```
// 系统：绿色智能化学解决方案系统
// 功能：化学品目录管理
// 描述：管理化学品基础信息库，记录化学品的标准名称、唯一标识 CAS 号及分子结构式，支持建立完整的化学品分类体系
import javax.persistence.*;
import java.time.LocalDateTime;
import java.util.List;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
/**
 * 化学品目录实体类
 */
@Entity
@Table(name = "chem_catalog")
public class ChemCatalog {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "chem_catalog_id")
    private Long id;  // 主键 ID
    @Column(name = "chemical_name", nullable = false, length = 255)
    private String chemicalName;  // 化学品名称
    @Column(name = "cas_number", nullable = false, length = 50, unique = true)
    private String casNumber;  // CAS 号
    @Column(name = "molecular_formula", nullable = false, length = 255)
    private String molecularFormula;  // 分子式
    @Column(name = "create_time", nullable = false)
    private LocalDateTime createTime;  // 创建时间
    @Column(name = "update_time")
    private LocalDateTime updateTime;  // 更新时间
    @Enumerated(EnumType.STRING)
    @Column(name = "record_status", nullable = false, length = 10)
    private RecordStatus recordStatus;  // 记录状态
    @Column(name = "category_code", length = 20)
    private String categoryCode;  // 分类代码
    @Enumerated(EnumType.STRING)
    @Column(name = "hazard_level", nullable = false, length = 10)
    private HazardLevel hazardLevel;  // 危险等级
    @Column(name = "storage_condition", length = 100)
    private String storageCondition;  // 存储条件
    // 枚举定义
    public enum RecordStatus {
        启用, 停用, 草稿
    }
    public enum HazardLevel {
        无危险, 低危, 中危, 高危
    }
    // Getter 和 Setter 方法
    public Long getId() { return id; }
    public void setId(Long id) { this.id = id; }
    public String getChemicalName() { return chemicalName; }
    public void setChemicalName(String chemicalName) { this.chemicalName = chemicalName; }
    public String getCasNumber() { return casNumber; }
    public void setCasNumber(String casNumber) { this.casNumber = casNumber; }
    public String getMolecularFormula() { return molecularFormula; }
    public void setMolecularFormula(String molecularFormula) { this.molecularFormula =
molecularFormula; }
    public LocalDateTime getCreateTime() { return createTime; }
    public void setCreateTime(LocalDateTime createTime) { this.createTime = createTime; }
    public LocalDateTime getUpdateTime() { return updateTime; }
    public void setUpdateTime(LocalDateTime updateTime) { this.updateTime = updateTime; }
    public RecordStatus getRecordStatus() { return recordStatus; }
    public void setRecordStatus(RecordStatus recordStatus) { this.recordStatus = recordStatus; }
    public String getCategoryCode() { return categoryCode; }
    public void setCategoryCode(String categoryCode) { this.categoryCode = categoryCode; }
    public HazardLevel getHazardLevel() { return hazardLevel; }
    public void setHazardLevel(HazardLevel hazardLevel) { this.hazardLevel = hazardLevel; }
    public String getStorageCondition() { return storageCondition; }
    public void setStorageCondition(String storageCondition) { this.storageCondition =
storageCondition; }
}
/**
 * 化学品目录数据访问接口
 */
public interface ChemCatalogRepository extends JpaRepository<ChemCatalog, Long> {
    // 根据 CAS 号查询
    ChemCatalog findByCasNumber(String casNumber);
    // 根据状态查询
    List<ChemCatalog> findByRecordStatus(ChemCatalog.RecordStatus status);
}
/**
 * 化学品目录服务实现
 */
@Service
@Transactional
public class ChemCatalogService {
    private final ChemCatalogRepository repository;
    public ChemCatalogService(ChemCatalogRepository repository) {
        this.repository = repository;
    }
    /**
     * 新增化学品目录
     * @param chemCatalog 化学品对象
     * @return 保存后的对象
     */
    public ChemCatalog createChemCatalog(ChemCatalog chemCatalog) {
        chemCatalog.setCreateTime(LocalDateTime.now());
        chemCatalog.setRecordStatus(ChemCatalog.RecordStatus.草稿);
        return repository.save(chemCatalog);
    }
    /**
     * 更新化学品目录属性
     * @param id 目标化学品 ID
     * @param updatedCatalog 更新数据对象
     * @return 更新后的对象
     */
    public ChemCatalog updateChemCatalog(Long id, ChemCatalog updatedCatalog) {
        return repository.findById(id).map(existing -> {
            if (updatedCatalog.getChemicalName() != null) {
                existing.setChemicalName(updatedCatalog.getChemicalName());
            }
            if (updatedCatalog.getMolecularFormula() != null) {
                existing.setMolecularFormula(updatedCatalog.getMolecularFormula());
            }
            if (updatedCatalog.getRecordStatus() != null) {
                existing.setRecordStatus(updatedCatalog.getRecordStatus());
            }
            if (updatedCatalog.getCategoryCode() != null) {
                existing.setCategoryCode(updatedCatalog.getCategoryCode());
            }
            if (updatedCatalog.getHazardLevel() != null) {
                existing.setHazardLevel(updatedCatalog.getHazardLevel());
            }
            if (updatedCatalog.getStorageCondition() != null) {
                existing.setStorageCondition(updatedCatalog.getStorageCondition());
            }
            existing.setUpdateTime(LocalDateTime.now());
            return repository.save(existing);
        }).orElseThrow(() -> new RuntimeException("化学品记录不存在"));
    }
    /**
     * 删除化学品目录
     * @param id 目标化学品 ID
     */
    public void deleteChemCatalog(Long id) {
        repository.deleteById(id);
    }
    /**
     * 获取所有化学品目录
     * @return 化学品列表
     */
    public List<ChemCatalog> getAllChemCatalogs() {
        return repository.findAll();
    }
    /**
     * 根据 ID 获取化学品详情
     * @param id 目标化学品 ID
     * @return 化学品对象
     */
    public ChemCatalog getChemCatalogById(Long id) {
        return repository.findById(id)
                .orElseThrow(() -> new RuntimeException("化学品记录不存在"));
    }
}
<!DOCTYPE html>
```

```
<html lang="zh-CN">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>化学品目录管理</title>
  <link rel="stylesheet" href="https://unpkg.com/element-plus/dist/index.css">
  <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
  <script src="https://unpkg.com/element-plus/dist/index.full.js"></script>
  <script src="https://unpkg.com/axios/dist/axios.min.js"></script>
  <style>
    .page-container {
      padding: 20px;
      background-color: #f5f7fa;
    }
    .toolbar {
      margin-bottom: 20px;
    }
    .form-container {
      padding: 20px;
    }
  </style>
</head>
<body>
  <div id="app">
    <el-container class="page-container">
      <!-- 主内容区 -->
      <el-main>
        <!-- 工具栏 -->
        <div class="toolbar">
          <el-button type="primary" @click="openCreateDialog">新增目录</el-button>
        </div>
        <!-- 数据表格 -->
        <el-table :data="chemCatalogs" border style="width: 100%">
          <el-table-column prop="chemicalName" label="化学品名称" width="180"></el-table-column>
          <el-table-column prop="casNumber" label="CAS 号" width="150"></el-table-column>
          <el-table-column prop="molecularFormula" label="分子式" width="120"></el-table-column>
          <el-table-column prop="recordStatus" label="状态" width="100">
            <template #default="scope">
              <el-tag :type="statusTagType(scope.row.recordStatus)">
                {{ scope.row.recordStatus }}
              </el-tag>
            </template>
          </el-table-column>
          <el-table-column prop="hazardLevel" label="危险等级" width="100">
            <template #default="scope">
              <el-tag :type="hazardTagType(scope.row.hazardLevel)">
                {{ scope.row.hazardLevel }}
              </el-tag>
            </template>
          </el-table-column>
          <el-table-column prop="categoryCode" label="分类代码" width="120"></el-table-column>
          <el-table-column prop="storageCondition" label="存储条件"></el-table-column>
          <el-table-column label="操作" width="200">
            <template #default="scope">
              <el-button size="small" @click="openEditDialog(scope.row)">编辑属性</el-button>
              <el-button size="small" type="danger" @click="confirmDelete(scope.row)">删除</el-
    button>
            </template>
          </el-table-column>
        </el-table>
      </el-main>
    </el-container>
    <!-- 新增/编辑弹窗 -->
    <el-dialog :title="dialogTitle" v-model="dialogVisible" width="600px">
      <div class="form-container">
        <el-form :model="currentChem" label-width="120px" ref="chemForm">
          <el-form-item label="化学品名称" prop="chemicalName" required>
            <el-input v-model="currentChem.chemicalName"></el-input>
          </el-form-item>
          <el-form-item label="CAS 号" prop="casNumber" required>
            <el-input v-model="currentChem.casNumber"></el-input>
          </el-form-item>
          <el-form-item label="分子式" prop="molecularFormula" required>
            <el-input v-model="currentChem.molecularFormula"></el-input>
          </el-form-item>
          <el-form-item label="记录状态" prop="recordStatus" required>
            <el-select v-model="currentChem.recordStatus">
              <el-option label="启用" value="启用"></el-option>
              <el-option label="停用" value="停用"></el-option>
              <el-option label="草稿" value="草稿"></el-option>
            </el-select>
          </el-form-item>
          <el-form-item label="危险等级" prop="hazardLevel" required>
            <el-select v-model="currentChem.hazardLevel">
              <el-option label="无危险" value="无危险"></el-option>
              <el-option label="低危" value="低危"></el-option>
              <el-option label="中危" value="中危"></el-option>
              <el-option label="高危" value="高危"></el-option>
            </el-select>
          </el-form-item>
          <el-form-item label="分类代码" prop="categoryCode">
            <el-input v-model="currentChem.categoryCode"></el-input>
          </el-form-item>
          <el-form-item label="存储条件" prop="storageCondition">
            <el-input v-model="currentChem.storageCondition"></el-input>
          </el-form-item>
        </el-form>
      </div>
      <template #footer>
        <el-button @click="dialogVisible = false">取消</el-button>
        <el-button type="primary" @click="saveChemCatalog">保存</el-button>
      </template>
    </el-dialog>
```

```
</div>
<script>
  const { createApp, ref, reactive, onMounted } = Vue;
  const { ElMessage, ElMessageBox } = ElementPlus;
  createApp({
    setup() {
      // 化学品数据
      const chemCatalogs = ref([]);
      const currentChem = reactive({
        chemCatalogId: null,
        chemicalName: '',
        casNumber: '',
        molecularFormula: '',
        recordStatus: '草稿',
        categoryCode: '',
        hazardLevel: '无危险',
        storageCondition: ''
      });
      // 弹窗控制
      const dialogVisible = ref(false);
      const dialogTitle = ref('新增化学品目录');
      const chemForm = ref(null);
      // 初始化加载数据
      const loadChemCatalogs = async () => {
        try {
          const response = await axios.get('/api/chemCatalogs');
          chemCatalogs.value = response.data;
        } catch (error) {
          ElMessage.error('加载化学品目录失败');
        }
      };
      // 打开新增弹窗
      const openCreateDialog = () => {
        resetForm();
        dialogTitle.value = '新增化学品目录';
        dialogVisible.value = true;
      };
      // 打开编辑弹窗
      const openEditDialog = (chem) => {
        Object.assign(currentChem, chem);
        dialogTitle.value = '编辑化学品属性';
        dialogVisible.value = true;
      };
      // 保存化学品目录
      const saveChemCatalog = async () => {
        try {
          if (currentChem.chemCatalogId) {
            await axios.put(`/api/chemCatalogs/${currentChem.chemCatalogId}`, currentChem);
            ElMessage.success('更新成功');
          } else {
            await axios.post('/api/chemCatalogs', currentChem);
            ElMessage.success('新增成功');
          }
          dialogVisible.value = false;
          loadChemCatalogs();
        } catch (error) {
          ElMessage.error('保存失败');
        }
      };
      // 删除确认
      const confirmDelete = (chem) => {
        ElMessageBox.confirm(`确定删除化学品 "${chem.chemicalName}"?`, '警告', {
          confirmButtonText: '确定',
          cancelButtonText: '取消',
          type: 'warning'
        }).then(async () => {
          try {
            await axios.delete(`/api/chemCatalogs/${chem.chemCatalogId}`);
            ElMessage.success('删除成功');
            loadChemCatalogs();
          } catch (error) {
            ElMessage.error('删除失败');
          }
        }).catch(() => {});
      };
      // 重置表单
      const resetForm = () => {
        currentChem.chemCatalogId = null;
        currentChem.chemicalName = '';
        currentChem.casNumber = '';
        currentChem.molecularFormula = '';
        currentChem.recordStatus = '草稿';
        currentChem.categoryCode = '';
        currentChem.hazardLevel = '无危险';
        currentChem.storageCondition = '';
      };
      // 状态标签样式
      const statusTagType = (status) => {
        switch(status) {
          case '启用': return 'success';
          case '停用': return 'danger';
          default: return 'info';
        }
      };
      // 危险等级标签样式
      const hazardTagType = (level) => {
        switch(level) {
          case '无危险': return 'success';
          case '低危': return '';
          case '中危': return 'warning';
          case '高危': return 'danger';
          default: return 'info';
        }
```

```
        };
        // 生命周期钩子
        onMounted(() => {
          loadChemCatalogs();
        });
        return {
          chemCatalogs,
          currentChem,
          dialogVisible,
          dialogTitle,
          chemForm,
          openCreateDialog,
          openEditDialog,
          saveChemCatalog,
          confirmDelete,
          statusTagType,
          hazardTagType
        };
      }
    }).use(ElementPlus).mount('#app');
  </script>
</body>
</html>
// 系统：绿色智能化学解决方案系统
// 功能：化学品库存管理
// 描述：跟踪化学品实时库存状态，记录存储位置和环境条件，管理化学品的入库、出库及库存盘点流程
import javax.persistence.*;
import java.time.LocalDateTime;
import java.util.List;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Service;
import org.springframework.beans.factory.annotation.Autowired;
/**
 * 化学品库存实体类
 * 对应数据库表 chem_inventory
 */
@Entity
