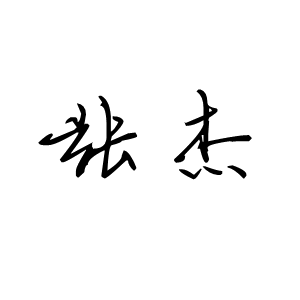
毕业设计（论文）原创性声明

本人郑重声明：所提交的毕业设计（论文），是本人在导师指导下，独立进行研究工作所取得的成果。除文中已注明引用的内容外，本毕业设计（论文）不包含任何其他个人或集体已经发表或撰写过的作品成果。对本研究做出过重要贡献的个人和集体，均已在文中以明确方式标明并表示了谢意。

作者签名：

日期： 2022 年 4 月 20 日

**基于二维码的展品介绍系统设计**

张杰，计算机与信息学院

摘 要：随着博物馆和科技馆在人们生活中的普及，展品介绍系统也在不断迭代升级。在移动互联网和二维码大规模应用的当下，将展品介绍与二维码结合设计一个面向移动终端的交互式App 具有重要意义。主要优势体现在，对于个人用户而言，不再拘泥于传统的人工讲解，交互方式变得更加便捷和智能化，节省了经济成本和时间成本。另一方面，对展品的宣传和推广而言也具有促进作用。

贯穿本文始终的是软件工程设计思想，先从市面上流行的展品介绍系统设计的发展现状入手，结合其背景意义和符合实际情况的需求分析来设计和完善整个系统。客户端用 Flutter 框架构建跨平台App，服务端业务采用 Go 语言开发并部署至 Docker 容器上，使用gRPC协议交换数据，并用Redis为MySQL做热点数据缓存。这些服务和技术是构建整套高可用系统的重要支撑。

关键词: 二维码；跨平台；微服务

**A Design of exhibit introduction system based on QR code**

Zhang Jie, School of Computer and Information

**Abstract:** With the popularity of museums and science and technology museums in people's lives, exhibit introduction systems are being iterated and upgraded. With the massive application of mobile internet and QR code, it is important to design an interactive app for mobile terminal by combining exhibit introduction with QR code. The main advantage is that, for individual users, they are no longer bound to the traditional manual explanation, and the interaction method becomes more convenient and intelligent, saving economic and time costs. On the other hand, it can also promote the publicity and promotion of exhibits.

Throughout this paper is the software engineering design ideas, first from the market popular exhibits to introduce the development of the system design status quo, combined with its background meaning and in line with the actual situation of the demand analysis to design and improve the whole system. The client uses Flutter framework to build cross-platform App. The server business is developed and deployed to Docker container using Go language, uses gRPC protocol to exchange data, and uses Redis to cache hot data for MySQL. These services and technologies are important support for building a full set of high availability systems.

**Keywords:** QR codes; cross-platform; container technology

**目 录**

[第1章 引言 1](#_Toc101037886)

[1.1 项目背景与意义 1](#_Toc101037887)

[1.2 研究现状 1](#_Toc101037888)

[1.3 研究内容 3](#_Toc101037889)

[第2章 相关技术简介 4](#_Toc101037890)

[2.1 项目架构 4](#_Toc101037891)

[2.2 C/S架构 5](#_Toc101037892)

[2.3 gRPC通信 5](#_Toc101037893)

[2.4 Flutter框架 5](#_Toc101037894)

[2.5 Redis缓存 6](#_Toc101037895)

[2.6 MySQL数据库 6](#_Toc101037896)

[2.7 MeiliSearch搜索引擎 6](#_Toc101037897)

[2.8 Docker容器化 6](#_Toc101037898)

[第3章 二维码展品介绍系统需求分析 7](#_Toc101037899)

[3.1 功能需求 7](#_Toc101037900)

[3.2 性能需求 7](#_Toc101037901)

[3.3 安全需求 7](#_Toc101037902)

[3.4 运维需求 8](#_Toc101037903)

[第4章 二维码展品介绍系统的功能描述与设计分析 9](#_Toc101037904)

[4.1 系统设计目标 9](#_Toc101037905)

[4.2 系统结构 9](#_Toc101037906)

[4.3 服务端结构 10](#_Toc101037907)

[第5章 二维码展品介绍系统 14](#_Toc101037908)

[5.1 移动端实现 14](#_Toc101037909)

[5.2 服务端实现 18](#_Toc101037910)

[5.3 性能测试 19](#_Toc101037911)

[第6章 总结 20](#_Toc101037912)

[参考文献 21](#_Toc101037913)

[致谢 22](#_Toc101037914)

