总体设计报告

网上书店

Online Bookstore

作者：

3160105470 车天一

3160104765 彭雅欣

3180400028 盛璇

1 体系结构设计

1.1 目的

体系结构设计（Architectural Design）表示建立计算机系统所需的数据结构和程序构件。它需要考虑系统采取的体系结构风格、系统组成构件的结构和属性以及系统中所有体系结构构件之间的相互关系。体系结构设计时构建软件的初始蓝图。

至于什么是软件体系结构呢？Bass、Clements 和 Kazman[Bas03]给出了一下定义：程序或计算系统的软件体系结构是指系统的一个或多个结构，它包括软件构件、构件的外部可见属性以及它们之间的相互关系。

对此，软件体系结构不仅提供了一种表示，有利于利益相关方的交流，而且突出了早期设计决策，最终构建一个相对小的、易于理解的模型来描述系统（网上书店）如何构建以及各构件之间如何协同工作。本章余下部分将从体系设计风格、层次结构、体系结构环境图以及原型四大方面着重研究并介绍我们的体系结构设计方法。

1.2 体系结构风格（Architectural Style）

体系结构风格是施加在整个系统构件上的一种变换，目的是为系统的所有构件建立一个结构。在对已有体系结构再工程时，体系结构风格的转化会导致软件结构的根本性变化，包括对软件构件的再分配等等。恰当的软件结构风格与模式合起来构成了软件的外形。

在过去数百万的计算机系统中，绝大多数可以归结成以下少数模式中的一种：

- 以数据为中心的体系结构
- 数据流体系结构
- 调用和返回体系结构
- 面向对象体系结构
- 层次体系结构

本次设计的软件网上书店虽然也会涉及到调用、返回以及面向对象的设计，但是从利益相关来看，它还是以数据库为中心，着重于对数据的更新、增加、删除、修改，此处的数据库类似于“黑板”，用户、管理员、游客独立的执行过程，必要时通过“黑板”来传送。此外在此基础上将层次体系结构与以数据为中心的体系结构结合使用，使软件的构成和构件的相互关系更清晰的呈现出来。

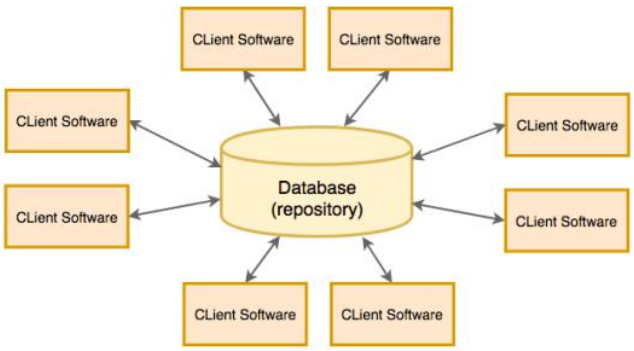


Figure1 以数据为中心的体系设计风格

1.3 层级结构（Hierarchy）

在之前的章节中，我们定义了软件所使用的体系结构风格。在此基础上，我们认为使用层次结构结合一数据为中心的体系结构可以使软件结构更加条理和明确，因此此处我们还需定义软件的层次结构。在顶层系统的基础上，我们可以先简单的分为两个子系统：服务器系统和客户端系统，这不仅是根据层次结构“中心”和“边缘”划为的，而且也符合 MVC 设计模型，简化了设计。这两个子系统具体将各自分为若干个相对独立的功能模块，其中每个功能模块依赖于更底层的功能模块，因此形成我们系统的四大层次：

