存管理功能流程图

如图2-1所示，整个库存管理流程以“录入—生成—入库—出库—统计”为主线，包括数量校验与预警，实现物料闭环管理。

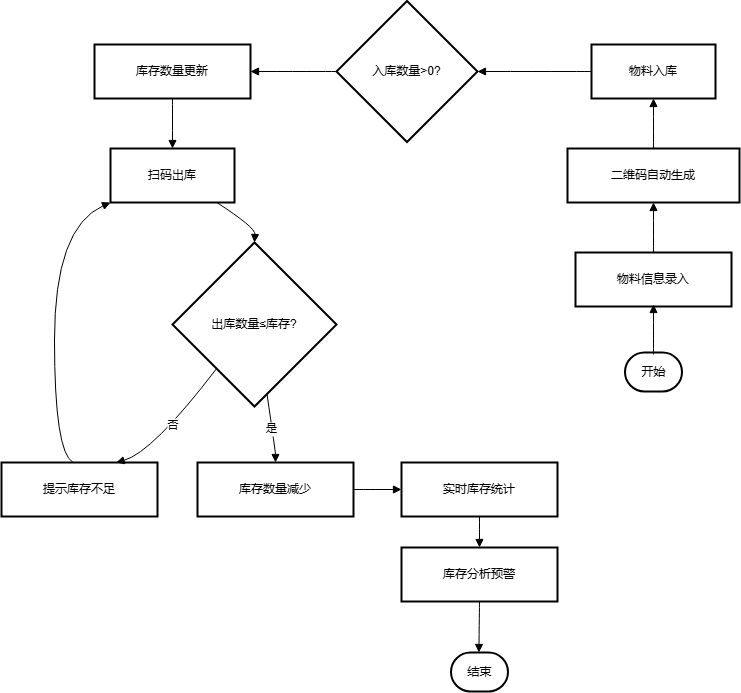


图2-1 功能流程图

# 3库存管理功能设计与实现

## 3.1 数据表设计

表3-1 inventory\_items表

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 约束及含义 |
| id  part\_code | Integer  String(50) | 主键，自增  物料编码 |
| name  spec  quantity  location  created\_at | String(100)  String(100)  Integer  String(50)  String(50) | 物料名称  规格型号  库存数量  存放位置  创建时间 |