@Table(name = "chem_inventory")
public class ChemInventory {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "chem_inventory_id")
    private Long id; // 库存记录 ID
    @Column(name = "chem_catalog_id", nullable = false)
    private Long catalogId; // 化学品目录 ID
    @Column(name = "inventory_quantity", nullable = false, precision = 15, scale = 3)
    private Double quantity; // 当前库存量
    @Column(name = "storage_location", nullable = false, length = 200)
    private String location; // 存储位置
    @Column(name = "storage_temp", length = 50)
    private String temperature; // 存储温度要求
```

```
@Column(name = "container_type", length = 100)
private String containerType; // 容器类型
@Enumerated(EnumType.STRING)
@Column(name = "inventory_status", nullable = false, length = 20)
private InventoryStatus status; // 库存状态
@Column(name = "last_update_time", nullable = false)
private LocalDateTime updateTime; // 最后更新时间
// 库存状态枚举
public enum InventoryStatus {
   NORMAL, BELOW_SAFE, EXPIRED
}
// 构造方法
public ChemInventory() {}
public ChemInventory(Long catalogId, Double quantity, String location,
                String temperature, String containerType,
                InventoryStatus status) {
   this.catalogId = catalogId;
   this.quantity = quantity;
   this.location = location;
   this.temperature = temperature;
   this.containerType = containerType;
   this.status = status;
   this.updateTime = LocalDateTime.now();
}
// Getter 和 Setter 方法
public Long getId() { return id; }
public void setId(Long id) { this.id = id; }
public Long getCatalogId() { return catalogId; }
public void setCatalogId(Long catalogId) { this.catalogId = catalogId; }
public Double getQuantity() { return quantity; }
public void setQuantity(Double quantity) {
   this.quantity = quantity;
   this.updateTime = LocalDateTime.now();
}
public String getLocation() { return location; }
public void setLocation(String location) {
   this.location = location;
   this.updateTime = LocalDateTime.now();
}
public String getTemperature() { return temperature; }
public void setTemperature(String temperature) {
   this.temperature = temperature;
   this.updateTime = LocalDateTime.now();
}
public String getContainerType() { return containerType; }
public void setContainerType(String containerType) {
   this.containerType = containerType;
   this.updateTime = LocalDateTime.now();
}
public InventoryStatus getStatus() { return status; }
public void setStatus(InventoryStatus status) {
   this.status = status;
   this.updateTime = LocalDateTime.now();
```

```
    }
    public LocalDateTime getUpdateTime() { return updateTime; }
    public void setUpdateTime(LocalDateTime updateTime) {
        this.updateTime = updateTime;
    }
}
/**
 * 库存数据访问接口
 * 提供基础 CRUD 操作
 */
public interface InventoryRepository extends JpaRepository<ChemInventory, Long> {
    // 根据化学品目录 ID 查询库存
    List<ChemInventory> findByCatalogId(Long catalogId);
    // 根据库存状态查询
    List<ChemInventory> findByStatus(ChemInventory.InventoryStatus status);
}
/**
 * 化学品库存服务实现
 * 包含核心业务逻辑
 */
@Service
public class InventoryService {
    @Autowired
    private InventoryRepository repository;
    /**
     * 创建新库存记录（入库登记）
     * @param inventory 库存实体对象
     * @return 保存后的库存记录
     */
    public ChemInventory createInventory(ChemInventory inventory) {
        inventory.setUpdateTime(LocalDateTime.now());
        return repository.save(inventory);
    }
    /**
     * 更新库存信息（库存调整）
     * @param id 库存记录 ID
     * @param updatedInventory 更新后的库存对象
     * @return 更新后的库存记录
     * @throws RuntimeException 当记录不存在时
     */
    public ChemInventory updateInventory(Long id, ChemInventory updatedInventory) {
        return repository.findById(id)
                .map(inventory -> {
                    inventory.setCatalogId(updatedInventory.getCatalogId());
                    inventory.setQuantity(updatedInventory.getQuantity());
                    inventory.setLocation(updatedInventory.getLocation());
                    inventory.setTemperature(updatedInventory.getTemperature());
                    inventory.setContainerType(updatedInventory.getContainerType());
                    inventory.setStatus(updatedInventory.getStatus());
                    inventory.setUpdateTime(LocalDateTime.now());
                    return repository.save(inventory);
                })
                .orElseThrow(() -> new RuntimeException("库存记录不存在: " + id));
    }
    /**
     * 删除库存记录
     * @param id 要删除的库存记录 ID
     */
    public void deleteInventory(Long id) {
        repository.deleteById(id);
    }
    /**
     * 执行出库操作
     * @param id 库存记录 ID
     * @param reduceQuantity 出库数量
     * @return 更新后的库存记录
     * @throws RuntimeException 当库存不足或记录不存在时
     */
    public ChemInventory reduceInventory(Long id, Double reduceQuantity) {
        ChemInventory inventory = repository.findById(id)
                .orElseThrow(() -> new RuntimeException("库存记录不存在: " + id));
        if (inventory.getQuantity() < reduceQuantity) {
            throw new RuntimeException("库存不足，当前库存: " + inventory.getQuantity());
        }
        inventory.setQuantity(inventory.getQuantity() - reduceQuantity);
        inventory.setUpdateTime(LocalDateTime.now());
        // 自动更新库存状态
        if (inventory.getQuantity() <= 0) {
            inventory.setStatus(ChemInventory.InventoryStatus.BELOW_SAFE);
        }
        return repository.save(inventory);
    }
    /**
     * 获取所有库存记录
     * @return 库存记录列表
     */
    public List<ChemInventory> getAllInventories() {
        return repository.findAll();
    }
    /**
     * 根据 ID 获取单个库存记录
     * @param id 库存记录 ID
     * @return 库存实体对象
     * @throws RuntimeException 当记录不存在时
     */
    public ChemInventory getInventoryById(Long id) {
        return repository.findById(id)
                .orElseThrow(() -> new RuntimeException("库存记录不存在: " + id));
    }
}
<!DOCTYPE html>
<html lang="zh-CN">
<head>
```

```
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>化学品库存管理系统</title>
  <link rel="stylesheet" href="https://unpkg.com/element-plus/dist/index.css">
  <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
  <script src="https://unpkg.com/element-plus/dist/index.full.js"></script>
  <script src="https://unpkg.com/axios/dist/axios.min.js"></script>
  <style>
    .app-container { padding: 20px; }
    .toolbar { margin-bottom: 20px; }
    .pagination { margin-top: 20px; text-align: right; }
  </style>
</head>
<body>
  <div id="app">
    <el-config-provider :locale="zhCn">
      <div class="app-container">
        <h2>化学品库存管理</h2>
        <!-- 工具栏 -->
        <div class="toolbar">
          <el-button type="primary" @click="openAddDialog">入库登记</el-button>
          <el-button @click="refreshData">刷新数据</el-button>
        </div>
        <!-- 库存列表 -->
        <el-table :data="inventoryList" border stripe>
          <el-table-column prop="id" label="ID" width="80" />
          <el-table-column prop="catalogId" label="化学品 ID" width="120" />
          <el-table-column prop="quantity" label="库存量" width="120">
            <template #default="scope">
              {{ scope.row.quantity }} kg
            </template>
          </el-table-column>
          <el-table-column prop="location" label="存储位置" width="180" />
          <el-table-column prop="temperature" label="存储温度" width="120" />
          <el-table-column prop="containerType" label="容器类型" width="150" />
          <el-table-column prop="status" label="库存状态" width="150">
            <template #default="scope">
              <el-tag :type="statusTagType(scope.row.status)">
                {{ statusText(scope.row.status) }}
              </el-tag>
            </template>
          </el-table-column>
          <el-table-column prop="updateTime" label="最后更新" width="180" />
          <el-table-column label="操作" width="280">
            <template #default="scope">
              <el-button size="small" @click="openEditDialog(scope.row)">库存调整</el-button>
              <el-button size="small" type="warning" @click="openReduceDialog(scope.row)">出库操作
</el-button>
              <el-button size="small" type="danger" @click="deleteInventory(scope.row.id)">删除</el-
button>
            </template>
```

```
     </el-table-column>
    </el-table>
    <!-- 分页 -->
    <div class="pagination">
     <el-pagination
      v-model:current-page="pagination.currentPage"
      v-model:page-size="pagination.pageSize"
      :total="pagination.total"
      layout="total, sizes, prev, pager, next, jumper"
      @current-change="fetchData"
      @size-change="fetchData"
     />
    </div>
    <!-- 入库登记对话框 -->
    <el-dialog v-model="addDialogVisible" title="入库登记">
     <el-form :model="addForm" label-width="120px">
      <el-form-item label="化学品 ID" required>
       <el-input v-model="addForm.catalogId" type="number" />
      </el-form-item>
      <el-form-item label="库存量(kg)" required>
       <el-input v-model="addForm.quantity" type="number" />
      </el-form-item>
      <el-form-item label="存储位置" required>
       <el-input v-model="addForm.location" />
      </el-form-item>
      <el-form-item label="存储温度">
       <el-input v-model="addForm.temperature" />
      </el-form-item>
      <el-form-item label="容器类型">
       <el-input v-model="addForm.containerType" />
      </el-form-item>
      <el-form-item label="库存状态" required>
       <el-select v-model="addForm.status">
        <el-option label="正常" value="NORMAL" />
        <el-option label="低于安全库存" value="BELOW_SAFE" />
        <el-option label="过期" value="EXPIRED" />
       </el-select>
      </el-form-item>
     </el-form>
     <template #footer>
      <el-button @click="addDialogVisible = false">取消</el-button>
      <el-button type="primary" @click="addInventory">提交</el-button>
     </template>
    </el-dialog>
    <!-- 库存调整对话框 -->
    <el-dialog v-model="editDialogVisible" title="库存调整">
     <el-form :model="editForm" label-width="120px">
      <el-form-item label="化学品 ID" required>
       <el-input v-model="editForm.catalogId" type="number" disabled />
      </el-form-item>
      <el-form-item label="库存量(kg)" required>
```

```
            <el-input v-model="editForm.quantity" type="number" />
          </el-form-item>
          <el-form-item label="存储位置" required>
            <el-input v-model="editForm.location" />
          </el-form-item>
          <el-form-item label="存储温度">
            <el-input v-model="editForm.temperature" />
          </el-form-item>
          <el-form-item label="容器类型">
            <el-input v-model="editForm.containerType" />
          </el-form-item>
          <el-form-item label="库存状态" required>
            <el-select v-model="editForm.status">
              <el-option label="正常" value="NORMAL" />
              <el-option label="低于安全库存" value="BELOW_SAFE" />
              <el-option label="过期" value="EXPIRED" />
            </el-select>
          </el-form-item>
        </el-form>
        <template #footer>
          <el-button @click="editDialogVisible = false">取消</el-button>
          <el-button type="primary" @click="updateInventory">保存</el-button>
        </template>
      </el-dialog>
      <!-- 出库操作对话框 -->
      <el-dialog v-model="reduceDialogVisible" title="出库操作">
        <el-form :model="reduceForm" label-width="120px">
          <el-form-item label="化学品 ID">
            <el-input v-model="reduceForm.catalogId" disabled />
          </el-form-item>
          <el-form-item label="当前库存(kg)">
            <el-input v-model="reduceForm.currentQuantity" disabled />
          </el-form-item>
          <el-form-item label="出库数量(kg)" required>
            <el-input v-model="reduceForm.reduceQuantity" type="number" />
          </el-form-item>
        </el-form>
        <template #footer>
          <el-button @click="reduceDialogVisible = false">取消</el-button>
          <el-button type="primary" @click="reduceInventory">确认出库</el-button>
        </template>
      </el-dialog>
    </div>
  </el-config-provider>
</div>
<script>
  const { createApp, ref, reactive, onMounted } = Vue;
  const { ElMessage, ElMessageBox } = ElementPlus;
  const zhCn = ElementPlus.locales.zhCn;
  createApp({
    setup() {
```

```
// 响应式数据
const inventoryList = ref([]);
const addDialogVisible = ref(false);
const editDialogVisible = ref(false);
const reduceDialogVisible = ref(false);
// 表单数据
const addForm = reactive({
  catalogId: null,
  quantity: null,
  location: '',
  temperature: '',
  containerType: '',
  status: 'NORMAL'
});
const editForm = reactive({
  id: null,
  catalogId: null,
  quantity: null,
  location: '',
  temperature: '',
  containerType: '',
  status: 'NORMAL'
});
const reduceForm = reactive({
  id: null,
  catalogId: null,
  currentQuantity: null,
  reduceQuantity: null
});
// 分页配置
const pagination = reactive({
  currentPage: 1,
  pageSize: 10,
  total: 0
});
// 状态文本映射
const statusText = (status) => {
  switch(status) {
    case 'NORMAL': return '正常';
    case 'BELOW_SAFE': return '低于安全库存';
    case 'EXPIRED': return '过期';
    default: return status;
  }
};
// 状态标签类型
const statusTagType = (status) => {
  switch(status) {
    case 'NORMAL': return 'success';
    case 'BELOW_SAFE': return 'warning';
    case 'EXPIRED': return 'danger';
    default: return 'info';
  }
```

```
};
// 获取库存数据
const fetchData = async () => {
  try {
    const response = await axios.get('/api/inventories', {
      params: {
        page: pagination.currentPage - 1,
        size: pagination.pageSize
      }
    });
    inventoryList.value = response.data.content;
    pagination.total = response.data.totalElements;
  } catch (error) {
    ElMessage.error('获取库存数据失败: ' + error.message);
  }
};
// 刷新数据
const refreshData = () => {
  pagination.currentPage = 1;
  fetchData();
};
// 打开新增对话框
const openAddDialog = () => {
  addForm.catalogId = null;
  addForm.quantity = null;
  addForm.location = '';
  addForm.temperature = '';
  addForm.containerType = '';
  addForm.status = 'NORMAL';
  addDialogVisible.value = true;
};
// 提交新增库存
const addInventory = async () => {
  try {
    await axios.post('/api/inventories', addForm);
    ElMessage.success('入库登记成功');
    addDialogVisible.value = false;
    refreshData();
  } catch (error) {
    ElMessage.error('入库失败: ' + error.response?.data?.message || error.message);