- 顶层：书店系统
- 次顶层：服务端子系统，客户端子系统
- 中间层：各大功能模块
- 底层：与外部支持的接口等

这四大层次之间相互独立，各自为上层服务，又采用下层系统或者外部接口的服务。层次分明又有机结合。如图所示

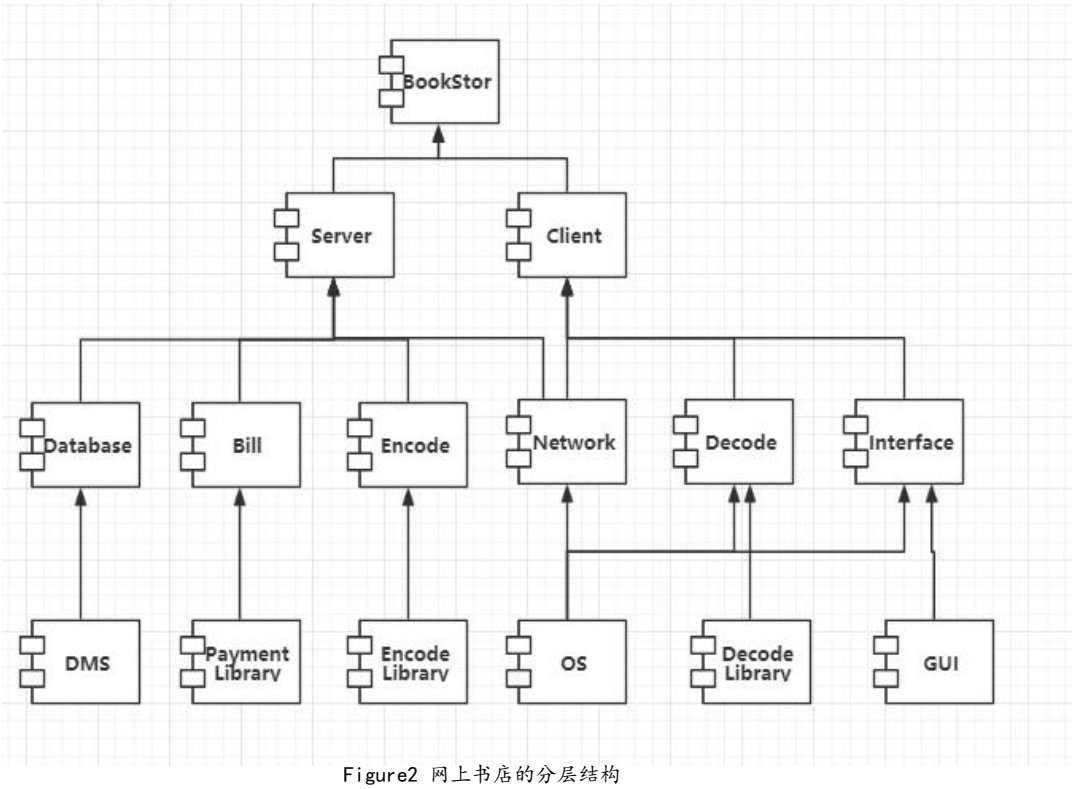


Figure2 网上书店的分层结构

1.4 体系结构环境图（Architectural Context Diagram）和原型 （Archetype）

在体系结构设计开始的时候，我们应该建立相应的环境。为了达成这一目的，应该定义与软件交互的外部实体和交互的特性，这些信息一般从需求模型中获取。一旦确认了软件的环境模型，并且描述出所有的外部软件接口，就可以确认体系结构原型集。

在明白外部的因素之后，我们对模型内部进行设计，还需要挖掘模块中核心的抽象类或者模式，此即原型。如果要是系统结构化，就必须要对这些原型进行结构化建模，而原型本身并不需要太多的实施细节。这些原型使逻辑关键环节，也是任务的实现支撑。在这些原型的基础上进行实例化和精细化，我们便能够构件具体功能和性能不同的软件系统。

在构建体系结构环境图时，我们选用教材中提到的一般结构来描述，如下图所示。它由上级系统、下级系统、参与者和同级构成，接下来将具体展示每一部分的体系结构环境图和原型设计。

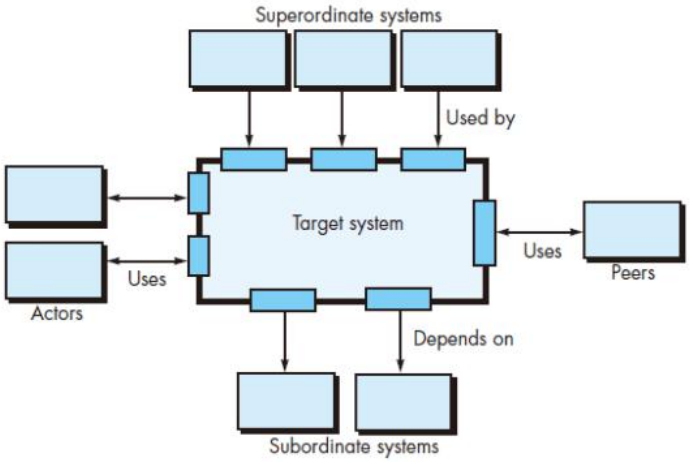


Figure3 体系结构环境图样例

1.4.1 顶层 - 书店系统（BookStore）

我们设计的书店系统只是模拟一个简单的电子商务系统，并不是其他系统的组成部分，也不被其他系统所使用，因此它没有上级系统，而且没有其他同级系统。因此这里它只有子系统：服务器子系统和客户端子系统，参与者：用户、管理员。

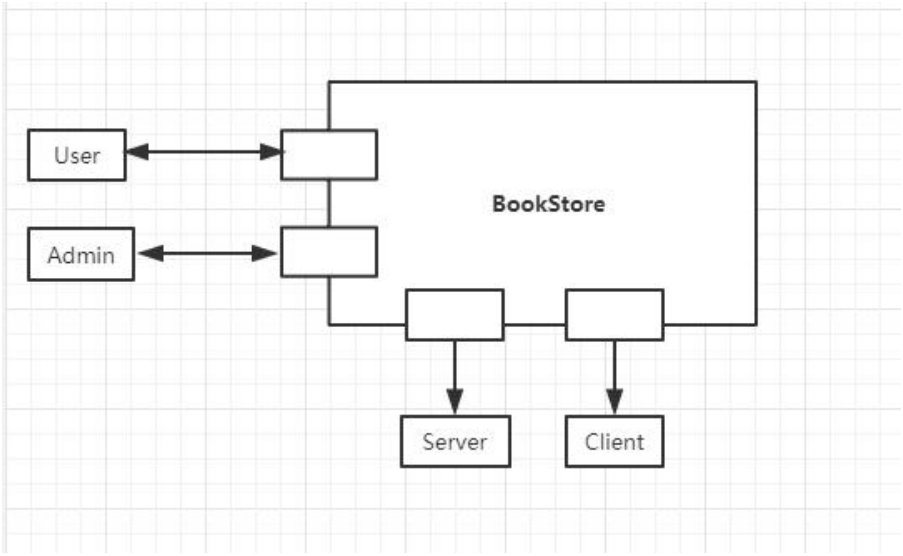


Figure4 书店模块体系结构环境图

整个系统相对比较简单，此处便不多加赘述了。接下来，将重点阐述一下各功能模块的环境设计。

1.4.2 次顶层 - 服务器端设计（Server）

服务器端设计一直运行在远程计算机上，负责处理来自客户端的请求，同时采用服务器端与数据库相连，这样有利于系统的维护、系统的扩展。大量的请求和响应时电子商务系统需要克服的最主要的难题，我们将其归入下一层的数据库模块，同时纳入网络模块和财务管理模块来完善所需要的服务。

