铁路售票系统架构评审文档

**组长：张付俊**

**组员：黄东鹏、孙帅**

目录

[铁路售票系统架构评审文档 1](#_Toc497122151)

[0.引言 2](#_Toc497122152)

[编写目的： 2](#_Toc497122153)

[背景： 2](#_Toc497122154)

[定义： 2](#_Toc497122155)

[三层架构软件设计 2](#_Toc497122156)

[ATAM架构评审模式 2](#_Toc497122157)

[参考资料： 3](#_Toc497122158)

[第0阶段：合作关系及准备工作 3](#_Toc497122159)

[第1阶段：评估阶段 3](#_Toc497122160)

[项目产品立项表述： 3](#_Toc497122161)

[架构方法分类： 4](#_Toc497122162)

[架构表述： 5](#_Toc497122163)

[架构类图： 6](#_Toc497122164)

[生成质量属性效用树： 6](#_Toc497122165)

[初步分析架构方法： 7](#_Toc497122166)

[第2阶段：评估阶段 8](#_Toc497122167)

[集体讨论并确定场景的优先级： 8](#_Toc497122168)

[再次分析架构方法： 8](#_Toc497122169)

[三层结构选择 8](#_Toc497122170)

[LRU缓冲技术分析 9](#_Toc497122171)

[MD5加密存储分析 9](#_Toc497122172)

[改进架构类图 10](#_Toc497122173)

[结果表述 10](#_Toc497122174)

[第3阶段：后续阶段 11](#_Toc497122175)

[附录 11](#_Toc497122176)

[拟采用架构评审方法中的ATAM方法 11](#_Toc497122177)

# 0.引言

## 编写目的：

本文档的编写目的是对铁路售票系统架构设计进行简略的评审，为后继的详细项目设计等工作提供参考和依据，本文档主要描述的内容有：

* 表述
* 调查和分析
* 测试
* 形成报告

本文档的预期读者为：系统设计人员、测试人员、用户及其它有权限查阅本文档的相关人员。

## 背景：

* 系统名称：铁路售票系统
* 任务提出者：黄东鹏、张付俊、孙帅
* 开发者（承接单位）：开发小组
* 用户：网上订购铁路车票的人

## 定义：

### 三层架构软件设计

所谓三层[体系结构](https://baike.baidu.com/item/%E4%BD%93%E7%B3%BB%E7%BB%93%E6%9E%84)，是在[客户端](https://baike.baidu.com/item/%E5%AE%A2%E6%88%B7%E7%AB%AF)与数据库之间加入了一个中间件层，也叫组件层。这里所 说的三层体系，不是指物理上的三层，不是简单地放置三台机器就是三层体系结构，也 不仅仅有B/S应用才是三层体系结构，三层是指逻辑上的三层，即使这三个层放置到一 台机器上。 三层体系的应用程序将业务规则、数据访问、合法性校验等工作放到了中 间层进行处理。通常情况下，客户端不直接与数据库进行交互，而是通过COM/DCOM通 讯与中间层建立连接，再经由中间层与数据库进行交换。

### ATAM架构评审模式

Architecture Tradeoff Analysis Method(构架权衡分析方法）。他是评价软件构架的一种综合全面的方法。这种方法不仅可以揭示出构架满足特定质量目标的情况，而且（因为它认识到了构架决策会影响多个质量属性）可以使我们更清楚地认识到质量目标之间的联系——即如何权衡诸多质量目标。

## 参考资料：

Software Architecture in Practical（第三版）

# 第0阶段：合作关系及准备工作

此次对项目的评估方法经小组协商讨论是采用ATAM架构评估综合方法。待评估的项目系统为铁路售票系统。这是一个基于B/S的体系的常见应用，基于网络连接实现铁路票务的相关业务。对其进行架构评估主要有如下几个原因：

1. 在架构搭建的过程中一定会碰见许多一致或者未知的问题和困难，如果在核心功能模块或者架构本身的设计根本上出现缺陷，尽早的发现对于晚发现，甚至完成项目后才发现的综合成本要低得多；
2. 由于该架构面向多个用户多平台，因此要有足够的健壮性，稳定性，可拓展性以及可修改性；

