

《数据库原理与技术》

课程设计报告

项目名称：在线小说阅读系统

院 系：航空航天学院自动化系

专 业：自动化

指导教师：洪文兴

组 员：王帅 张成成 谢岑 周贯中

2021年 月 日

1. **选题情况**

系统选题情况以及任务分工情况说明（团队合作需细化每个队员负责的具体功能模块，工作量占比）

本系统是为喜爱小说阅读的受众提供的具有便捷、易于阅读、内容涵盖广等有点的在线小说阅读系统。

团队成员为王帅、张成成、周贯中、谢岑。

具体分工：

①王帅：负责数据库整体设计，系统运行环境配置、前端与后台数据库连接。

②张成成：参与数据库设计，负责概念模型设计，数据库逻辑及物理结构设计。

③周贯中：参与数据库设计，负责前端编写，网站页面设计。

④谢岑：参与数据库设计，负责项目报告编写整理，系统演示。

1. **需求分析**

1.背景

读者阅读小说往往需要在线下花费大量时间查找自身所需要的作品集，不易操作且一般效率较低。本系统致力于提升读者搜寻小说的效率，更加便捷的找到需要的书籍。

2.系统功能

本网站系统致力于为广大读者提供各类型的小说作品，在提供各类小说的同时，允许注册，允许小说创作者提交自己的原创或改编小说内容，或者更新自己的作品。

本网站的阅读区分为玄幻、仙侠等分区，读者也可以通过订阅将喜欢的作品加入自己的收藏以及通过消耗积分购买相关付费章节。

3.系统用例图

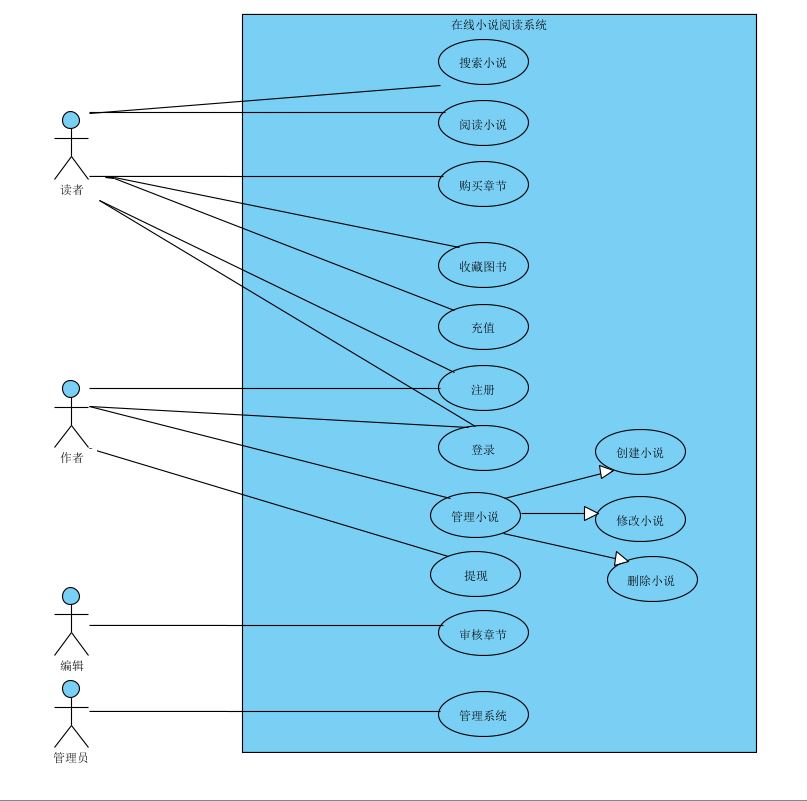


图 1小说系统用例

3.1读者用例分析

对于读者来说，其主要需求正如上述背景分析中提到的那样，主要集中于小说作品的搜索和阅读，读者可以在本网站注册登陆并搜索小说，如果读者对所阅读的作品章节感到满意则可以选择购买该章节，或者进行收藏。

3.2作者用例分析

作者可以通过在本网站注册登陆，管理自己的小说作品以及提现收益。其中，管理小说的功能包含创建新的小说作品，修改已经存在的作品以及删除作品。作者的提现收益与网站平台的收益比例为2:3。

3.3编辑用例分析

编辑的主要功能在于审核已上传的小说章节，可以根据内容选择审核通过或者不通过。编辑若审核不通过，则作者可以选择修改后再次提交。

3.4管理员用例分析

网站管理员的主要需求是在登陆后可以对该系统：录入用户、作者、编辑的基本的信息，维护和修改各类数据字典。

1. **数据库设计**

**3.1概念模型（E-R 图）设计**

基于对小说网站的需求分析以及网站系统用例图，我们提取出了NOVEL,READER,AUTHOR,EDITOR这四个实体；对于NOVEL而言CHAPTER需要被分离以消除多值依赖。 COLLECTION,INCHARGEOF分别是NOVLE和READER以及AUTHOR和EDITOR的交表，SUBSCRIBE是READER和CHAPTER的关联表。其中CHAPTER表中的Type用于表示小说是否需要付费，当Type取0时表示免费，取1时表示收费；NOVEL表中的State用来表示小说是否审核通过。

AUTHOR和EDITOR存在这样的联系：一个AUTHOR可以由多个EDITOR负责审核，但至少有一位EDITOR对其进行审核。相似地，NOVEL可以对应0个或1个或多个READER。

具体的实体联系参考以下E-R模型：

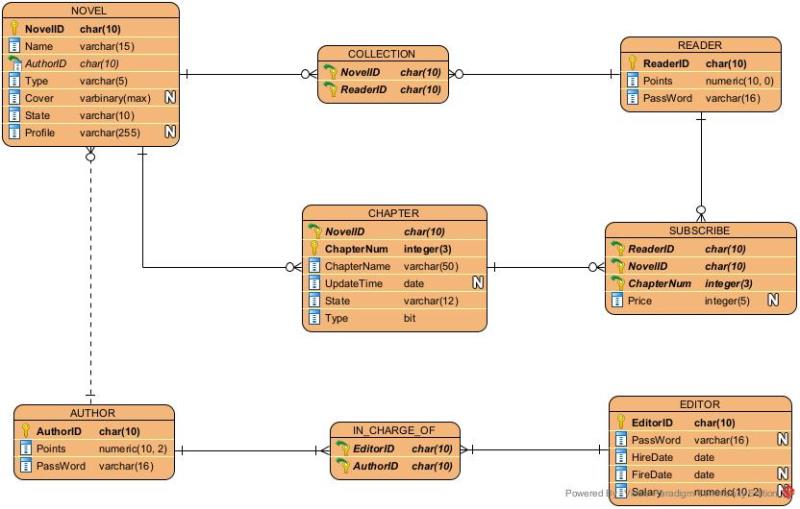


图 2小说阅读网页E-R图

**3.2数据库逻辑结构设计**

将E-R图转化为表，实现E-R概念模型到关系（数据）模型的转变，并且在基本表的基础上建立视图（view）

3.2.1 E-R模型转化为表

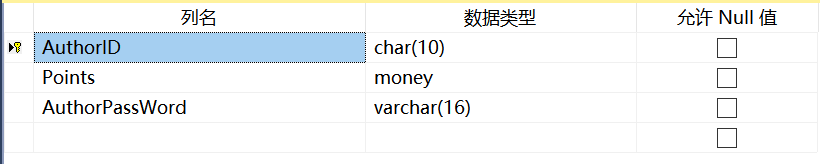


图 3 Author表

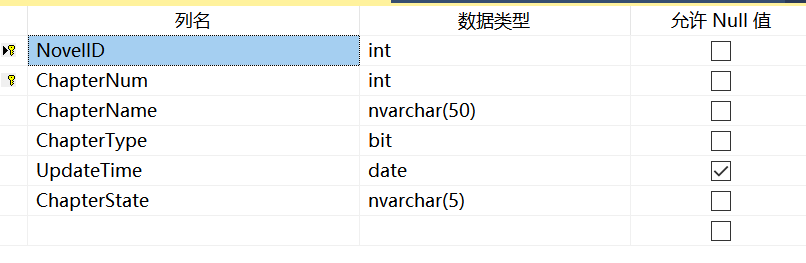


图 4 Chapter表

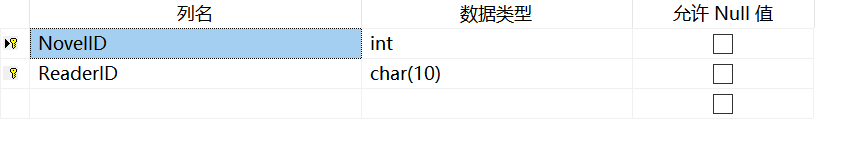


图 5 Collections表

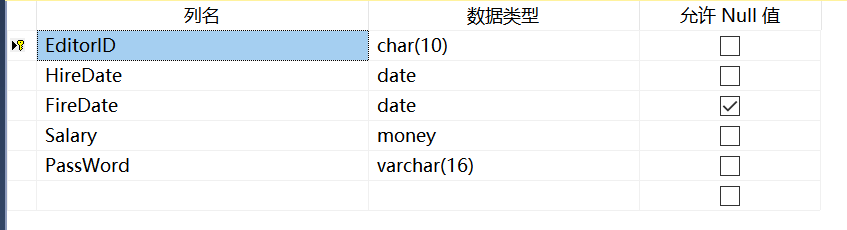


图 6 Editor表

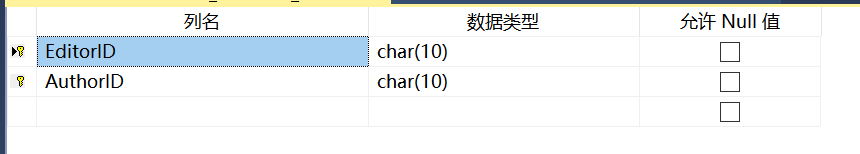


图 7 IN\_CHARGE\_OF表



图 8 Novel表

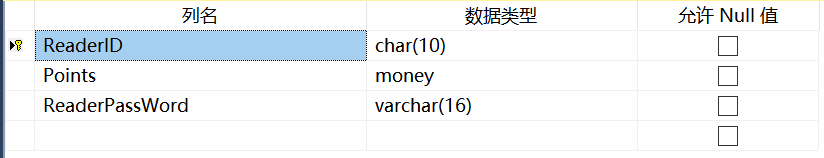


图 9 Reader表

3.2.2 建立视图

Author视图的建立：

USE NovelWebsite

GO

IF OBJECT\_ID('AuthorView','V') IS NOT NULL--V视图

DROP VIEW AuthorView

GO

CREATE VIEW AuthorView WITH SCHEMABINDING

AS

SELECT A.AuthorID,A.Points,N.NovelName,N.NovelType,N.NovelState,N.Cover,N.NovelProfile,CP.ChapterNum,CP.UpdateTime,CP.ChapterState

