 燕山大学

Java Web开发技术

三级项目报告

题 目：  图书馆管理系统

学 院：   信息科学与工程学院

专业班级：   计算机科学与技术

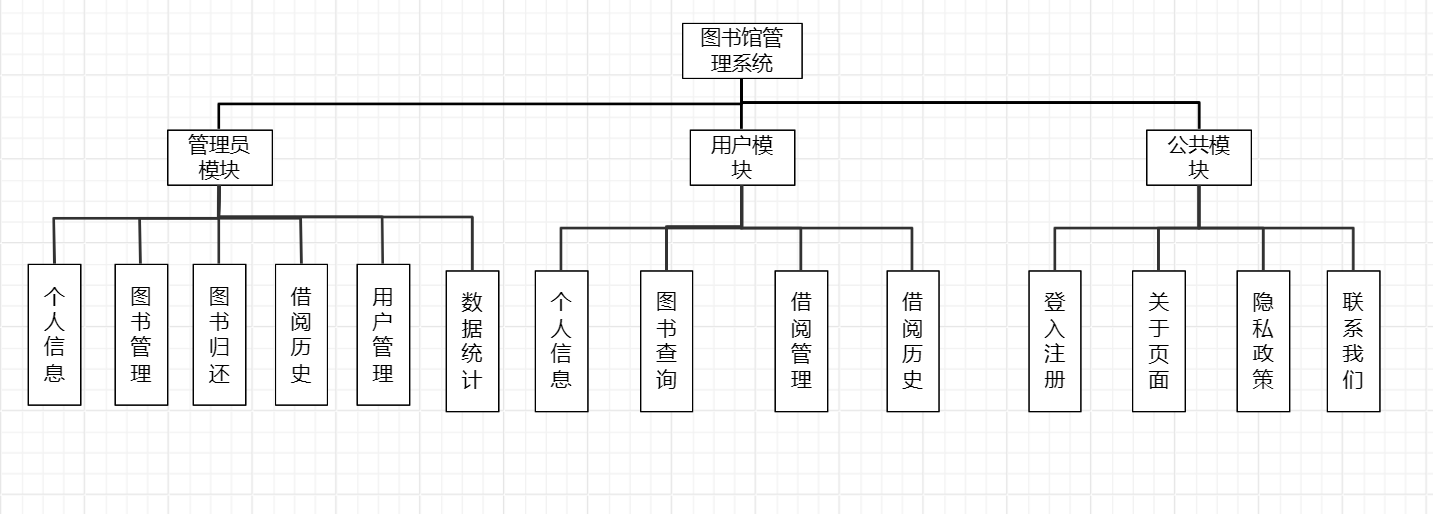
姓 名：  那圣奇

学 号：   202311040130

报告日期 2025年 4 月 12 日

## 1系统功能

系统功能图如图1所示。



图表 1系统功能结构图

图书馆管理系统分为三个模块组成，分别是管理员模块、用户模块、公共模块。

1. **公共模块**

登入注册：管理员和用户登入注册的页面，用于验证账号密码和注册新用户，并有记住和忘记密码功能，有关于密码的信息在前后端都是以哈希加密存储，并在后端用私钥对前端的哈希值二次加密存入数据库中。

关于页面：用于介绍图书馆的建造、扩建、发展历史，向用户展示图书馆的相关信息。

隐私政策：向用户展示相关的隐私政策，用于告知用户的相关隐私信息。

联系我们：展示图书馆的联系信息，用于用户联系图书馆管理员。

1. 用户模块

个人信息：展示用户的学号、姓名、性别、联系电话、身份、状态等，

并提供个人信息的编辑修改保存，和密码的更改。

图书查询：通过图书编号、书名、作者查找图书的信息，包含图ID、书名、作者、出版社、出版时间等，用户并可以看见图书是否被借阅的状态，可以进行批量借阅实现快捷地借阅。

借阅管理：用户在借阅图书后可以看到相关借阅信息，包含借阅图书ID、书名、借阅时间和还书时间，而且用户可以在还书时间前进行续约，当超出还书时间没有还书时，将会先时逾期，并无法再次续约。

借阅历史：用户还书后可以看到借阅历史，借阅历史有着借阅图书的相关信息以及用户借阅的时间、还书时间和实际的还书时间。系统可以根据还书时间和实际还书时间进行比对，来确认用户是否逾期，并可以专门显示逾期归还的图书。用户对借阅历史实现逻辑删除，不对用户本人显示。

1. 管理员模块

个人信息：个人信息：展示管理员的工号、姓名、性别、联系电话、身份、状态等，并提供个人信息的编辑修改保存，和密码的更改。

图书管理：管理员可以查看所有图书，并实时对图书进行信息修改保存，并可以批量删除图书，当然图书已被借阅时后端将会检查到这一状态，阻止管理员对已借阅图书进行删除。

图书归还：管理员可以根据图书ID、读者ID对已借阅图书进行搜索，管理员也可以搜索全部被借阅的图书，并有批量归还图书功能。

借阅历史：管理员可以查看所有用户的借阅历史（即使用户本人进行逻辑删除），借阅历史可以显示用户借阅的时间、还书时间和实际的还书时间。系统可以根据还书时间和实际还书时间进行比对，来确认用户是否逾期，并可以专门显示逾期归还的图书。

用户管理：管理员可以根据用户ID和姓名进行搜索，结果以信息卡片的形式进行展示，卡片包含用户的相关信息，管理员可以对用户进行管理，包括冻结和注销账户的功能，被冻结的用户无法实现借阅图书等操作，管理也可以对冻结的用户进行解冻。

数据统计：为图书管理员提供了强大的工具，能够对图书和用户数据进行深入的可视化分析。通过直观、美观的图表和图形，管理员可以轻松地掌握图书馆的运营情况，包括图书的借阅情况、用户的分布这些数据以视觉化的方式呈现，使得管理员能够快速识别出趋势、模式和潜在问题，从而做出更加明智的决策。

