作业四：软件体系结构设计

小组分工

包图（A-分层模式、B-管理过滤器模式） 2人

设计资产（C-分层模式包图的设计资产，D-管理过滤器模式包图的设计资产） 2人

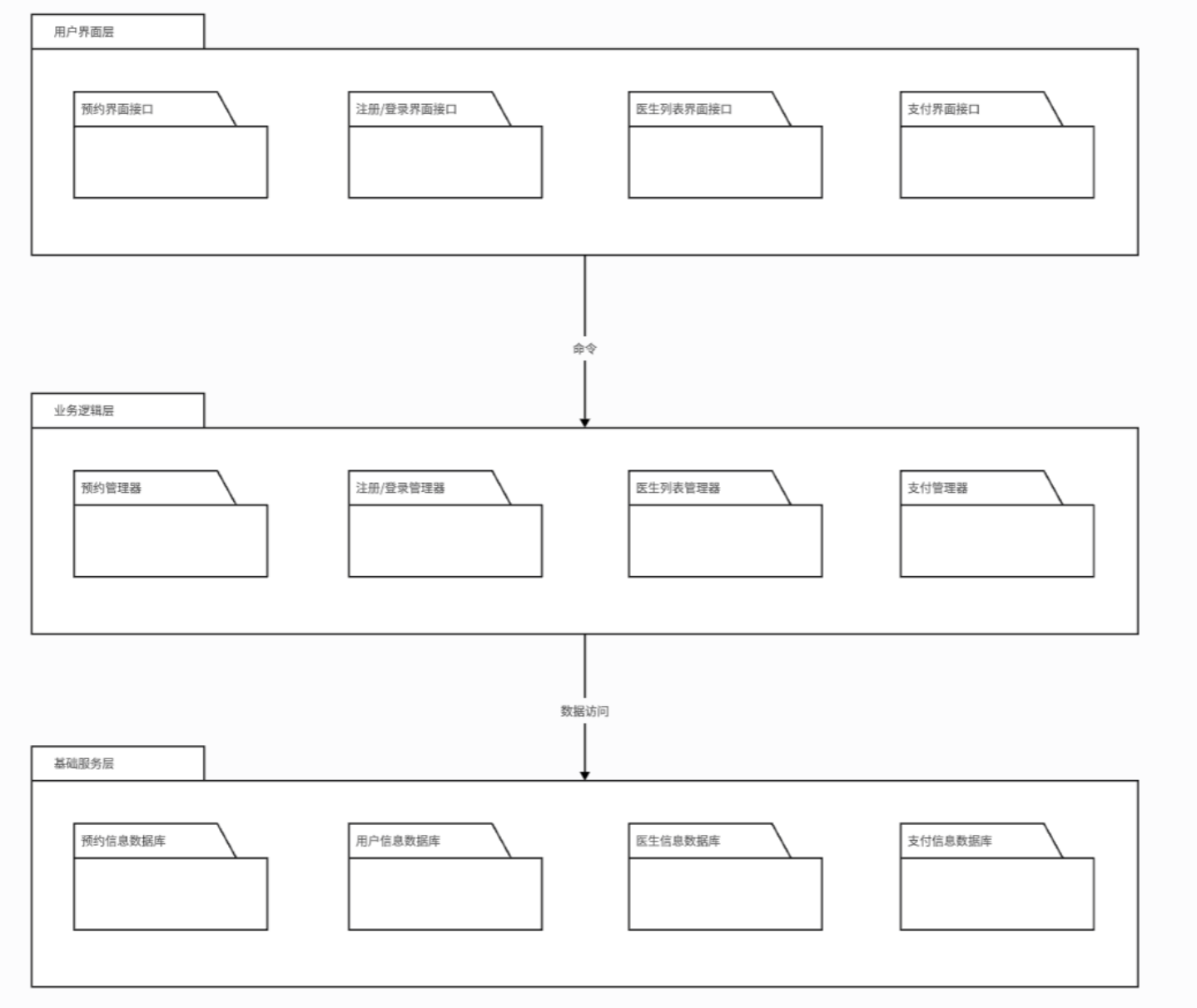
学号	姓名	分工内容	工作量
202233432302 9	张雅瑞	A-患者系统包图-分层模式	25%
202233286403 3	马召洋	B-管理者包图-管理过滤器模式	25%
202233287101 9	郭奇	C-患者系统包图的设计资产-分层模式-	25%
202233120113 1	孔令哲	D-管理者包图的设计资产-管理过滤器模式	25%

