**设计原则：**

1. **模块化**

模块化也被称作关注点分离，也就是把系统中各不相关的部分进行分离的原则，这样利于各部分能够进行独立研究。为了实现模块化，我们要辨析系统不相关的关注点来分解系统，并把他们放入不同的模块中。这样可以是运行速度大大加快，也可以使维护和开发变得非常便利。评估模块化有两个概念，

其一就是耦合度，我们在这方面做的比较好，就比如我们上面设计方法提到了，我们在这方面使用了parnas方法，实现了大部分情况下，各个模块之间的数据互不相关，各个模块之间也基本没有联系。

二就是内聚度。也就是指模块内部各个数据之间的“粘合”程度。模块内部粘合度越高，我们总体目标就越相关，整体性能就越好。为了实现高内聚度，我们在实现代码之前，先画出了数据库各个部分之间的联系，通过他们之间的联系，我们来设计模块，尽量的使模块内部之间的数据具有高内聚度的特性，减少独立数据。

评估：在实现模块化上，我们整体做的还是不错的。为了实现耦合度，我们采用parnas方法来实现高耦合度，但也不可避免的存在了模块之间具有数据互通的弊端，我们思考很久没有找到很好的解决方法，所以最后采用了自底向上的方法来解决这个问题。在内聚度方面，我们刚开始也是没想到很好的方法，后来想到了大二上的数据库结构课设。所以我们先画出数据库结构，根据各个部分之间的联系来设计模块，尽量减少模块之间的联系，并且加大模块内的内聚度。

**2．接口**

接口，这个概念很好理解，为系统其他部分定义了该软件单元的服务， 以及如何获取这些服务。它制定了操作性名称，参数和可能返回的值。当然， 为了数据的安全，我们只有在上下文环境符合一定条件时才被调用，一旦不 满足其中任意一个，都不会得预期的功能。也就是说软件单元的接口描述了 不同环境的需求和不同的服务。

评估：我们的接口实现的比较简单，指示在调用模块内，写入了所有可 能被调用的数据接口，这样从数据库中获取信息，或者向数据库中写入信息 的时候，只需要调用这些接口就可以了。为了保护数据的安全，防止数据库 数据错误导致崩溃，我们在调用的时候返回一个值，若调用错误会返回一个 固定的值，这时就不会得到预期的服务。

**信息隐藏**

概念：

对使用模块的用户隐藏模块实现相关的信息。模块内有哪些数据，函数是用何种逻辑实现功能的，这些信息全部对外隐藏。我们要让外部无法直接访问模块内的数据。模块的函数也尽量不要公开。用户只能通过最低限度公开的函数来操作模块。

优点：

将用户不必知晓的内部详细信息隐藏起来，可以减少接口的代码量，让信息交互变得更简洁，降低代码的复杂程度。从用户的角度来看，由于排除了多余信息的干扰，模块的使用方法变得更加简单。另外，公开的部分越少，模块内部的修改就越不容易影响到外部、这样可以将修改代码的范围控制到最小。

评估：本设计采用信息隐藏的方式，隐藏后端数据操作指令。外部无法直接访问模块内数据。

可能的解决方案：

实现信息隐藏需要使用封装的手段，分组可以整理“关系网”，降低隐藏信息的难度。

**增量式开发：**

评估：

从6.2.4来看，关于增量式开发的评估主要应该从使用图应是否具有树型结构或者是树型结构的森林方面来进行，实际实现过程中并未采用增量式开发的方法进行，而是采用了scrum这一基于增量式开发和迭代式开发的方法。

增量开发基于一个原则：先构建部分，在构建整体。我们避免到最后才冒出一个大的、爆发式的活动，集成所有组件和交付所有产品。相反，我们把产品分解成更小的特性，先构建一部分，再来做出调整，构建更多的特性。增量开发使我们能够适应开发工作并改变工作方式。

Scrum中增量开发缺点的解决方案：

增量开发中最大的缺点是逐步构建的过程中，有迷失全局的风险Scrum综合了迭代和增量两种开发模式的有点，消除了单独使用其中一种模式的缺点。Scrum使用一系列固定时长的适应性迭代来同时利用这两种方法的思想，这种迭代便是冲刺。Scrum并不是每次做一个阶段的工作，而是每次做一个特性 。这样一来，在冲刺结束的时候就可以创建一个有价值的产品增量。在收到冲刺结果的反馈后 ，我们可以进行调整。在接下来的冲刺中可以选择开发其他特性或是修改用于构建下一组特性的过程。

**抽象:**

采用分层次抽象，自顶向下、逐层细化的办法控制软件开发过程的复杂性。同java面向对象化的设计相似，抽象就是忽略一个主题中与当前目标无关的那些方面,以便更充分地注意与当前目标有关的方面。抽象并不打算了解全部问题,而只是选择其中的一部分,暂时不用部分细节。

评估：在本vue项目的开发中我们使用vue的组件这一功能，该功能体现了抽象化的精神。

Vue就是由一个一个的组件构成的。 一般来说，组件大致可以分为三类：页面级别的组件、业务上可复用的基础组件、与业务无关的独立组件。

我们主要用到了pages下的页面级别组件、components下的功能复用的基础组件以及utils下的与业务无关的组件。

**通用性：**

软件工程原则之一就是可复用性(reusability): 构造在将来软件产品中仍可使用的软件单元，旨在通过复用来分摊开发的成本。通用性是这样一种设计原则:在开发软件单元时，使它尽可能地能够成为通用的软件，来加强它在将来某个系统中能够被使用的可能性。

评估及方案：

通用性往往和需求规格不可兼容，在本项目中我们要对决策做出相应调整来适应特定用户的需要。

由于本项目课设需求如下：

1.系统中有多个电影院，系统管理员可以完成电影院的维护，系统管理员可以为每个电影院

指派1各电影院管理员;

2.电影院管理员定期维护本电影院即将.上映的电影信息;

4.网民可以根据时间、电影名称、电影院名称进行查询，查询到自己中意的电影后，注册的

网民可以在网上完成订票，并进行网上支付;

5.系统能够对指定时间、电影院、电影名字进行统计分析，以便分析出受欢迎的电影

这种客户化定制降低了软件的通用性，在通用性和客户化之间权衡后，我们选择一个更满足客户化而不是通用性。但是在一些后端基础功能vue组件上以及前端的登录pages等可以达到通用的目的。