3.由于该系统借助了网络的传播性，可以随时随地的对系统进行管理和维护，但是网络的泛滥使得网络环境总是充斥着有意无意的攻击，为了避免系统所部属的服务器沦为肉鸡的下场，或者内部数据被恶意破坏造成重大损失，所以系统应保证相对的安全性，使得入侵者所花费的入侵成本>入侵系统的获利成本或客户损失。

主要目标协商完成后，我们组成了一个虚拟的一人多角色的评估小组，成员列表如下：

|  |  |
| --- | --- |
| 成员 | 角色 |
| 黄东鹏 | 评估小组负责人、评估总结者、提问者、场景书记员、时间管理者 |
| 张付俊 | 评估负责人、提问者、架构设计师、提问者、进展书记员、数据收集人、提问者、领域专家、资料员 |
| 孙帅 | 时间管理者、提问者、场景书记员、资料员 |

表1：评估小组成员列表

# 第1阶段：评估阶段

## 项目产品立项表述：

随着现代交通的发展，在基于经济以及便利的考虑基础上，铁路出行成为广大人民首选的性价比最高的交通方式。但随着经济的发展，人工售票逐渐不能满足庞大人口数量的基本购票需求。随着互联网的发展，网络购票的普及解决了这个主要矛盾。

## 架构方法分类：

进行了架构表述后，评估小组列出他们曾听到的架构方法，以及那些在对文档进行评估前的评审中所了解到的方法：

1. 分层架构

这种架构将软件分成若干个水平曾，每一层都有清晰的角色和分工，不需要知道其他层的细节，层与层之间通过接口通信。

1. 事件驱动架构

事件是状态发生变化时，软件发出的通知。事件驱动架构就是通过事件进行通信的软件架构。分为：事件队列、分发器、事件通道、事件处理器。

1. 微核架构

又称为“插件架构”，指的是软件的内核相对较小，只要功能和业务逻辑都通过插件实现。

内核通常只包含系统运行的最小功能。插件则是相互独立的，插件之间的通信，应该减少到最低，避免出现相互依赖的问题。

1. 微服务架构

是服务导向的架构的升级。

每一个服务都是一个独立的部署单元。这些单元都是分布式的，互相解耦，通过远程通信协议联系。

1. 云架构

云架构主要解决扩展性和并发问题，是最容易扩展的架构。

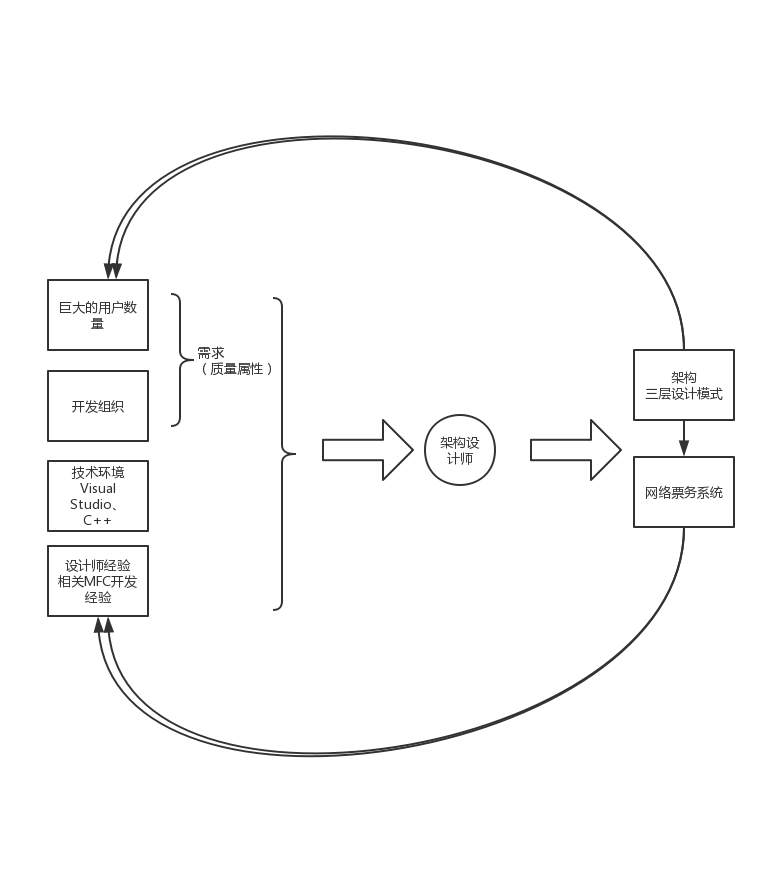
它的高扩展，主要原因是没使用中央数据库，而是把数据都复制到内存中，变成可复制的内存数据单元。然后，业务处理能力封装成一个个单元。比如访问量增加，就新建单元处理；访问量减少，就关闭但处理单元。由于没有中央数据库，所以扩展性的最大瓶颈消失了。由于每个处理单元的数据都在内存里，最好要进行数据持久化。