包图1-分层模式

撰写人：张雅瑞

分层模式的好处

1. **模块化**：每个层次都有明确的职责，便于理解和维护。
2. **可扩展性**：可以独立地对某一层进行修改或扩展，而不影响其他层次。
3. **可重用性**：业务逻辑层和数据访问层的组件可以在不同的用户界面层中重用。
4. **易于测试**：各层次可以独立测试，确保系统的各个部分都能正常工作。



分层

1. 用户界面层（Presentation Layer）

任务：负责与用户的交互，处理用户输入并显示相应的输出。用户界面层主要包括各种用户界面组件和界面逻辑。

主要模块：

- **预约界面接口**：处理预约医生和取消预约的用户界面。
- **注册/登录界面接口**：处理用户注册和登录的界面。
- **医生列表界面接口**：显示医生列表的界面。

- **支付界面接口**：处理支付预约费用的界面。

2. 业务逻辑层（Business Logic Layer）

任务：处理系统的核心业务逻辑和规则。业务逻辑层接收来自用户界面层的请求，执行相应的业务逻辑，然后与数据访问层交互以存取数据。

主要模块：

- **预约管理器**：处理预约医生和取消预约的业务逻辑。
- **注册/登录管理器**：处理用户的注册和登录逻辑。
- **医生列表管理器**：处理医生列表查询的业务逻辑。
- **支付管理器**：处理支付预约费用的业务逻辑。

3. 数据访问层（Data Access Layer）

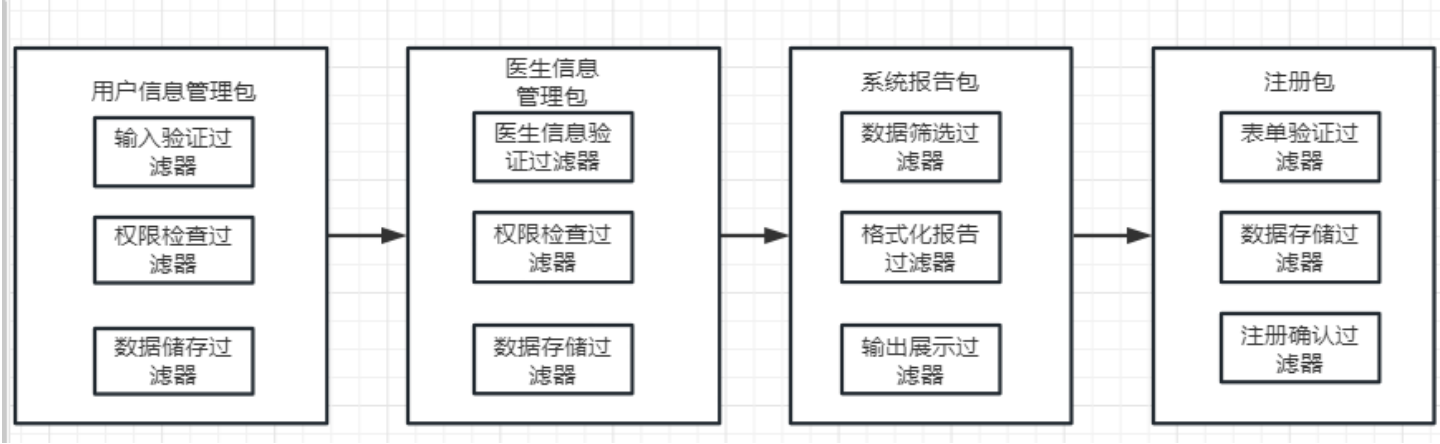
任务：负责与数据库或其他数据存储系统的交互。数据访问层提供数据的增删改查功能，并将数据传递给业务逻辑层。

主要模块：

- **预约信息数据库**：存储和管理预约信息的数据。
- **用户信息数据库**：存储和管理用户信息的数据。
- **医生信息数据库**：存储和管理医生信息的数据。
- **支付信息数据库**：存储和管理支付信息的数据。

包图2-管理过滤器模式

撰写人：马召洋



1.用户信息管理包

任务：完成对用户信息的管理，如完成新注册用户的信息植入数据库等。

主要模块：

- **输入验证过滤器**：负责对输入的数据进行基本的格式检查或规则验证。
- **权限检查过滤器**：验证当前操作是否符合权限要求。
- **数据存储过滤器**：将经过处理的数据存储到数据库中。