## 3.2库存列表功能实现

（1）实现图

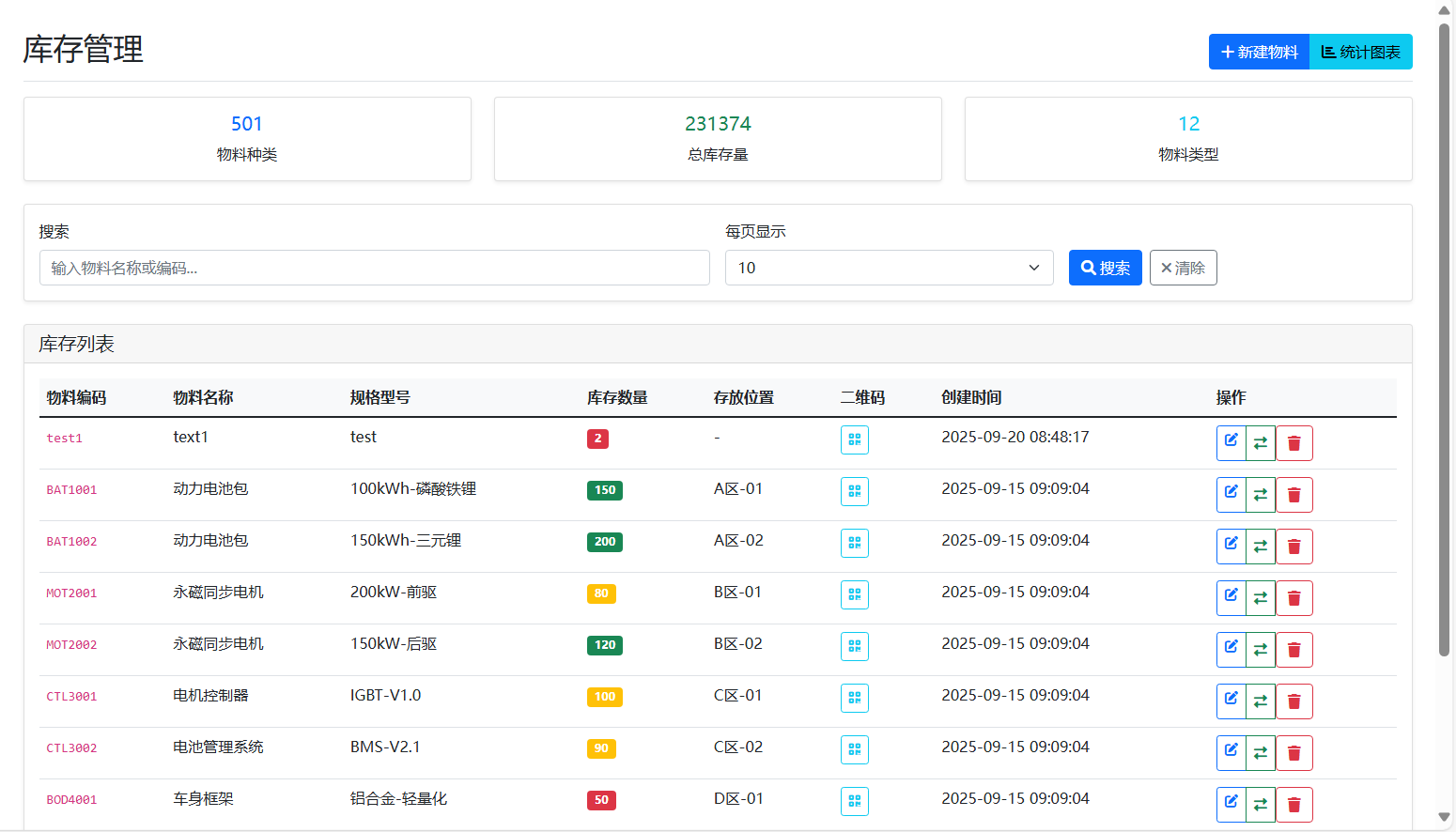


图3-1 库存列表功能

（2）核心代码

|  |
| --- |
| @inventory\_bp.route('/')  def page\_inventory\_list():  try:  db = session\_factory()  inventory\_service = InventoryService(db)    page = int(request.args.get('page', 1))  per\_page = int(request.args.get('per\_page', 10))  search = request.args.get('search', '')    result = inventory\_service.get\_items(page=page, per\_page=per\_page, search=search)    stats = inventory\_service.get\_inventory\_statistics()    return render\_template('inventory/list.html',  items=result['items'],  pagination=result,  search=search,  stats=stats)  except Exception as e:  flash(f'获取库存列表失败: {str(e)}', 'error')  return render\_template('inventory/list.html', items=[], pagination={}, search='', stats={})  finally:  db.close() |