FROM dbo.AUTHOR A INNER JOIN dbo.Novel N ON A.AuthorID=N.AuthorID

INNER JOIN dbo.CHAPTER CP ON N.NovelID=CP.NovelID

GO

--TEST

--SELECT \* FROM AuthorView

Editor视图的建立：

USE NovelWebsite

GO

IF OBJECT\_ID('EditorView','V') IS NOT NULL--V视图

DROP VIEW EditorView

GO

CREATE VIEW EditorView WITH SCHEMABINDING

AS

SELECT E.EditorID,E.Salary,A.AuthorID,N.NovelName,CP.ChapterNum,CP.ChapterState

FROM dbo.EDITOR E INNER JOIN dbo.IN\_CHARGE\_OF ICF ON E.EditorID=ICF.EditorID

INNER JOIN dbo.AUTHOR A ON A.AuthorID=ICF.AuthorID

INNER JOIN dbo.Novel N ON A.AuthorID=N.AuthorID

INNER JOIN dbo.CHAPTER CP ON CP.NovelID=N.NovelID

GO

--TEST

--SELECT \* FROM EditorView

Novel视图的建立：

USE NovelWebsite

GO

IF OBJECT\_ID('NovelView','V') IS NOT NULL--V视图

DROP VIEW NovelView

GO

CREATE VIEW NovelView WITH SCHEMABINDING

AS

SELECT NovelName,AuthorID,NovelType,Cover,NovelState,NovelProfile,ChapterNum

FROM dbo.NOVEL INNER JOIN dbo.CHAPTER ON NOVEL.NovelID=CHAPTER.NovelID

GO

--TEST

--SELECT \* FROM NovelView

Reader视图的建立：

USE NovelWebsite

GO

IF OBJECT\_ID('ReaderView','V') IS NOT NULL--V视图

DROP VIEW ReaderView

GO

CREATE VIEW ReaderView WITH SCHEMABINDING

AS

SELECT R.ReaderID,Points,C.NovelID,ChapterNum

FROM dbo.READER R INNER JOIN dbo.COLLECTIONS C ON R.ReaderID=C.ReaderID

INNER JOIN dbo.CHAPTER CP ON C.NovelID=CP.NovelID

GROUP BY R.ReaderID,C.NovelID,CP.ChapterNum,Points

GO

--TEST

--SELECT \* FROM ReaderView

**3.3存储过程与触发器设计**

3.3.1InseterChapter触发器：

USE NovelWebsite

GO

IF OBJECT\_ID('trigger\_InsertChapter','TR') IS NOT NULL

DROP TRIGGER trigger\_InsertChapter

GO

CREATE TRIGGER trigger\_InsertChapter

ON CHAPTER AFTER INSERT

AS

UPDATE CHAPTER SET UpdateTime=GETDATE()

WHERE ChapterNum IN (SELECT ChapterNum FROM inserted)

GO

--该触发器用于在插入新的章节后更新该章节的更新时间

3.3.2 UpdateChapter触发器：

USE NovelWebsite

GO

IF OBJECT\_ID('trigger\_UpdateChapter','TR') IS NOT NULL

DROP TRIGGER trigger\_UpdateChapter

GO

CREATE TRIGGER trigger\_UpdateChapter

ON CHAPTER AFTER UPDATE

AS

UPDATE CHAPTER SET UpdateTime=GETDATE()

WHERE ChapterNum IN (SELECT ChapterNum FROM inserted)

GO

--该触发器用于在更新新的章节后更新该章节的更新时间

3.3.3 InsertSubscribe触发器：

USE NovelWebsite

GO

IF OBJECT\_ID('tigger\_InsertSubscribe','TR') IS NOT NULL

DROP TRIGGER tigger\_InsertSubscribe

GO

CREATE TRIGGER tigger\_InsertSubscribe

ON SUBSCRIBE AFTER INSERT

AS

UPDATE READER SET Points=Points-(SELECT Price FROM SUBSCRIBE)

\*(SELECT COUNT(ChapterNum)

FROM inserted

GROUP BY ReaderID

HAVING READER.ReaderID=inserted.ReaderID)

WHERE READER.Points>(SELECT Price FROM SUBSCRIBE)

\*(SELECT COUNT(ChapterNum)

FROM inserted

GROUP BY ReaderID

HAVING READER.ReaderID=inserted.ReaderID)

--该触发器用于在插入读者订阅信息后，扣除读者的积分

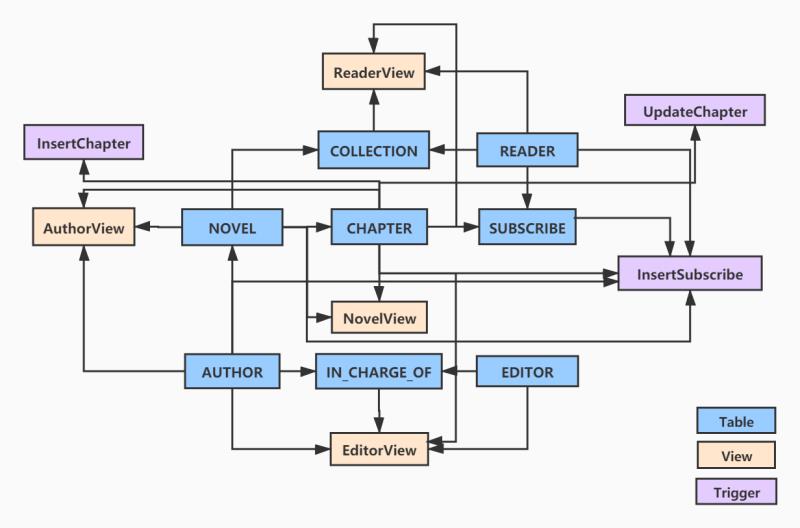
****

图 10触发器设计

1. **系统说明**

系统运行环境配置、前端与后台数据库连接用到的关键语句说明：

1. '''''
2. 数据库连接demo:
3. Sqlsever 配置为通过网络连接，python 需要满足下列依赖
4. pymssql==2.2.1
5. Flask==1.1.2
6. DBUtils==2.0.1
7. '''
8. **from** DBUtils.PersistentDB **import** PersistentDB
9. **import** pymssql
11. POOL = PersistentDB(
12. creator=pymssql, # 使用链接数据库的模块
13. maxusage=None, # 一个链接最多被重复使用的次数，None表示无限制
14. setsession=[], # 开始会话前执行的命令列表。如：["set datestyle to ...", "set time zone ..."]
15. ping=0,
16. # ping MySQL服务端，检查是否服务可用。# 如：0 = None = never, 1 = default = whenever it is requested, 2 = when a cursor is created, 4 = when a query is executed, 7 = always
17. closeable=False,
18. # 如果为False时， conn.close() 实际上被忽略，供下次使用，再线程关闭时，才会自动关闭链接。如果为True时， conn.close()则关闭链接，那么再次调用pool.connection时就会报错，因为已经真的关闭了连接（pool.steady\_connection()可以获取一个新的链接）
19. threadlocal=None, # 本线程独享值得对象，用于保存链接对象，如果链接对象被重置
20. host='127.0.0.1',
21. port=1433,
22. user='sa',
23. password='123456',
24. database='NovelWebsite',
25. charset='utf8'
26. )
27. conn = POOL.connection()
28. cursor = conn.cursor()
29. **try**:
30. cursor.execute('select \* from AUTHOR')#执行SQL语句
31. result = cursor.fetchone()#取结果或cursor.fetchall()
32. conn.commit()
33. **except** Exception as e:
34. **print**(e)
35. #异常处理
36. **finally**:
37. cursor.close()
38. conn.close()
39. **功能界面介绍**