数据变换：

在用户信息管理包中，输入验证过滤器会接收用户输入，进行验证后，将数据传递给权限检查过滤器，之后由数据存储过滤器将数据保存。

2.医生信息管理包

任务：完成对医生信息的管理，如完成新注册的医生的信息植入数据库等。

主要模块：

- **输入验证过滤器**：负责对输入的数据进行基本的格式检查或规则验证。
- **权限检查过滤器**：验证当前操作是否符合权限要求。
- **数据存储过滤器**：将经过处理的数据存储到数据库中。

数据变换：

在医生信息管理包中，输入验证过滤器会接收用户输入，进行验证后，将数据传递给权限检查过滤器，之后由数据存储过滤器将数据保存。

3.系统报告包

任务：完成医生开处方的系统报告的传递等。

主要模块：

- **数据筛选过滤器**：根据不同条件筛选报告数据。
- **格式化报告过滤器**：对报告数据进行格式化，生成可视化结果。

- **输出展示过滤器**：负责将格式化后的报告数据展示给用户。

数据变换：

在系统报告包中，数据筛选过滤器根据条件筛选出相应的数据，然后交给格式化报告过滤器，格式化报告过滤器将数据进行格式化，生成可视化结果后交给输出展示过滤器，由输出展示过滤器展示给用户。

4.注册包

任务：完成医生或者患者新注册账号信息植入数据库等。

主要模块：

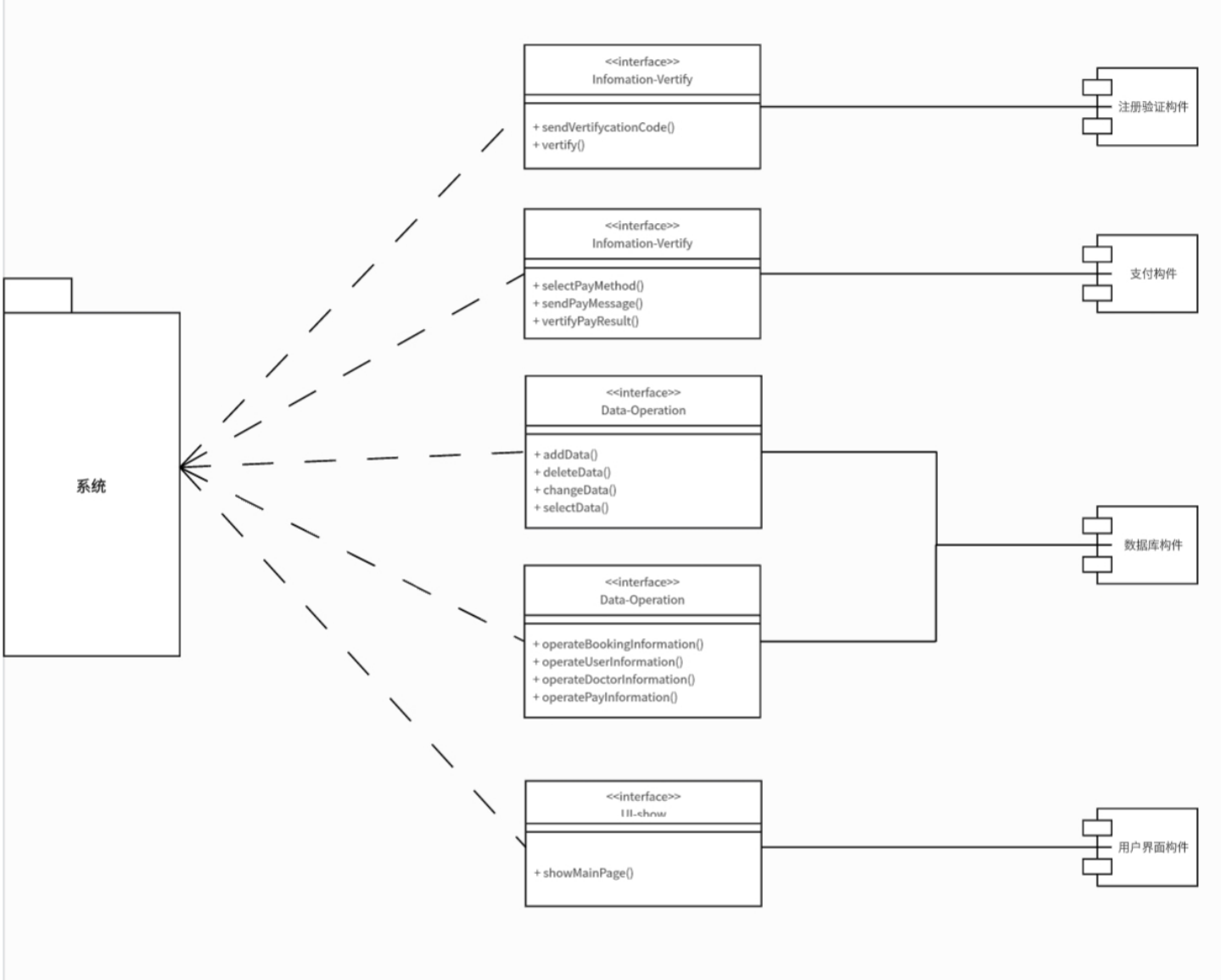
- **数据存储过滤器**：将经过处理的数据存储到数据库中。
- **表单验证过滤器**：将表单的信息进行审核和确认，发给指定人。
- **注册确认过滤器**：将数据库中新植入的信息再次与注册信息对比，审核是否正确植入。