## 2 数据库设计

1. 用户及管理员表

用户及管理员表如图二所示。  
user＿id：学号/公号，varchar类型

password＿hash：用于存储后端加密过的密码哈希值

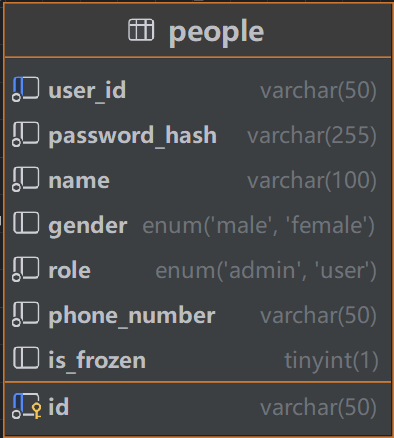
name：人物姓名

gender：人物性别，枚举类型生成

role：人物身份，分为管理员和用户

phone＿number：电话号码，使用了正则检查，确保格式正确

is\_frozen：boolean类型，用于判断用户是否被冻结

id:主键id，由雪花算法生成，确保了主键的唯一性

图表 2 用户及管理员表

2.图书表

图书表如图三所示

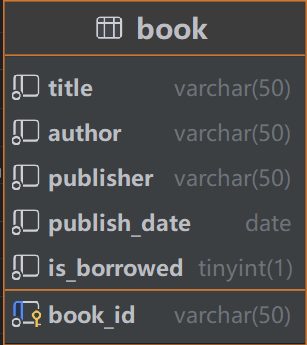
title：书名，varchar类型

author：作者名，varchar类型

publisher：出版社，varchar类型

publish\_date：出版日期，date类型

is\_borrowed：是否被借阅，boolean类型

book\_id:图书编号，varchar类型  


图表 3 图书表

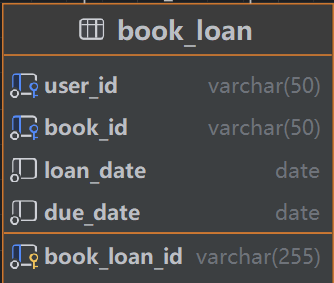
3.图书借阅表

图书借阅表如图四所示  
user\_id：用户ID

book\_id：图书ID

loan\_date：借阅日期，date类型

due\_date：预计归还时间，date类型

book\_loan\_id：借阅流水号，varchar类型

图表 4 图书借阅表

4.图书借阅历史表

图书借阅历史表如图五所示

user\_id：用户ID

book\_id：图书ID

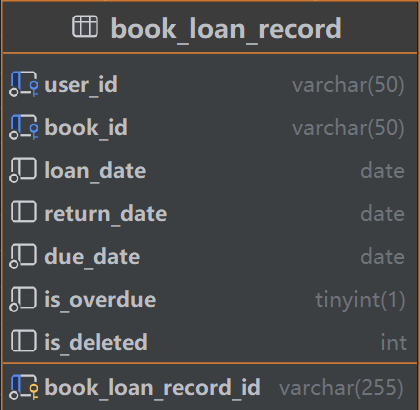
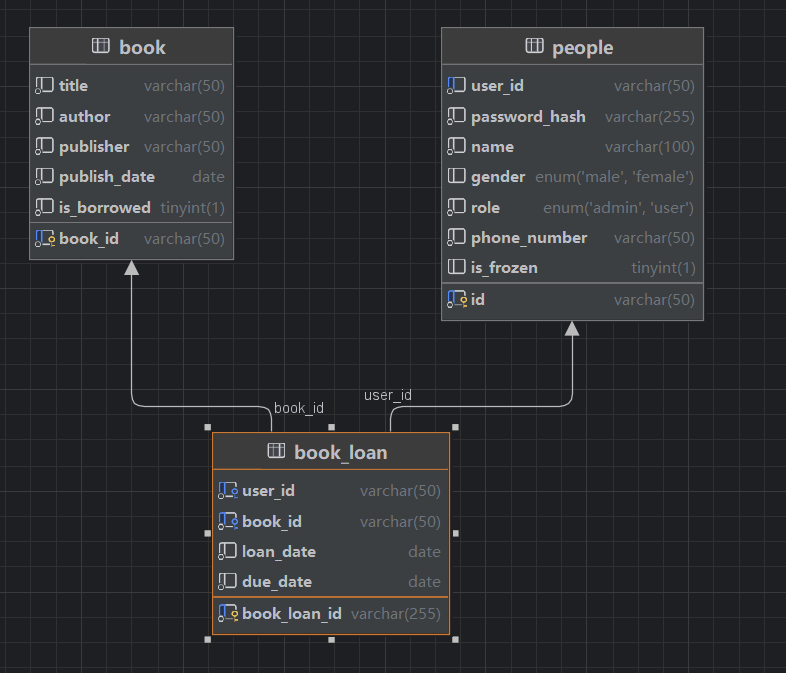
loan\_date：借阅日期，date类型

due\_date：预计归还时间，date类型

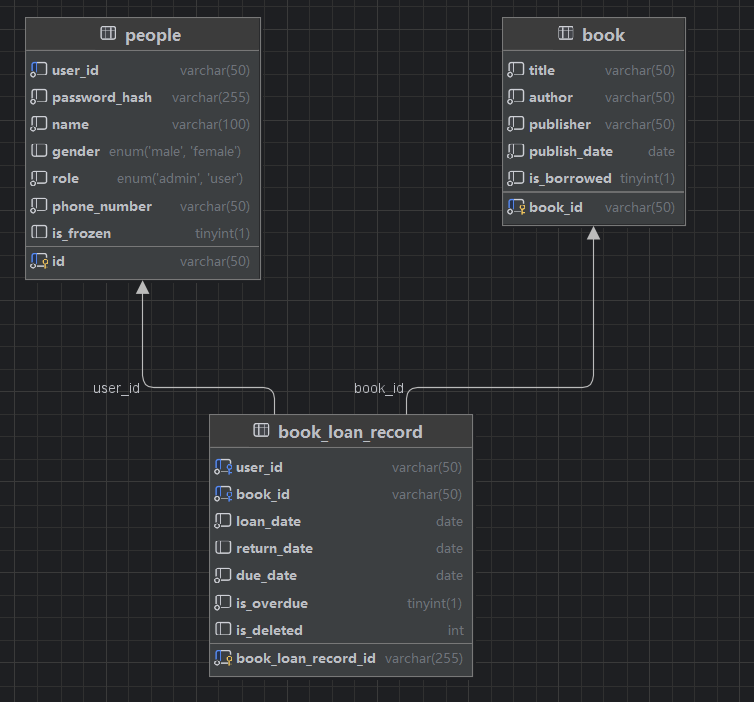
return\_date：实际还书时间，date类型

is\_overdue:是否逾期，boolean类型

is\_deleted:是否逻辑删除，用来判断用户是否进行了逻辑删除，逻辑删除后，数据仅管理员可见，integer类型

book\_loan\_record\_id:借阅历史ID，varchar类型

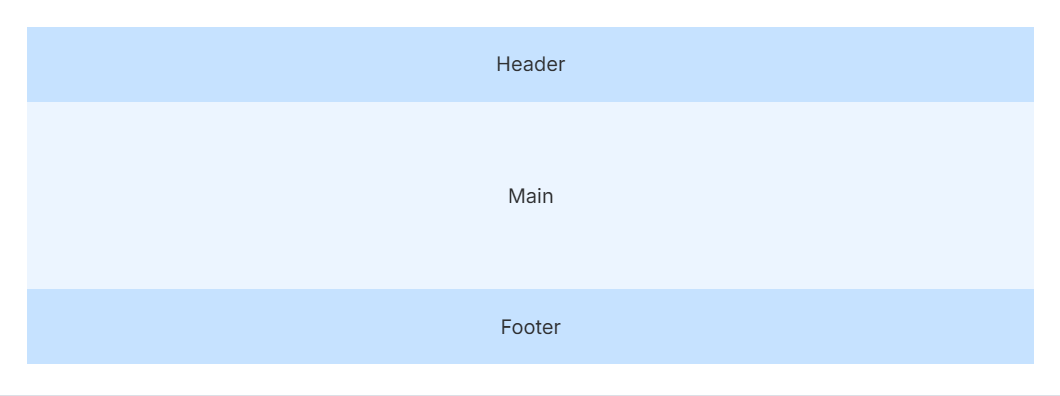
图表 5 图书借阅历史表

相关表的ER 图如图六、图七所示

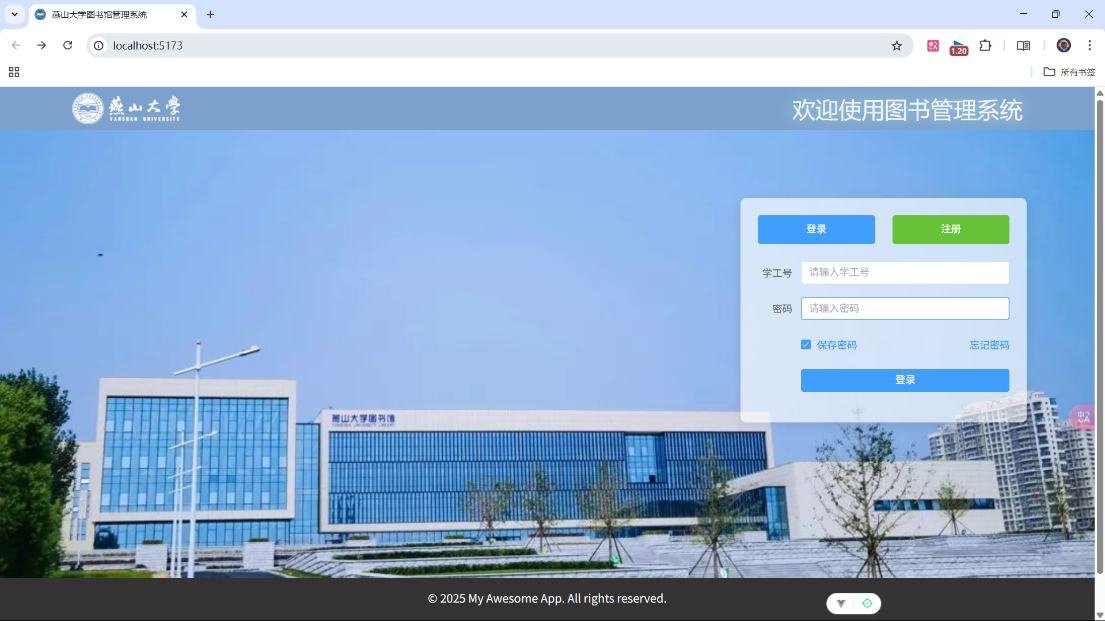
图表 6 people、book、boo\_loan\_record表ER关系图