系统主要功能展示及实现，需要截图展示

**5.1编辑主要界面展示**

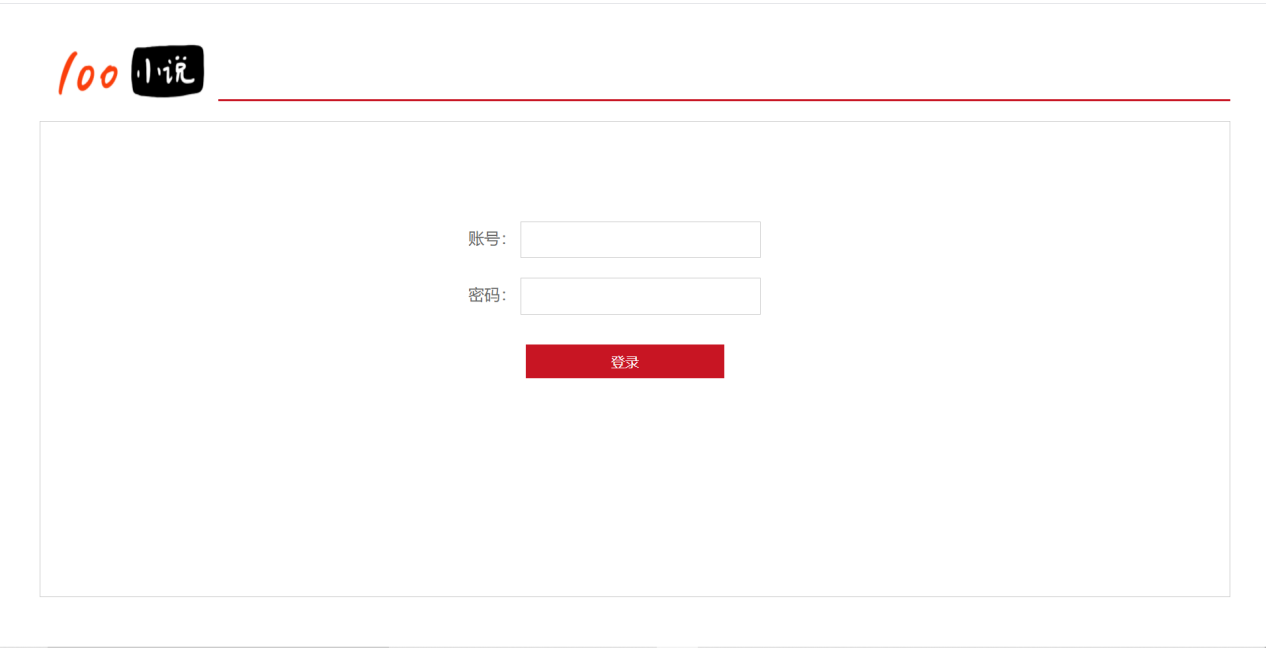


图 11编辑登陆界面

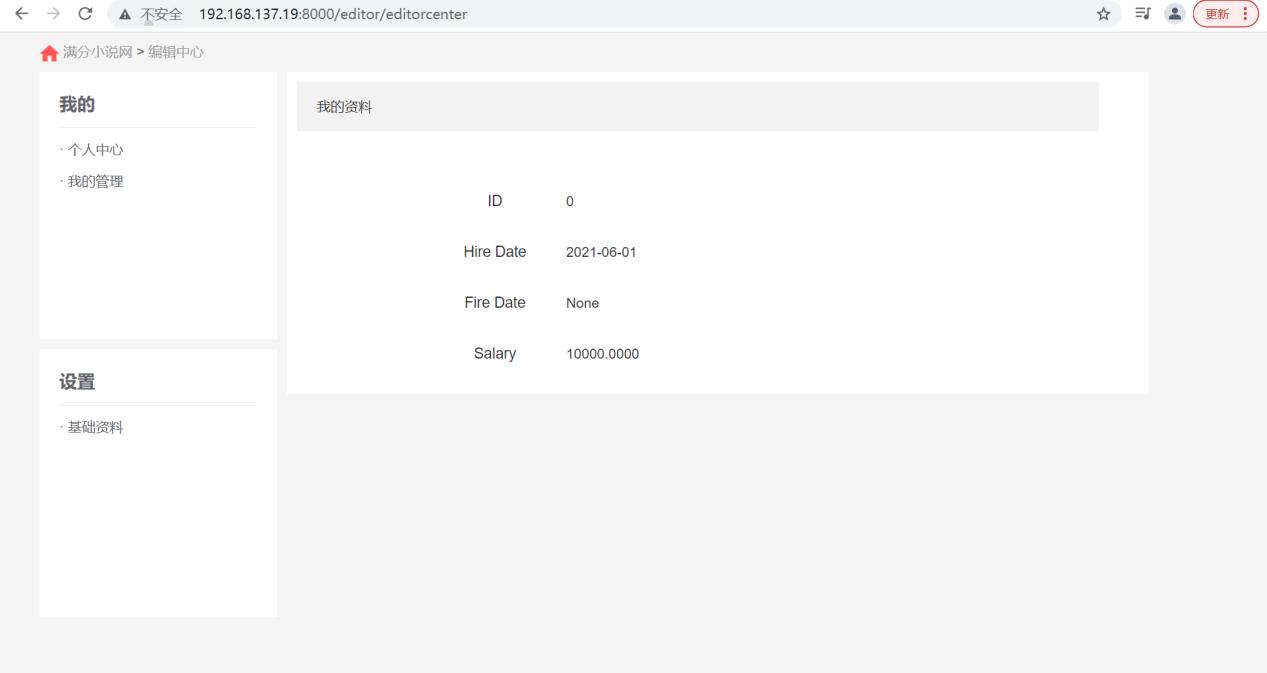


图 12编辑个人信息界面



图 13编辑审核界面