数据变换：

在注册包中，表单验证过滤器对用户输入进行验证，验证通过后，数据被传递到数据存储过滤器保存，然后由注册确认过滤器生成注册确认信息。

设计资产1-分层模式的设计资产

撰写人:郭奇



系统目的：提供高效、可扩展的服务，通过组件化设计实现不同功能的分离和集成。
主要组件：注册验证构件、支付构件、数据库构件、用户界面构件。

1. 注册验证构件

接口名称：Information-Verify

功能描述：处理用户注册时的验证逻辑。

接口方法：

- sendVerificationCode(email: String): Boolean：发送验证码到用户邮箱。
- verifyRegistration(code: String, userInputCode: String): Boolean：验证用户输入的验证码是否正确。

2. 支付构件

接口名称：Information-Verify

功能描述：处理支付相关的验证逻辑。

接口方法：

- selectPayMethod(): String：选择支付方式。
- sendPayMessage(message: String): Boolean：发送支付消息。
- verifyPayResult(result: String): Boolean：验证支付结果。

3. 数据库构件

接口名称：Data-Operation

功能描述：处理数据库操作。

接口方法：

- addData(data: Object): Boolean：添加数据。
- deleteData(id: String): Boolean：删除数据。
- changeData(id: String, newData: Object): Boolean：修改数据。
- selectData(query: String): List<Object>：查询数据。