  }
};
// 打开编辑对话框
const openEditDialog = (row) => {
  Object.assign(editForm, row);
  editDialogVisible.value = true;
};
// 更新库存信息
const updateInventory = async () => {
  try {
    await axios.put(`/api/inventories/${editForm.id}`, editForm);
    ElMessage.success('库存调整成功');
```

```
      editDialogVisible.value = false;
      refreshData();
    } catch (error) {
      ElMessage.error('更新失败: ' + error.response?.data?.message || error.message);
    }
  };
  // 打开出库对话框
  const openReduceDialog = (row) => {
    reduceForm.id = row.id;
    reduceForm.catalogId = row.catalogId;
    reduceForm.currentQuantity = row.quantity;
    reduceForm.reduceQuantity = null;
    reduceDialogVisible.value = true;
  };
  // 执行出库操作
  const reduceInventory = async () => {
    try {
      await axios.post(`/api/inventories/${reduceForm.id}/reduce`, {
        reduceQuantity: parseFloat(reduceForm.reduceQuantity)
      });
      ElMessage.success('出库操作成功');
      reduceDialogVisible.value = false;
      refreshData();
    } catch (error) {
      ElMessage.error('出库失败: ' + error.response?.data?.message || error.message);
    }
  };
  // 删除库存记录
  const deleteInventory = (id) => {
    ElMessageBox.confirm('确定要删除这条库存记录吗?', '警告', {
      confirmButtonText: '确定',
      cancelButtonText: '取消',
      type: 'warning'
    }).then(async () => {
      try {
        await axios.delete(`/api/inventories/${id}`);
        ElMessage.success('删除成功');
        refreshData();
      } catch (error) {
        ElMessage.error('删除失败: ' + error.message);
      }
    }).catch(() => {});
  };
  // 初始化加载数据
  onMounted(() => {
    fetchData();
  });
  return {
    zhCn,
    inventoryList,
    addDialogVisible,
    editDialogVisible,
          reduceDialogVisible,
          addForm,
          editForm,
          reduceForm,
          pagination,
          statusText,
          statusTagType,
          fetchData,
          refreshData,
          openAddDialog,
          addInventory,
          openEditDialog,
          updateInventory,
          openReduceDialog,
          reduceInventory,
          deleteInventory
        };
      }
    }).use(ElementPlus).mount('#app');
  </script>
</body>
</html>
// 系统：绿色智能化学解决方案系统
// 功能：安全数据表管理
// 描述：存储和管理化学品安全数据表(SDS)文档，确保文档版本有效性，提供标准化安全操作指引
// 安全数据表实体类
import javax.persistence.*;
import java.util.Date;
@Entity
@Table(name = "safety_datasheet")
public class SafetyDatasheet {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "safety_datasheet_id")
    private Long safetyDatasheetId; // 主键 ID
    @Column(name = "chem_id", nullable = false)
    private Long chemId; // 关联化学品 ID
    @Column(name = "version", nullable = false, length = 20)
    private String version; // 版本号
    @Column(name = "effective_date", nullable = false)
    @Temporal(TemporalType.DATE)
    private Date effectiveDate; // 生效日期
    @Column(name = "document_path", nullable = false, length = 255)
    private String documentPath; // 文档存储路径
    @Column(name = "file_format", nullable = false, length = 10)
    private String fileFormat; // 文件格式(PDF/DOC/XLS)
    @Enumerated(EnumType.STRING)
    @Column(name = "status", nullable = false, length = 10)
    private Status status; // 状态(生效/失效/草稿)
    @Column(name = "created_by", nullable = false, length = 50)
    private String createdBy; // 创建人
    @Column(name = "created_time", nullable = false)
    @Temporal(TemporalType.TIMESTAMP)
    private Date createdTime; // 创建时间
    @Column(name = "updated_by", length = 50)
    private String updatedBy; // 更新人
    @Column(name = "updated_time")
    @Temporal(TemporalType.TIMESTAMP)
    private Date updatedTime; // 更新时间
    // 状态枚举定义
    public enum Status {
        生效, 失效, 草稿
    }
    // 构造方法
    public SafetyDatasheet() {}
    // Getter 和 Setter 方法
    public Long getSafetyDatasheetId() { return safetyDatasheetId; }
    public void setSafetyDatasheetId(Long safetyDatasheetId) { this.safetyDatasheetId =
safetyDatasheetId; }
    public Long getChemId() { return chemId; }
    public void setChemId(Long chemId) { this.chemId = chemId; }
    public String getVersion() { return version; }
    public void setVersion(String version) { this.version = version; }
    public Date getEffectiveDate() { return effectiveDate; }
    public void setEffectiveDate(Date effectiveDate) { this.effectiveDate = effectiveDate; }
    public String getDocumentPath() { return documentPath; }
    public void setDocumentPath(String documentPath) { this.documentPath = documentPath; }
    public String getFileFormat() { return fileFormat; }
    public void setFileFormat(String fileFormat) { this.fileFormat = fileFormat; }
    public Status getStatus() { return status; }
    public void setStatus(Status status) { this.status = status; }
    public String getCreatedBy() { return createdBy; }
    public void setCreatedBy(String createdBy) { this.createdBy = createdBy; }
    public Date getCreatedTime() { return createdTime; }
    public void setCreatedTime(Date createdTime) { this.createdTime = createdTime; }
    public String getUpdatedBy() { return updatedBy; }
    public void setUpdatedBy(String updatedBy) { this.updatedBy = updatedBy; }
    public Date getUpdatedTime() { return updatedTime; }
    public void setUpdatedTime(Date updatedTime) { this.updatedTime = updatedTime; }
}
// 数据访问层
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import java.util.List;
@Repository
public interface SafetyDatasheetRepository extends JpaRepository<SafetyDatasheet, Long> {
    // 根据化学品 ID 查询安全数据表
    List<SafetyDatasheet> findByChemId(Long chemId);
    // 根据状态查询安全数据表
    List<SafetyDatasheet> findByStatus(SafetyDatasheet.Status status);
}
// 服务层
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import java.util.Date;
import java.util.List;
@Service
@Transactional
public class SafetyDatasheetService {
   @Autowired
   private SafetyDatasheetRepository repository;
   // 创建安全数据表记录
   public SafetyDatasheet createDatasheet(SafetyDatasheet datasheet, String creator) {
      datasheet.setCreatedBy(creator);
      datasheet.setCreatedTime(new Date());
      datasheet.setStatus(SafetyDatasheet.Status.草稿);
      return repository.save(datasheet);
   }
   // 更新安全数据表记录
   public SafetyDatasheet updateDatasheet(Long id, SafetyDatasheet updatedDatasheet, String updater)
{
      return repository.findById(id).map(datasheet -> {
         datasheet.setVersion(updatedDatasheet.getVersion());
         datasheet.setEffectiveDate(updatedDatasheet.getEffectiveDate());
         datasheet.setDocumentPath(updatedDatasheet.getDocumentPath());
         datasheet.setFileFormat(updatedDatasheet.getFileFormat());
         datasheet.setStatus(updatedDatasheet.getStatus());
         datasheet.setUpdatedBy(updater);
         datasheet.setUpdatedTime(new Date());
         return repository.save(datasheet);
      }).orElseThrow(() -> new RuntimeException("安全数据表记录不存在"));
   }
   // 删除安全数据表记录
   public void deleteDatasheet(Long id) {
      repository.deleteById(id);
   }
   // 获取单个安全数据表记录
   public SafetyDatasheet getDatasheetById(Long id) {
      return repository.findById(id)
            .orElseThrow(() -> new RuntimeException("安全数据表记录不存在"));
   }
   // 获取所有安全数据表记录
   public List<SafetyDatasheet> getAllDatasheets() {
      return repository.findAll();
   }
   // 更新文档版本状态
   public SafetyDatasheet updateStatus(Long id, SafetyDatasheet.Status newStatus, String updater) {
      return repository.findById(id).map(datasheet -> {
         datasheet.setStatus(newStatus);
         datasheet.setUpdatedBy(updater);
         datasheet.setUpdatedTime(new Date());
         return repository.save(datasheet);
      }).orElseThrow(() -> new RuntimeException("安全数据表记录不存在"));
   }
}
<!DOCTYPE html>
<html lang="zh-CN">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>安全数据表管理系统</title>
  <link rel="stylesheet" href="https://unpkg.com/element-plus/dist/index.css">
  <script src="https://unpkg.com/vue@3"></script>
  <script src="https://unpkg.com/element-plus"></script>
  <script src="https://unpkg.com/axios/dist/axios.min.js"></script>
  <style>
    .app-container {
      padding: 20px;
      max-width: 1200px;
      margin: 0 auto;
    }
    .header-actions {
      margin-bottom: 20px;
      display: flex;
      justify-content: space-between;
    }
    .form-row {
      margin-bottom: 18px;
    }
    .status-tag {
      margin-left: 10px;
    }
  </style>
</head>
<body>
  <div id="app">
    <el-container class="app-container">
      <el-header>
        <h1>安全数据表管理系统</h1>
      </el-header>
      <el-main>
        <!-- 搜索和操作区域 -->
        <div class="header-actions">
          <el-button type="primary" @click="openCreateDialog">新增安全数据表</el-button>
          <el-input v-model="searchKeyword" placeholder="搜索化学品/版本号" style="width: 300px"
clearable>
            <template #append>
              <el-button icon="el-icon-search" @click="fetchDatasheets"></el-button>
            </template>
          </el-input>
        </div>
        <!-- 数据表格 -->
        <el-table :data="datasheets" border stripe height="calc(100vh - 250px)">
          <el-table-column prop="safetyDatasheetId" label="ID" width="80" sortable></el-table-column>
          <el-table-column label="化学品名称">
            <template #default="{row}">
              {{ getChemName(row.chemId) }}
              <el-tag :type="statusTagType(row.status)" class="status-tag">
                {{ row.status }}
              </el-tag>
            </template>
          </el-table-column>
          <el-table-column prop="version" label="版本号" width="120"></el-table-column>
          <el-table-column prop="effectiveDate" label="生效日期" width="120" sortable>
            <template #default="{row}">
              {{ formatDate(row.effectiveDate) }}
            </template>
          </el-table-column>
          <el-table-column prop="fileFormat" label="文件类型" width="100"></el-table-column>
          <el-table-column prop="createdBy" label="创建人" width="120"></el-table-column>
          <el-table-column prop="createdTime" label="创建时间" width="160">
            <template #default="{row}">
              {{ formatDateTime(row.createdTime) }}
            </template>
          </el-table-column>
          <el-table-column label="操作" width="220" fixed="right">
            <template #default="{row}">
              <el-button size="small" @click="downloadFile(row)">下载</el-button>
              <el-button size="small" type="primary" @click="openEditDialog(row)">编辑</el-button>
              <el-button size="small" type="danger" @click="confirmDelete(row.safetyDatasheetId)">删
除</el-button>
            </template>
          </el-table-column>
        </el-table>
        <!-- 分页控件 -->
        <el-pagination
          v-model:currentPage="currentPage"
          v-model:page-size="pageSize"
          :total="total"
          layout="total, sizes, prev, pager, next, jumper"
          @current-change="fetchDatasheets"
          @size-change="handleSizeChange"
        ></el-pagination>
      </el-main>
    </el-container>
    <!-- 新增/编辑对话框 -->
    <el-dialog v-model="dialogVisible" :title="dialogTitle" width="600px">
      <el-form :model="formData" ref="datasheetForm" label-width="120px">
        <el-form-item label="化学品" prop="chemId" required>
          <el-select v-model="formData.chemId" placeholder="请选择化学品" style="width:100%">
            <el-option
              v-for="chem in chemicals"
              :key="chem.chemId"
              :label="chem.chemName"
              :value="chem.chemId"
            ></el-option>
          </el-select>
        </el-form-item>
        <div class="form-row">
          <el-form-item label="版本号" prop="version" required style="width:48%;display:inline-
block">
            <el-input v-model="formData.version" placeholder="如 v1.0.2"></el-input>
          </el-form-item>
          <el-form-item label="生效日期" prop="effectiveDate" required
style="width:48%;display:inline-block;margin-left:4%">
            <el-date-picker
              v-model="formData.effectiveDate"
              type="date"
              placeholder="选择日期"
              style="width:100%"
            ></el-date-picker>
          </el-form-item>
        </div>
        <div class="form-row">
          <el-form-item label="文件类型" prop="fileFormat" required style="width:48%;display:inline-
block">
            <el-select v-model="formData.fileFormat" placeholder="选择文件格式">
              <el-option label="PDF" value="PDF"></el-option>
              <el-option label="DOC" value="DOC"></el-option>
              <el-option label="XLS" value="XLS"></el-option>
            </el-select>
          </el-form-item>
          <el-form-item label="状态" prop="status" required style="width:48%;display:inline-
block;margin-left:4%">
            <el-select v-model="formData.status" placeholder="选择状态">
              <el-option label="生效" value="生效"></el-option>
              <el-option label="失效" value="失效"></el-option>
              <el-option label="草稿" value="草稿"></el-option>
            </el-select>
          </el-form-item>
        </div>
        <el-form-item label="安全数据表" prop="documentPath" required>
          <el-upload
            action="/api/upload"
            :on-success="handleUploadSuccess"
            :show-file-list="false"
          >
            <el-button type="primary">点击上传</el-button>
            <span v-if="formData.documentPath" style="margin-left:10px">
              {{ formData.documentPath.split('/').pop() }}
            </span>
          </el-upload>
        </el-form-item>
      </el-form>
      <template #footer>
        <el-button @click="dialogVisible = false">取消</el-button>
        <el-button type="primary" @click="submitForm">保存</el-button>
      </template>
    </el-dialog>
```

```
</div>
<script>
 const { createApp, ref, reactive, onMounted } = Vue;
 const { ElMessage, ElMessageBox } = ElementPlus;
 createApp({
  setup() {
   // 状态管理
   const datasheets = ref([]);
   const chemicals = ref([]);
   const dialogVisible = ref(false);
   const isEditMode = ref(false);
   const currentPage = ref(1);
   const pageSize = ref(10);
   const total = ref(0);
   const searchKeyword = ref('');
   // 表单数据模型
   const formData = reactive({
    safetyDatasheetId: null,