图 14作品审核界面

**5.2作者界面展示**

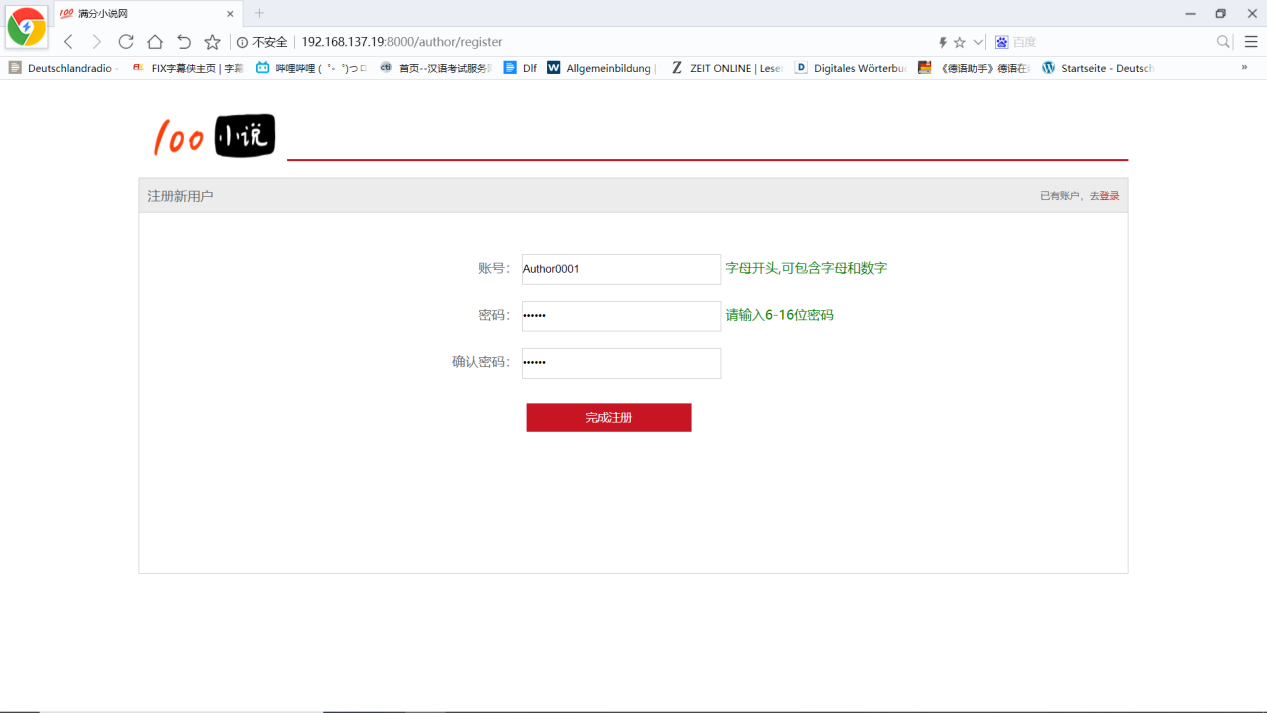


图 15 作者注册

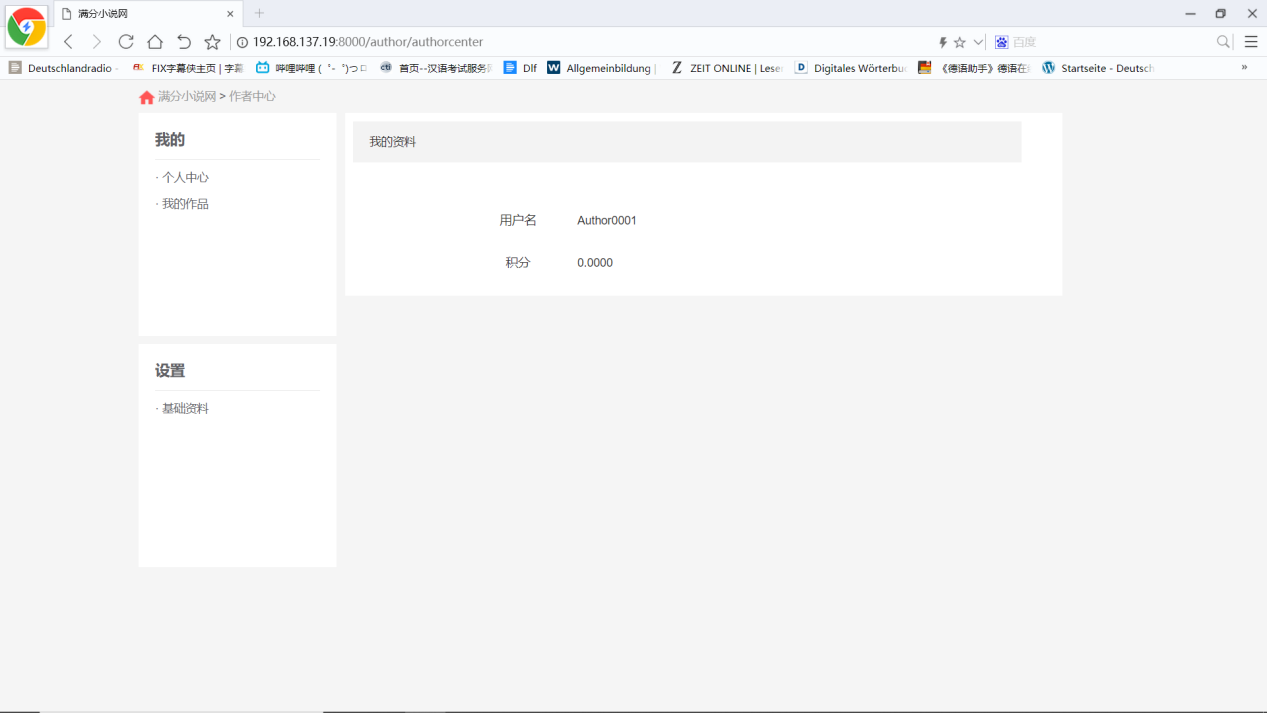


图 16作者个人中心

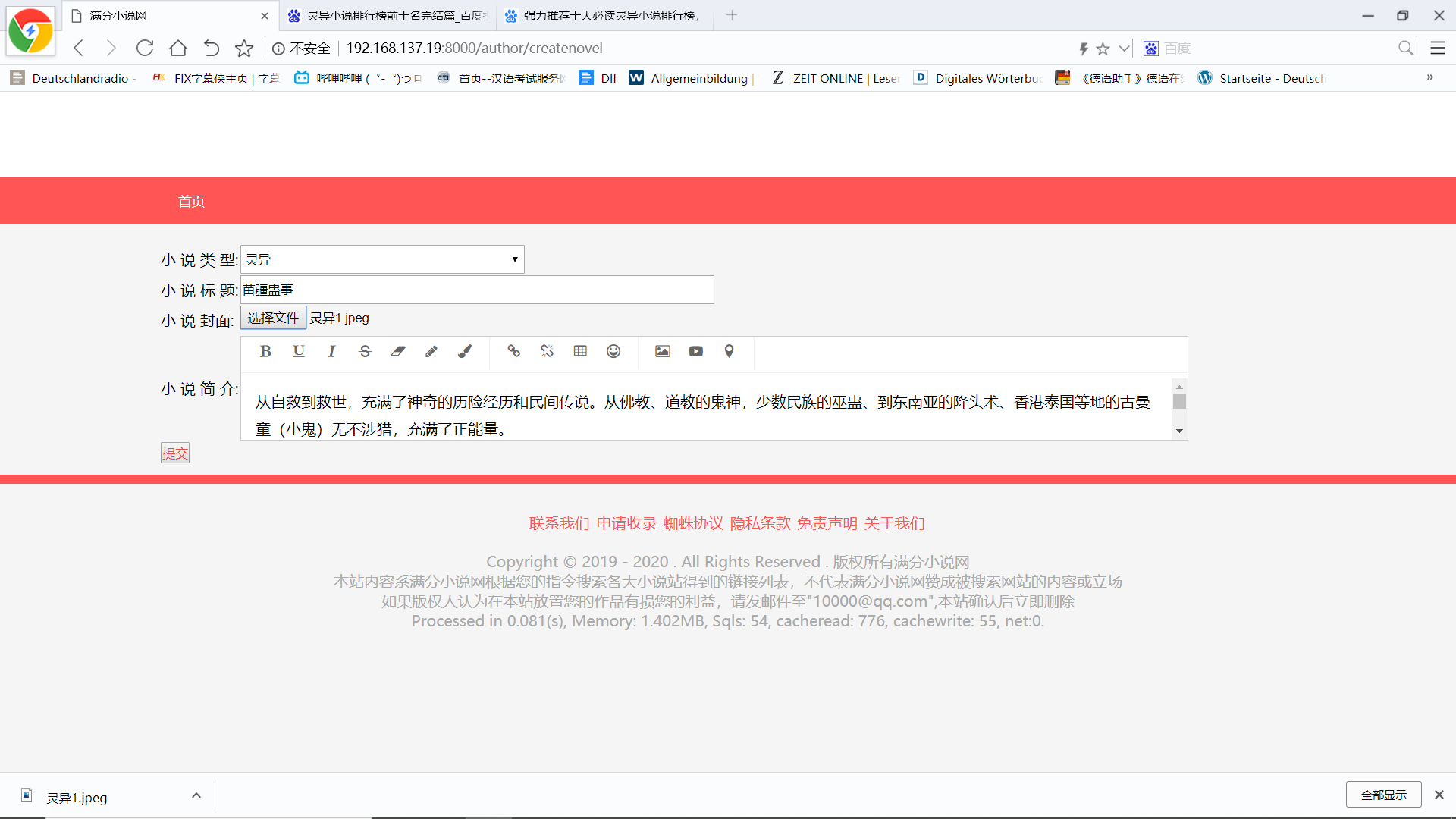