- 网络模块：主要负责服务器端系统与外界的通信
- 数据库模块：主要服务服务器端系统与数据库的通信
- 编码模块：主要负责适应网络传输过程且提高性能的传输问题
- 财务管理模块：考虑到其本质上交易系统，因此需要其在账目处理上清晰准确因此比较有必要单独设计

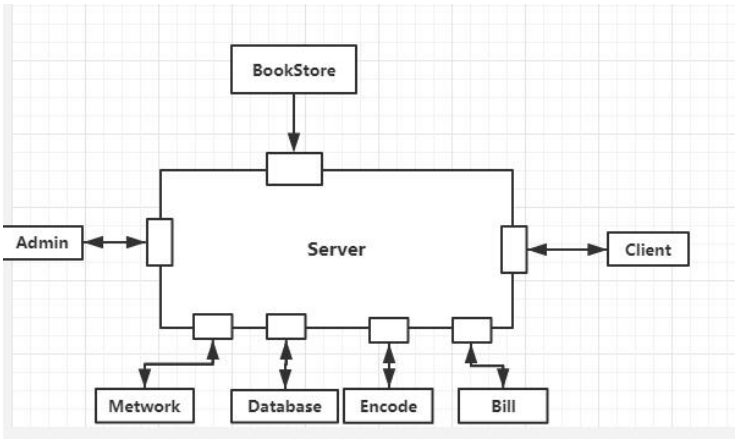


Figure5 服务器端子系统体系结构环境图

1.4.3 次顶层 - 客户端子系统（Client）

因为要实现与服务器的通信，因此客户端子系统也拥有着网络模块、解码模块，此外在需求中提到的优美的 IU 设计对于电商系统是非常重要的，因此也需要用户界面模块，当然还包括用户行为的逻辑分析。因此客户端子系统包括以下子模块：