    chemId: null,
    version: '',
    effectiveDate: null,
    documentPath: '',
    fileFormat: '',
    status: '草稿',
    createdBy: '',
    createdTime: null,
    updatedBy: '',
    updatedTime: null
   });
   // 计算属性
   const dialogTitle = Vue.computed(() =>
    isEditMode.value ? '编辑安全数据表' : '新增安全数据表'
   );
   // 初始化加载数据
   onMounted(() => {
    fetchDatasheets();
    fetchChemicals();
   });
   // 获取化学品列表
   const fetchChemicals = async () => {
    try {
     const response = await axios.get('/api/chemicals');
     chemicals.value = response.data;
    } catch (error) {
     ElMessage.error('获取化学品列表失败');
    }
   };
   // 获取安全数据表列表
   const fetchDatasheets = async () => {
    try {
     const params = {
      page: currentPage.value - 1,
      size: pageSize.value,
      keyword: searchKeyword.value
    };
    const response = await axios.get('/api/safety-datasheets', { params });
    datasheets.value = response.data.content;
    total.value = response.data.totalElements;
  } catch (error) {
    ElMessage.error('获取安全数据表失败');
  }
};
// 分页大小变化处理
const handleSizeChange = (size) => {
  pageSize.value = size;
  fetchDatasheets();
};
// 打开创建对话框
const openCreateDialog = () => {
  resetForm();
  isEditMode.value = false;
  dialogVisible.value = true;
};
// 打开编辑对话框
const openEditDialog = (row) => {
  Object.assign(formData, {
    ...row,
    effectiveDate: new Date(row.effectiveDate)
  });
  isEditMode.value = true;
  dialogVisible.value = true;
};
// 重置表单
const resetForm = () => {
  Object.assign(formData, {
    safetyDatasheetId: null,
    chemId: null,
    version: '',
    effectiveDate: null,
    documentPath: '',
    fileFormat: '',
    status: '草稿',
    createdBy: '当前用户',
    createdTime: null,
    updatedBy: '',
    updatedTime: null
  });
};
// 文件上传成功处理
const handleUploadSuccess = (response) => {
  formData.documentPath = response.data.path;
};
// 提交表单
const submitForm = async () => {
    try {
      if (isEditMode.value) {
        await axios.put(`/api/safety-datasheets/${formData.safetyDatasheetId}`, formData);
        ElMessage.success('更新成功');
      } else {
        formData.createdTime = new Date();
        await axios.post('/api/safety-datasheets', formData);
        ElMessage.success('创建成功');
      }
      dialogVisible.value = false;
      fetchDatasheets();
    } catch (error) {
      ElMessage.error('操作失败');
    }
  };
  // 删除确认
  const confirmDelete = (id) => {
    ElMessageBox.confirm('确定删除此安全数据表?', '警告', {
      confirmButtonText: '确定',
      cancelButtonText: '取消',
      type: 'warning'
    }).then(async () => {
      try {
        await axios.delete(`/api/safety-datasheets/${id}`);
        ElMessage.success('删除成功');
        fetchDatasheets();
      } catch (error) {
        ElMessage.error('删除失败');
      }
    }).catch(() => {});
  };
  // 下载文件
  const downloadFile = (row) => {
    window.open(`/api/download?path=${encodeURIComponent(row.documentPath)}`, '_blank');
  };
  // 根据化学品 ID 获取名称
  const getChemName = (chemId) => {
    const chem = chemicals.value.find(c => c.chemId === chemId);
    return chem ? chem.chemName : '未知化学品';
  };
  // 状态标签样式
  const statusTagType = (status) => {
    switch(status) {
      case '生效': return 'success';
      case '失效': return 'danger';
      case '草稿': return 'warning';
      default: return 'info';
    }
  };
  // 日期格式化
  const formatDate = (dateStr) => {
          if (!dateStr) return '';
          const date = new Date(dateStr);
          return `${date.getFullYear()}-${(date.getMonth()+1).toString().padStart(2, '0')}-
${date.getDate().toString().padStart(2, '0')}`;
        };
        // 日期时间格式化
        const formatDateTime = (dateStr) => {
          if (!dateStr) return '';
          const date = new Date(dateStr);
          return `${formatDate(dateStr)} ${date.getHours().toString().padStart(2,
'0')}:${date.getMinutes().toString().padStart(2, '0')}`;
        };
        return {
          datasheets,
          chemicals,
          dialogVisible,
          dialogTitle,
          formData,
          currentPage,
          pageSize,
          total,
          searchKeyword,
          fetchDatasheets,
          handleSizeChange,
          openCreateDialog,
          openEditDialog,
          submitForm,
          confirmDelete,
          downloadFile,
          handleUploadSuccess,
          getChemName,
          statusTagType,
          formatDate,
          formatDateTime
        };
      }
    }).use(ElementPlus).mount('#app');
  </script>
</body>
</html>
// 系统：绿色智能化学解决方案系统
// 功能：实验方案设计
// 描述：创建标准化的实验操作流程，定义实验步骤、所需试剂及安全注意事项
import javax.persistence.*;
import java.time.LocalDateTime;
/**
 * 实验方案设计实体类
 * 对应数据库表：experiment_design
 */
@Entity
@Table(name = "experiment_design")
public class ExperimentDesign {
    @Id
```

```
@GeneratedValue(strategy = GenerationType.IDENTITY)
@Column(name = "experiment_design_id")
private Long id; // 方案设计 ID
@Column(name = "design_name", nullable = false, length = 255)
private String designName; // 方案名称
@Column(name = "creator_id", nullable = false)
private Long creatorId; // 创建人 ID
@Column(name = "created_date", nullable = false)
private LocalDateTime createdDate; // 创建日期
@Enumerated(EnumType.STRING)
@Column(name = "status", nullable = false, length = 20)
private Status status; // 方案状态
@Lob
@Column(name = "experiment_steps")
private String experimentSteps; // 实验步骤
@Lob
@Column(name = "reagents")
private String reagents; // 所需试剂
@Lob
@Column(name = "safety_notes")
private String safetyNotes; // 安全注意事项
@Column(name = "submitted_date")
private LocalDateTime submittedDate; // 提交时间
@Column(name = "reviewer_id")
private Long reviewerId; // 审核人 ID
@Column(name = "reviewed_date")
private LocalDateTime reviewedDate; // 审核时间
@Lob
@Column(name = "review_comment")
private String reviewComment; // 审核意见
// 状态枚举定义
public enum Status {
   草稿, 已提交, 已审核
}
// 无参构造函数
public ExperimentDesign() {}
// 带参构造函数
public ExperimentDesign(String designName, Long creatorId) {
   this.designName = designName;
   this.creatorId = creatorId;
   this.createdDate = LocalDateTime.now();
   this.status = Status.草稿;
}
// Getter 和 Setter 方法
public Long getId() { return id; }
public void setId(Long id) { this.id = id; }
public String getDesignName() { return designName; }
public void setDesignName(String designName) { this.designName = designName; }
public Long getCreatorId() { return creatorId; }
public void setCreatorId(Long creatorId) { this.creatorId = creatorId; }
public LocalDateTime getCreatedDate() { return createdDate; }
```

```
    public void setCreatedDate(LocalDateTime createdDate) { this.createdDate = createdDate; }
    public Status getStatus() { return status; }
    public void setStatus(Status status) { this.status = status; }
    public String getExperimentSteps() { return experimentSteps; }
    public void setExperimentSteps(String experimentSteps) { this.experimentSteps =
experimentSteps; }
    public String getReagents() { return reagents; }
    public void setReagents(String reagents) { this.reagents = reagents; }
    public String getSafetyNotes() { return safetyNotes; }
    public void setSafetyNotes(String safetyNotes) { this.safetyNotes = safetyNotes; }
    public LocalDateTime getSubmittedDate() { return submittedDate; }
    public void setSubmittedDate(LocalDateTime submittedDate) { this.submittedDate =
submittedDate; }
    public Long getReviewerId() { return reviewerId; }
    public void setReviewerId(Long reviewerId) { this.reviewerId = reviewerId; }
    public LocalDateTime getReviewedDate() { return reviewedDate; }
    public void setReviewedDate(LocalDateTime reviewedDate) { this.reviewedDate = reviewedDate; }
    public String getReviewComment() { return reviewComment; }
    public void setReviewComment(String reviewComment) { this.reviewComment =
reviewComment; }
}
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>实验方案设计</title>
    <link rel="stylesheet" href="https://unpkg.com/element-plus/dist/index.css">
    <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
    <script src="https://unpkg.com/element-plus/dist/index.full.js"></script>
    <script src="https://unpkg.com/axios/dist/axios.min.js"></script>
    <style>
        .app-container { padding: 20px; }
        .header-actions { margin-bottom: 20px; }
        .form-container { max-width: 800px; }
        .status-tag { margin-left: 10px; }
    </style>
</head>
<body>
    <div id="app">
        <el-container class="app-container">
            <el-main>
                <div class="header-actions">
                    <el-button type="primary" @click="openCreateDialog">新增目录</el-button>
                </div>
                <!-- 方案列表 -->
                <el-table :data="designList" border stripe>
                    <el-table-column prop="designName" label="方案名称" width="180" />
                    <el-table-column prop="creatorId" label="创建人 ID" width="100" />
                    <el-table-column prop="createdDate" label="创建日期" width="180" />
                    <el-table-column label="状态" width="120">
                        <template #default="{row}">
```

```
              <el-tag :type="getStatusTagType(row.status)" class="status-tag">
                {{ row.status }}
              </el-tag>
            </template>
          </el-table-column>
          <el-table-column prop="submittedDate" label="提交时间" width="180" />
          <el-table-column fixed="right" label="操作" width="200">
            <template #default="{row}">
              <el-button size="small" @click="openEditDialog(row)">编辑属性</el-button>
              <el-button size="small" type="danger" @click="deleteDesign(row.id)">删除</el-
button>
            </template>
          </el-table-column>
        </el-table>
        <!-- 创建/编辑对话框 -->
        <el-dialog v-model="dialogVisible" :title="dialogTitle" width="50%">
          <el-form ref="designForm" :model="currentDesign" label-width="120px" class="form-
container">
            <el-form-item label="方案名称" prop="designName" required>
              <el-input v-model="currentDesign.designName" />
            </el-form-item>
            <el-form-item label="创建人 ID" prop="creatorId" required>
              <el-input-number v-model="currentDesign.creatorId" :min="1" />
            </el-form-item>
            <el-form-item label="状态" prop="status" required>
              <el-select v-model="currentDesign.status">
                <el-option label="草稿" value="草稿" />
                <el-option label="已提交" value="已提交" />
                <el-option label="已审核" value="已审核" />
              </el-select>
            </el-form-item>
            <el-form-item label="实验步骤" prop="experimentSteps">
              <el-input v-model="currentDesign.experimentSteps" type="textarea" :rows="4" />
            </el-form-item>
            <el-form-item label="所需试剂" prop="reagents">
              <el-input v-model="currentDesign.reagents" type="textarea" :rows="3" />
            </el-form-item>
            <el-form-item label="安全注意事项" prop="safetyNotes">
              <el-input v-model="currentDesign.safetyNotes" type="textarea" :rows="3" />
            </el-form-item>
            <el-form-item label="审核人 ID" prop="reviewerId">
              <el-input-number v-model="currentDesign.reviewerId" :min="1" />
            </el-form-item>
            <el-form-item label="审核意见" prop="reviewComment">
              <el-input v-model="currentDesign.reviewComment" type="textarea" :rows="3" />
            </el-form-item>
          </el-form>
          <template #footer>
            <el-button @click="dialogVisible = false">取消</el-button>
            <el-button v-if="isEditMode" type="primary" @click="saveDraft">保存草稿</el-
button>
```

```
            <el-button type="success" @click="submitForReview">提交审核</el-button>
          </template>
        </el-dialog>
      </el-main>
    </el-container>
  </div>
  <script>
    const { createApp, ref, reactive, onMounted } = Vue;
    const { ElMessage, ElMessageBox } = ElementPlus;
    createApp({
      setup() {
        // 状态管理
        const designList = ref([]);
        const dialogVisible = ref(false);
        const isEditMode = ref(false);
        const dialogTitle = ref('');
        const designForm = ref(null);
        // 当前方案数据模型
        const currentDesign = reactive({
          id: null,
          designName: '',
          creatorId: null,
          createdDate: '',
          status: '草稿',
          experimentSteps: '',
          reagents: '',
          safetyNotes: '',
          submittedDate: '',
          reviewerId: null,
          reviewedDate: '',
          reviewComment: ''
        });
        // API 端点
        const API_BASE = '/api/experiment-design';
        // 初始化加载数据
        const fetchDesigns = async () => {
          try {
            const response = await axios.get(API_BASE);
            designList.value = response.data;
          } catch (error) {
            ElMessage.error('加载方案列表失败');
          }
        };
        // 打开创建对话框
        const openCreateDialog = () => {
          resetCurrentDesign();
          dialogTitle.value = '创建新方案';
          isEditMode.value = false;
          dialogVisible.value = true;
        };
        // 打开编辑对话框
        const openEditDialog = (design) => {
```

```
    Object.assign(currentDesign, design);
    dialogTitle.value = '编辑方案属性';
    isEditMode.value = true;
    dialogVisible.value = true;
};
// 重置当前方案
const resetCurrentDesign = () => {
    Object.assign(currentDesign, {
        id: null,
        designName: '',
        creatorId: null,
        createdDate: '',
        status: '草稿',
        experimentSteps: '',
        reagents: '',
        safetyNotes: '',
        submittedDate: '',
        reviewerId: null,
        reviewedDate: '',
        reviewComment: ''
    });
};
// 保存草稿
const saveDraft = async () => {
    try {
        const method = currentDesign.id ? 'put' : 'post';
        const url = currentDesign.id ? `${API_BASE}/${currentDesign.id}` : API_BASE;
        const payload = { ...currentDesign };
        payload.status = '草稿';
        await axios[method](url, payload);
        ElMessage.success('方案草稿保存成功');
        dialogVisible.value = false;
        fetchDesigns();
    } catch (error) {