图 17小说编辑

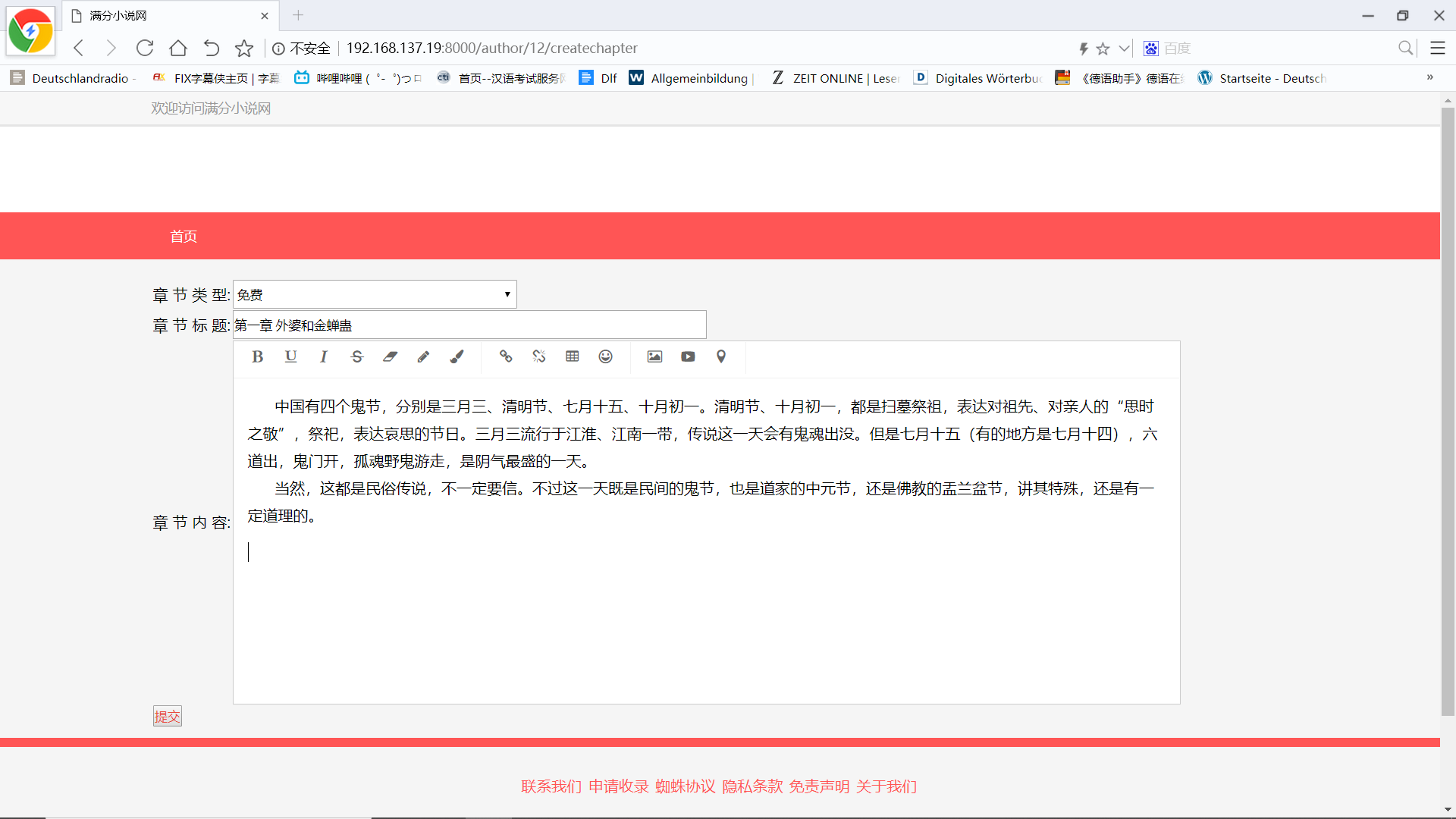


图 18章节编辑

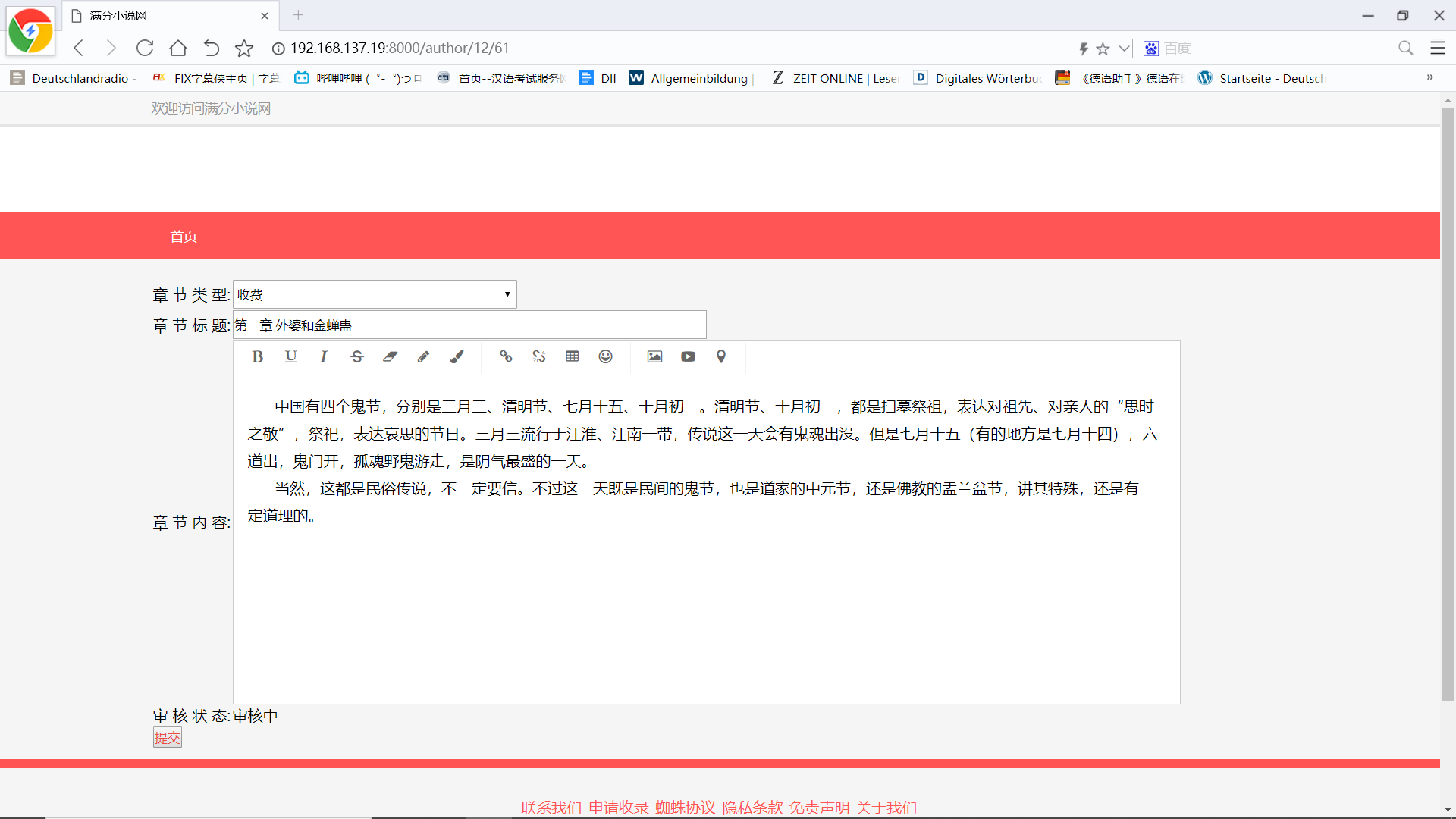


图 19章节审核

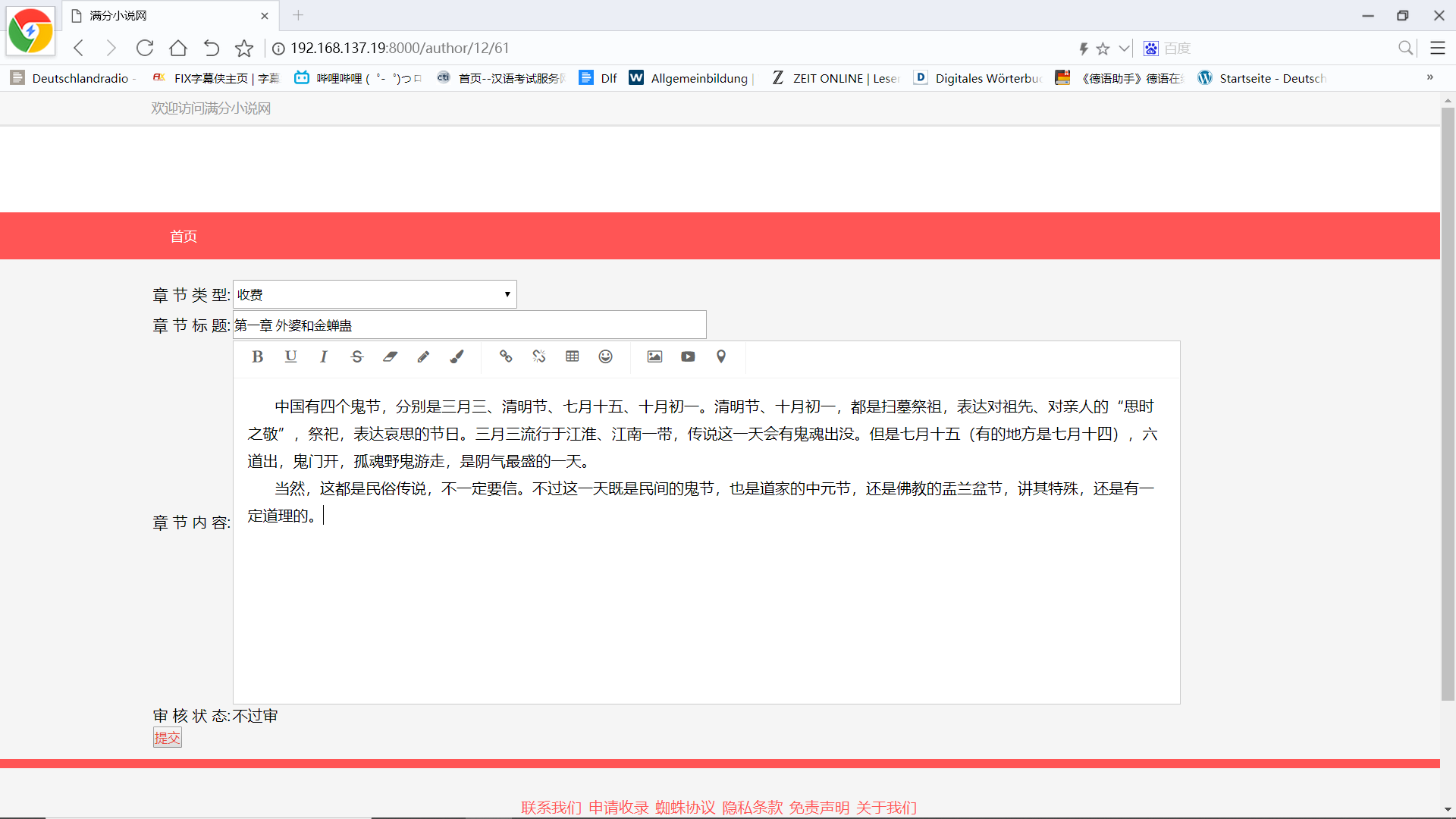