这个模式主要分成两部分：处理单元和虚拟中间件。

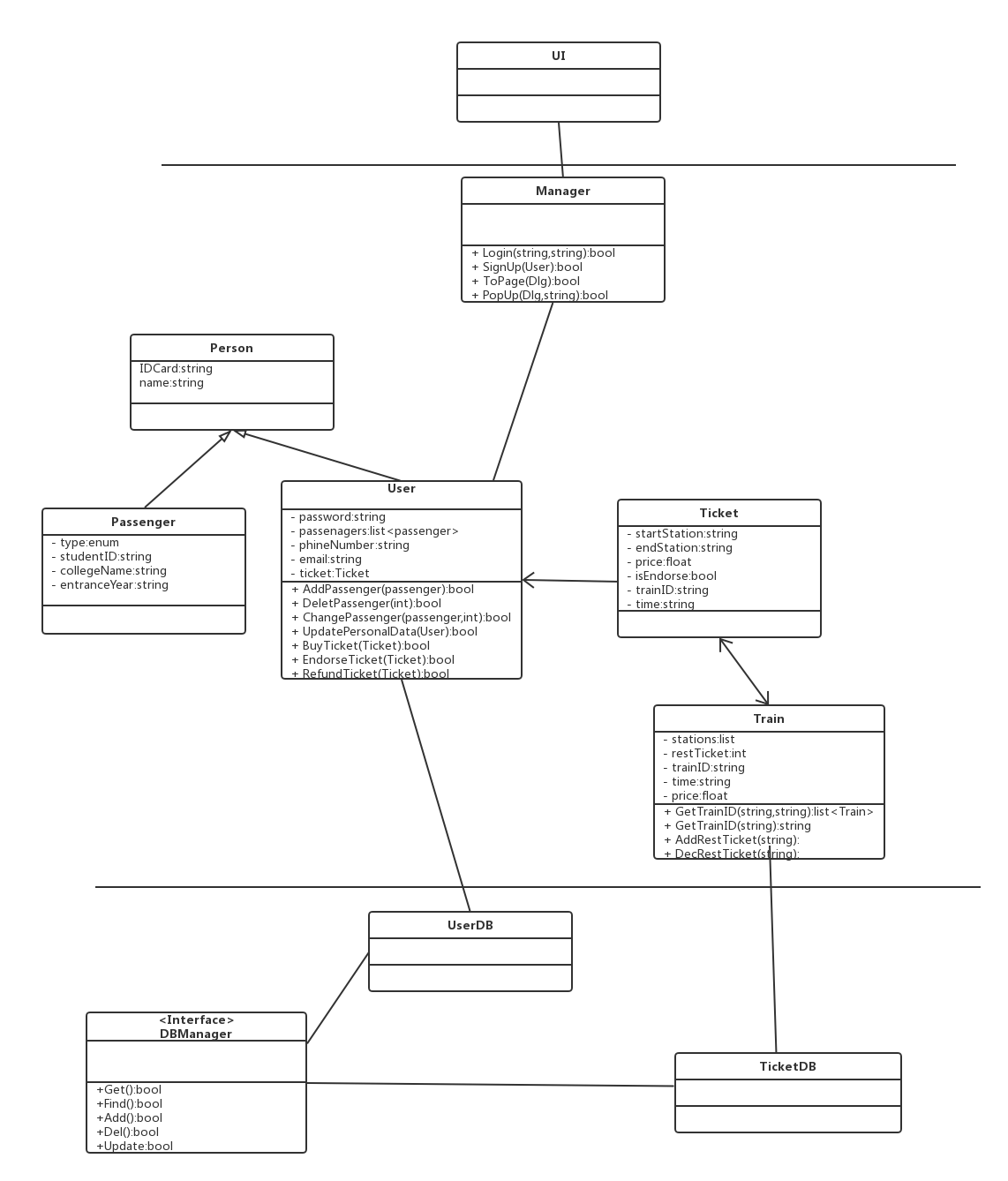
## 架构表述：

[软件架构](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E6%9E%B6%E6%9E%84" \t "_blank)是一系列相关的抽象模式，用于指导大型[软件系统](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E7%B3%BB%E7%BB%9F)各个方面的设计。软件架构是一个系统的[草图](https://baike.baidu.com/item/%E8%8D%89%E5%9B%BE)。软件[体系结构](https://baike.baidu.com/item/%E4%BD%93%E7%B3%BB%E7%BB%93%E6%9E%84)是构建[计算机软件](https://baike.baidu.com/item/%E8%AE%A1%E7%AE%97%E6%9C%BA%E8%BD%AF%E4%BB%B6)实践的基础。

考虑到票务系统的特点，将使用三层结构进行系统的架构。



### 架构类图：



## 生成质量属性效用树：

下表给出了在对铁路售票系统评估期间生成的质量属性效用树，有几个质量属性求精没有与之相关的场景。这种情况经常出现，这并不是问题，对于某个质量属性，人们有时能够想出一个合理的求精，但当让他们在自己的系统的上下文中对该质量属性的用例进行说明时，却发现该求精实际上并不适用。

表 2：对铁路售票系统进行ATAM评估的效用树表格

|  |  |  |
| --- | --- | --- |
| **质量属性** | **属性求精** | **场景** |