        ElMessage.error('保存失败');
    }
};
// 提交审核
const submitForReview = async () => {
    try {
        const method = currentDesign.id ? 'put' : 'post';
        const url = currentDesign.id ? `${API_BASE}/${currentDesign.id}` : API_BASE;
        const payload = { ...currentDesign };
        payload.status = '已提交';
        payload.submittedDate = new Date().toISOString();
        await axios[method](url, payload);
        ElMessage.success('方案已提交审核');
        dialogVisible.value = false;
        fetchDesigns();
    } catch (error) {
        ElMessage.error('提交失败');
    }
```

```
            };
            // 删除方案
            const deleteDesign = (id) => {
                ElMessageBox.confirm('确定删除此方案吗?', '警告', {
                    confirmButtonText: '确定',
                    cancelButtonText: '取消',
                    type: 'warning'
                }).then(async () => {
                    try {
                        await axios.delete(`${API_BASE}/${id}`);
                        ElMessage.success('方案已删除');
                        fetchDesigns();
                    } catch (error) {
                        ElMessage.error('删除失败');
                    }
                });
            };
            // 状态标签样式
            const getStatusTagType = (status) => {
                switch (status) {
                    case '草稿': return 'info';
                    case '已提交': return 'warning';
                    case '已审核': return 'success';
                    default: return '';
                }
            };
            // 初始化
            onMounted(fetchDesigns);
            return {
                designList,
                dialogVisible,
                isEditMode,
                dialogTitle,
                currentDesign,
                designForm,
                openCreateDialog,
                openEditDialog,
                saveDraft,
                submitForReview,
                deleteDesign,
                getStatusTagType
            };
        }
    }).use(ElementPlus).mount('#app');
  </script>
</body>
</html>
// 系统：绿色智能化学解决方案系统
// 功能：实验执行记录
// 描述：实时记录实验操作过程和数据结果，跟踪实验进度并自动生成操作日志
import javax.persistence.*;
import java.time.LocalDateTime;
```

```
/**
 * 实验执行记录实体类
 */
@Entity
@Table(name = "experiment_record")
public class ExperimentRecord {
    /**
     * 主键 ID
     */
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "experiment_record_id")
    private Long experimentRecordId;
    /**
     * 实验名称
     */
    @Column(name = "experiment_name", length = 100, nullable = false)
    private String experimentName;
    /**
     * 执行人 ID（关联用户表主键）
     */
    @Column(name = "executor_id", nullable = false)
    private Long executorId;
    /**
     * 计划执行日期
     */
    @Column(name = "execute_date", nullable = false)
    private LocalDateTime executeDate;
    /**
     * 操作过程日志
     */
    @Column(name = "operation_log", columnDefinition = "TEXT")
    private String operationLog;
    /**
     * 实验数据（JSON 格式）
     */
    @Column(name = "experiment_data", columnDefinition = "JSON")
    private String experimentData;
    /**
     * 当前实验状态
     */
    @Enumerated(EnumType.STRING)
    @Column(name = "current_status", nullable = false)
    private ExperimentStatus currentStatus;
    /**
     * 关联实验方案 ID（可为空）
     */
    @Column(name = "design_id")
    private Long designId;
    /**
     * 实际开始时间
     */
    @Column(name = "start_time")
    private LocalDateTime startTime;
    /**
     * 实际完成时间
     */
    @Column(name = "end_time")
    private LocalDateTime endTime;
    // 状态枚举定义
    public enum ExperimentStatus {
        未开始, 记录中, 已完成
    }
    // 构造方法
    public ExperimentRecord() {}
    // Getter 和 Setter 方法
    public Long getExperimentRecordId() { return experimentRecordId; }
    public void setExperimentRecordId(Long experimentRecordId) { this.experimentRecordId =
experimentRecordId; }
    public String getExperimentName() { return experimentName; }
    public void setExperimentName(String experimentName) { this.experimentName =
experimentName; }
    public Long getExecutorId() { return executorId; }
    public void setExecutorId(Long executorId) { this.executorId = executorId; }
    public LocalDateTime getExecuteDate() { return executeDate; }
    public void setExecuteDate(LocalDateTime executeDate) { this.executeDate = executeDate; }
    public String getOperationLog() { return operationLog; }
    public void setOperationLog(String operationLog) { this.operationLog = operationLog; }
    public String getExperimentData() { return experimentData; }
    public void setExperimentData(String experimentData) { this.experimentData = experimentData; }
    public ExperimentStatus getCurrentStatus() { return currentStatus; }
    public void setCurrentStatus(ExperimentStatus currentStatus) { this.currentStatus = currentStatus; }
    public Long getDesignId() { return designId; }
    public void setDesignId(Long designId) { this.designId = designId; }
    public LocalDateTime getStartTime() { return startTime; }
    public void setStartTime(LocalDateTime startTime) { this.startTime = startTime; }
    public LocalDateTime getEndTime() { return endTime; }
    public void setEndTime(LocalDateTime endTime) { this.endTime = endTime; }
}
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>实验执行记录管理</title>
    <link rel="stylesheet" href="https://unpkg.com/element-plus/dist/index.css">
    <script src="https://unpkg.com/vue@3"></script>
    <script src="https://unpkg.com/element-plus"></script>
    <script src="https://unpkg.com/axios/dist/axios.min.js"></script>
    <style>
        .page-container {
            padding: 20px;
            max-width: 1200px;
            margin: 0 auto;
        }
        .form-row {
            margin-bottom: 20px;
        }
        .status-tag {
            margin-left: 10px;
        }
        .operation-buttons {
            margin-top: 20px;
            display: flex;
            gap: 10px;
        }
    </style>
</head>
<body>
    <div id="app">
        <div class="page-container">
            <el-card>
                <template #header>
                    <div class="card-header">
                        <span>实验执行记录管理</span>
                        <el-button type="primary" @click="openCreateDialog">新建记录</el-button>
                    </div>
                </template>
                <!-- 数据表格 -->
                <el-table :data="recordList" border stripe>
                    <el-table-column prop="experimentRecordId" label="记录 ID" width="100" />
                    <el-table-column prop="experimentName" label="实验名称" min-width="150" />
                    <el-table-column label="执行人">
                        <template #default="scope">
                            {{ getUserName(scope.row.executorId) }}
                        </template>
                    </el-table-column>
                    <el-table-column prop="executeDate" label="计划执行日期" width="180" />
                    <el-table-column label="当前状态" width="120">
                        <template #default="scope">
                            <el-tag :type="getStatusTagType(scope.row.currentStatus)" class="status-tag">
                                {{ scope.row.currentStatus }}
                            </el-tag>
                        </template>
                    </el-table-column>
                    <el-table-column label="关联方案">
                        <template #default="scope">
                            {{ getDesignName(scope.row.designId) }}
                        </template>
                    </el-table-column>
                    <el-table-column label="操作" width="220" fixed="right">
                        <template #default="scope">
                            <el-button size="small" @click="openEditDialog(scope.row)">编辑</el-button>
                            <el-button
                                size="small"
                                type="primary"
                                v-if="scope.row.currentStatus === '未开始'"
```

```
              @click="startRecording(scope.row.experimentRecordId)"
            >开始记录</el-button>
            <el-button
              size="small"
              type="warning"
              v-if="scope.row.currentStatus === '记录中'"
              @click="openAddDataDialog(scope.row)"
            >添加数据</el-button>
            <el-button
              size="small"
              type="success"
              v-if="scope.row.currentStatus === '记录中'"
              @click="completeRecord(scope.row.experimentRecordId)"
            >完成</el-button>
            <el-button
              size="small"
              type="danger"
              @click="deleteRecord(scope.row.experimentRecordId)"
            >删除</el-button>
          </template>
        </el-table-column>
      </el-table>
    </el-card>
    <!-- 创建/编辑对话框 -->
    <el-dialog
      v-model="recordDialogVisible"
      :title="currentRecord.experimentRecordId ? '编辑实验记录' : '新建实验记录'"
      width="600px"
    >
      <el-form :model="currentRecord" label-width="120px">
        <el-form-item label="实验名称" required>
          <el-input v-model="currentRecord.experimentName" />
        </el-form-item>
        <el-form-item label="执行人" required>
          <el-select v-model="currentRecord.executorId" placeholder="请选择执行人">
            <el-option
              v-for="user in userList"
              :key="user.sysUserId"
              :label="user.userName"
              :value="user.sysUserId"
            />
          </el-select>
        </el-form-item>
        <el-form-item label="计划执行日期" required>
          <el-date-picker
            v-model="currentRecord.executeDate"
            type="datetime"
            placeholder="选择日期时间"
          />
        </el-form-item>
        <el-form-item label="关联实验方案">
```

```
                <el-select v-model="currentRecord.designId" placeholder="请选择实验方案">
                    <el-option
                        v-for="design in designList"
                        :key="design.experimentDesignId"
                        :label="design.schemeName"
                        :value="design.experimentDesignId"
                    />
                </el-select>
            </el-form-item>
            <el-form-item label="操作日志">
                <el-input
                    v-model="currentRecord.operationLog"
                    type="textarea"
                    :rows="4"
                    placeholder="记录实验操作过程"
                />
            </el-form-item>
        </el-form>
        <template #footer>
            <span class="dialog-footer">
                <el-button @click="recordDialogVisible = false">取消</el-button>
                <el-button type="primary" @click="saveRecord">保存</el-button>
            </span>
        </template>
    </el-dialog>
    <!-- 添加数据对话框 -->
    <el-dialog v-model="addDataDialogVisible" title="添加实验数据" width="600px">
        <el-form :model="dataForm" label-width="100px">
            <el-form-item label="实验数据" required>
                <el-input
                    v-model="dataForm.jsonData"
                    type="textarea"
                    :rows="6"
                    placeholder="请输入 JSON 格式的实验数据"
                />
            </el-form-item>
            <el-alert title="JSON 格式{'temperature':25.5, 'pressure':101.3}" type="info" />
        </el-form>
        <template #footer>
            <span class="dialog-footer">
                <el-button @click="addDataDialogVisible = false">取消</el-button>
                <el-button type="primary" @click="addExperimentData">确认添加</el-button>
            </span>
        </template>
    </el-dialog>
</div>
</div>
<script>
const { createApp, ref, reactive, onMounted } = Vue;
const { ElMessage, ElMessageBox } = ElementPlus;
createApp({
    setup() {
```

```
// 数据状态
const recordList = ref([]);
const userList = ref([]);
const designList = ref([]);
const recordDialogVisible = ref(false);
const addDataDialogVisible = ref(false);
// 当前操作的记录
const currentRecord = reactive({
  experimentRecordId: null,
  experimentName: '',
  executorId: null,
  executeDate: null,
  operationLog: '',
  experimentData: null,
  currentStatus: '未开始',
  designId: null,
  startTime: null,
  endTime: null
});
// 添加数据表单
const dataForm = reactive({
  recordId: null,
  jsonData: ''
});
// 初始化加载数据
const loadData = async () => {
  try {
    // 加载实验记录
    const recordRes = await axios.get('/api/experiment-records');
    recordList.value = recordRes.data;
    // 加载用户列表
    const userRes = await axios.get('/api/users');
    userList.value = userRes.data;
    // 加载实验方案
    const designRes = await axios.get('/api/experiment-designs');
    designList.value = designRes.data;
  } catch (error) {
    ElMessage.error('数据加载失败: ' + error.message);
  }
};
// 打开创建对话框
const openCreateDialog = () => {
  Object.assign(currentRecord, {
    experimentRecordId: null,
    experimentName: '',
    executorId: null,
    executeDate: new Date(),
    operationLog: '',
    experimentData: null,
    currentStatus: '未开始',
    designId: null,
    startTime: null,
          endTime: null
        });
        recordDialogVisible.value = true;
      };
      // 打开编辑对话框
      const openEditDialog = (record) => {
        Object.assign(currentRecord, record);
        recordDialogVisible.value = true;
      };
      // 保存记录
      const saveRecord = async () => {
        try {
          if (currentRecord.experimentRecordId) {
            await axios.put(`/api/experiment-records/${currentRecord.experimentRecordId}`,
currentRecord);
            ElMessage.success('记录更新成功');
          } else {
            await axios.post('/api/experiment-records', currentRecord);
            ElMessage.success('记录创建成功');
          }
          recordDialogVisible.value = false;
          await loadData();
        } catch (error) {
          ElMessage.error('保存失败: ' + error.message);
        }
      };
      // 删除记录
      const deleteRecord = async (id) => {
        try {
          await ElMessageBox.confirm('确定要删除此实验记录吗？', '警告', {
            confirmButtonText: '确定',
            cancelButtonText: '取消',
            type: 'warning'
          });
          await axios.delete(`/api/experiment-records/${id}`);