图 20章节不过审

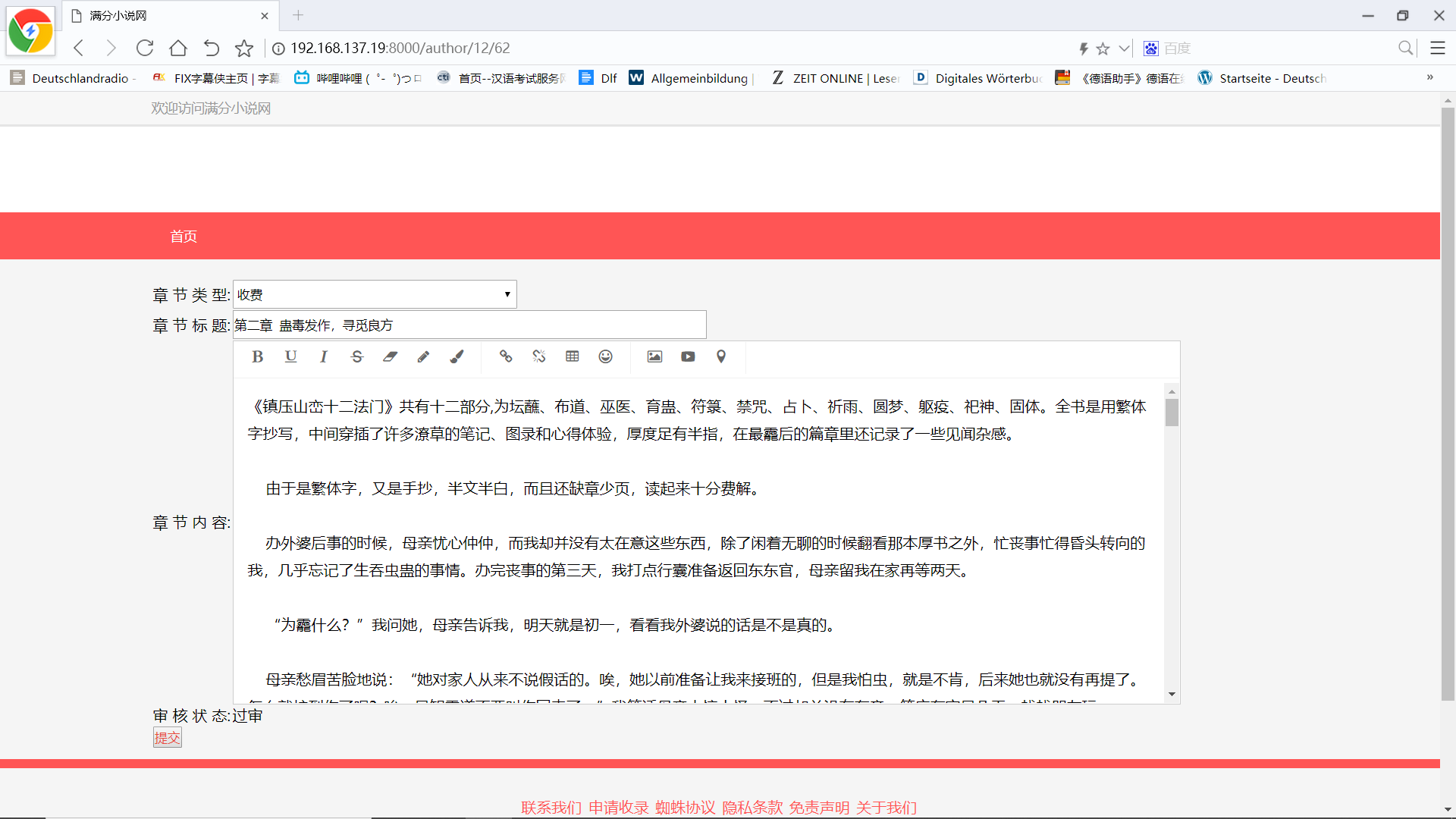


图 21章节审核通过

**5.3管理员界面**

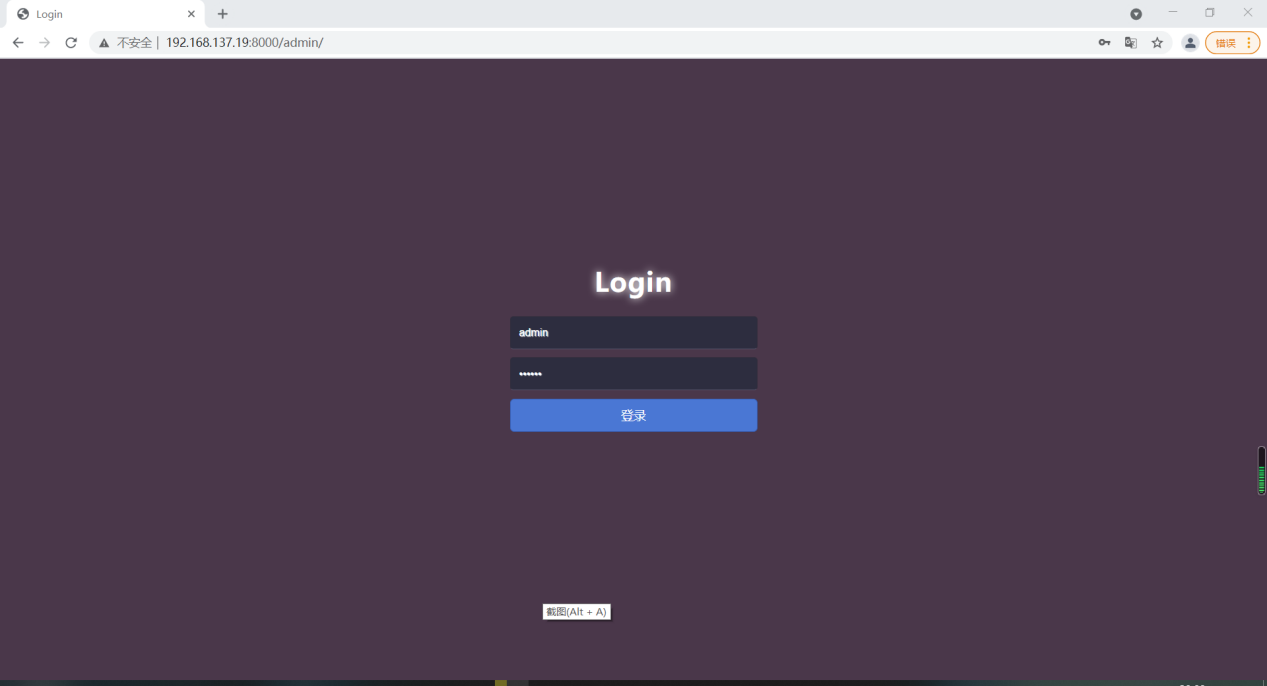
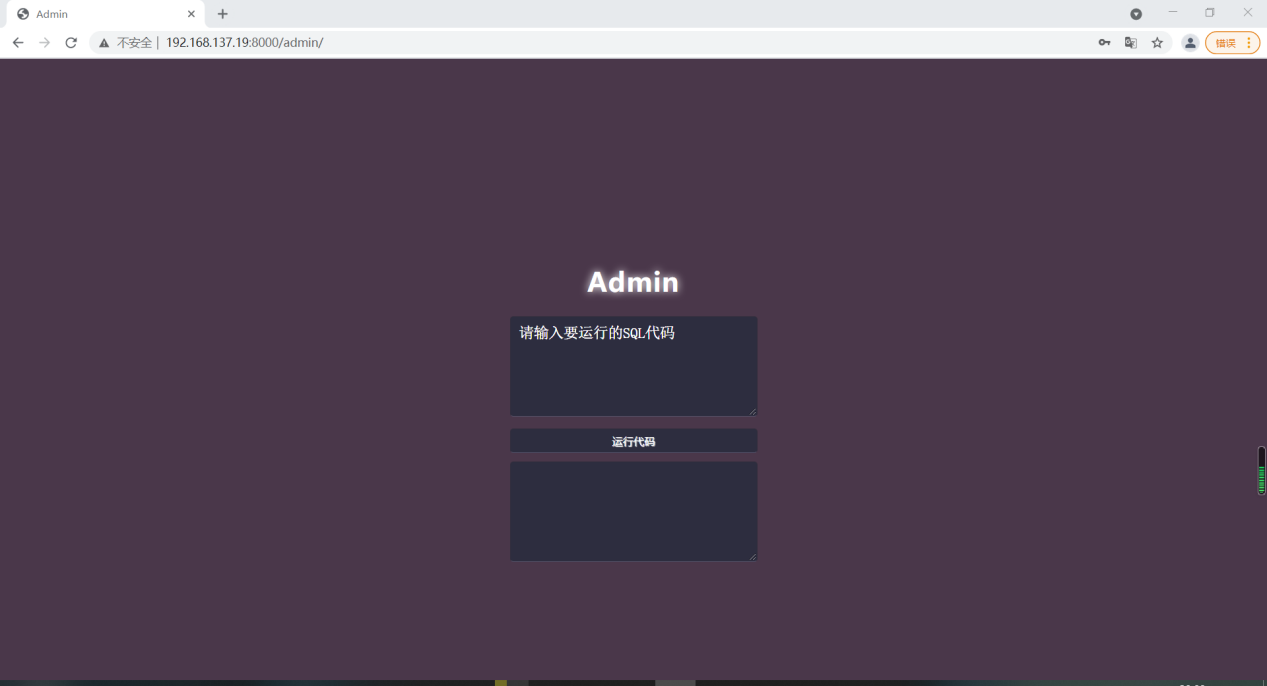