| 性能 | 最大负载  响应时间  吞吐量 | 当开票时，用户量剧增，能够同时负载至少500的用户同时访问（H,H）  用户输入数据后在网络畅通的情况下应能在s内给出相关信息（H,M）  日最高订票量500万张（按目前网络订票系统工作18小时算，每秒处理订单量为78张）（H,H） |
| 安全性 | 数据存储  注册验证  登录验证  密码强度 | 当用户注册时，系统将用户的信息加密后存入数据库；当用户登录时，系统将数据加密后再与数据库的内容进行比较，防止传输过程被窃取泄露。（H,L）  当用户注册时,为确认为真人操作，存在手机信息验证码验证，并且60s内不允许重复获取验证码。（H,L）  当用户登录时，连续输入两次错误密码后，再次登录需要根据系统给出图片验证码输入正确的验证码才能完成登录，图片验证内容随机生成，并且随机生成条纹遮挡字符，防止机器验证。（H,L）  当用户注册时系统根据用户的密码显示相应的密码强度以提示用户增强密码强度。密码强度根据密码内容字符类型以及长度确定。为了确保基本的安全性以及防止用户遗忘密码，用户密码长度范围限制为6-16位。（H,L） |
| 易用性 |  | 用户通过输入简单的查询信息就能够得到对应的相关数据并让用户轻易完成购买（H,M） |
| 兼容性 | 多系统支持 | 在相同平台的不同系统上也能够正常运行（H,H） |
| 可维护性 | 管理员功能  维护管理  系统自动报错 | 管理员能够在铁路信息、用户信息需要更新时进行及时更新，并同步数据给用户（H,M）  当出现了不可避免的错误时，可以及时进行维护修复（H,M）  可以定位出系统报错内容、报错位置（M,L） |
| 可扩展性 | 添加新功能 | 在出现新需求时能够添加新功能，如支付渠道的增加支付渠道的选择（M,L） |
| 可扩充性 | 功能业务的子模块 | 随着发展和意见的收集，能够根据情况添加新的业务功能，如外卖预定（M,M） |
| 可靠性 | 不易出错 | 在程序的使用过程中出错概率要尽量小，出错了要能够及时修复（H,H） |