          ElMessage.success('记录已删除');
          await loadData();
        } catch (error) {
          if (error !== 'cancel') {
            ElMessage.error('删除失败: ' + error.message);
          }
        }
      };
      // 开始记录
      const startRecording = async (id) => {
        try {
          await axios.post(`/api/experiment-records/${id}/start`);
          ElMessage.success('实验记录已开始');
          await loadData();
        } catch (error) {
          ElMessage.error('操作失败: ' + error.message);
        }
```

```
};
// 打开添加数据对话框
const openAddDataDialog = (record) => {
  dataForm.recordId = record.experimentRecordId;
  dataForm.jsonData = '';
  addDataDialogVisible.value = true;
};
// 添加实验数据
const addExperimentData = async () => {
  try {
    // 验证 JSON 格式
    try {
      JSON.parse(dataForm.jsonData);
    } catch (e) {
      throw new Error('请输入有效的 JSON 格式数据');
    }
    await axios.post(`/api/experiment-records/${dataForm.recordId}/data`, {
      newData: dataForm.jsonData
    });
    ElMessage.success('实验数据已添加');
    addDataDialogVisible.value = false;
    await loadData();
  } catch (error) {
    ElMessage.error('添加失败: ' + error.message);
  }
};
// 完成记录
const completeRecord = async (id) => {
  try {
    await axios.post(`/api/experiment-records/${id}/complete`);
    ElMessage.success('实验记录已完成');
    await loadData();
  } catch (error) {
    ElMessage.error('操作失败: ' + error.message);
  }
};
// 根据状态获取标签类型
const getStatusTagType = (status) => {
  switch(status) {
    case '未开始': return 'info';
    case '记录中': return 'warning';
    case '已完成': return 'success';
    default: return '';
  }
};
// 获取执行人姓名
const getUserName = (userId) => {
  const user = userList.value.find(u => u.sysUserId === userId);
  return user ? user.userName : '未知用户';
};
// 获取方案名称
```

```
            const getDesignName = (designId) => {
              if (!designId) return '未关联';
              const design = designList.value.find(d => d.experimentDesignId === designId);
              return design ? design.schemeName : '未知方案';
            };
            // 组件挂载时加载数据
            onMounted(loadData);
            return {
              recordList,
              userList,
              designList,
              recordDialogVisible,
              addDataDialogVisible,
              currentRecord,
              dataForm,
              openCreateDialog,
              openEditDialog,
              saveRecord,
              deleteRecord,
              startRecording,
              openAddDataDialog,
              addExperimentData,
              completeRecord,
              getStatusTagType,
              getUserName,
              getDesignName
            };
          }
        }).use(ElementPlus).mount('#app');
    </script>
</body>
</html>
// 系统：绿色智能化学解决方案系统
// 功能：实验历史查询
// 描述：查询已完成实验的完整历史记录，包括操作过程、原始数据和最终结果
import javax.persistence.*;
import java.time.LocalDateTime;
/**
 * 实验历史记录实体类
 */
@Entity
@Table(name = "experiment_history")
public class ExperimentHistory {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "experiment_history_id")
    private Long id;  // 历史记录 ID
    @Column(name = "experiment_name", nullable = false, length = 255)
    private String experimentName;  // 实验名称
    @Column(name = "complete_date", nullable = false)
    private LocalDateTime completeDate;  // 完成日期
    @Enumerated(EnumType.STRING)
    @Column(name = "record_status", nullable = false, length = 20)
    private RecordStatus recordStatus;  // 记录状态
    @Column(name = "executor_id", nullable = false)
    private Long executorId;  // 执行人 ID
    @Column(name = "experiment_type", nullable = false, length = 100)
    private String experimentType;  // 实验类型
    @Column(name = "operation_summary", columnDefinition = "TEXT")
    private String operationSummary;  // 操作过程摘要
    @Column(name = "raw_data_path", length = 500)
    private String rawDataPath;  // 原始数据路径
    @Column(name = "final_conclusion", columnDefinition = "TEXT")
    private String finalConclusion;  // 最终结论
    @Column(name = "created_time", nullable = false)
    private LocalDateTime createdTime;  // 创建时间
    @Column(name = "last_updated")
    private LocalDateTime lastUpdated;  // 最后更新时间
    // 枚举类型定义
    public enum RecordStatus {
        有效, 已归档
    }
    // 构造方法
    public ExperimentHistory() {
        this.createdTime = LocalDateTime.now();
        this.recordStatus = RecordStatus.有效;
    }
    // Getter 和 Setter 方法
    public Long getId() { return id; }
    public void setId(Long id) { this.id = id; }
    public String getExperimentName() { return experimentName; }
    public void setExperimentName(String experimentName) { this.experimentName =
experimentName; }
    public LocalDateTime getCompleteDate() { return completeDate; }
    public void setCompleteDate(LocalDateTime completeDate) { this.completeDate = completeDate; }
    public RecordStatus getRecordStatus() { return recordStatus; }
    public void setRecordStatus(RecordStatus recordStatus) { this.recordStatus = recordStatus; }
    public Long getExecutorId() { return executorId; }
    public void setExecutorId(Long executorId) { this.executorId = executorId; }
    public String getExperimentType() { return experimentType; }
    public void setExperimentType(String experimentType) { this.experimentType = experimentType; }
    public String getOperationSummary() { return operationSummary; }
    public void setOperationSummary(String operationSummary) { this.operationSummary =
operationSummary; }
    public String getRawDataPath() { return rawDataPath; }
    public void setRawDataPath(String rawDataPath) { this.rawDataPath = rawDataPath; }
    public String getFinalConclusion() { return finalConclusion; }
    public void setFinalConclusion(String finalConclusion) { this.finalConclusion = finalConclusion; }
    public LocalDateTime getCreatedTime() { return createdTime; }
    public void setCreatedTime(LocalDateTime createdTime) { this.createdTime = createdTime; }
    public LocalDateTime getLastUpdated() { return lastUpdated; }
    public void setLastUpdated(LocalDateTime lastUpdated) { this.lastUpdated = lastUpdated; }
}
<!DOCTYPE html>
```

```
<html lang="zh-CN">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>实验历史查询</title>
    <link rel="stylesheet" href="https://unpkg.com/element-plus/dist/index.css">
    <script src="https://unpkg.com/vue@3"></script>
    <script src="https://unpkg.com/element-plus"></script>
    <script src="https://unpkg.com/axios/dist/axios.min.js"></script>
    <style>
        .container {
            max-width: 1200px;
            margin: 20px auto;
            padding: 20px;
        }
        .header {
            display: flex;
            justify-content: space-between;
            margin-bottom: 20px;
        }
        .form-container {
            margin-bottom: 20px;
        }
        .operation-buttons {
            margin-bottom: 15px;
        }
    </style>
</head>
<body>
    <div id="app">
        <div class="container">
            <div class="header">
                <h2>实验历史查询</h2>
                <el-button type="primary" @click="showCreateDialog">新增记录</el-button>
            </div>
            <!-- 查询条件 -->
            <div class="form-container">
                <el-form :inline="true" :model="queryParams">
                    <el-form-item label="实验名称">
                        <el-input v-model="queryParams.experimentName" placeholder="请输入实验名称"
clearable></el-input>
                    </el-form-item>
                    <el-form-item label="实验类型">
                        <el-select v-model="queryParams.experimentType" placeholder="请选择实验类型"
clearable>
                            <el-option label="合成实验" value="合成实验"></el-option>
                            <el-option label="分析实验" value="分析实验"></el-option>
                            <el-option label="测试实验" value="测试实验"></el-option>
                        </el-select>
                    </el-form-item>
                    <el-form-item label="记录状态">
                        <el-select v-model="queryParams.recordStatus" placeholder="请选择状态" clearable>
                <el-option label="有效" value="有效"></el-option>
                <el-option label="已归档" value="已归档"></el-option>
              </el-select>
            </el-form-item>
            <el-form-item>
              <el-button type="primary" @click="fetchData">查询</el-button>
            </el-form-item>
          </el-form>
        </div>
        <!-- 操作按钮 -->
        <div class="operation-buttons">
          <el-button type="success" @click="exportRecords">导出记录</el-button>
          <el-button type="warning" @click="batchArchive">批量归档</el-button>
        </div>
        <!-- 数据表格 -->
        <el-table
          :data="tableData"
          border
          stripe
          style="width: 100%"
          @selection-change="handleSelectionChange">
          <el-table-column type="selection" width="55"></el-table-column>
          <el-table-column prop="experimentName" label="实验名称" width="180"></el-table-
column>
          <el-table-column prop="completeDate" label="完成日期" width="180">
            <template #default="scope">
              {{ formatDate(scope.row.completeDate) }}
            </template>
          </el-table-column>
          <el-table-column prop="recordStatus" label="记录状态" width="100"></el-table-column>
          <el-table-column prop="experimentType" label="实验类型" width="120"></el-table-
column>
          <el-table-column prop="executorName" label="执行人" width="120"></el-table-column>
          <el-table-column label="操作" width="250">
            <template #default="scope">
              <el-button size="small" @click="viewDetail(scope.row)">查看详情</el-button>
              <el-button size="small" type="primary" @click="editRecord(scope.row)">编辑</el-
button>
              <el-button size="small" type="danger"
@click="deleteRecord(scope.row.experimentHistoryId)">删除</el-button>
              <el-button size="small" type="warning"
@click="archiveRecord(scope.row.experimentHistoryId)"
                v-if="scope.row.recordStatus === '有效'">归档
              </el-button>
            </template>
          </el-table-column>
        </el-table>
        <!-- 分页 -->
        <el-pagination
          @size-change="handleSizeChange"
          @current-change="handleCurrentChange"
          :current-page="currentPage"
          :page-sizes="[10, 20, 50]"
          :page-size="pageSize"
          layout="total, sizes, prev, pager, next, jumper"
          :total="total">
        </el-pagination>
        <!-- 详情对话框 -->
        <el-dialog v-model="detailDialogVisible" title="实验详情" width="60%">
          <el-descriptions :column="2" border>
            <el-descriptions-item label="实验名称">{{ detailData.experimentName }}</el-
  descriptions-item>
            <el-descriptions-item label="完成日期">{{ formatDate(detailData.completeDate) }}</el-
  descriptions-item>
            <el-descriptions-item label="记录状态">{{ detailData.recordStatus }}</el-descriptions-
  item>
            <el-descriptions-item label="执行人">{{ detailData.executorName }}</el-descriptions-
  item>
            <el-descriptions-item label="实验类型">{{ detailData.experimentType }}</el-
  descriptions-item>
            <el-descriptions-item label="创建时间">{{ formatDate(detailData.createdTime) }}</el-
  descriptions-item>
            <el-descriptions-item label="操作过程摘要" :span="2">
              <div style="white-space: pre-line;">{{ detailData.operationSummary }}</div>
            </el-descriptions-item>
            <el-descriptions-item label="原始数据路径" :span="2">
              <el-link type="primary" :href="detailData.rawDataPath" target="_blank">
                {{ detailData.rawDataPath }}
              </el-link>
            </el-descriptions-item>
            <el-descriptions-item label="最终结论" :span="2">
              <div style="white-space: pre-line;">{{ detailData.finalConclusion }}</div>
            </el-descriptions-item>
          </el-descriptions>
          <template #footer>
            <el-button @click="detailDialogVisible = false">关闭</el-button>
          </template>
        </el-dialog>
        <!-- 编辑/新增对话框 -->
        <el-dialog v-model="editDialogVisible" :title="isEdit ? '编辑实验记录' : '新增实验记录'"
  width="50%">
          <el-form :model="editForm" label-width="120px" ref="editFormRef">
            <el-form-item label="实验名称" prop="experimentName" required>
              <el-input v-model="editForm.experimentName"></el-input>
            </el-form-item>
            <el-form-item label="完成日期" prop="completeDate" required>
              <el-date-picker
                v-model="editForm.completeDate"
                type="datetime"
                placeholder="选择日期时间">
              </el-date-picker>
            </el-form-item>
```

```
                <el-form-item label="实验类型" prop="experimentType" required>
                    <el-select v-model="editForm.experimentType" placeholder="请选择实验类型">
                        <el-option label="合成实验" value="合成实验"></el-option>
                        <el-option label="分析实验" value="分析实验"></el-option>
                        <el-option label="测试实验" value="测试实验"></el-option>
                    </el-select>
                </el-form-item>
                <el-form-item label="执行人 ID" prop="executorId" required>
                    <el-input v-model.number="editForm.executorId"></el-input>
                </el-form-item>
                <el-form-item label="操作过程摘要" prop="operationSummary">
                    <el-input
                        v-model="editForm.operationSummary"
                        type="textarea"
                        :rows="4"
                        placeholder="请输入实验操作过程摘要">
                    </el-input>
                </el-form-item>
                <el-form-item label="原始数据路径" prop="rawDataPath">
                    <el-input v-model="editForm.rawDataPath" placeholder="请输入原始数据存储路径
"></el-input>
                </el-form-item>
                <el-form-item label="最终结论" prop="finalConclusion">
                    <el-input
                        v-model="editForm.finalConclusion"
                        type="textarea"
                        :rows="4"
                        placeholder="请输入实验最终结论">