- 网络模块：主要负责客户端系统与外界的通信
- 解码模块：主要负责解析数据并充分展示
- 用户界面模块：充分的界面展示，用户行为的逻辑分析

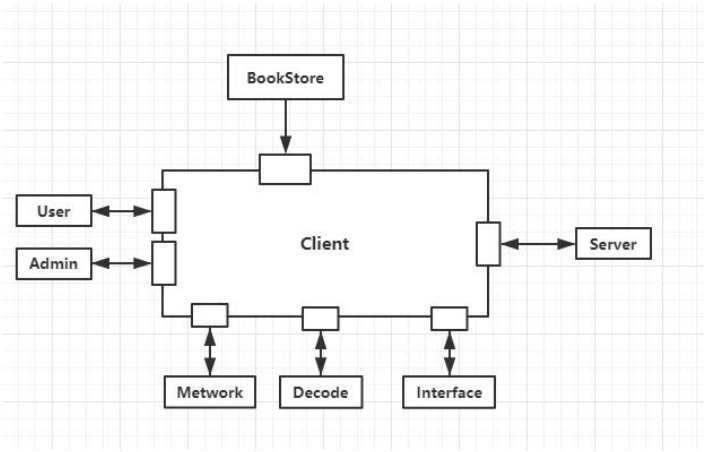


Figure6 客户端子系统体系结构环境图

完成次顶层的环境搭建之后，我们需要完成更细致的中间层的功能模块的环境和原型设计。

1.4.4 中间层 - 服务器端子系统的网络模块（Network）

从模块是基于操作系统提供的网络接口实现的

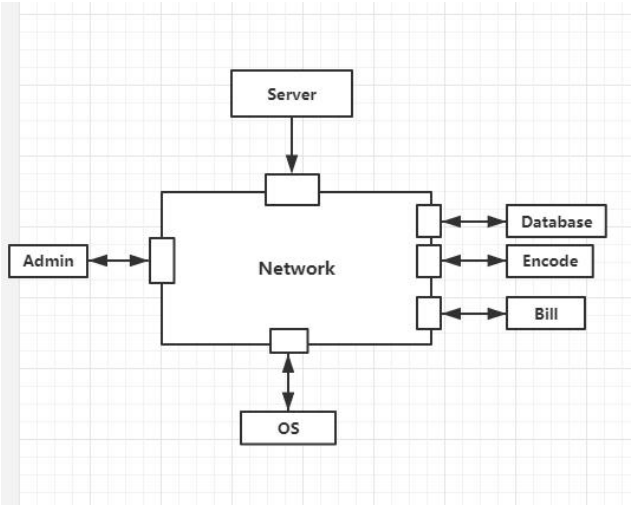


Figure7 服务器端子系统网络模块环境图

此模块是负责请求和响应的功能组件，联系操作系统和计算机网络相关知识，我们暂定设计该模块原型如图所示：

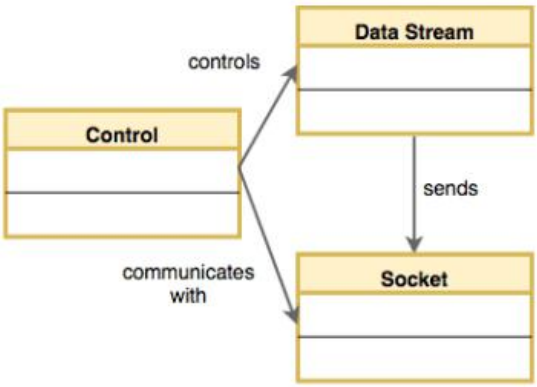


Figure8 服务器端子系统网络模块原型图

1.4.5 中间层 - 服务器端子系统的数据库模块（Database）

此模块是基于数据库提供的数据接口。

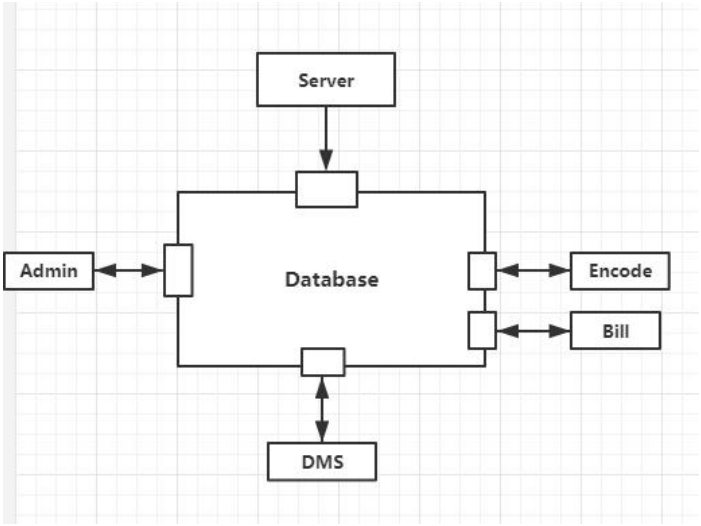


Figure9 服务器端子系统数据库模块环境图

此模块负责响应客户端的数据查询、增减、修改等操作，但是因为涉及到大数据的性能问题，所以该模块还有很多需要优化的地方，它的大致原型抽象为下图所示

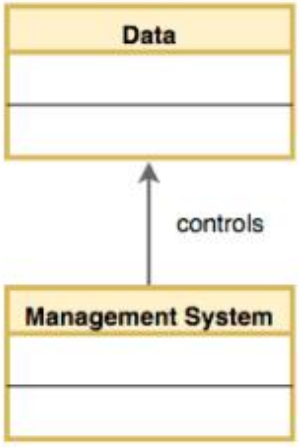


Figure10 服务器端子系统数据库模块原型图

1.4.6 中间层 - 服务器端子系统的编码模块（Encode）

此模块是配合网络传输的特定场景所需要功能完善性模块，在之后的性能优化中同样起到非常重要的作用。为了稳定的、安全的、高效的传递数据，我们基于成熟的已有的库来达成这一目的。