表中的场景给出了决策者所分配的优先级。在每一对有序的字母中，第一个代表能力的重要性，第二个代表设计师对实现该质量属性的难度的估计。

我们需要注意到，一些场景已经很完备了，具备了刺激、环境和响应三个部分；一些场景没有刺激，还有一些场景没有响应。在这个阶段，只要涉众能够理解场景的含义，不明确的场景说明是允许的。如果所选择的场景用于进行分析，那么该场景中的刺激和响应必须得到足够的明确。

## 初步分析架构方法：

评估小组首先分析最重要而且最难实现的场景，每次分析一个最高优先级的场景，同时我们的设计师详细地解释了构架如何支持每个场景。小组成员探查设计师用来实现场景的架构方法，把相关架构决策编成文档，一共确定了6个敏感点，4个权衡点，3个风险点以及2个机遇点。分别如下：

敏感点：各种人员对数据库的操作等级，网上支付的安全性衔接，系统安全性，查询便捷性多样性，用户支付的便捷程度，用户支付的多样性

权衡点：查询结果的响应与查询的便捷性多样性，系统的安全性与管理员的监管难度，网上支付的安全性衔接与用户支付的便捷程度，防止恶意抢票和普通用户的购票难度

风险点：网上支付的安全性衔接，系统安全性，查询结果的响应时间

机遇点：查询购买的便捷性，系统安全性

# 第2阶段：评估阶段

## 集体讨论并确定场景的优先级：

下表给出了在本步骤中提出的某些部分感兴趣场景进行重点分析。按重点次序罗列，由于篇幅有限，有些细微场景没有列出，只列出了认为重要场景。

表 3：集体讨论确定的场景

|  |  |
| --- | --- |
| **场景号** | **场景** |
| 1 | 为了防止软件抢票，应在对一个相同的用户的多次请求进行分析。如：验证码两次输入错误之间的之间间隔。当判定不是软件抢票时，应当随着验证错误次数的增加降低验证难度 |
| 2 | 用户账户与付款的绑定、多种支付方式、以及安全性的要求。在保证安全性的前提下应该能够让用户通过最简洁的流程选择自己合适的支付流程。 |
| 3 | 突然激增的流量导致服务器处理缓慢，甚至崩溃异常，要求对有害信息进行过滤，使用LRU缓冲计数减少服务器负担，增加服务器工作效率。 |
| 4 | 用户访问的相关车次查询、买票、下订单、付款、改签、退款的流程应直观、简洁。 |
| 5 | 支持新车次添加、旧车次删除、用户信息修改操作等。 |
| 6 | 如果存在多个管理员时怎么并发管理系统，如果多个管理员对同一数据进行修改时应如何保护数据不被多次修改。这里考虑到参考锁机制。 |

## 再次分析架构方法：

### 三层结构选择

由于票务数据以及用户数据量庞大，而三层结构的特点将数据层、逻辑层以及表现层分隔开，在开发上降低了复杂度。并且考虑到系统的开发效率，三层结构使项目结构更清楚，分工更明确。

考虑到用户数量多，并且票务信息随时都有可能发生变化，使用三层结构有利于后期的维护、更新和升级。并且数据的差异性大，将三层架构中的控制层进行了细分，实现低耦合、并行开发。

为解决访问流量大的问题，服务器考虑使用LRU缓冲计数以提高服务器工作效率和改善用户体验。并且出于对大量用户信息安全的角度考虑，数据的传输和校验采用MD5加密方式。