图 22管理员登陆界面



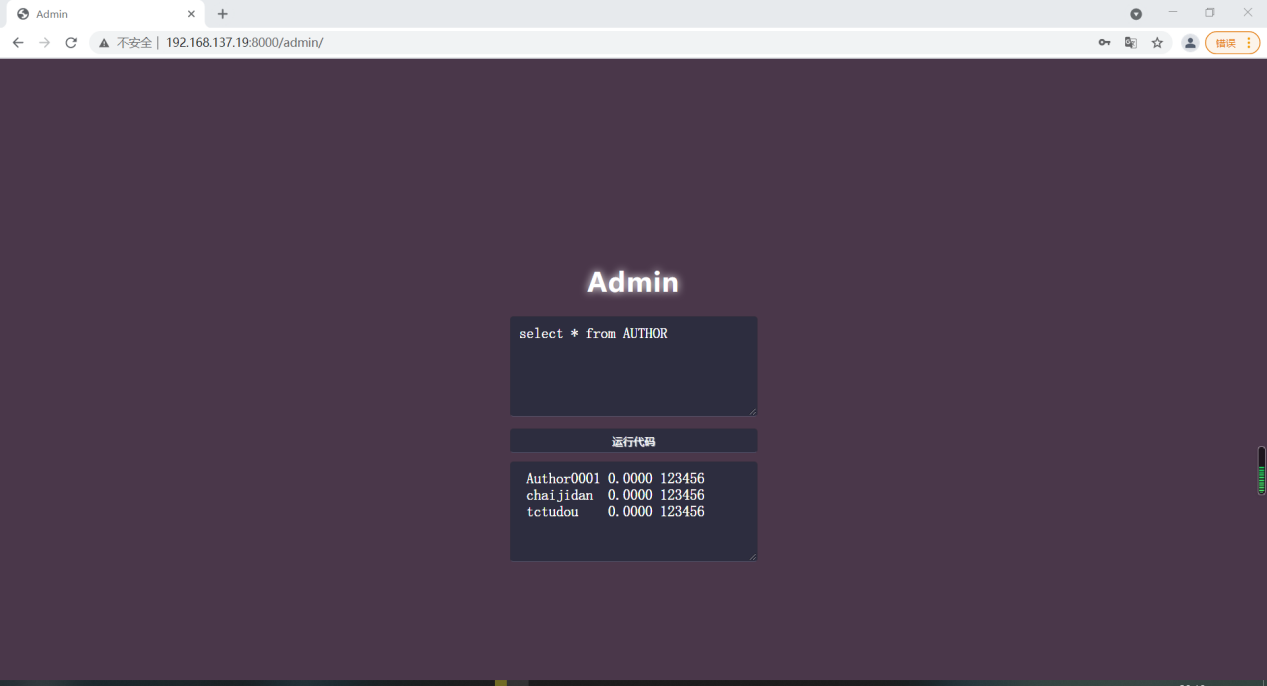


图 23管理员SQL输入界面

**5.4**

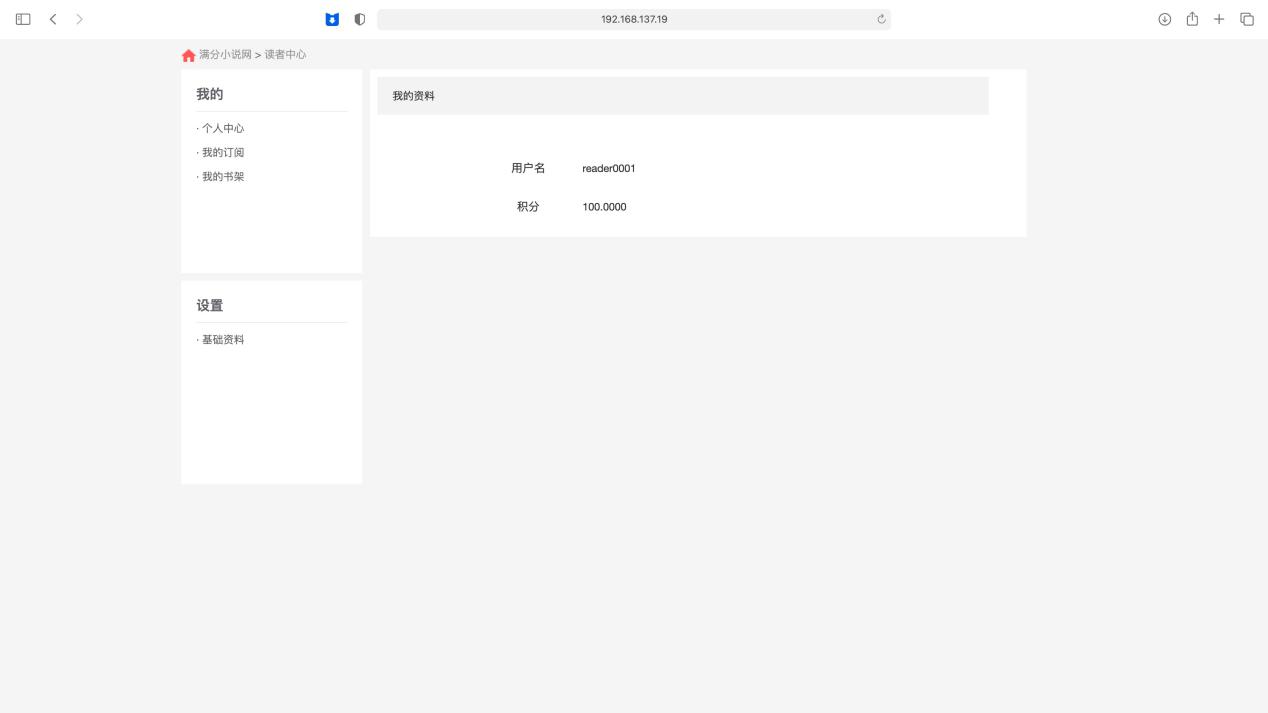


图 24读者个人中心界面

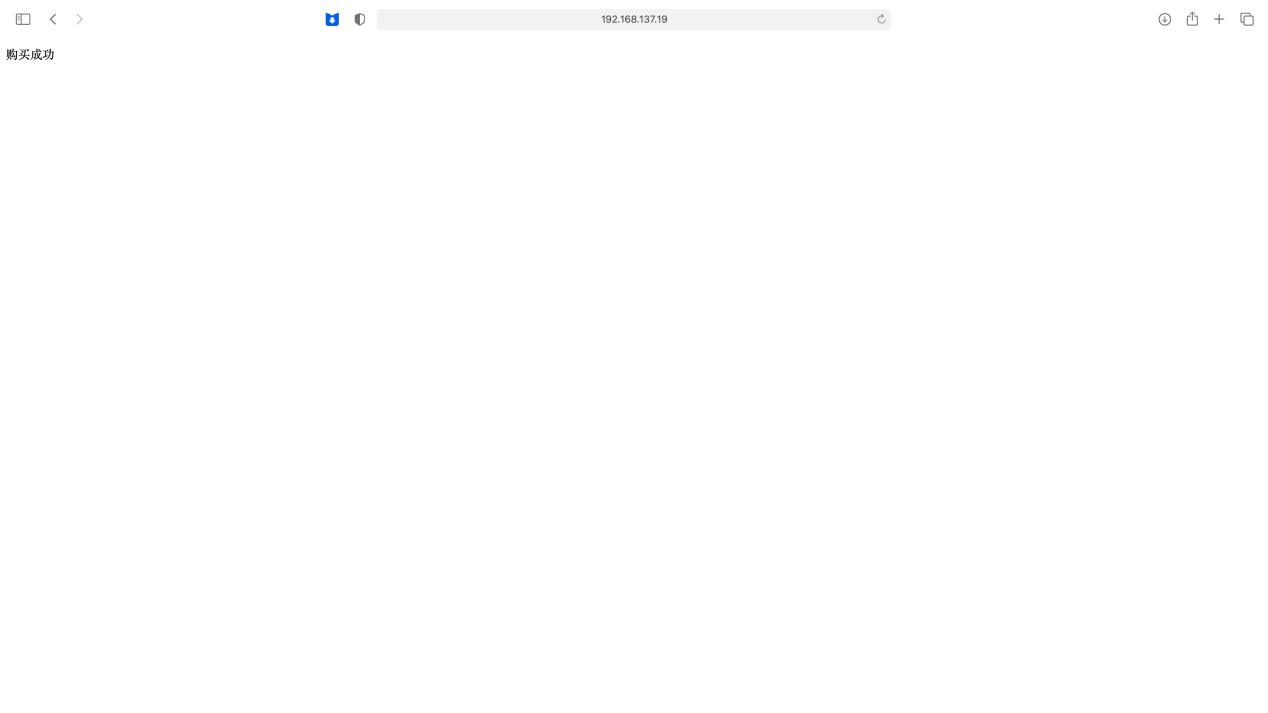
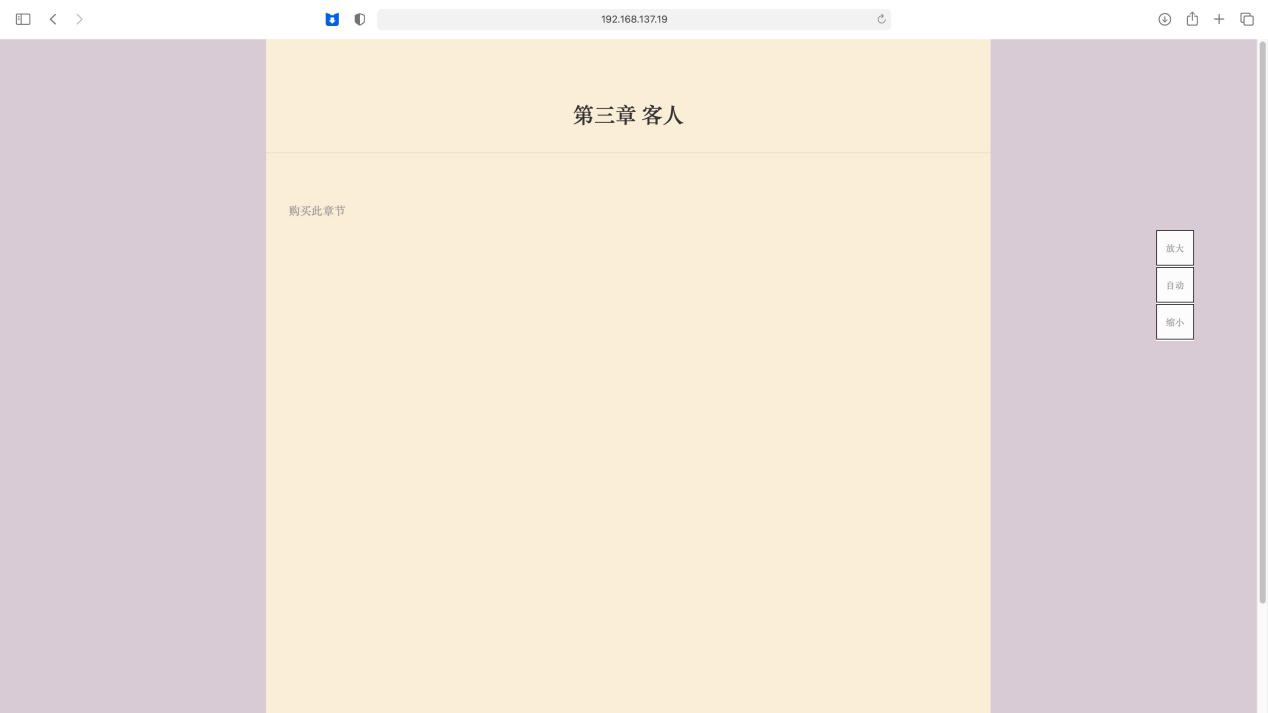