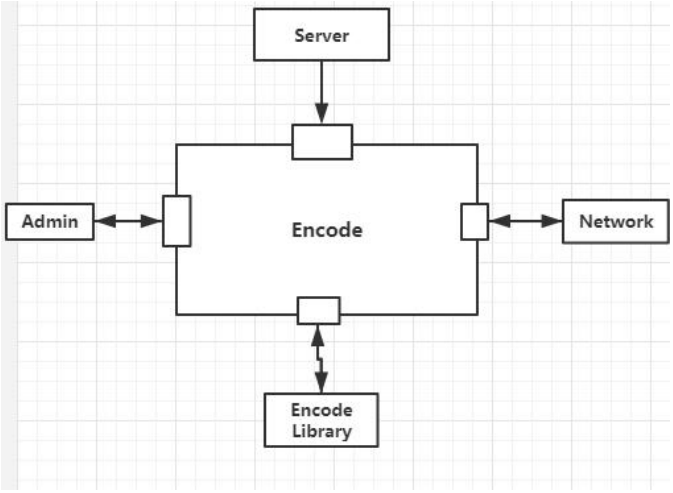


Figure11 服务器端子系统编码模块环境图

这个模块本质上是对数据流的加工转换和传输，因此同样参考数据流模型，将此模型的原型集合设计为数据 - 编码 - 缓冲。原型图如下所示：

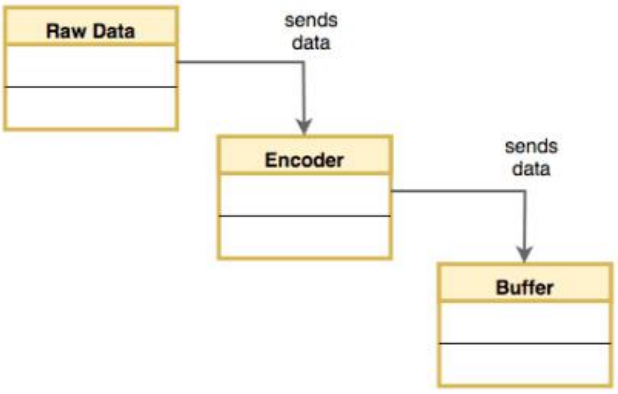


Figure12 服务器端子系统编码模块原型图

1.4.7 中间层 - 服务器端子系统的财务管理模块（Bill）

此部分可以基于提供的移动端的支付库实现

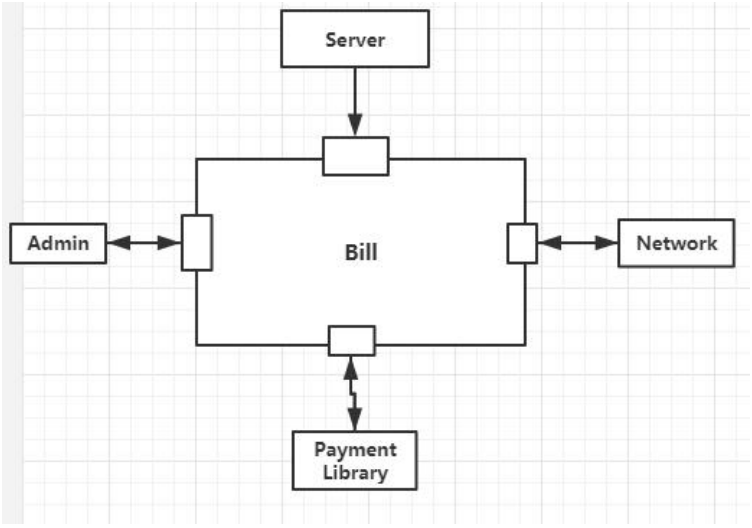
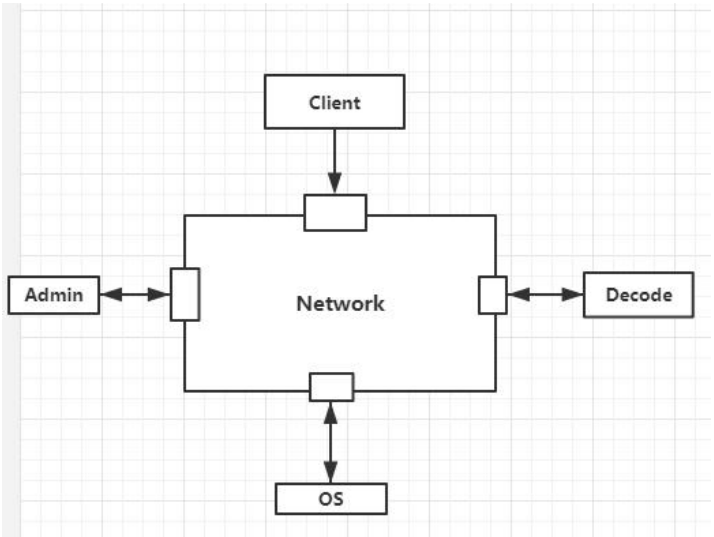


Figure13 服务器端子系统财务管理模块环境图

此模块封装了所有与支付相关的方法，保障在电商中最核心的安全、准确的支付流程，抽象成原型即为不同账户之间的交易。

1.4.8 中间层 - 客户端子系统的网络模块（Network）

该设计与大部分设计一样，此模块与服务器端的设计模块大致相同。只需要注意此处需解码而非编码，因此将原型图中数据流向反过来即可



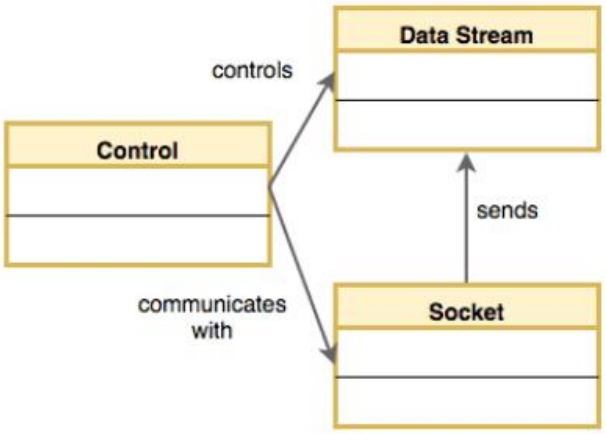
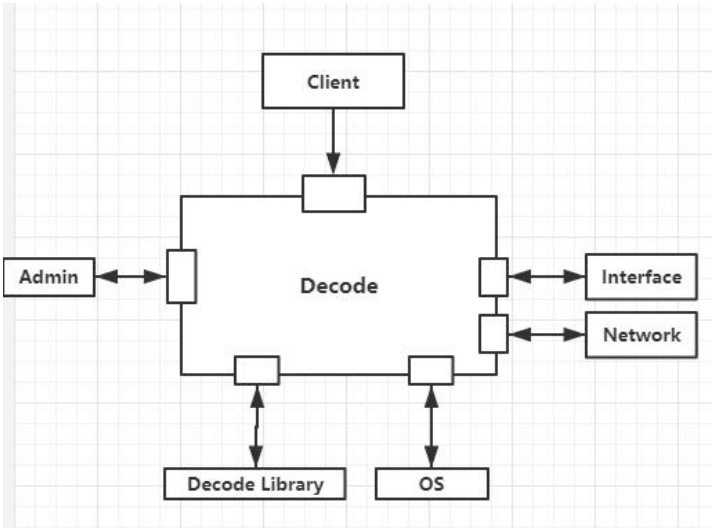