在集中的讨论中，我们重点讨论了上述问题的解决方案，考虑到时间和空间的限制，我们就第一二个详细展开说明，以及分析和权衡。

### LRU缓冲技术分析

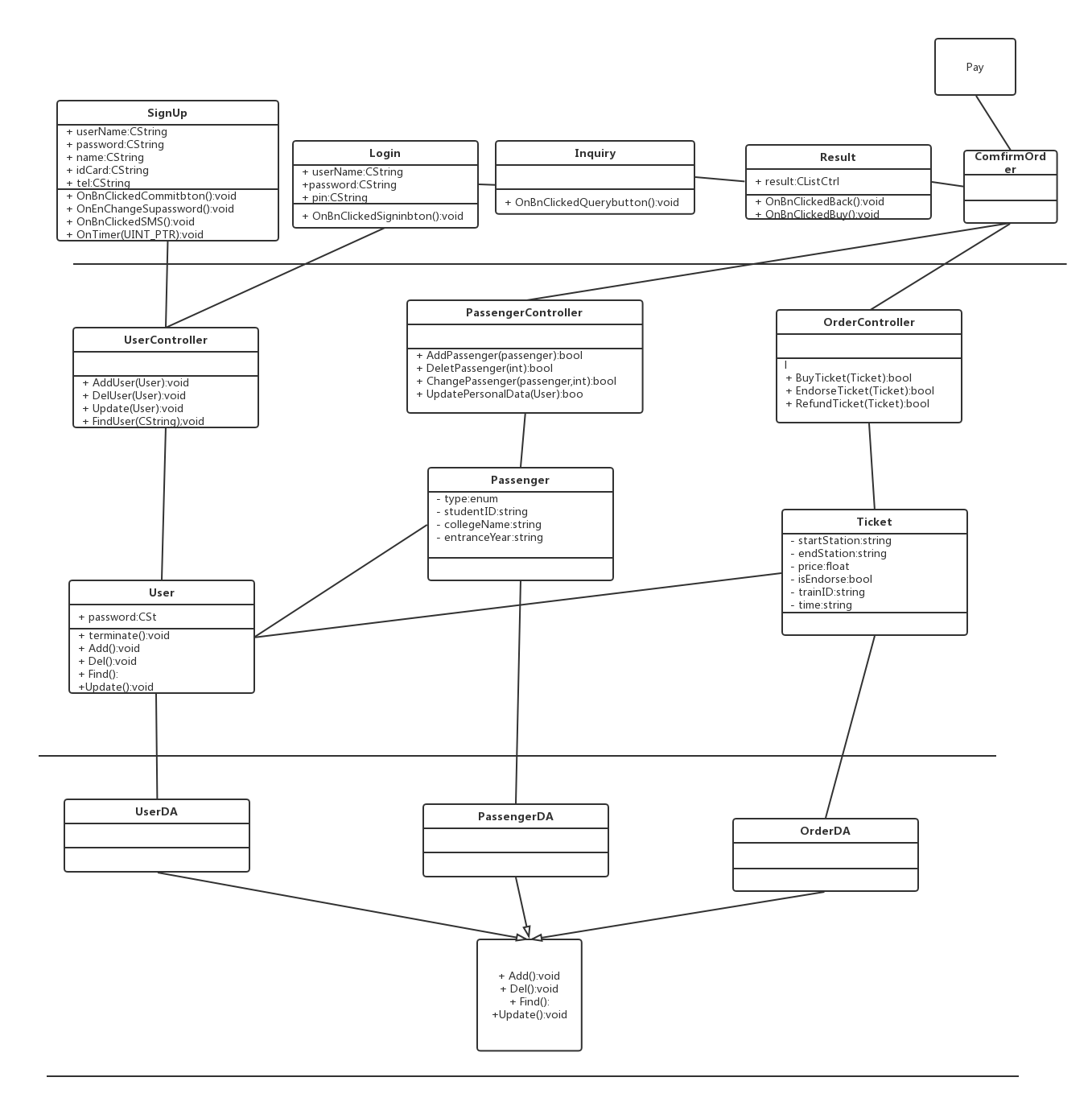
比如说一些系统登录的操作，不可能每次你访问系统都去调用数据库的东西，如果能划出一些空间来，比如说500M，用来缓存这些东西，这样用户访问的时候先在缓存里找，找不到，再去访问数据库，同时把被访问的内容放到缓存里面（我们可以假设这些东西还会经常被访问）。然而，我们分配用来做缓存（Cache）的空间肯定是有限的，总不可能从数据库读的东西全部放到缓存里，所以，当缓存里的内容达到上限值的时候，我们就要把最少使用的东西写回数据库，再将新的访问内容从数据库暂存到缓存里面。以此保证服务端的效率和使用度。