图表 7 book、people、book\_loan表的ER关系图

## 3 结果展示

如图八所示，初始页面由header、main、footer三个布局组成。

图表 8 初始页面布局

main包含登入、注册组件，用户和管理员保存密码进行下次快捷登入，如果忘记密码可以点击忘记密码的连接，跳转到密码重置页面。footer 是有关网站的介绍包含关于我们、隐私政策、联系我们三个跳转链接，帮助用户提前了解图书馆的相关信息。

图表 9 初始页面

点击忘记密码跳转密码重置页面如图九所示，用户通过输入学工号和数字验证码，就可以重置自己的密码，密码重置成功后跳转到初始页面登入。

点击footer中的关于我们跳转到about页面，页面效果如图十一所示，页面简单介绍了图书馆的概况，对图书馆也个简短概括。

图表 10 关于我们页面

图表 11 忘记密码页面

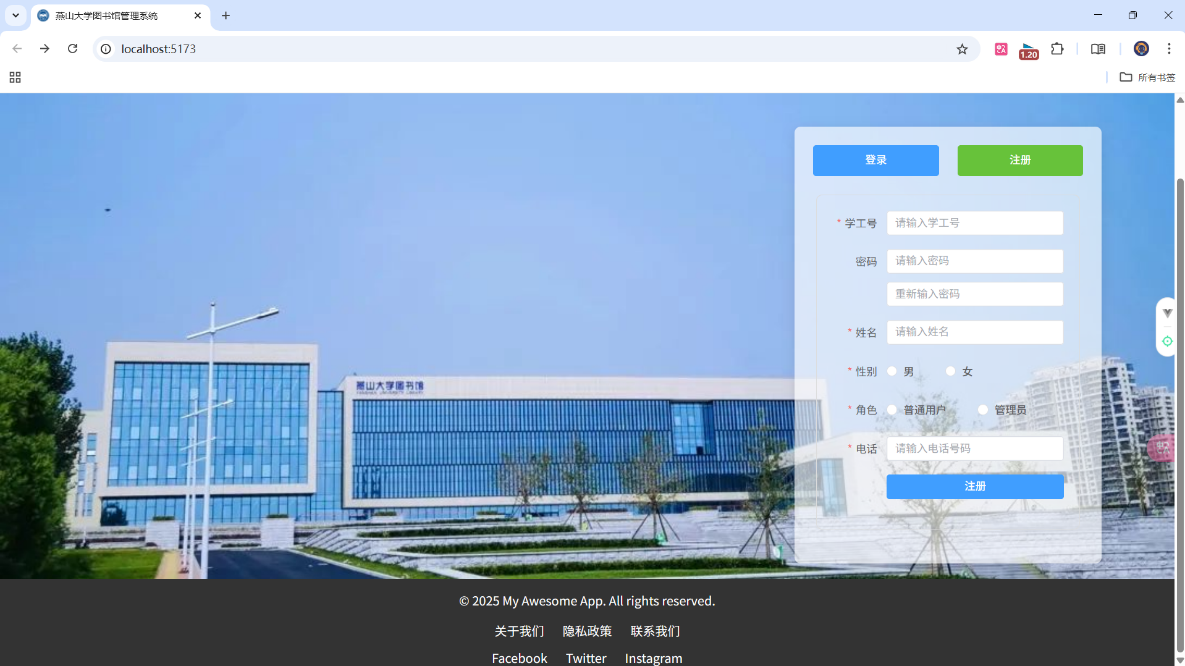
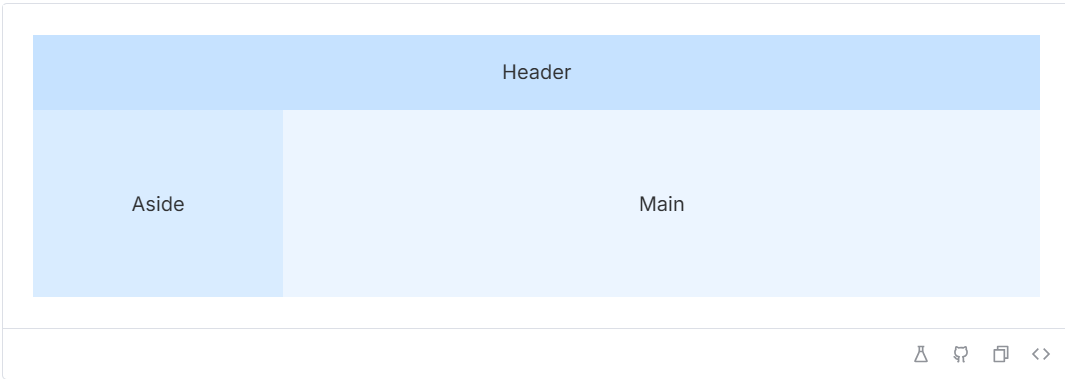
点击隐私政策跳转到privacy页面，效果如图十二所示，是关于网站隐私信息采用的说明，提前告知用户相关注意事项。

点击了联系我们，就会跳转contact页面，页面效果如图十三所示，告知用户图书馆的联系方式，方便联系。

图表 12 联系页面

图表 13 隐私页面

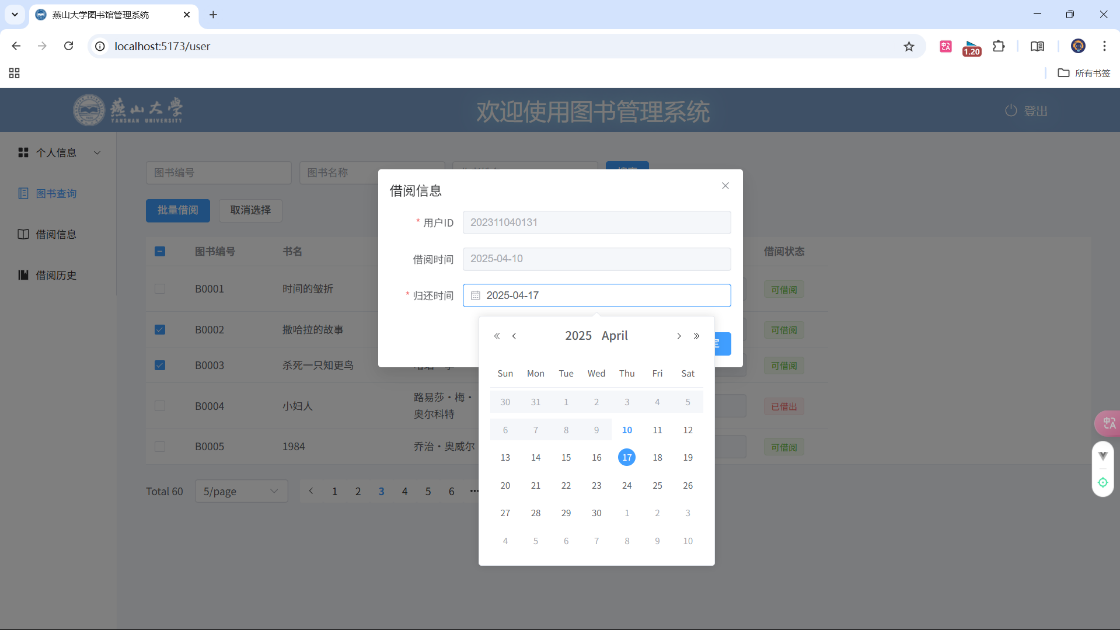
点击注册页面就会切换到注册组件，效果如图十四所示，用户填完相关信息，即可完成账号注册，注册后自动切换到登入组件进行登入。

登入后页面会自动先跳转到图书查询，用户页面分为header、container、main组件。

图表 14 主页面布局

图表 13 注册组件视图

图书查询如图十五所示，用户可以根据图书编号，图书名称，作者姓名进行查询，过多的数据会被分页存储。用户可以实时查看图书的借阅情况，也可以进行批量图书借阅。借阅时会弹出对话框，让用户选择还书时间，使用了element-plus的时间选择器组件，并限制了日期稚嫩给指定当前日期之后，提高准确行，借阅成功后则会弹出success消息通知。

 个人信息如图十六所示，个人信息包含个人信息编辑和更改密码两个子组件。用户可以查看个人信息，并且支持编辑功能，点击编辑按钮编辑后点击保存按钮，会弹出确认消息框，确认后保存信息，实时查看已修改信息。