Figure14 客户端子系统网络模块环境图及原型图

1.4.9 中间层 - 客户端子系统的解码模块（Decode）

此部分亦与服务器端子系统的编码模块大致相同，同样是为了配合网络传输的功能性模块，细微差别在于数据流方向的不同。



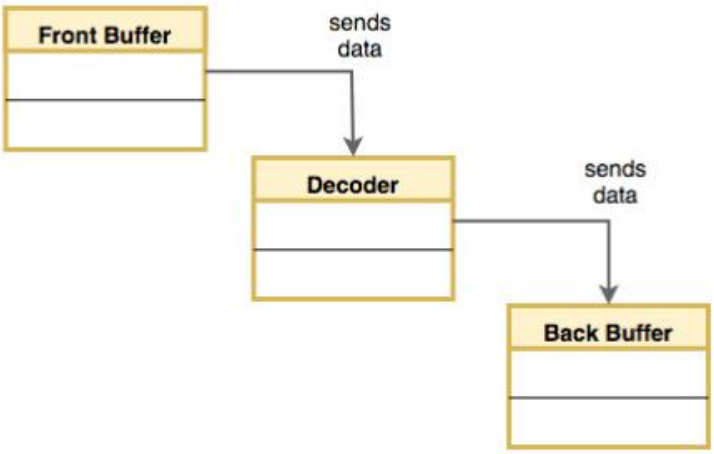


Figure15 客户端子系统解码模块环境图及原型图

1.4.10 中间层 - 客户端子系统的用户界面模块（Interface）

此模块使用到的底层模块是特定的 GUI 库。这是用户和管理员都可以交互的模块，因此也需要根据用户需求进行测试和维护，达到引人注目的效果。

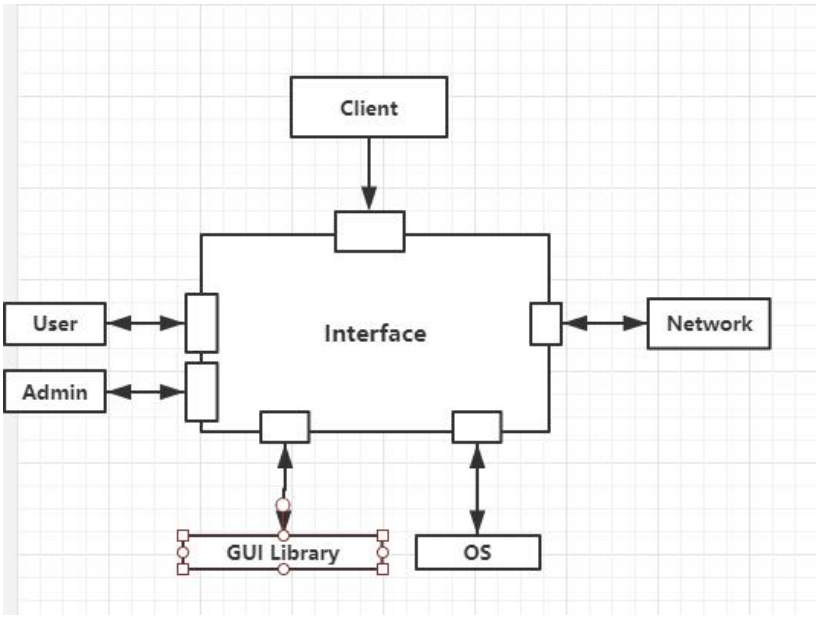


Figure16 客户端子系统用户界面模块环境图

原型方面，我们使用经典的 MVC 模式接管我们的设计方案，具体关系如下图所示

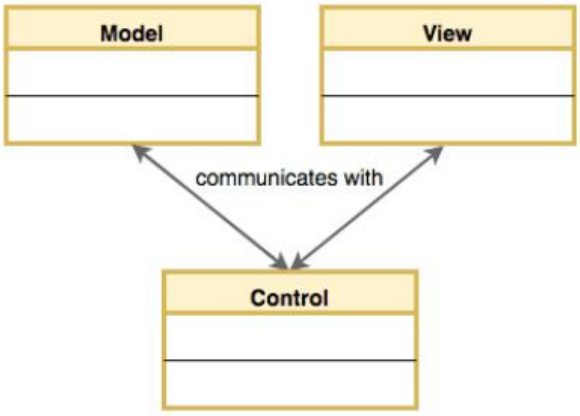


Figure17 客户端子系统用户界面模块原型图

2 服务器端设计

2.1 服务器架构（Server Flamework）

本部分将简述对于服务器端整体架构的设计。

可以将服务器端分为三个层次：表现层、业务逻辑层和数据访问层。

2.1.1 表示层（Presentation Layer）

用于显示数据和接收用户输入的数据，为用户提供一种交互式操作的界面。比如用 JSP 设计页面时，可以展示页面，也可以根据用户的操作收集到其数据。

2.1.2 业务逻辑层（Business Logic Layer）

处于数据访问层与表示层中间，对于数据访问层而言，它是调用者；对于表示层而言，它却是被调用者。负责系统领域业务的处理，负责逻辑性数据的生成、处理及转换。对所输入的逻辑性数据的正确性及有效性负责。

2.1.3 数据访问层（Data Access Layer）

负责对数据库数据的访问、删改等操作，可以访问数据库系统、二进制文件、文本文档或是 XML 文档。

2.2 服务器处理流程（Process）

根据服务器端的分层，可以将其处理流程大致划分如下图。其中客户发出请求发生在表示层；处理请求时，会需要进行数据的访问等操作，这就涉及到了数据访问层；而其他的操作都需要业务逻辑层的参与。

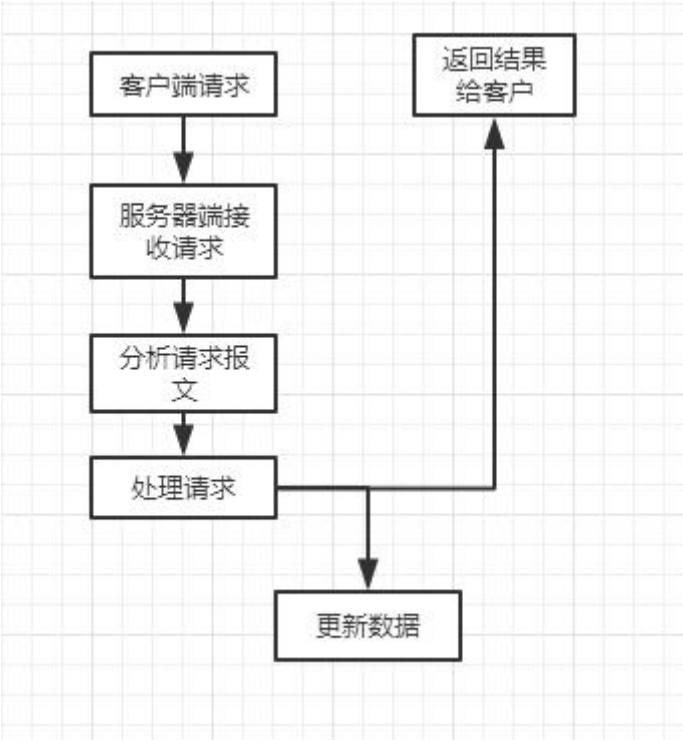


Figure18 服务器处理流程图

用户通过界面操作发送请求给服务器。服务器接收到请求后，会对其进行分析和处理，而这正是服务器的核心部分，需要用于处理的时间也相较于更长。网站在运行的过程中，可能在同一时间段内接收到多位用户的请求，所以我们在服务器端需要是多线程，以便能在监听到用户请求后，能及时响应。