### MD5加密存储分析

MD5将任意长度的“字节串”变换成一个128bit的大整数，并且它是一个不可逆的字符串变换算法，换句话说就是，即使你看到源程序和算法描述，也无法将一个MD5的值变换回原始的字符串，从数学原理上说，是因为原始的字符串有无穷多个，这有点象不存在反函数的数学函数。

将客户端的内容数据进行加密后传输到数据库进行校验，这样可以防止在传输过程中数据被盗取的安全性问题。

### 改进架构类图



## 结果表述

1. 由于开发人员对架构缺乏一定的熟练度，我们决定边学边做，碰见一些解决不了的问题采用在文档中总结报告，记录下来，优先完成功能模块的实现环节；
2. 平台开发入门槛较低，产品容易被模仿，需要及时更新设计以摆脱竞争对手，所以应该预留系统API接口，为不管是以后管理方设计更改界面还是可以由用户自定义开发界面都能起到良好的促进作用。

## 第3阶段：后续阶段

ATAM评估的一个具体结果就是生成了最终报告，该报告包括一个有风险决策、误风险决策、敏感点和权衡点的列表。还包括一个涉及如下内容的目录：所使用的架构方法、效用树、经过集体讨论确定的场景以及所选择的每个场景的分析记录。最后，最终报告还包括由该评估小组所确定的风险主题的集合，并指出了每个风险主题所危及的商业动机。

# 附录

## 拟采用架构评审方法中的ATAM方法

对ATAM模型方法的简略描述：

软件构架的评估方法：SAAM和ATAM。这里只详细说明ATAM方法。

ATAM一种进行构架评估的综合方法，ATAM是评估软件构架的一个健壮的方法。在该方法中，项目决策者和涉众要清晰地阐述一个准确的质量属性需求列表（以场景的方式），并说明与实现每个高优先场景相关的构架决策。然后，把这些决策确定为有风险决策或无风险决策，以找到构架中任何存在问题的地方。

ATAM不是需求评估。ATAM不是代码评估。ATAM不包括实际的系统测试。ATAM不是一个准确的手段，但它识别了构架中可能存在风险的区域。这些风险包含在敏感点和权衡中。

ATAM活动的4个阶段：

在第0阶段（合作关系和准备）确定细节：人员名单，时间，地点；评估小组获取资料并进行初步了解分析。

第1阶段，评估阶段，决策者参与，小组开始信息收集与分析；耗时约1周。1～2周中断期，评估小组进一步以非正式方式了解构架。

第2阶段，评估阶段，涉众参与，分析继续；约2天。

第3阶段，后续阶段，生成最终报告，进行评估活动总结；1周。

评估阶段的步骤：

第1步：ATAM方法的表述。评估负责人向决策者表述ATAM方法，使大家理解其过程，了解角色布局。

第2步：商业动机的表述。决策者介绍系统商业动机、重要功能、各种限制（任何相关的技术、管理、经济和政治限制）、商业目标和上下文、主要的涉众、驱动因素等。

第3步：构架的表述。首席设计师或架构小组介绍构架，技术限制、所用模式等。

第4步：对构架方法进行分类。评估小组利用所有已知信息对构架方法进行分类。

第5步：生成质量属性效用树。生成质量属性效用树，捕获详细的需求信息，为每个场景分配一个级别，如（高，中），前者为重要度，后者为实现难易度，重点放在（高，高）的场景；此处场景具备刺激、环境、响应三要素就可以了。

第6步：分析构架方法。评估小组分析所有重要场景，设计师解释如何支持该场景，检查所用构架方法，分析风险点、权衡点、敏感点。

经过一段中断期，第2阶段开始，此时涉众开始参与；首先仍然需要一个对ATAM方法的介绍，并使涉众了解已有的成果。

第7步：集体讨论并确定场景的优先级。集体讨论并分析场景的优先级，以了解更广泛的涉众的想法；该过程可能产生新的场景；使用“有限票数法”投票确定每个场景的优先级——此处不考虑实现难度。

第8步：分析构架方法。分析新的高优先级的场景，构架师解释构架是怎么满足各场景的。

第9步：结果表述。总结评估结果，评估负责人展示该结果。