图表 15 个人信息修改

图表 16 图书查询页面

密码修改如图十七所示，用户重复输入两次相同的密码,进行密码更改，如果两次密码相同，则密码成功更改。否则，则会表单验证不成功。

 借阅信息页面如图十八所示，将会搜寻用户所有已借阅图书的信息，用户可以对借阅图书进行续借，点击续借按钮弹出图书归还时间的重新选择，并增加了便捷的延长时间选项进行快速选择。

图表 18 借阅信息页面

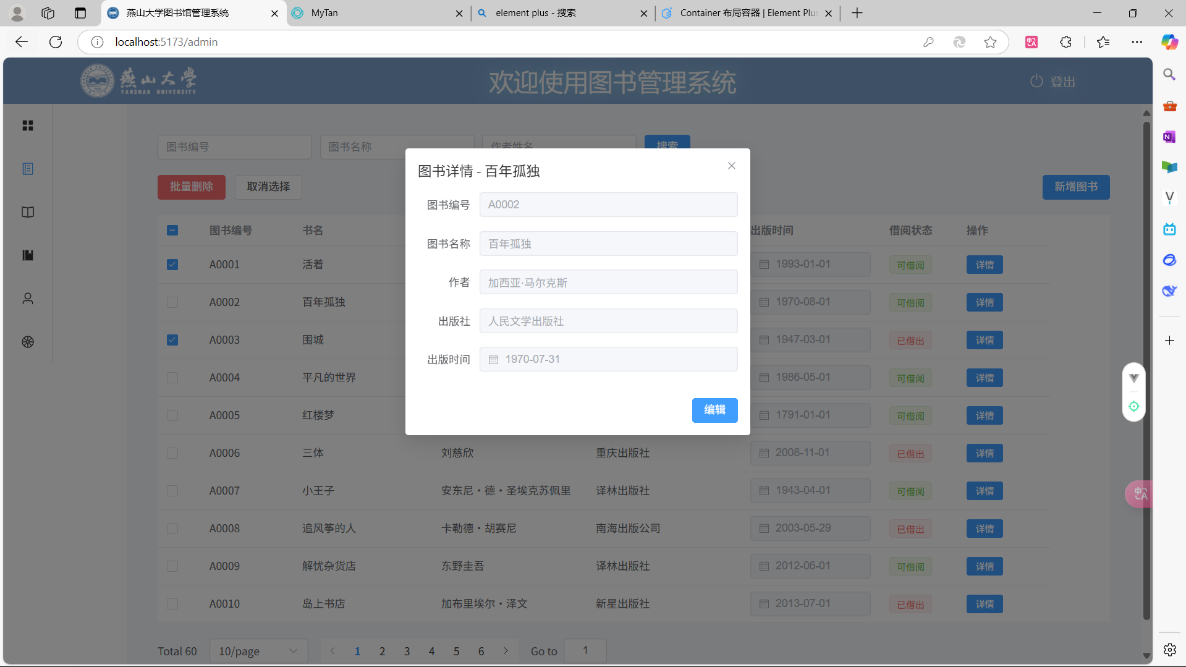
图表 17 更改密码页面

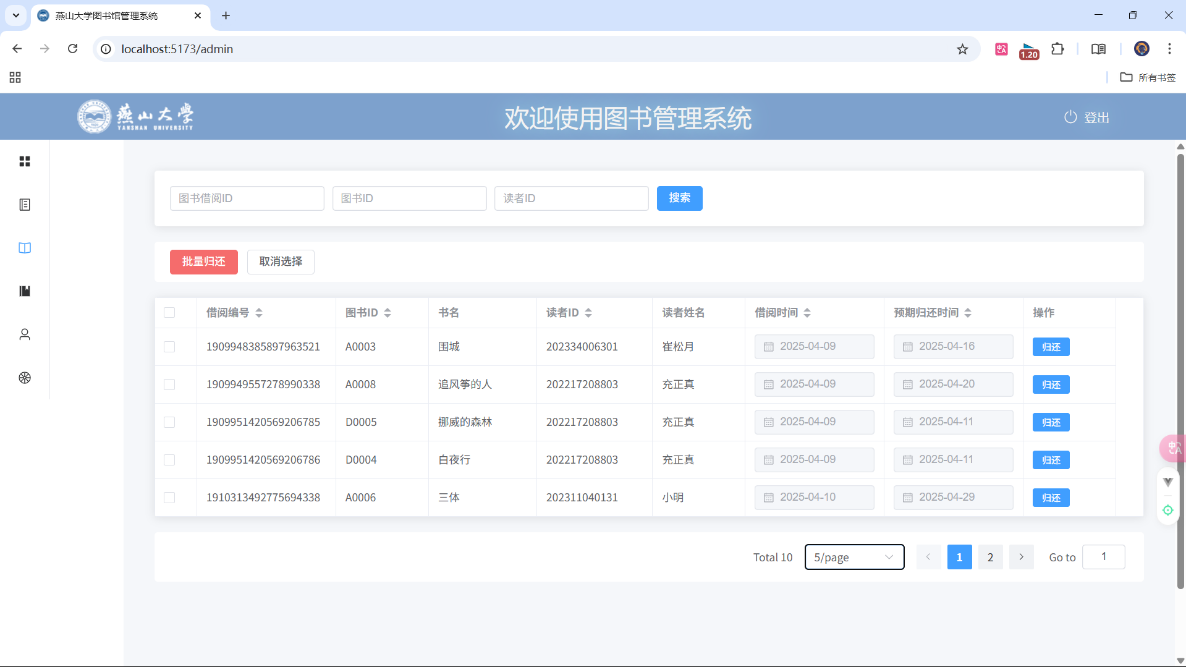
借阅历史页面如图十九所示，将会搜索用户所有借阅历史记录，历史记录有着借阅图书的相关信息，并且会显示实际归还时间是否早于预定归时间，即是否逾期，如果逾期则显示红色，没有则显示绿色。用户可以对借阅历史进行批量逻辑删除，逻辑删除后的数据不会对用户本人显示，但是会对管理员可见。

图表 19 用户借阅历史页面

管理员登入进去后会默认进入图书管理页面，如图二十所示，管理员可以查看所有图书的信息和借阅情况。并可以对图书进行批量的删除，当删除的图书已被借阅时，对被组织删除，并返回相关的错误消息通知。管理员也可以实时对图书进行信息编辑并保存，保存的图书信息会及时更新