                    </el-input>
                </el-form-item>
            </el-form>
            <template #footer>
                <el-button @click="editDialogVisible = false">取消</el-button>
                <el-button type="primary" @click="submitForm">保存</el-button>
            </template>
        </el-dialog>
    </div>
</div>
<script>
    const { createApp, ref, reactive, onMounted } = Vue;
    const { ElMessage, ElMessageBox } = ElementPlus;
    createApp({
        setup() {
            // 状态管理
            const tableData = ref([]);
            const currentPage = ref(1);
            const pageSize = ref(10);
            const total = ref(0);
            const detailDialogVisible = ref(false);
            const editDialogVisible = ref(false);
            const isEdit = ref(false);
            const selectedRows = ref([]);
            // 查询参数
            const queryParams = reactive({
                experimentName: '',
                experimentType: '',
                recordStatus: ''
            });
            // 详情数据
            const detailData = reactive({
                experimentHistoryId: null,
                experimentName: '',
                completeDate: null,
                recordStatus: '',
                executorId: null,
                executorName: '',
                experimentType: '',
                operationSummary: '',
                rawDataPath: '',
                finalConclusion: '',
                createdTime: null,
                lastUpdated: null
            });
            // 编辑表单
            const editForm = reactive({
                experimentHistoryId: null,
                experimentName: '',
                completeDate: new Date(),
                recordStatus: '有效',
                executorId: null,
                experimentType: '',
                operationSummary: '',
                rawDataPath: '',
                finalConclusion: ''
            });
            // 格式化日期
            const formatDate = (dateStr) => {
                if (!dateStr) return '';
                const date = new Date(dateStr);
                return `${date.getFullYear()}-${(date.getMonth()+1).toString().padStart(2, '0')}-
${date.getDate().toString().padStart(2, '0')} ${date.getHours().toString().padStart(2,
'0')}:${date.getMinutes().toString().padStart(2, '0')}`;
            };
            // 获取实验历史数据
            const fetchData = async () => {
                try {
                    const params = {
                        ...queryParams,
                        page: currentPage.value - 1,
                        size: pageSize.value
                    };
                    const response = await axios.get('/api/experiment-history', { params });
                    tableData.value = response.data.content;
                    total.value = response.data.totalElements;
            } catch (error) {
                ElMessage.error('获取数据失败: ' + error.message);
            }
          };
          // 查看详情
          const viewDetail = async (row) => {
            try {
                const response = await axios.get(`/api/experiment-
history/${row.experimentHistoryId}`);
                Object.assign(detailData, response.data);
                // 获取执行人姓名（假设有用户服务）
                const userResponse = await axios.get(`/api/users/${detailData.executorId}`);
                detailData.executorName = userResponse.data.name;
                detailDialogVisible.value = true;
            } catch (error) {
                ElMessage.error('获取详情失败: ' + error.message);
            }
          };
          // 新增记录
          const showCreateDialog = () => {
            isEdit.value = false;
            Object.assign(editForm, {
                experimentHistoryId: null,
                experimentName: '',
                completeDate: new Date(),
                recordStatus: '有效',
                executorId: null,
                experimentType: '',
                operationSummary: '',
                rawDataPath: '',
                finalConclusion: ''
            });
            editDialogVisible.value = true;
          };
          // 编辑记录
          const editRecord = (row) => {
            isEdit.value = true;
            Object.assign(editForm, {
                experimentHistoryId: row.experimentHistoryId,
                experimentName: row.experimentName,
                completeDate: new Date(row.completeDate),
                recordStatus: row.recordStatus,
                executorId: row.executorId,
                experimentType: row.experimentType,
                operationSummary: row.operationSummary,
                rawDataPath: row.rawDataPath,
                finalConclusion: row.finalConclusion
            });
            editDialogVisible.value = true;
          };
          // 提交表单
          const submitForm = async () => {
```

```
    try {
      if (isEdit.value) {
        await axios.put(`/api/experiment-history/${editForm.experimentHistoryId}`,
editForm);
        ElMessage.success('更新成功');
      } else {
        await axios.post('/api/experiment-history', editForm);
        ElMessage.success('创建成功');
      }
      editDialogVisible.value = false;
      fetchData();
    } catch (error) {
      ElMessage.error('操作失败: ' + error.message);
    }
  };
  // 删除记录
  const deleteRecord = async (id) => {
    try {
      await ElMessageBox.confirm('确定要删除这条记录吗?', '警告', {
        confirmButtonText: '确定',
        cancelButtonText: '取消',
        type: 'warning'
      });
      await axios.delete(`/api/experiment-history/${id}`);
      ElMessage.success('删除成功');
      fetchData();
    } catch (error) {
      if (error !== 'cancel') {
        ElMessage.error('删除失败: ' + error.message);
      }
    }
  };
  // 归档记录
  const archiveRecord = async (id) => {
    try {
      await ElMessageBox.confirm('确定要归档这条记录吗?', '确认', {
        confirmButtonText: '确定',
        cancelButtonText: '取消',
        type: 'warning'
      });
      await axios.put(`/api/experiment-history/archive/${id}`);
      ElMessage.success('归档成功');
      fetchData();
    } catch (error) {
      if (error !== 'cancel') {
        ElMessage.error('归档失败: ' + error.message);
      }
    }
  };
  // 批量归档
  const batchArchive = async () => {
          if (selectedRows.value.length === 0) {
            ElMessage.warning('请至少选择一条记录');
            return;
          }
          try {
            await ElMessageBox.confirm(`确定要归档选中的 ${selectedRows.value.length} 条记
录吗?`, '确认', {
              confirmButtonText: '确定',
              cancelButtonText: '取消',
              type: 'warning'
            });
            const ids = selectedRows.value.map(row => row.experimentHistoryId);
            await axios.post('/api/experiment-history/batch-archive', { ids });
            ElMessage.success('批量归档成功');
            fetchData();
          } catch (error) {
            if (error !== 'cancel') {
              ElMessage.error('归档失败: ' + error.message);
            }
          }
        };
        // 导出记录
        const exportRecords = async () => {
          try {
            const params = {
              ...queryParams
            };
            const response = await axios.get('/api/experiment-history/export', {
              params,
              responseType: 'blob'
            });
            const url = window.URL.createObjectURL(new Blob([response.data]));
            const link = document.createElement('a');
            link.href = url;
            link.setAttribute('download', `实验历史记录_${new
Date().toISOString().slice(0,10)}.xlsx`);
            document.body.appendChild(link);
            link.click();
            document.body.removeChild(link);
          } catch (error) {
            ElMessage.error('导出失败: ' + error.message);
          }
        };
        // 表格选择变化
        const handleSelectionChange = (selection) => {
          selectedRows.value = selection;
        };
        // 分页处理
        const handleSizeChange = (val) => {
          pageSize.value = val;
          fetchData();
        };
```

```
            const handleCurrentChange = (val) => {
               currentPage.value = val;
               fetchData();
            };
            // 初始化加载数据
            onMounted(() => {
               fetchData();
            });
            return {
               tableData,
               currentPage,
               pageSize,
               total,
               queryParams,
               detailDialogVisible,
               editDialogVisible,
               isEdit,
               detailData,
               editForm,
               formatDate,
               fetchData,
               viewDetail,
               showCreateDialog,
               editRecord,
               submitForm,
               deleteRecord,
               archiveRecord,
               batchArchive,
               exportRecords,
               handleSelectionChange,
               handleSizeChange,
               handleCurrentChange
            };
         }
      }).use(ElementPlus).mount('#app');
   </script>
</body>
</html>
// 系统：绿色智能化学解决方案系统
// 功能：风险分析工具
// 描述：提供化学品危险性评估工具，根据化学特性自动生成风险评级建议
// 风险分析工具实体类
package com.example.chemicalsystem.domain;
import com.baomidou.mybatisplus.annotation.*;
import lombok.Data;
import java.util.Date;
@Data
@TableName("risk_analyzer")
public class RiskAnalyzer {
   @TableId(value = "risk_analyzer_id", type = IdType.AUTO)
   private Long riskAnalyzerId; // 风险分析 ID
   @TableField("analysis_name")
```

```
    private String analysisName; // 分析名称
    @TableField("creator_id")
    private Long creatorId; // 创建人 ID（关联用户表）
    @TableField("analysis_type")
    private String analysisType; // 分析类型枚举值
    @TableField("analysis_parameter")
    private String analysisParameter; // JSON 格式分析参数
    @TableField("evaluation_result")
    private String evaluationResult; // 评估结果文本
    @TableField(value = "created_time", fill = FieldFill.INSERT)
    private Date createdTime; // 创建时间
    @TableField(value = "updated_time", fill = FieldFill.INSERT_UPDATE)
    private Date updatedTime; // 最后更新时间
    @TableField("analysis_status")
    private String analysisStatus; // 分析状态枚举值
    @TableField("executed_time")
    private Date executedTime; // 执行时间
}
<!DOCTYPE html>
<html lang="zh-CN">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>风险分析工具</title>
  <link rel="stylesheet" href="https://unpkg.com/element-plus/dist/index.css">
  <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
  <script src="https://unpkg.com/element-plus/dist/index.full.js"></script>
  <script src="https://unpkg.com/axios/dist/axios.min.js"></script>
  <style>
    .container { padding: 20px; }
    .toolbar { margin-bottom: 20px; }
    .json-editor { height: 200px; }
    .status-tag { margin-left: 10px; }
  </style>
</head>
<body>
  <div id="app">
    <el-container class="container">
      <el-header>
        <h1>风险分析工具</h1>
      </el-header>
      <el-main>
        <!-- 工具栏 -->
        <div class="toolbar">
          <el-button type="primary" @click="openCreateDialog">新建分析</el-button>
          <el-button @click="refreshList">刷新列表</el-button>
        </div>
        <!-- 分析列表 -->
        <el-table :data="analysisList" border>
          <el-table-column prop="analysisName" label="分析名称" width="180"></el-table-column>
          <el-table-column label="分析类型" width="150">
          <template #default="scope">
            {{ scope.row.analysisType }}
            <el-tag v-if="scope.row.analysisStatus === '草稿'" type="warning" class="status-tag">草稿
</el-tag>
            <el-tag v-if="scope.row.analysisStatus === '已执行'" type="success" class="status-tag">已
执行</el-tag>
          </template>
        </el-table-column>
        <el-table-column prop="creatorName" label="创建人" width="120"></el-table-column>
        <el-table-column prop="createdTime" label="创建时间" width="180"></el-table-column>
        <el-table-column label="操作" width="280">
          <template #default="scope">
            <el-button v-if="scope.row.analysisStatus === '草稿'"
                    type="primary" size="small"
                    @click="openExecuteDialog(scope.row)">执行评估</el-button>
            <el-button v-if="scope.row.analysisStatus === '草稿'"
                    type="warning" size="small"
                    @click="openEditDialog(scope.row)">编辑</el-button>
            <el-button type="info" size="small"
                    @click="viewDetails(scope.row)">详情</el-button>
            <el-button type="danger" size="small"
                    @click="deleteAnalysis(scope.row.riskAnalyzerId)">删除</el-button>
          </template>
        </el-table-column>
      </el-table>
    </el-main>
  </el-container>
  <!-- 新建/编辑对话框 -->
  <el-dialog v-model="showFormDialog" :title="formTitle" width="600px">
    <el-form :model="currentAnalysis" label-width="100px">
      <el-form-item label="分析名称" required>
        <el-input v-model="currentAnalysis.analysisName" placeholder="请输入分析名称"></el-
input>
      </el-form-item>
      <el-form-item label="分析类型" required>
        <el-select v-model="currentAnalysis.analysisType" placeholder="请选择分析类型">
          <el-option label="化学品危险性" value="化学品危险性"></el-option>
          <el-option label="实验操作风险" value="实验操作风险"></el-option>
          <el-option label="存储风险" value="存储风险"></el-option>
          <el-option label="废弃物风险" value="废弃物风险"></el-option>
        </el-select>
      </el-form-item>
      <el-form-item label="分析参数">
        <el-input
          v-model="currentAnalysis.analysisParameter"
          type="textarea"
          :rows="5"
          class="json-editor"
          placeholder="请输入 JSON 格式的分析参数"></el-input>
      </el-form-item>
      <el-form-item>
        <el-button type="primary" @click="saveAnalysis">保存</el-button>
        <el-button @click="showFormDialog = false">取消</el-button>
      </el-form-item>
    </el-form>
  </el-dialog>
  <!-- 执行评估对话框 -->
  <el-dialog v-model="showExecuteDialog" title="执行风险评估" width="800px">
    <el-form :model="executeForm" label-width="100px">
      <el-form-item label="分析名称">
        <span>{{ executeForm.analysisName }}</span>
      </el-form-item>
      <el-form-item label="分析类型">
        <span>{{ executeForm.analysisType }}</span>