## 3.3 编辑库存功能实现

（1）实现图

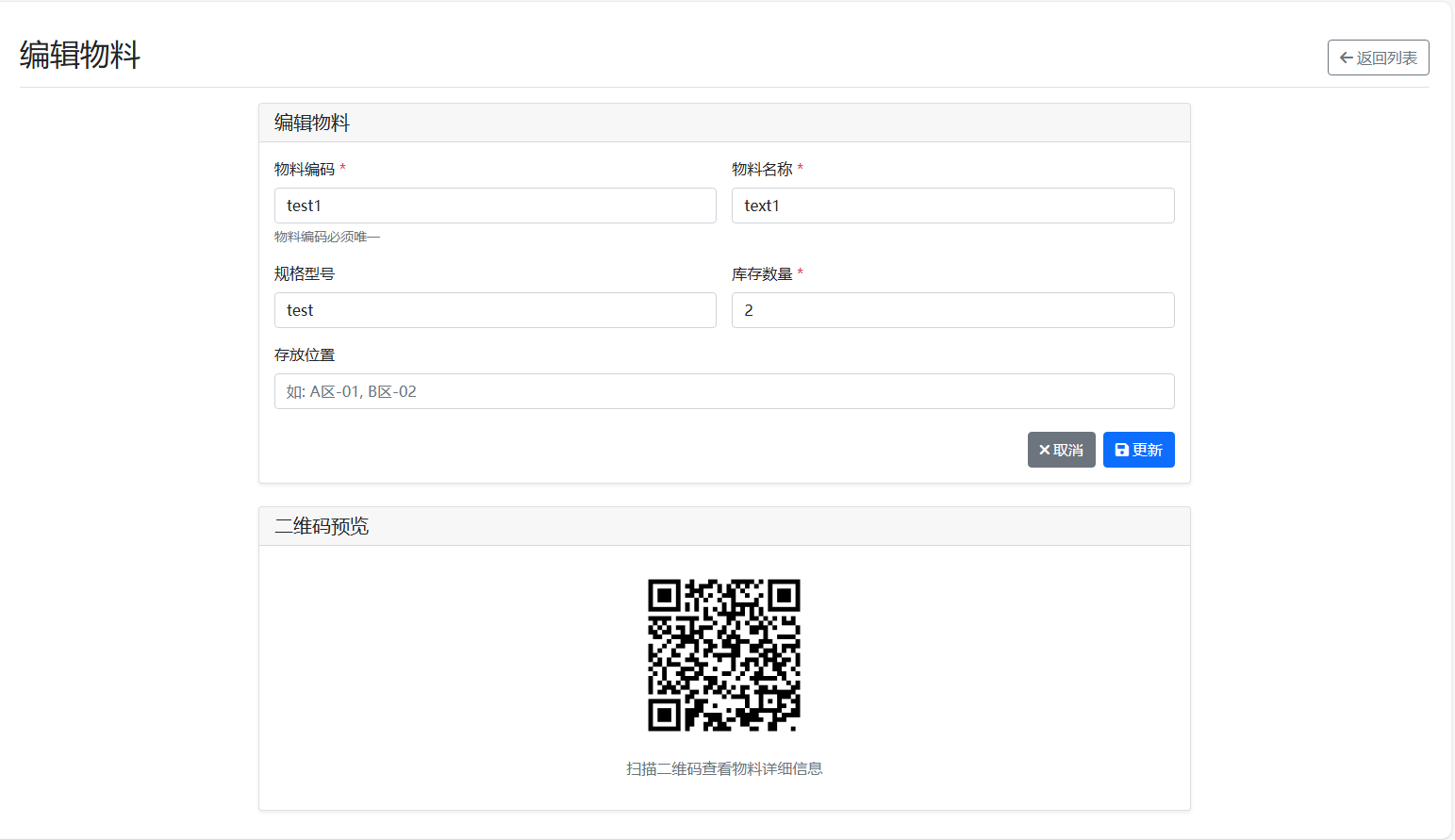


图3-2 编辑库存功能

（2）核心代码

|  |
| --- |
| @inventory\_bp.route('/<int:item\_id>/edit', methods=['GET', 'POST'])  def page\_inventory\_edit(item\_id):  try:  db = session\_factory()  inventory\_service = InventoryService(db)    if request.method == 'GET':  item = inventory\_service.get\_item\_by\_id(item\_id)  if not item:  flash('库存物料不存在', 'error')  return redirect(url\_for('inventory.page\_inventory\_list'))    return render\_template('inventory/form.html', item=item.to\_dict())    item\_data = {  'part\_code': request.form.get('part\_code'),  'name': request.form.get('name'),  'spec': request.form.get('spec'),  'quantity': int(request.form.get('quantity', 0)),  'location': request.form.get('location')  }    item = inventory\_service.update\_item(item\_id, item\_data)  if not item:  flash('库存物料不存在', 'error')  return redirect(url\_for('inventory.page\_inventory\_list'))    flash('库存物料更新成功', 'success')  return redirect(url\_for('inventory.page\_inventory\_list'))    except Exception as e:  flash(f'更新库存物料失败: {str(e)}', 'error')  return redirect(url\_for('inventory.page\_inventory\_edit', item\_id=item\_id))  finally:  db.close() |