图表 20 图书管理页面

到用户的图书搜索界面当中。

借阅管理页面如图二十一所示，管理员可以根据相关参数进行查询，可以观察到相关的借阅信息，支持管理员进行批量归还操作。

图表 21 借阅管理页面

 借阅历史页面如图二十二所示，管理员可以查看全部借阅历史，无论用户是否删除。并可以查看是否逾期，支持仅显示逾期的借阅图书数据。

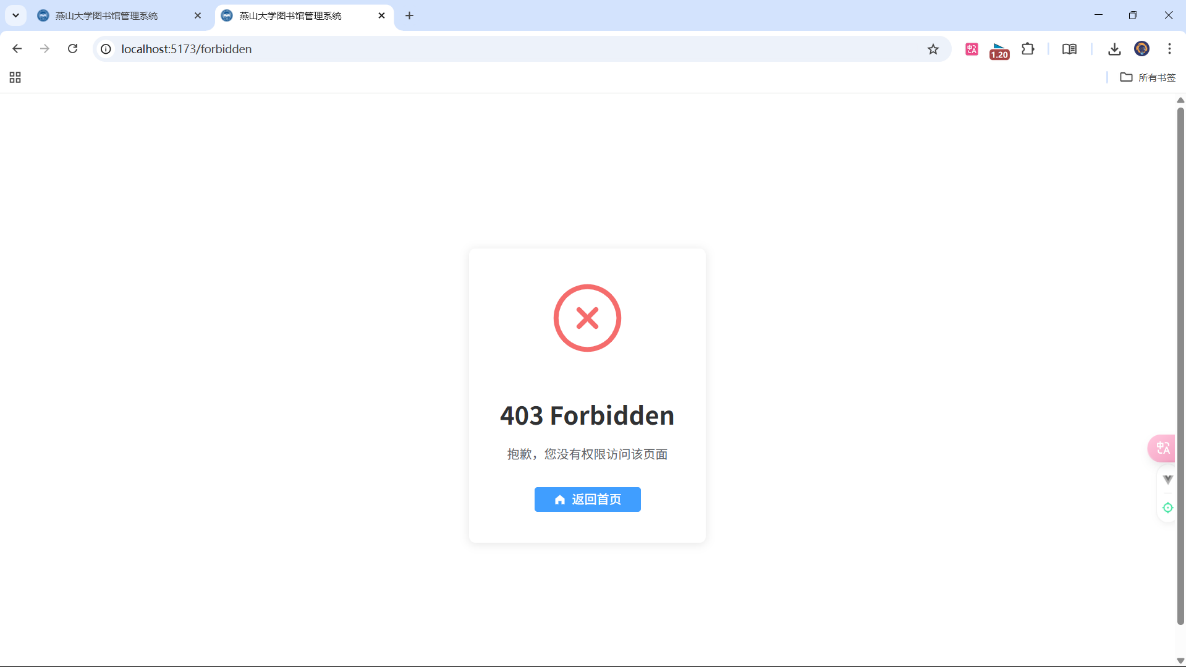
图表 22 借阅历史页面

用户管理页面如图二十三所示，管理员可以查看所有用户的数据，并可以对用户进行管理，如冻结、解冻和注销，被冻结的用户无法进行图书的借阅，注销用户则会把账号密码删除。

图表 23 用户管理页面

数据统计页面如图二十四所示，图书馆相关数据会以可视化图表的形式展现出来，方便管理员对数据进行查看和管理，更加直观清晰地展示数据。

图表 24 数据统计

404 Forbidden页面如图二十五所示，当用户或管理员访问到对方的页面时，路由会监听到，并启动路由守卫阻止该行为，确保了用户和管理员的权限隔离，保证了系统的安全性。

图表 25 404页面

## 4项目代码

**前端部分代码**

**LoginView.vue:**

<template>

  <div class="login-view">

    <el-container>

      <el-header class="header">

        <WebHeader />

      </el-header>

      <el-main class="main">

        <div class="log-wrapper">

          <div class="log-container">

            <div class="button-group">

              <el-button

                type="primary"

                size="large"

                @click="showLoginBox"

                style="flex: 1">

                登录

              </el-button>

              <el-button

                type="success"

                size="large"

                @click="showRegistrationBox"

                style="flex: 1">

                注册

              </el-button>

            </div>

            <component

              :is="currentComponent as ComponentType"

              @registration-success="showLoginBox"

            />

          </div>

        </div>

      </el-main>

      <el-footer class="footer">

        <WebFooter />

      </el-footer>

    </el-container>

  </div>

</template>

<script setup lang="ts">

import { ref } from 'vue'

import LoginBox from '../components/LoginBox.vue'

import RegistrationBox from '../components/RegistrationBox.vue'

import WebHeader from '../components/WebHeader.vue'

import WebFooter from '../components/WebFooter.vue'

type ComponentType = typeof LoginBox | typeof RegistrationBox

const currentComponent = ref<ComponentType>(LoginBox)

const showLoginBox = () => {

  currentComponent.value = LoginBox

}

const showRegistrationBox = () => {

  currentComponent.value = RegistrationBox

}

</script>

<style scoped>

.login-view {

  min-height: 100vh;

  display: flex;

  flex-direction: column;

}

.header {

  height: 60px;

  background-color: rgb(125, 161, 205);

  padding: 0;

}

.main {

  background-image: url('../assets/login-background.png');

  background-size: cover;

  background-position: center;

  min-height: 60vh;

  padding: 20px;

  flex: 1;

}

.log-wrapper {

  display: flex;

  justify-content: flex-end;

  height: 100%;

}

.log-container {

  background-color: rgba(255, 255, 255, 0.65);

  padding: 24px;

  border-radius: 8px;

  box-shadow: 0 4px 16px rgba(0,0,0,0.1);

  width: 350px;

  margin-right: 5%;

  margin-top: 5%;

}

.button-group {

  display: flex;

  gap: 12px;

  margin-bottom: 24px;

}

.footer {

  background-color: #333;

  color: white;

  padding: 0;

  margin-top: auto;

}

</style>

UserHomeView.vue:

<template>

  <div class="common-layout">

    <el-container direction="vertical">

      <el-header class="header">

        <div class="header-title">

          <img src="../assets/header.png" alt="logo" />

          <el-text class="header-text">欢迎使用图书管理系统</el-text>

          <el-space class="header-text">

              <el-icon :size = 20><SwitchButton /></el-icon>

              <el-text class="header-text"  @click = quit() style="font-size: 16px;">登出</el-text>

            </el-space>

        </div>

      </el-header>

      <el-container class="main-container">

        <el-aside width="160px" class="aside">

          <UserFoldingPanel @menu-change="handleMenuChange" />

        </el-aside>

        <el-main>

          <component :is="currentComponent" />

        </el-main>

      </el-container>

    </el-container>

  </div>

</template>

<script setup lang="ts">

import { ref, shallowRef, defineAsyncComponent } from 'vue'

import UserFoldingPanel from '../components/usercomponents/UserFoldingPanel.vue'

import { useRouter } from 'vue-router'

import {

  SwitchButton

} from '@element-plus/icons-vue'

const router = useRouter()

*// 异步加载组件*

const UserInfo = defineAsyncComponent(() => import('@/components/usercomponents/UserInfo.vue'))

const PasswordUpdate= defineAsyncComponent(() => import('@/components/usercomponents/PasswordUpdate.vue'))

const BookSearch = defineAsyncComponent(() => import('@/components/usercomponents/BookSearch.vue'))

const BorrowInfo = defineAsyncComponent(() => import('@/components/usercomponents/BorrowInfo.vue'))

const BorrowHistory = defineAsyncComponent(() => import('@/components/usercomponents/BorrowHistory.vue'))

*// 组件映射表*

const componentMap: Record<string, any> = {

  '1-1': UserInfo,        *// 个人信息*

  '1-2': PasswordUpdate,  *// 修改密码*

  '2': BookSearch,        *// 图书查询*

  '3': BorrowInfo,        *// 借阅信息*

  '4': BorrowHistory      *// 借阅历史*

}

*// 当前显示的组件（默认显示图书查询）*

const currentComponent = shallowRef(BookSearch)

const handleMenuChange = (menuKey: string) => {

  const component = componentMap[menuKey]

  if (component) {

    currentComponent.value = component

  }

}

const quit  = () => {

  router.push('/')

}

</script>

<style scoped>

.header {

  height: 60px;

  background-color: rgb(125, 161, 205);

  padding: 0;

}

.header-title {

  display: flex;

  align-items: center;

  height: 100%;

  padding: 0 100px;

  justify-content: space-between;

}

.header-text {

  font-family: "Microsoft Yahei", sans-serif;

  font-weight: 500;

  font-size: 2em;

  text-shadow: 0 0 25px #a2e2ef;

  color: #f3f3f3;

  text-align: center;

}

.aside {

  height: calc(100vh - 60px);

}

.main-container {

  flex: 1;

  overflow: hidden;

}

.common-layout,

.el-container {

  height: 100vh;

}

*/\* 主内容区域样式优化 \*/*

.el-main {

  padding: 20px;

  height: calc(100vh - 60px);

  overflow-y: auto;

  background-color: #f5f7fa;

}

.fade-enter-active,

.fade-leave-active {

  transition: opacity 0.3s ease;

}

.fade-enter-from,

.fade-leave-to {

  opacity: 0;

}

</style>

AdminHomeView.vue:

<template>

    <div class="common-layout">

      <el-container direction="vertical">

        <el-header class="header">

          <div class="header-title">

            <img src="../assets/header.png" alt="logo" />

            <el-text class="header-text">欢迎使用图书管理系统</el-text>

            <el-space class="header-text">

              <el-icon :size = 20><SwitchButton /></el-icon>

              <el-text class="header-text"  @click = quit() style="font-size: 16px;">登出</el-text>

            </el-space>

          </div>

        </el-header>

        <el-container class="main-container">

          <el-aside width="160px" class="aside">

            <AdminFoldingPanel @menu-change="handleMenuChange" />

          </el-aside>

          <el-main>

            <component :is="currentComponent" />

          </el-main>

        </el-container>

      </el-container>

    </div>

  </template>

  <script setup lang="ts">

  import { ref, shallowRef, defineAsyncComponent } from 'vue'

  import AdminFoldingPanel from '../components/admincomponents/AdminFoldingPanel.vue'

  import { useRouter } from 'vue-router'

  import {

  SwitchButton

} from '@element-plus/icons-vue'