4. 用户界面构件

接口名称：UI-Show

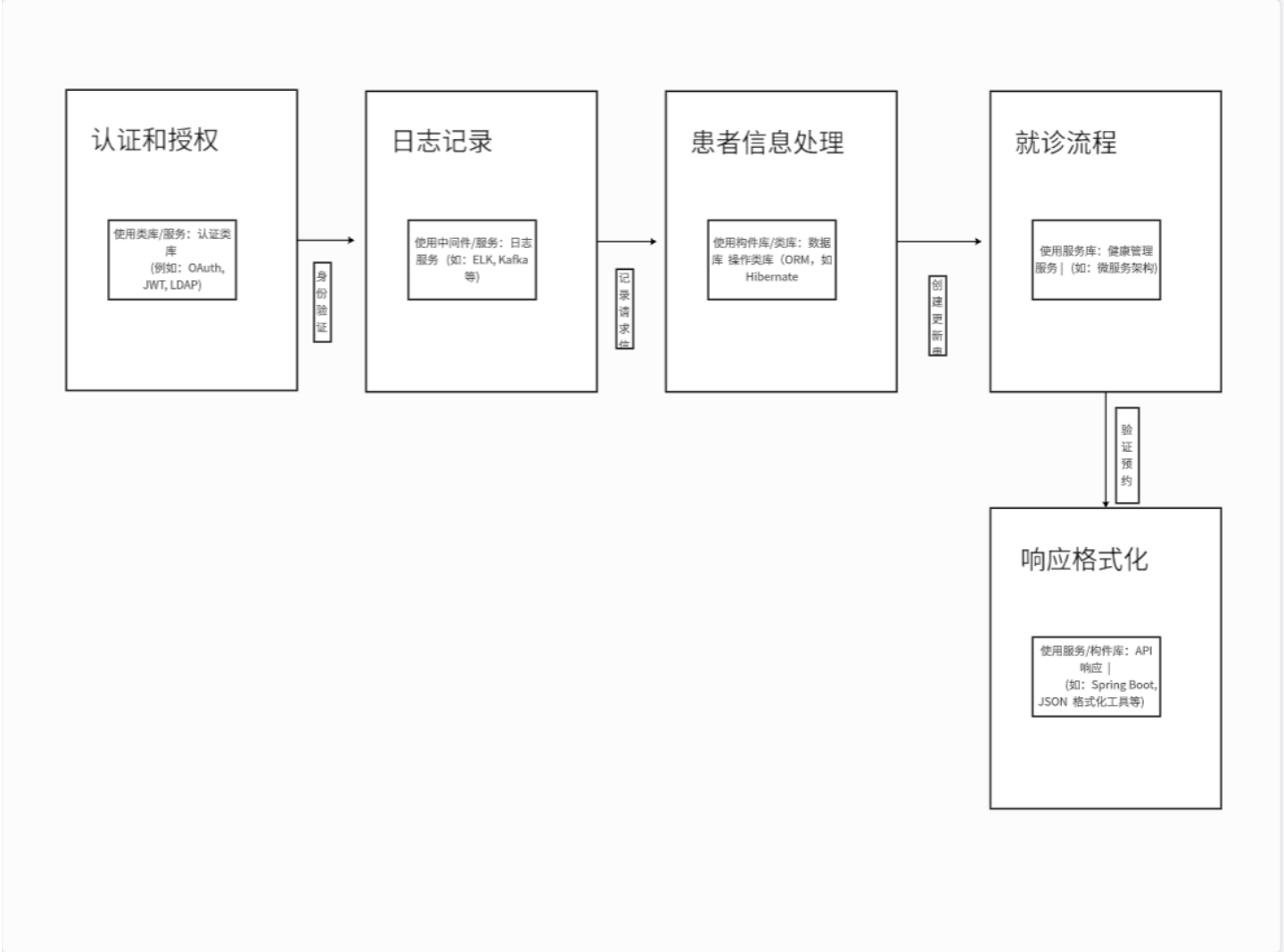
功能描述：处理用户界面的展示逻辑。

接口方法：

- showMainPage(): void：展示主页面。

设计资产2-管理过滤器模式的设计资产

撰写人:孔令哲



1. 构件库（Component Libraries）：

- **数据库操作类库（ORM）**：如Hibernate，处理数据库的增删改查操作。患者信息、预约等数据的存取都需要依赖这些库。
- **库存管理类库**：如使用消息队列（如Kafka或RabbitMQ）来管理药品库存变化的事件流。

2. 服务库（Service Libraries）：

- **健康管理服务**：就诊流程中可能包括对病人的健康信息的管理，这些服务通常会封装为微服务或通过类似Spring框架来处理。
- **预约服务**：为患者预约、挂号提供专门的服务，如调度系统、医生排班等。

3. 类库（Libraries）：

- **数据验证类库**：例如Java中的JSR-303 Validation API，用来验证输入数据是否符合要求。
- **认证类库**：如OAuth、JWT，处理用户的身份认证和授权。

4. 中间件（Middleware）：

    - **日志服务**：如ELK Stack（Elasticsearch, Logstash, Kibana）或Kafka等，用于记录和分析日志数据。
    - **消息队列中间件**：如RabbitMQ或Kafka，用于异步处理如库存更新、药品配送等操作。

过滤器：每个过滤器作为请求处理的中间层，调用相应的库或服务来完成其功能，确保系统的安全性、数据完整性和高效性。

体系结构特征：

- **模块化设计**：通过使用类库、服务库和构件库，将不同功能的代码进行分离，每个功能模块独立管理。
- **中间件支持**：通过引入中间件（如日志服务、消息队列等），支持高并发、大数据量的处理和分析。
- **可扩展性**：系统可通过添加新的服务库或更改构件库来进行扩展，适应未来需求的变化。

这种结构有效地将系统功能解耦，并通过设计资产提供了更高的复用性和维护性，便于进行后期扩展和优化。