      </el-form-item>
      <el-form-item label="评估结果" required>
        <el-input
          v-model="executeForm.evaluationResult"
          type="textarea"
          :rows="8"
          placeholder="请输入评估结果文本"></el-input>
      </el-form-item>
      <el-form-item>
        <el-button type="success" @click="confirmExecute">确认执行</el-button>
        <el-button @click="showExecuteDialog = false">取消</el-button>
      </el-form-item>
    </el-form>
  </el-dialog>
  <!-- 详情对话框 -->
  <el-dialog v-model="showDetailDialog" title="分析详情" width="800px">
    <el-descriptions :column="2" border>
      <el-descriptions-item label="分析 ID">{{ currentAnalysis.riskAnalyzerId }}</el-descriptions-
item>
      <el-descriptions-item label="分析名称">{{ currentAnalysis.analysisName }}</el-descriptions-
item>
      <el-descriptions-item label="分析类型">{{ currentAnalysis.analysisType }}</el-descriptions-
item>
      <el-descriptions-item label="分析状态">
        <el-tag v-if="currentAnalysis.analysisStatus === '草稿'" type="warning">草稿</el-tag>
        <el-tag v-if="currentAnalysis.analysisStatus === '已执行'" type="success">已执行</el-tag>
      </el-descriptions-item>
      <el-descriptions-item label="创建人">{{ currentAnalysis.creatorName }}</el-descriptions-item>
      <el-descriptions-item label="创建时间">{{ currentAnalysis.createdTime }}</el-descriptions-
item>
      <el-descriptions-item label="执行时间">{{ currentAnalysis.executedTime || '未执行' }}</el-
descriptions-item>
      <el-descriptions-item label="最后更新时间">{{ currentAnalysis.updatedTime }}</el-
descriptions-item>
    </el-descriptions>
    <el-divider></el-divider>
```

```
      <el-descriptions title="分析参数" :column="1" border>
        <el-descriptions-item>
          <pre>{{ formatJSON(currentAnalysis.analysisParameter) }}</pre>
        </el-descriptions-item>
      </el-descriptions>
      <el-descriptions title="评估结果" :column="1" border v-if="currentAnalysis.evaluationResult">
        <el-descriptions-item>
          {{ currentAnalysis.evaluationResult }}
        </el-descriptions-item>
      </el-descriptions>
    </el-dialog>
  </div>
  <script>
    const { createApp, ref, reactive, onMounted } = Vue;
    const { ElMessage, ElMessageBox } = ElementPlus;
    createApp({
      setup() {
        // 状态管理
        const analysisList = ref([]);
        const showFormDialog = ref(false);
        const showExecuteDialog = ref(false);
        const showDetailDialog = ref(false);
        const formTitle = ref('新建风险分析');
        const currentAnalysis = reactive({
          riskAnalyzerId: null,
          analysisName: '',
          analysisType: '',
          analysisParameter: '',
          analysisStatus: '草稿'
        });
        const executeForm = reactive({
          riskAnalyzerId: null,
          analysisName: '',
          analysisType: '',
          evaluationResult: ''
        });
        // 初始化加载数据
        const loadAnalysisList = async () => {
          try {
            const response = await axios.get('/api/risk-analyzer');
            analysisList.value = response.data.map(item => ({
              ...item,
              createdTime: formatDateTime(item.createdTime),
              updatedTime: formatDateTime(item.updatedTime),
              executedTime: formatDateTime(item.executedTime)
            }));
          } catch (error) {
            ElMessage.error('加载分析列表失败: ' + error.message);
          }
        };
        // 打开新建对话框
        const openCreateDialog = () => {
```

```
  resetCurrentAnalysis();
  formTitle.value = '新建风险分析';
  showFormDialog.value = true;
};
// 打开编辑对话框
const openEditDialog = (analysis) => {
  Object.assign(currentAnalysis, analysis);
  formTitle.value = '编辑风险分析';
  showFormDialog.value = true;
};
// 打开执行对话框
const openExecuteDialog = (analysis) => {
  executeForm.riskAnalyzerId = analysis.riskAnalyzerId;
  executeForm.analysisName = analysis.analysisName;
  executeForm.analysisType = analysis.analysisType;
  executeForm.evaluationResult = '';
  showExecuteDialog.value = true;
};
// 查看详情
const viewDetails = (analysis) => {
  Object.assign(currentAnalysis, analysis);
  showDetailDialog.value = true;
};
// 保存分析（新建/更新）
const saveAnalysis = async () => {
  try {
    if (currentAnalysis.riskAnalyzerId) {
      await axios.put('/api/risk-analyzer', currentAnalysis);
      ElMessage.success('分析草稿保存成功');
    } else {
      const id = await axios.post('/api/risk-analyzer', currentAnalysis);
      currentAnalysis.riskAnalyzerId = id;
      ElMessage.success('新建分析创建成功');
    }
    showFormDialog.value = false;
    loadAnalysisList();
  } catch (error) {
    ElMessage.error('保存失败: ' + error.message);
  }
};
// 执行评估
const confirmExecute = async () => {
  try {
    await axios.put(`/api/risk-analyzer/execute/${executeForm.riskAnalyzerId}`,
                executeForm.evaluationResult, {
                  headers: { 'Content-Type': 'text/plain' }
                });
    ElMessage.success('风险评估执行成功');
    showExecuteDialog.value = false;
    loadAnalysisList();
  } catch (error) {
    ElMessage.error('执行失败: ' + error.message);
    }
  };
  // 删除分析
  const deleteAnalysis = (id) => {
    ElMessageBox.confirm('确定要删除此分析记录吗?', '警告', {
      confirmButtonText: '确定',
      cancelButtonText: '取消',
      type: 'warning'
    }).then(async () => {
      try {
        await axios.delete(`/api/risk-analyzer/${id}`);
        ElMessage.success('分析记录已删除');
        loadAnalysisList();
      } catch (error) {
        ElMessage.error('删除失败: ' + error.message);
      }
    }).catch(() => {});
  };
  // 刷新列表
  const refreshList = () => {
    loadAnalysisList();
    ElMessage.info('列表已刷新');
  };
  // 重置当前分析对象
  const resetCurrentAnalysis = () => {
    Object.assign(currentAnalysis, {
      riskAnalyzerId: null,
      analysisName: '',
      analysisType: '',
      analysisParameter: '',
      analysisStatus: '草稿'
    });
  };
  // 格式化日期时间
  const formatDateTime = (timestamp) => {
    if (!timestamp) return '';
    const date = new Date(timestamp);
    return date.toLocaleString();
  };
  // 格式化 JSON 显示
  const formatJSON = (jsonStr) => {
    if (!jsonStr) return '无参数';
    try {
      return JSON.stringify(JSON.parse(jsonStr), null, 2);
    } catch (e) {
      return jsonStr;
    }
  };
  // 初始化加载数据
  onMounted(() => {
    loadAnalysisList();
  });
```

```
        return {
          analysisList,
          showFormDialog,
          showExecuteDialog,
          showDetailDialog,
          formTitle,
          currentAnalysis,
          executeForm,
          openCreateDialog,
          openEditDialog,
          openExecuteDialog,
          viewDetails,
          saveAnalysis,
          confirmExecute,
          deleteAnalysis,
          refreshList,
          formatJSON
        };
      }
    }).use(ElementPlus).mount('#app');
  </script>
</body>
</html>
// 系统：绿色智能化学解决方案系统
// 功能：风险状态监控
// 描述：持续追踪高风险化学品和实验过程的状态变化，设置预警阈值
// 枚举类型定义
public enum RiskStatus {
    NORMAL, WARNING, DANGER
}
public enum NotifyMethod {
    EMAIL, SMS, INNER_MSG
}
public enum CheckCycle {
    DAILY, WEEKLY, MONTHLY
}
// 实体类
import javax.persistence.*;
import java.time.LocalDateTime;
@Entity
@Table(name = "risk_monitor")
public class RiskMonitor {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "risk_monitor_id")
    private Long id;  // 监控记录 ID
    @Column(name = "monitor_item", nullable = false, length = 255)
    private String monitorItem;  // 监控项名称
    @Enumerated(EnumType.STRING)
    @Column(name = "current_status", nullable = false, length = 20)
    private RiskStatus currentStatus;  // 当前状态（正常/警告/危险）
    @Column(name = "last_check_time", nullable = false)
    private LocalDateTime lastCheckTime;  // 最后检查时间
    @Column(name = "creator_id", nullable = false)
    private Long creatorId;  // 创建人 ID
    @Column(name = "create_time", nullable = false)
    private LocalDateTime createTime;  // 创建时间
    @Column(name = "last_updater_id")
    private Long lastUpdaterId;  // 最后更新人 ID
    @Column(name = "last_update_time")
    private LocalDateTime lastUpdateTime;  // 最后更新时间
    @Column(name = "related_chemical_id")
    private Long relatedChemicalId;  // 关联化学品 ID
    @Column(name = "threshold_value", precision = 10, scale = 2)
    private Double thresholdValue;  // 监控阈值
    @Enumerated(EnumType.STRING)
    @Column(name = "check_cycle", nullable = false, length = 20)
    private CheckCycle checkCycle;  // 监控周期（每日/每周/每月）
    @Enumerated(EnumType.STRING)
    @Column(name = "notify_method", nullable = false, length = 20)
    private NotifyMethod notifyMethod;  // 预警通知方式（邮件/短信/站内信）
    // 构造方法
    public RiskMonitor() {}
    // Getter 和 Setter 方法
    public Long getId() { return id; }
    public void setId(Long id) { this.id = id; }
    public String getMonitorItem() { return monitorItem; }
    public void setMonitorItem(String monitorItem) { this.monitorItem = monitorItem; }
    public RiskStatus getCurrentStatus() { return currentStatus; }
    public void setCurrentStatus(RiskStatus currentStatus) { this.currentStatus = currentStatus; }
    public LocalDateTime getLastCheckTime() { return lastCheckTime; }
    public void setLastCheckTime(LocalDateTime lastCheckTime) { this.lastCheckTime =
lastCheckTime; }
    public Long getCreatorId() { return creatorId; }
    public void setCreatorId(Long creatorId) { this.creatorId = creatorId; }
    public LocalDateTime getCreateTime() { return createTime; }
    public void setCreateTime(LocalDateTime createTime) { this.createTime = createTime; }
    public Long getLastUpdaterId() { return lastUpdaterId; }
    public void setLastUpdaterId(Long lastUpdaterId) { this.lastUpdaterId = lastUpdaterId; }
    public LocalDateTime getLastUpdateTime() { return lastUpdateTime; }
    public void setLastUpdateTime(LocalDateTime lastUpdateTime) { this.lastUpdateTime =
lastUpdateTime; }
    public Long getRelatedChemicalId() { return relatedChemicalId; }
    public void setRelatedChemicalId(Long relatedChemicalId) { this.relatedChemicalId =
relatedChemicalId; }
    public Double getThresholdValue() { return thresholdValue; }
    public void setThresholdValue(Double thresholdValue) { this.thresholdValue = thresholdValue; }
    public CheckCycle getCheckCycle() { return checkCycle; }
    public void setCheckCycle(CheckCycle checkCycle) { this.checkCycle = checkCycle; }
    public NotifyMethod getNotifyMethod() { return notifyMethod; }
    public void setNotifyMethod(NotifyMethod notifyMethod) { this.notifyMethod = notifyMethod; }
}
// 数据访问层
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
@Repository
public interface RiskMonitorRepository extends JpaRepository<RiskMonitor, Long> {
    // 继承 JPA 基础 CRUD 方法
}
// 服务层
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import javax.transaction.Transactional;
import java.time.LocalDateTime;
import java.util.List;
@Service
@Transactional
public class RiskMonitorService {
    @Autowired
    private RiskMonitorRepository repository;
    /**
     * 添加监控项
     * @param monitor 监控项实体
     * @param creatorId 操作人 ID
     * @return 保存后的实体
     */
    public RiskMonitor addMonitor(RiskMonitor monitor, Long creatorId) {
        monitor.setCreateTime(LocalDateTime.now());
        monitor.setCreatorId(creatorId);
        monitor.setLastCheckTime(LocalDateTime.now());
        return repository.save(monitor);
    }
    /**
     * 更新监控状态
     * @param id 监控记录 ID
     * @param status 新状态
     * @param updaterId 操作人 ID
     * @return 更新后的实体
     */
    public RiskMonitor updateStatus(Long id, RiskStatus status, Long updaterId) {
        RiskMonitor monitor = repository.findById(id).orElseThrow();
        monitor.setCurrentStatus(status);
        monitor.setLastUpdaterId(updaterId);
        monitor.setLastUpdateTime(LocalDateTime.now());
        return repository.save(monitor);
    }
    /**
     * 删除监控项
     * @param id 监控记录 ID
     */
    public void deleteMonitor(Long id) {
        repository.deleteById(id);
    }
    /**
     * 获取所有监控项
     * @return 监控项列表
     */
    public List<RiskMonitor> getAllMonitors() {
        return repository.findAll();
    }
    /**
     * 更新监控阈值
     * @param id 监控记录 ID
     * @param threshold 新阈值
     * @param updaterId 操作人 ID
     * @return 更新后的实体
     */
    public RiskMonitor updateThreshold(Long id, Double threshold, Long updaterId) {
        RiskMonitor monitor = repository.findById(id).orElseThrow();
        monitor.setThresholdValue(threshold);
        monitor.setLastUpdaterId(updaterId);
        monitor.setLastUpdateTime(LocalDateTime.now());
        return repository.save(monitor);
    }
    /**
     * 获取监控趋势数据
     * @param chemicalId 关联化学品 ID
     * @return 相关监控项列表
     */
    public List<RiskMonitor> getTrendData(Long chemicalId) {
        return repository.findByRelatedChemicalId(chemicalId);
    }
}
// 控制器层
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
import java.util.List;
@RestController
@RequestMapping("/api/risk-monitor")
public class RiskMonitorController {
    @Autowired
    private RiskMonitorService service;
    // 添加监控项
    @PostMapping("/add")
    public RiskMonitor addMonitor(@RequestBody RiskMonitor monitor, @RequestParam Long
creatorId) {
        return service.addMonitor(monitor, creatorId);
    }
    // 更新状态
    @PutMapping("/update-status/{id}")
    public RiskMonitor updateStatus(@PathVariable Long id, @RequestParam RiskStatus status,
                                  @RequestParam Long updaterId) {
        return service.updateStatus(id, status, updaterId);
    }
    // 删除监控项
    @DeleteMapping("/delete/{id}")
    public void deleteMonitor(@PathVariable Long id) {
        service.deleteMonitor(id);
    }
    // 获取所有监控项
    @GetMapping("/all")
    public List<RiskMonitor> getAllMonitors() {
        return service.getAllMonitors();
    }
    // 更新阈值
    @PutMapping("/update-threshold/{id}")
    public RiskMonitor updateThreshold(@PathVariable Long id, @RequestParam Double threshold,
                                       @RequestParam Long updaterId) {
        return service.updateThreshold(id, threshold, updaterId);
    }
    // 查看趋势
    @GetMapping("/trend/{chemicalId}")
    public List<RiskMonitor> getTrendData(@PathVariable Long chemicalId) {
        return service.getTrendData(chemicalId);
    }
}
```