*// 异步加载组件*

  const router = useRouter()

  const AdminInfo = defineAsyncComponent(() => import('@/components/admincomponents/AdminInfo.vue'))

  const PasswordUpdate= defineAsyncComponent(() => import('@/components/admincomponents/PasswordUpdate.vue'))

  const BookManagement = defineAsyncComponent(() => import('@/components/admincomponents/BookManagement.vue'))

  const BorrowManagement = defineAsyncComponent(() => import('@/components/admincomponents/BorrowManagement.vue'))

  const BorrowHistory = defineAsyncComponent(() => import('@/components/admincomponents/BorrowHistory.vue'))

  const UserManagement = defineAsyncComponent(() => import('@/components/admincomponents/UserManagement.vue'))

  const DataStatistics = defineAsyncComponent(() => import('@/components/admincomponents/DataStatistics.vue'))

*// 组件映射表*

  const componentMap: Record<string, any> = {

    '1-1': AdminInfo,       *// 个人信息*

    '1-2': PasswordUpdate,  *// 修改密码*

    '2': BookManagement,    *// 图书管理*

    '3': BorrowManagement,  *// 借阅管理*

    '4': BorrowHistory,      *// 借阅历史*

    '5': UserManagement,   *// 用户管理*

    '6': DataStatistics   *// 数据统计*

  }

*// 当前显示的组件（默认显示图书查询）*

  const currentComponent = shallowRef(BookManagement)

  const handleMenuChange = (menuKey: string) => {

    const component = componentMap[menuKey]

    if (component) {

      currentComponent.value = component

    }

  }

  const quit = () => {

    router.push('/')

  }

  </script>

  <style scoped>

  .header {

    height: 60px;

    background-color: rgb(125, 161, 205);

    padding: 0;

  }

  .header-title {

    display: flex;

    align-items: center;

    height: 100%;

    padding: 0 100px;

    justify-content: space-between;

  }

  .header-text {

    font-family: "Microsoft Yahei", sans-serif;

    font-weight: 500;

    font-size: 2em;

    text-shadow: 0 0 25px #a2e2ef;

    color: #f3f3f3;

    text-align: center;

  }

  .aside {

    height: calc(100vh - 60px);

  }

  .main-container {

    flex: 1;

    overflow: hidden;

  }

  .common-layout,

  .el-container {

    height: 100vh;

  }

  .el-main {

    padding: 20px;

    height: calc(100vh - 60px);

    overflow-y: auto;

    background-color: #f5f7fa;

  }

  .fade-enter-active,

  .fade-leave-active {

    transition: opacity 0.3s ease;

  }

  .fade-enter-from,

  .fade-leave-to {

    opacity: 0;

  }

  </style>

BookSearch.vue:

<template>

  <div class="book-management">

*<!-- 搜索区域 -->*

    <div class="search-box">

      <el-input v-model="searchParams.bookId" placeholder="图书编号" style="width: 200px" />

      <el-input v-model="searchParams.title" placeholder="图书名称" style="width: 200px; margin-left: 10px" />

      <el-input v-model="searchParams.author" placeholder="作者姓名" style="width: 200px; margin-left: 10px" />

      <el-button type="primary" @click="handleSearch" style="margin-left: 10px">搜索</el-button>

    </div>

*<!-- 操作按钮 -->*

    <div class="action-buttons" style="margin: 20px 0">

      <el-button type="primary" @click="handleBatchBorrow">批量借阅</el-button>

      <el-button @click="handleCancelSelect">取消选择</el-button>

    </div>

*<!-- 表格 -->*

    <el-table

      ref="bookTableRef"

      :data="pagedData"

      style="width: 100%"

      @selection-change="handleSelectionChange"

      row-key="bookId"

    >

      <el-table-column type="selection" width="55" :reserve-selection="true" />

      <el-table-column prop="bookId" label="图书编号" width="120" />

      <el-table-column prop="title" label="书名" width="180" />

      <el-table-column prop="author" label="作者" width="120" />

      <el-table-column prop="publisher" label="出版社" width="180" />

      <el-table-column prop="publishDate" label="出版时间" width="180">

        <template #default="{ row }">

          <el-date-picker

            v-model="row.publishDate"

            type="date"

            value-format="YYYY-MM-DD"

            placeholder="选择日期"

            style="width: 100%"

            readonly

            :disabled="true"

          />

        </template>

      </el-table-column>

      <el-table-column prop="isBorrowed" label="借阅状态" width="100">

        <template #default="{ row }">

          <el-tag :type="row.isBorrowed ? 'danger' : 'success'">

            {{ row.isBorrowed ? '已借出' : '可借阅' }}

          </el-tag>

        </template>

      </el-table-column>

    </el-table>

*<!-- 分页 -->*

    <el-pagination

      v-model:current-page="currentPage"

      v-model:page-size="pageSize"

      :total="filteredData.length"

      :page-sizes="[5, 10, 20]"

      layout="total, sizes, prev, pager, next, jumper"

      style="margin-top: 20px"

      @current-change="handlePageChange"

    />

*<!-- 借阅对话框 -->*

    <el-dialog

      v-model="dialogVisible"

      title="借阅信息"

      width="500px"

      @closed="handleDialogClose"

    >

      <el-form

        ref="formRef"

        :model="formData"

        :rules="formRules"

        label-width="100px"

      >

        <el-form-item label="用户ID" prop="userId">

          <el-input v-model="formData.userId" disabled />

        </el-form-item>

        <el-form-item label="借阅时间" prop="loanDate">

          <el-input :value="formData.loanDate" disabled />

        </el-form-item>

        <el-form-item label="归还时间" prop="dueDate">

          <el-date-picker

            v-model="formData.dueDate"

            type="date"

            placeholder="选择归还日期"

            :disabled-date="disabledDueDate"

            value-format="YYYY-MM-DD"

            style="width: 100%"

          />

        </el-form-item>

      </el-form>

      <template #footer>

        <el-button @click="dialogVisible = false">取消</el-button>

        <el-button type="primary" @click="handleConfirmBorrow">确定</el-button>

      </template>

    </el-dialog>

  </div>

</template>

<script setup>

import { ref, reactive, computed, nextTick } from 'vue'

import { ElMessage, ElMessageBox } from 'element-plus'

import dayjs from 'dayjs'

import { borrowBooks, getBookList } from '@/api/book'

import { createBookLoan } from '@/api/bookLoan'

import { onMounted } from 'vue'

import { useRouter } from 'vue-router'

const router = useRouter()

const bookTableRef = ref()

const formRef = ref()

const selectedIds = ref(new Set())

const mockData = ref([])

const dialogVisible = ref(false)

*// 表单数据*

const formData = reactive({

  userId: localStorage.getItem('userId') || '',

  loanDate: dayjs().format('YYYY-MM-DD'),

  dueDate: ''

})

*// 表单验证规则*

const formRules = reactive({

  userId: [{ required: true, message: '获取用户ID失败', trigger: 'blur' }],

  dueDate: [{ required: true, message: '请选择归还日期', trigger: 'change' }]

})

*// 搜索参数*

const searchParams = reactive({

  bookId: '',

  title: '',

  author: ''

})

*// 分页参数*

const currentPage = ref(1)

const pageSize = ref(10)

*// 计算属性*

const filteredData = computed(() => {

  return mockData.value.filter(book => {

    return (

      book.bookId.includes(searchParams.bookId) &&

      book.title.toLowerCase().includes(searchParams.title.toLowerCase()) &&

      book.author.toLowerCase().includes(searchParams.author.toLowerCase())

    )

  })

})

const pagedData = computed(() => {

  const start = (currentPage.value - 1) \* pageSize.value

  return filteredData.value.slice(start, start + pageSize.value)

})

*// 方法*

const handleSearch = () => {

  currentPage.value = 1

  selectedIds.value.clear()

}

const handleSelectionChange = (selection) => {

  const currentPageIds = new Set(pagedData.value.map(item => item.bookId))

  const newSelected = new Set(selection.map(item => item.bookId))

  currentPageIds.forEach(bookId => {

    newSelected.has(bookId) ? selectedIds.value.add(bookId) : selectedIds.value.delete(bookId)

  })

}

const handleBatchBorrow = () => {

  if (selectedIds.value.size === 0) {

    ElMessage.warning('请选择要借阅的图书')

    return

  }

  dialogVisible.value = true

}

const handleConfirmBorrow = async () => {

  try {

*// 验证表单*

    await formRef.value.validate()