下面以将商品加入服务器的过程为例，来解释服务器的处理流程。注册用户和游客都可以使用该功能，但是服务器接收到的请求中的数据信息有所不同，故服务器会根据其身份对其请求进行不同的处理操作。

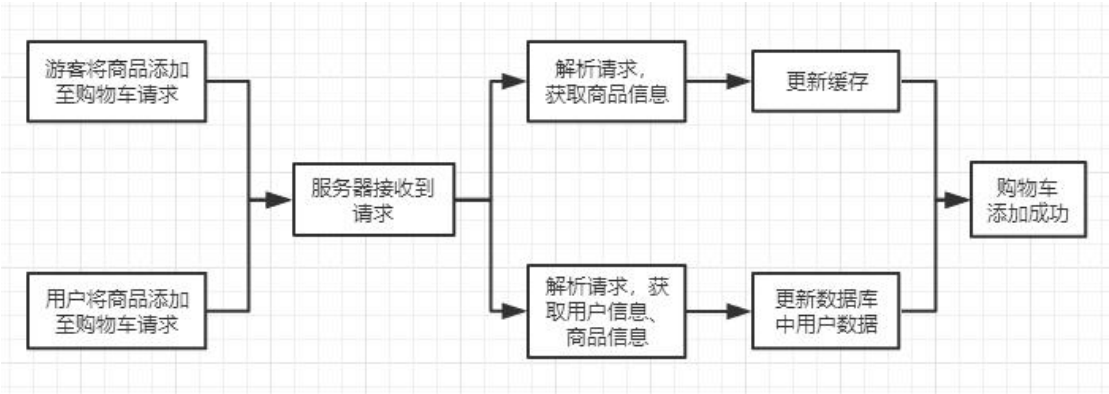


Figure19 添加购物车处理流程图

2.3 数据库设计（Database）

对系统中的书籍、用户等实体进行分析，可以画出如下实体关系图（ER 图）。图中仅表示了注册用户与其他实体类间的关系，游客与管理员与其仅略有不同。

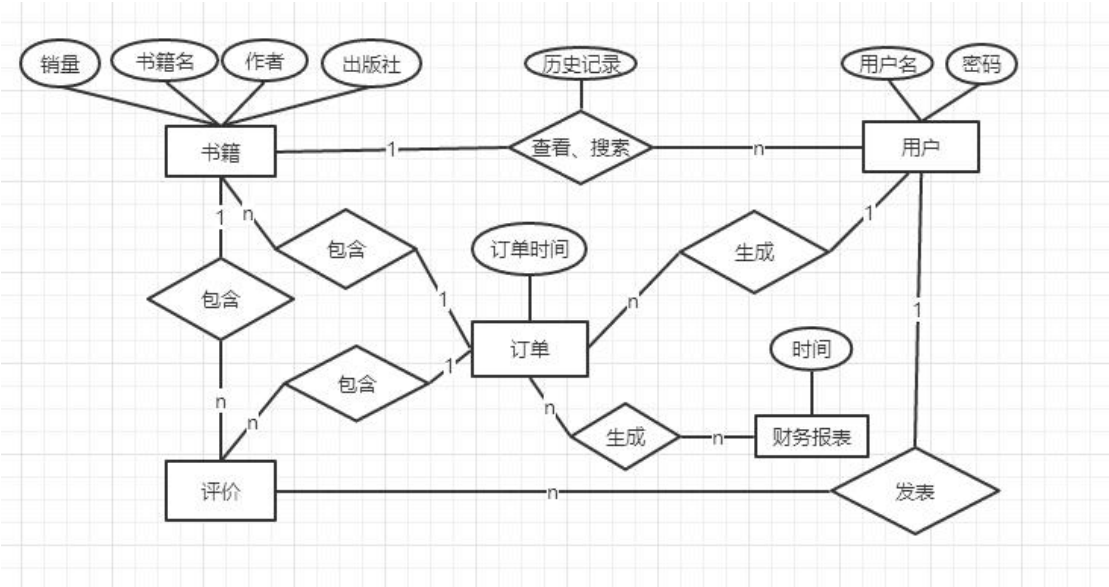


Figure20 数据库 ER 图

基于 ER 图，可以设计出系统所需的数据库表：

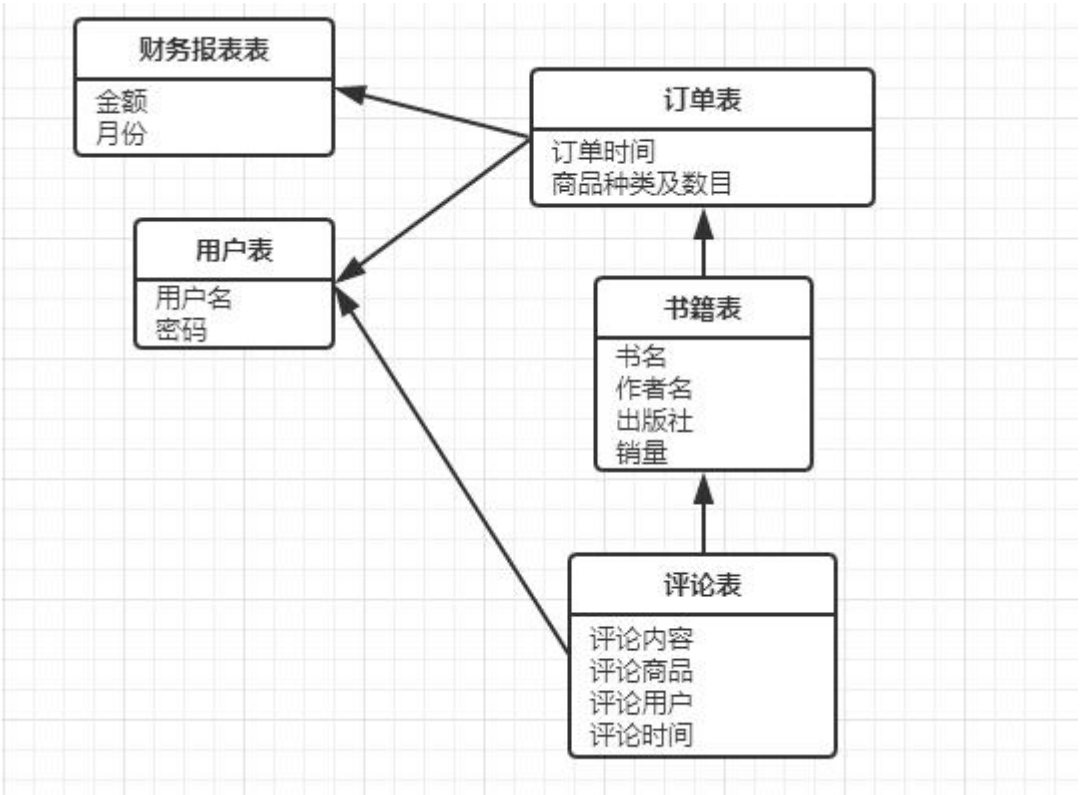


Figure21 数据库表图

3 UI 设计

3.1 设计原则（Principle）

在进行 UI 设计时，需要充分考虑用户的需求，需要将以下原则融入 UI 的设计中：

1. 设计简洁明了，考虑用户习惯。

界面的简洁有利于用户了解和使用产品，减少用户出错的可能。同时可以考虑用户的爱好和习惯来设计界面中功能的位置，方便用户使用。

2. 美观大方