## 3.4 可视化统计功能实现

（1）实现图

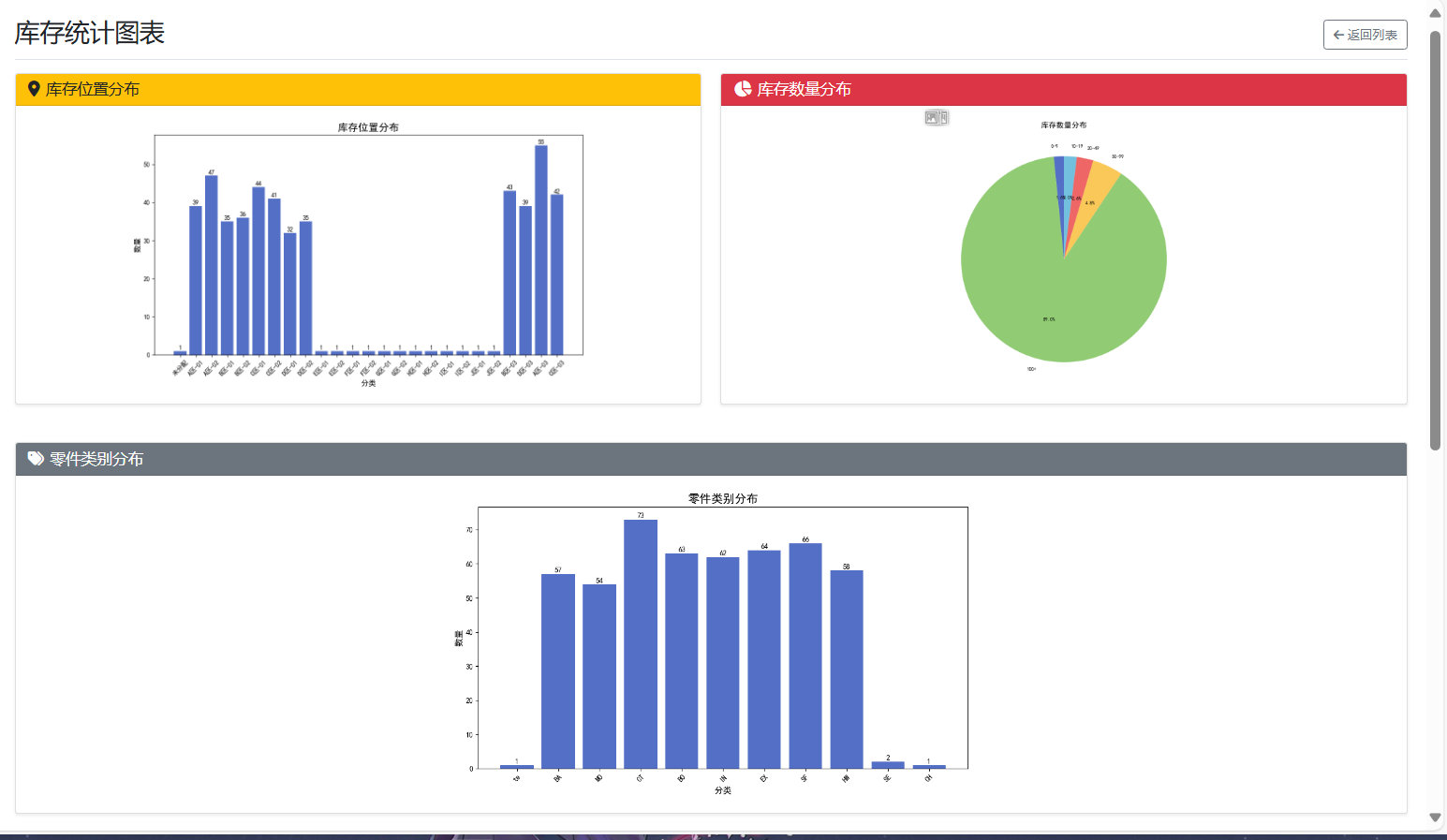


图3-3 可视化统计功能

（2）核心代码

|  |
| --- |
| @inventory\_bp.route('/charts')  def page\_inventory\_charts():  try:  db = session\_factory()  inventory\_service = InventoryService(db)    result = inventory\_service.get\_items(page=1, per\_page=1000)  items = result['items']    charts = create\_inventory\_charts(items)    return render\_template('inventory/charts.html',  location\_chart=charts.get('location\_chart', ''),  quantity\_chart=charts.get('quantity\_chart', ''),  category\_chart=charts.get('category\_chart', ''),  items=items)    except Exception as e:  flash(f'获取统计图表失败: {str(e)}', 'error')  return render\_template('inventory/charts.html',  location\_chart='',  quantity\_chart='',  category\_chart='',  items=[])  finally:  db.close() top\_customers = dict(sorted(customer\_stats.items(), key=lambda x: x[1], reverse=True)[:10])    status\_chart = ''  customer\_chart = ''    if translated\_status\_stats:  status\_chart = MatplotlibCharts.create\_pie\_chart(  title='订单状态分布',  data=translated\_status\_stats  )    if top\_customers:  customer\_chart = MatplotlibCharts.create\_bar\_chart(  title='客户订单数量TOP10',  data=top\_customers  )    return {  'status\_chart': status\_chart,  'customer\_chart': customer\_chart  }    except Exception as e:  print(f"创建订单图表失败: {e}")  return {'status\_chart': '', 'customer\_chart': ''} |