*// 构造请求参数*

    const loans = Array.from(selectedIds.value).map(bookId => ({

      userId: formData.userId,

      bookId: bookId,

      loanDate: formData.loanDate,

      dueDate: formData.dueDate

    }))

*// 调用API*

    const bookIds = Array.from(selectedIds.value)

    const response = await createBookLoan(loans)

    if(response.code === 200) {

      try{

        const response1 = await borrowBooks(bookIds)

        if (response1.code === 200) {

*// 更新本地状态*

          mockData.value = mockData.value.map(book => {

            if (selectedIds.value.has(book.bookId)) {

              return { ...book, isBorrowed: true }

            }

            return book

          })

          ElMessage.success('借阅成功')

          dialogVisible.value = false

          handleCancelSelect()

        } else {

          ElMessage.error(response1.message)

        }

      }catch (error) {

        console.error('借阅失败:', error)

      }} else {

        ElMessage.error(response.message)

      }

    } catch (error) {

    console.error('借阅失败:', error)

  }

}

const disabledDueDate = (date) => {

  return dayjs(date).isBefore(dayjs(), 'day')

}

const handleDialogClose = () => {

  formRef.value.resetFields()

  formData.loanDate = dayjs().format('YYYY-MM-DD')

}

const handleCancelSelect = () => {

  selectedIds.value.clear()

  bookTableRef.value?.clearSelection()

}

const handlePageChange = () => {

  nextTick(() => {

    if (!bookTableRef.value) return

      pagedData.value.forEach(row => {

      bookTableRef.value.toggleRowSelection(row, selectedIds.value.has(row.bookId))

    })

  })

}

*// 初始化数据*

onMounted(async () => {

  try {

    if (!localStorage.getItem('userId')) {

        ElMessage.error('用户ID未找到,请重新登录')

        router.push('/')

        return

      }

    const response = await getBookList()

    if (response.code === 200) {

      mockData.value = response.data

    } else {

      ElMessage.error(response.message)

    }

  } catch (error) {

    ElMessage.error('获取图书列表失败')

  }

})

</script>

<style scoped>

.book-management {

  padding: 20px;

}

.search-box {

  margin-bottom: 20px;

}

</style>

**后端部分代码：**

**BookController.jave:**

**package org.example.bookmanagement\_springboot.controller;**

**import jakarta.validation.Valid;**

**import jakarta.validation.constraints.NotBlank;**

**import lombok.RequiredArgsConstructor;**

**import org.example.bookmanagement\_springboot.common.ApiResponse;**

**import org.example.bookmanagement\_springboot.dto.BookDTO;**

**import org.example.bookmanagement\_springboot.entity.Book;**

**import org.example.bookmanagement\_springboot.entity.People;**

**import org.example.bookmanagement\_springboot.enums.ResultEnum;**

**import org.example.bookmanagement\_springboot.exception.BusinessException;**

**import org.example.bookmanagement\_springboot.service.BookService;**

**import org.springframework.beans.factory.annotation.Autowired;**

**import org.springframework.beans.factory.config.BeanFactoryPostProcessor;**

**import org.springframework.validation.annotation.Validated;**

**import org.springframework.web.bind.annotation.\*;**

**import java.util.List;**

**@CrossOrigin**

**@RequiredArgsConstructor**

**@RestController**

**@RequestMapping("/book")**

**public class BookController {**

**private final BookService bookService;**

**private final BeanFactoryPostProcessor forceAutoProxyCreatorToUseClassProxying;**

**@PostMapping("/add")**

**public ApiResponse<?> addBook(@Valid @RequestBody BookDTO bookDTO) {**

**try{**

**Book book = bookService.add(bookDTO);**

**return ApiResponse.success(book);**

**}catch (BusinessException e){**

**return ApiResponse.error(e.getCode());**

**}**

**}**

**@DeleteMapping("/delete/{BookId}")**

**public ApiResponse<?> deleteBook(@Valid @PathVariable String BookId) {**

**try {**

**if(bookService.deleteById(BookId)){**

**return ApiResponse.success(ResultEnum.SUCCESS);**

**}else{**

**return ApiResponse.error(ResultEnum.BOOK\_NOT\_FOUND);**

**}**

**} catch (BusinessException e) {**

**return ApiResponse.error(e.getCode());**

**}**

**}**

**@DeleteMapping("/deleteBooks")**

**public ApiResponse<?> deleteBooks(@RequestBody List<String> BookIds) {**

**try {**

**if(bookService.deleteBooks(BookIds)){**

**return ApiResponse.success(ResultEnum.SUCCESS);**

**}else{**

**return ApiResponse.error(ResultEnum.BOOK\_DELETE\_ERROR);**

**}**

**} catch (BusinessException e) {**

**return ApiResponse.error(e.getCode());**

**}**

**}**

**@PutMapping("/update")**

**public ApiResponse<?> updateBook(@Valid @RequestBody BookDTO bookDTO) {**

**try {**

**Book book = bookService.update(bookDTO);**

**return ApiResponse.success(book);**

**} catch (BusinessException e) {**

**return ApiResponse.error(e.getCode());**

**}**

**}**

**@PutMapping("/borrowBooks")**

**public ApiResponse<?> borrowBook( @RequestBody List<String> bookIds) {**

**try {**

**List<Book> books = bookService.borrowBooks(bookIds);**

**return ApiResponse.success(books);**

**} catch (BusinessException e) {**

**return ApiResponse.error(e.getCode());**

**}**

**}**

**@PutMapping("/returnBooks")**

**public ApiResponse<?> returnBook(@RequestBody List<String> bookIds) {**

**try {**

**List<Book> books = bookService.returnBooks(bookIds);**

**return ApiResponse.success(books);**

**} catch (BusinessException e) {**

**return ApiResponse.error(e.getCode());**

**}**

**}**

**@GetMapping("/getAll")**

**public ApiResponse<?> getBookList() {**

**try {**

**List<Book> list = bookService.getAllBooks();**

**return ApiResponse.success(list);**

**} catch (BusinessException e) {**

**return ApiResponse.error(e.getCode());**

**}**

**}**

**@GetMapping("/getBook/{BookId}")**

**public ApiResponse<?> getBook(@PathVariable String BookId) {**

**try {**

**Book book = bookService.getBookById(BookId);**

**return ApiResponse.success(book);**

**} catch (BusinessException e) {**

**return ApiResponse.error(e.getCode());**

**}**

**}**

**}**

**BookService.java:**

**package org.example.bookmanagement\_springboot.service;  
  
import com.baomidou.mybatisplus.extension.service.IService;  
import org.example.bookmanagement\_springboot.dto.BookDTO;  
import org.example.bookmanagement\_springboot.entity.Book;  
  
import java.util.List;  
  
*/\*\*  
\* @author DELL  
\* @description 针对表【book】的数据库操作Service  
\* @createDate 2025-03-27 22:04:17  
\*/*public interface BookService extends IService<Book> {  
 Book add(BookDTO bookDTo);  
  
 Boolean deleteById(String bookId);  
  
 Boolean deleteBooks(List<String> bookIds);  
  
 Book update(BookDTO bookDTo);  
  
 List<Book> borrowBooks(List<String> bookIds);  
  
 List<Book> returnBooks(List<String> bookIds);  
  
 List<Book> getAllBooks();  
  
 Book getBookById(String bookId);  
  