系统界面的设计在视觉效果上便于理解和使用。同时，也可以让界面更为美观，让用户在使用感觉赏心悦目，提高用户的使用体验。

3. 一致性

界面的结构必须清晰且一致，风格必须与产品内容相一致。

3.2 UI 原型设计（Prorotype）

产品原型的主界面有包括主界面、订单页面、搜索页面等。这些页面中的功能与许多传统的电子商务系统的页面类似，这样便于用户的使用，无需花费过多时间用于摸索页面。

例如主页，集中了许多功能，如搜索、登录、订单查询等功能。对于用户不常用的功能，会对其进行整合，不进行单独展示。如在客户服务中，会有帮助中心等功能，为用户提供系统使用等过程中会遇到的问题的答案汇总。

Figure22 主界面原型图

对于搜索图书，可以无需用户登录，就可查看图书的信息、加入购物车。不过在后续购买时会提示游客登录，以及游客登录后可以将在游客状态下添加的购物车中的商品更新至用户信息中。

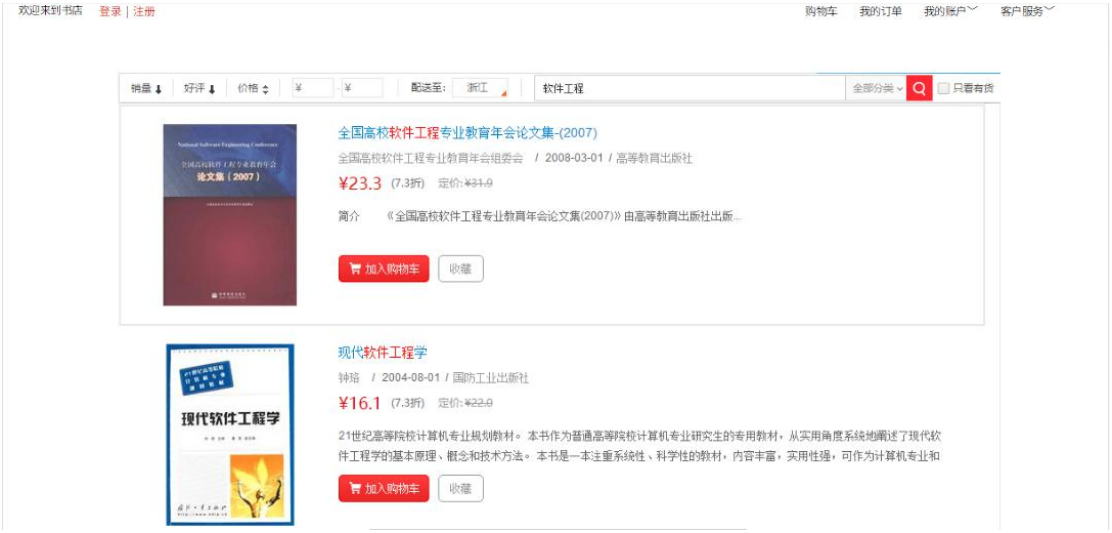


Figure23 搜索结果原型图

对于许多功能，游客、注册用户和管理员的权限都有所不同。其中展示给游客和注册用户的页面基本是相同的，但是当游客使用需要用户权限的功能时，会提示其进行注册和登录。而管理员与这两种类型的用户的权限有所不同，如对于书籍信息的管理、财务报表的查看等，这些都不会展示给普通的用户。