# 三层架构

摘要【300-320，不要超过330】：

2023年11月份，我所在公司承担了某银行“智能终端管理系统”的开发工作，我作为该项目的系统架构设计师，负责该系统的需求分析与架构设计工作。该系统**主要提升银行对智能收银终端软硬件的管理及维护能力**，业务模块包括应用APP管理、设备管理、远程协助以及屏媒广告等多个模块。本文以该平台为例，论述系统三层架构在该项目中的具体应用。表现层，包括Web管理后台和Android客户端，前者采用Vue、iView等前端技术，构建友好的用户操作界面；业务逻辑层，负责处理系统的核心业务，采用SpringBoot框架，将不同业务功能进行模块化，按功能模块进行开发和维护；数据层，采用Mybatis作为持久层框架，数据库选择MySql，引入数据库读写分离技术提升访问效率。整个项目历时11个月开发完成并上线，到目前运行稳定。实践证明，三层架构的使用极大提升了开发效率，减少了开发成本，也为后续的持续优化奠定了基础。

正文【2200-2500】：

随着互联网技术的飞速发展，传统的收银设备操作系统简单，功能单一，无法远程升级（OTA），设备的状态也无从得知，所以智能设备替换传统收银设备是一个趋势。为了增加商户的粘性，提升服务质量，各大银行开始加大智能收银终端的投放。不同于传统的收银设备，智能设备联网能力强，操作系统强大，能够实现更多更复杂的业务。为了充分发挥智能设备的优势，**并进一步提高银行对设备的监管**，行方提出建设一个智能终端管理系统的需求，实时监控设备状态，并对设备的软硬件生命周期进行有效的管理。

2023年11月份，我司承担了某银行的智能终端管理系统的开发工作，我作为该项目的系统架构设计师，参与整个开发过程。该系统的主要业务模块包括应用APP管理、设备管理、远程协助以及屏媒广告等。建设该**系统旨在加大银行对设备的管理，提升对商户设备的监管能力，以及对设备的维护效率**。应用APP管理模块，主要包括应用上架，应用管理等功能。银行或商户可定制专属应用市场，上传和发布自己的APP。银行作为管理方，还可以针对不同业态的商户配置应用的可见性。设备管理模块，包括设备总览、定制设备以及远程更新功能。银行通过平台的可视化界面，能够清晰、实时的关注到设备当前的软件使用情况和硬件状态。另外，银行可通过平台配置装机初始化任务，设备首次激活时，会联网获取装机任务，包括定制开机动画、应用预装以及系统配置等。远程协助模块，主要包括文件传送接送，录屏以及远程控制等功能。针对软件常见的BUG问题，在商家授权后，通过该平台的远程协助功能，可远程操控设备，获取设备日志，便于软件开发人员排查问题。屏媒广告模块，主要包括媒体资源管理、广告数据统计等功能。银行或商户可通过平台上传广告资源，如优惠活动宣传等，并分析广告播放的数据统计。

由于银行系统对安全性、可靠性、可用性以及响应速度要求很高，我选择了三层架构作为该系统的软件体系结构。三层架构作为一种经典的系统架构设计模式，应用广泛，各层次间职责分离，相对独立，便于系统的维护和扩展。下面我将分层次详细介绍三层架构在该系统的应用过程。

首先，表现层负责为用户提供直观的操作界面，包括Web管理后台以及Android客户端。Web管理平台采用Vue、iView等前端技术进行开发设计。Vue是一个轻量级的框架，易于上手，项目组中的新人能够快速掌握。另外，它采用虚拟DOM和组件化的方式来提高渲染效率，性能较高，表现层和用户直接交互，流畅的操作体验非常重要，所以我最终选择Vue作为前端技术。iView是一个开源的高质量UI组件库，它提供了覆盖导航、布局、数据输入等多个方面的组件，组件库丰富，几乎覆盖了所有常见web应用场景。同时，它的API设计非常友好，开发者能够快熟上手，所以我选择了它作为前端UI组件库。

其次，业务逻辑层是系统的核心部分，采用主流的SpringBoot框架进行开发。使得开发人员可以更专注于表示逻辑和业务逻辑的开发工作。通过框架的使用，缩短了开发周期，减少了开发费用和维护费用，提高了开发的成功率。为了实现智能管理系统和Android端应用的长连接及消息推送场景，我们引入了MQTT消息中间件。通过MQTT，系统能够将银行或者商户的配置，实时推送到设备，这非常重要，比如远程更新任务。当系统修复了一个重大问题时，需要对所有设备进行强制升级，稳定可靠的连接是实现这些的基础前提。同时考虑业务逻辑层对系统性能要求比较高，为了避免频繁访问数据库，我们增加了Redis缓存机制，将系统配置、设备信息等热点数据缓存到redis，从而提高系统性能和访问效率。另外，为了增强系统的可用性，防止过多的请求对系统造成过载，保护后端资源，我们还实现了请求限制，当客户端短时间内频繁请求超过设定的阈值时，服务器会拒绝该请求并返回错误。

最后，数据层选择Mybatis作为持久层框架，MySQL作为数据库。Mybatis简单易学，灵活并且解除了sql语句与程序代码的耦合，通过DAO层，将业务逻辑和数据访问逻辑分离，使得系统的设计更加合理、清晰，并具备更好的维护性。为了应对高并发数据访问可能导致的数据库压力过大，我们使用数据库读写分离技术，主数据库负责写操作，从数据库负责读操作，减轻主库的压力。另外，为了提升设备信息、应用APP信息的查询速度，我们对SQL语句和索引进行优化，并使用批量操作的SQL语句减少数据库的访问次数。例如在更新设备状态或应用信息时，将多条语句合并成一条批量更新语句。

整个项目历时11个月开发完成，并于2024年10月顺利上线，到目前运行稳定，市场反馈良好。尽管该架构带来了许多优势，但实际应用中仍然存在一些不足之处。系统目前使用的缓存策略是针对热点数据进行缓存，但在某些场景下，缓存的命中率仍然偏低，导致频繁访问数据库。针对这个问题，一方面可以通过延长热点数据的过期时间，另一方面针对未命中的数据，在缓存中存储一个空值，并设置一个较短的过期时间，避免同样的请求频繁打到数据库，减少数据库的压力。

实践证明，通过基于三层架构设计的智能终端管理平台，不仅降低了开发成本，提高了团队开发效率，同时，系统在大规模设备管理中展现出了良好的稳定性。通过表示层、业务逻辑层和数据访问层的分工和协作，实现了系统的高效、灵活和安全，有效提升了银行智能终端管理效率，减少了设备维护成本，从项目完工后的使用效果来看，达到了预期目的，也获得了行方的一致好评。

最后，经过这次参与系统三层架构的设计实施后，我也看到了自己身上的不足之处，在未来还会不断地更新知识，完善本系统在各方面的设计，我们相信能够更好地满足不断变化的业务需求，并提升系统的整体性能和稳定性。在为公司创造价值的同时，也能够为客户开发出更稳定可靠的系统。