  
}**

**BookServiceImpl.java:**

**package org.example.bookmanagement\_springboot.service.impl;**

**import com.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;**

**import com.baomidou.mybatisplus.extension.conditions.query.LambdaQueryChainWrapper;**

**import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;**

**import lombok.RequiredArgsConstructor;**

**import lombok.extern.slf4j.Slf4j;**

**import org.example.bookmanagement\_springboot.dto.BookDTO;**

**import org.example.bookmanagement\_springboot.entity.Book;**

**import org.example.bookmanagement\_springboot.enums.ResultEnum;**

**import org.example.bookmanagement\_springboot.exception.BusinessException;**

**import org.example.bookmanagement\_springboot.mapper.BookMapper;**

**import org.example.bookmanagement\_springboot.service.BookService;**

**import org.springframework.stereotype.Service;**

**import org.springframework.transaction.annotation.Transactional;**

**import java.util.List;**

***/\*\****

**\* @author DELL**

**\* @description 针对表【book】的数据库操作Service实现**

**\* @createDate 2025-03-27 22:04:17**

**\*/**

**@Service**

**@Transactional(rollbackFor = Exception.class)**

**@RequiredArgsConstructor**

**@Slf4j**

**public class BookServiceImpl extends ServiceImpl<BookMapper, Book> implements BookService {**

**private final BookMapper bookMapper;**

**public Book add(BookDTO bookDTO){**

***// 1. 检查图书ID是否存在***

**if (lambdaQuery()**

**.eq(Book::getBookId, bookDTO.getBookId())**

**.exists()) {**

**log.warn("图书已存在，图书ID: {}", bookDTO.getBookId());**

**throw new BusinessException(ResultEnum.BOOK\_ID\_CONFLICT);**

**}**

**try {**

***// 2. 创建新图书***

**Book book = new Book()**

**.setBookId(bookDTO.getBookId())**

**.setTitle(bookDTO.getTitle())**

**.setAuthor(bookDTO.getAuthor())**

**.setPublisher(bookDTO.getPublisher())**

**.setPublishDate(bookDTO.getPublishDate());**

***// 3. 保存图书***

**if (!save(book)) {**

**log.error("图书保存失败，图书信息: {}", bookDTO);**

**throw new BusinessException(ResultEnum.BOOK\_ADD\_ERROR);**

**}**

**return book;**

**} catch (BusinessException e) {**

**throw e; *// 已知业务异常直接抛出***

**} catch (Exception e) {**

**log.error("添加图书时发生系统异常", e);**

**throw new BusinessException(ResultEnum.SYSTEM\_ERROR, "图书添加失败");**

**}**

**}**

**@Override**

**public Boolean deleteById(String bookId){**

**try {**

**Book book = lambdaQuery().eq(Book::getBookId, bookId).one();**

**if (book != null) {**

**bookMapper.deleteById(book.getBookId());**

**return true;**

**}else{**

**return false;**

**}**

**} catch (Exception e) {**

**log.error("删除图书时发生系统异常", e);**

**throw new BusinessException(ResultEnum.BOOK\_DELETE\_ERROR, "图书删除失败");**

**}**

**}**

**@Override**

**public Boolean deleteBooks(List<String> bookIds){**

**try {**

**for(String bookId : bookIds){**

**Book book = lambdaQuery().eq(Book::getBookId, bookId).one();**

**if (book != null ) {**

**if(book.getIsBorrowed()){**

**throw new BusinessException(ResultEnum.BOOK\_IS\_BORROWED, "图书已被借阅，无法删除");**

**}**

**}else{**

**throw new BusinessException(ResultEnum.BOOK\_NOT\_FOUND, "图书不存在");**

**}**

**}**

**int deletedCount = bookMapper.deleteByIds(bookIds);**

**return deletedCount>0;**

**} catch (BusinessException e){**

**throw e;**

**}catch (Exception e) {**

**log.error("删除图书时发生系统异常", e);**

**throw new BusinessException(ResultEnum.BOOK\_DELETE\_ERROR, "图书删除失败");**

**}**

**}**

**@Override**

**public Book update(BookDTO bookDTO){**

**try{**

**Book book = lambdaQuery().eq(Book::getBookId, bookDTO.getBookId()).one();**

**if (book != null) {**

***// 2. 更新图书信息***

**book.setBookId(bookDTO.getBookId())**

**.setTitle(bookDTO.getTitle())**

**.setPublisher(bookDTO.getPublisher())**

**.setPublishDate(bookDTO.getPublishDate())**

**.setIsBorrowed(bookDTO.getIsBorrowed());**

***// 3. 保存图书***

**bookMapper.updateById(book);**

**return book;**

**}else{**

**throw new BusinessException(ResultEnum.BOOK\_NOT\_FOUND, "图书不存在");**

**}**

**} catch (BusinessException e) {**

**throw e; *// 已知业务异常直接抛出***

**} catch (Exception e) {**

**log.error("更新图书时发生系统异常", e);**

**throw new BusinessException(ResultEnum.SYSTEM\_ERROR, "图书更新失败");**

**}**

**}**

**@Override**

**public List<Book> borrowBooks(List<String> bookIds) {**

**try {**

***// 1. 查询所有要借阅的图书***

**List<Book> books = bookMapper.selectByIds(bookIds);**

***// 2. 检查图书是否存在并更新状态***

**for (Book book : books) {**

**if (book != null) {**

**book.setIsBorrowed(true); *// 将图书状态改为已借阅***

**} else {**

**throw new BusinessException(ResultEnum.BOOK\_NOT\_FOUND);**

**}**

**}**

***// 3. 批量更新图书状态***

**boolean updated = this.updateBatchById(books);**

**if (!updated) {**

**log.error("更新图书借阅状态失败，图书信息: {}", books);**

**throw new BusinessException(ResultEnum.BOOK\_LOAN\_ADD\_ERROR, "图书借阅失败");**

**}**

**return books; *// 返回更新后的图书列表***

**} catch (Exception e) {**

**log.error("借阅图书时发生系统异常", e);**

**throw new BusinessException(ResultEnum.BOOK\_LOAN\_ADD\_ERROR, "图书借阅失败");**

**}**

**}**

**public List<Book> returnBooks(List<String> bookIds){**

**try {**

***// 1. 查询所有要借阅的图书***

**List<Book> books = bookMapper.selectByIds(bookIds);**

***// 2. 检查图书是否存在并更新状态***

**for (Book book : books) {**

**if (book != null) {**

**book.setIsBorrowed(false);**

**} else {**

**throw new BusinessException(ResultEnum.BOOK\_NOT\_FOUND);**

**}**

**}**

***// 3. 批量更新图书状态***

**boolean updated = this.updateBatchById(books);**

**if (!updated) {**

**log.error("更新图书借阅状态失败，图书信息: {}", books);**

**throw new BusinessException(ResultEnum.BOOK\_LOAN\_DELETE\_ERROR, "图书归还失败");**

**}**

**return books; *// 返回更新后的图书列表***

**} catch (Exception e) {**

**log.error("借阅图书时发生系统异常", e);**

**throw new BusinessException(ResultEnum.BOOK\_LOAN\_ADD\_ERROR, "图书借阅失败");**

**}**

**}**

**@Override**

**public List<Book> getAllBooks(){**

**try{**

**return bookMapper.selectList(null);**

**}catch (Exception e) {**

**log.error("获取图书列表时发生系统异常", e);**

**throw new BusinessException(ResultEnum.SYSTEM\_ERROR, "获取图书列表失败");**

**}**

**}**

**@Override**

**public Book getBookById(String bookId){**

**try {**

**Book book = bookMapper.selectById(bookId);**

**if (book == null) {**

**throw new BusinessException(ResultEnum.BOOK\_NOT\_FOUND, "图书不存在");**

**}**

**return book;**

**} catch (Exception e) {**

**log.error("获取图书时发生系统异常", e);**

**throw new BusinessException(ResultEnum.SYSTEM\_ERROR, "获取图书失败");**

**}**

**}**

**}**

由于篇幅有限，文档无法展示全部代码，完整代码已上传到Github上<https://github.com/n1425362023/Undergraduate-Program/tree/main/JavaWeb>

## 5心得体会

在开发图书馆管理系统的过程中，我积累了许多宝贵的经验和心得体会。以下是我主要的几点体会：

## **需求分析的重要性**

在项目初期，进行详细的需求分析至关重要。明确项目的需求，帮助我更好地设计系统功能，如借书、还书、查询和管理图书等。

## **技术选型**

选择合适的技术栈是项目成功的关键。我使用了Java作为后端语言，结合Spring Boot框架进行开发，使用MySQL作为数据库。这样不仅提高了开发效率，也使得系统更加稳定和可维护。

## **模块化设计**

将系统分为多个模块（如用户管理、图书管理、借阅管理等），使得代码结构更加清晰，便于后期维护。每个模块独立开发，减少了相互之间的耦合。

## **用户体验**

在界面设计上，我注重用户体验，力求简洁明了。通过合理的布局和交互设计，使得用户能够快速上手，减少学习成本。

## **测试与反馈**

在开发过程中，持续进行单元测试和集成测试，及时发现并修复bug。同时，邀请用户进行体验，收集反馈并进行迭代改进，使系统更符合实际需求。

## **8. 持续学习**

在项目中遇到许多技术难题，通过查阅资料和请教老师、同学，我意识到持续学习的重要性。技术更新迅速，保持学习的态度能帮助我跟上行业的步伐。

## **总结**

通过这个项目，我提升了自己的编程能力和项目管理能力，这些经验对我未来的学习和工作都将产生积极的影响