图 25读者购买章节

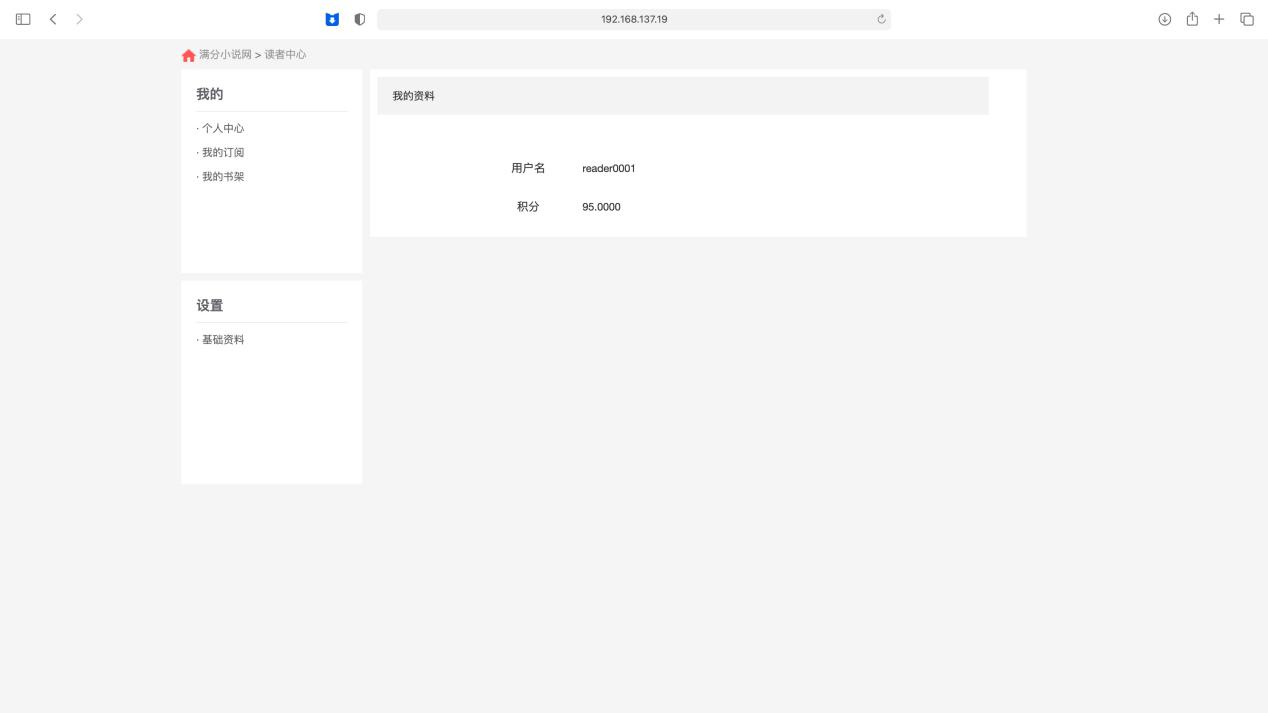


图 26读者购买成功[[1]](#footnote-0)

**5.5整体页面展示**

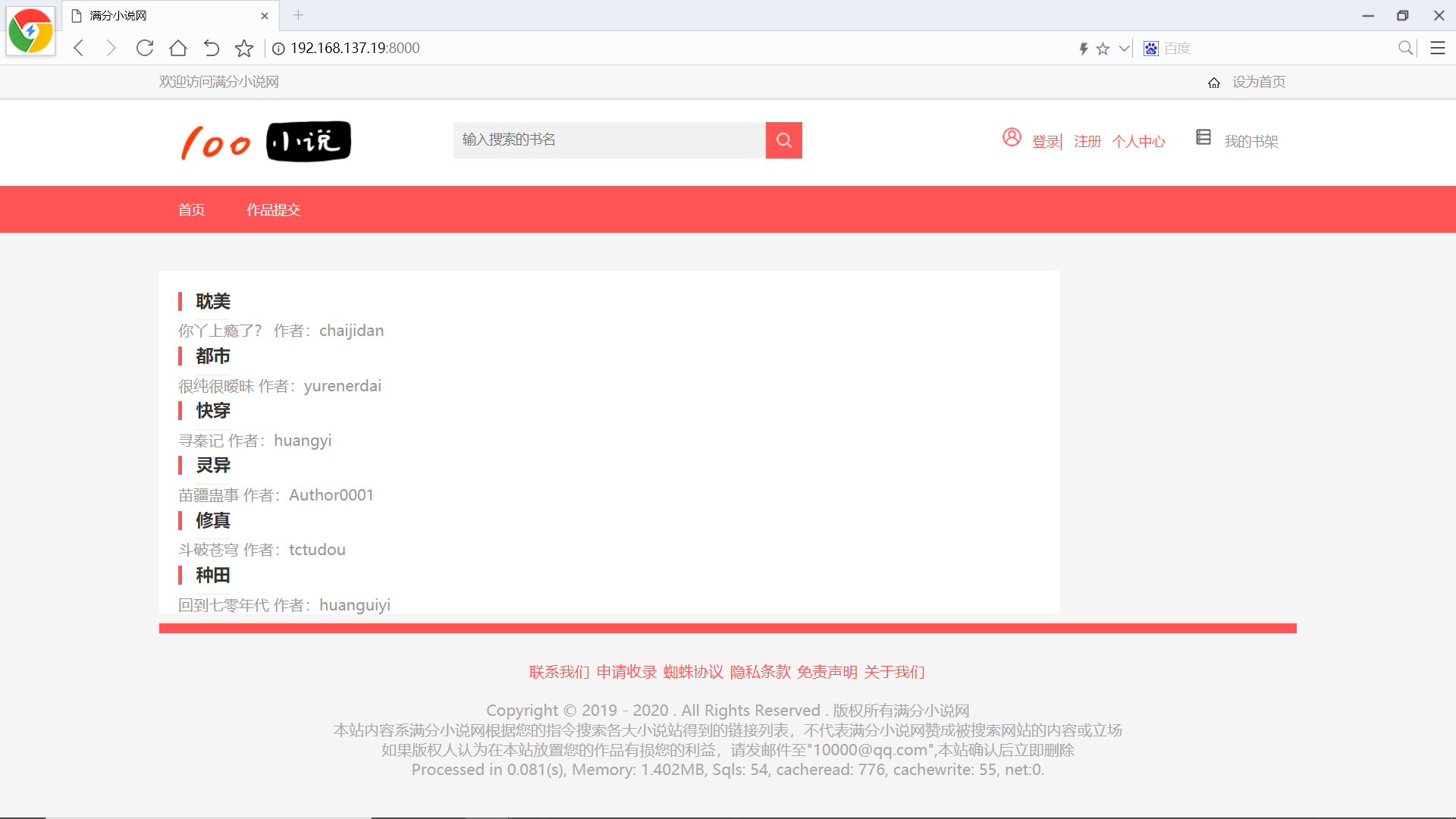


图 27首页

1. **实验总结**

系统开发所付出的努力、面临的困难，自学了哪些相关知识；自己负责什么模块，遇到什么问题怎么解决的，有没有自己创新的设计，开发的体会与收获等：

系统开发面临的最主要困难主要在于数据库模型的建立以及将原始的模型转化成为表，并且确保它在实际运行的过程中符合实际需求以及实际的逻辑。小组最初始设计的E-R模型和迭代后最终的E-R模型对比如下：

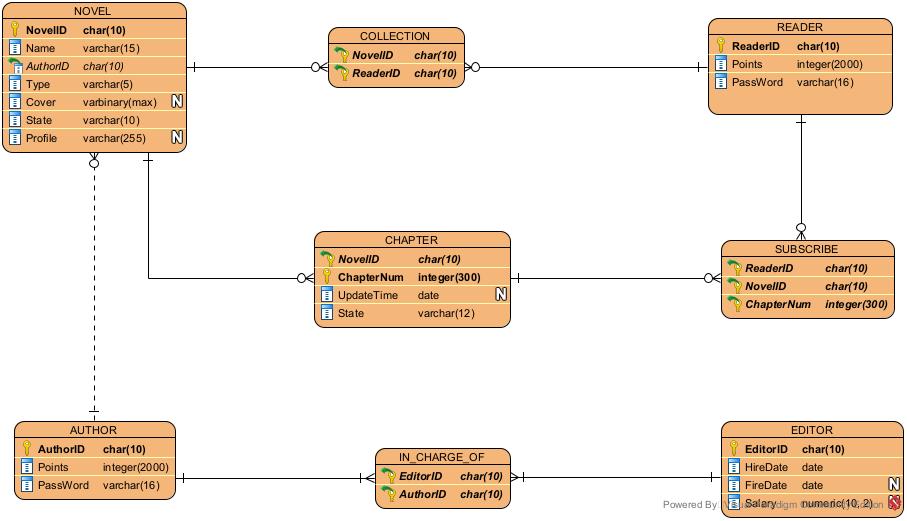


图 28 更改前

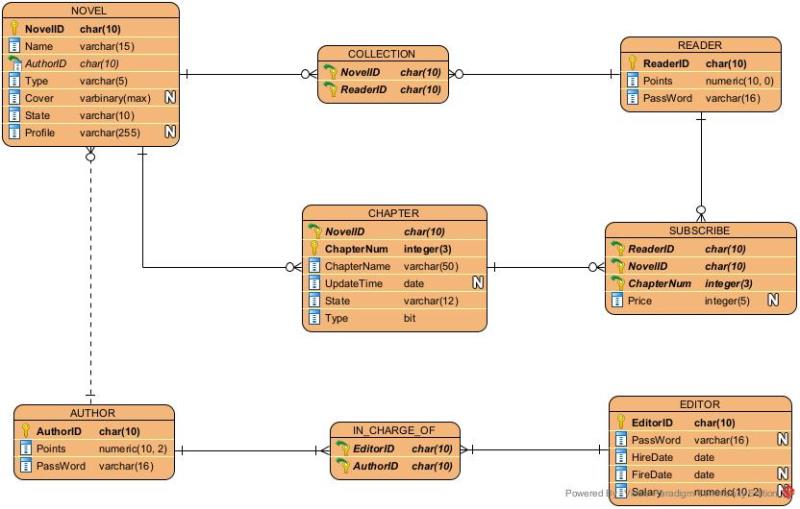


图 29 更改后

可以看到，我们在CHAPTER这张表内增加了State和Type用以说明该章节正处于何种审核状态以及该章节是否收费。

此外，由于小组成员对前端页面的编写不是非常熟悉，因此部分成员也花费了大量时间自学HTML，CSS等技术，最终成功地实现的这个网站的页面。

通过本次系统的设计与开发，我们收获了一定的项目经验，各个成员也或多或少学习了额外的知识并且巩固了已学的数据库相关知识；同时，这次开发增进了同学之间的合作沟通能力，这对于未来从事任何类型的工作都将是有益的。

1. 此处读者购买扣除5积分，作者获得2积分。 [↑](#footnote-ref-0)