# 第1章 引言

## 1.1 项目背景与意义

数字化生活影响了人们的方方面面，移动网络的高速发展是推动这种现象产生的重要原因之一。使用二维码进行支付交易在中国已成为经常的事情。事实上，二维码作为信息的载体，可以提供更加广泛的服务接口而不仅限于移动支付。例如，在一些诸如博物馆等旅游景点，往往需要专业的导游进行引领与讲解，但是导游仅能进行文字性描述，而不能进行视觉再现。为此，基于二维码进行个性化的声视觉呈现，可以有效提升游客的旅游体验。在每个展品周围附上其专属二维码，用户可在有网络的环境下自由地选择了解他们感兴趣的展品。基于二维码技术，以移动网络为载体设计一个展品介绍系统具有重要的意义。

伴随互联网发展一道出现的是新媒体， 展品介绍系统也部分承担了这样的职责，为人们提供多姿多彩的知识服务，这对于丰富他们的闲暇时光并且提高精神素养具有积极影响。然而，设计这一系统存在诸多挑战与未知的风险，设计者和工程师需要解决性能问题，研究表明当有一群人在狭小密闭的环境中使用手机时，网络信号会比平时弱许多，这是因为基站的频段所能容纳的数量是有限的。在一个展厅或博物馆这样一个处于弱网通信的环境中，游客扫了某展品厅的二维码却长时间刷新不出来有用的信息，很可能会尝试多次重复操作，这无疑提高了服务器的压力负载，我们既要保证多次请求的幂等性，还要保证高可用性。为此，以用户体验为出发点，如何为这样的场景设计一个符合人体工程学的人机交互的系统已经成为当下一个重要的问题。

## 1.2 研究现状

上个世纪末到本世纪初，一维条码是用来标识物体的最佳手段，然而，这种条码能够承载的信息不多，而且受限于特别的设备。随着互联网浪潮的到来，人们对条码的能力提出了更高的要求，迫切地寻找能满足大容量和高精度的条码表达，在这种渴求和编码技术的推动之下，Interface Mechanisms于1970年发明了二维码。

Wahsheh和Bani-Hani等发表的Secure QR Code system[1]这篇文章指出了如何构建一个安全的二维码系统为将来的大规模推广使用打下了坚实的基础。

二维码，又称快速响应矩阵码，顾名思义，就是在有限的平面空间上用纵横两轴来进行编解码，这种信息的记录依赖于特定规律的几何图形分布[2]，早期的二维码是黑白的，目前，大量使用的二维码也是黑白的，但也出现了其他各式各样的二维码（如微信支付，QQ钱包）。由于记录能力在另一方向上的延展，所以二维码所能承载的信息容量得到了极大的提升，这为我们实现加密和冗余纠错机制搭建了舞台。

信息就是比特流和上下文的组合，如果提前规定一些几何图形与信息的映射，那么我们就可以使用一些光学设备来读取这些图形然后翻译成对应的信息[3]。一个典型的二维码结构如图1-1所示。

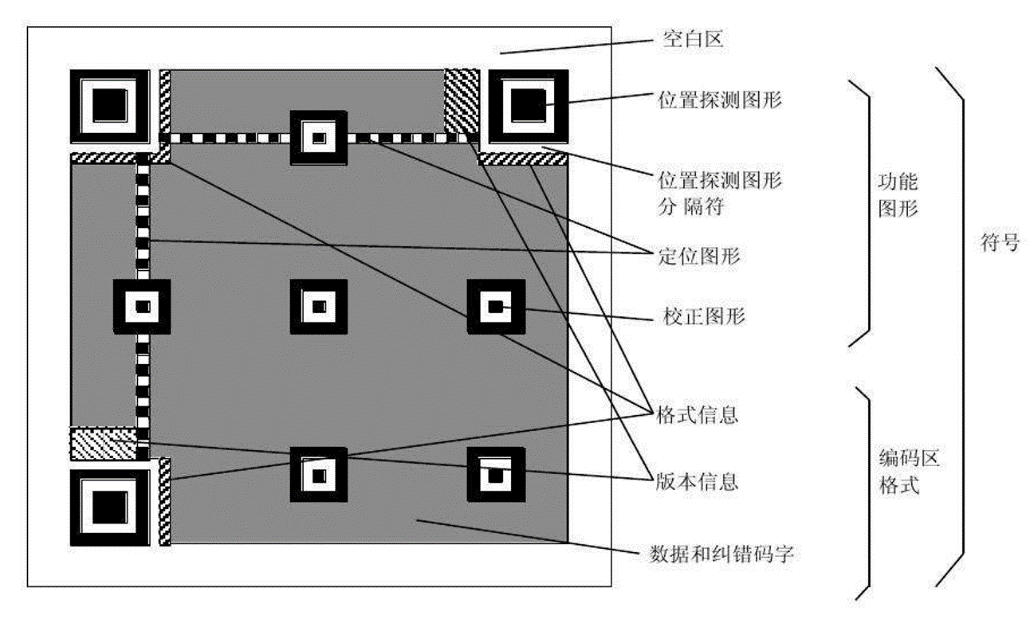


图 1-1 二维码结构图

世界各地的博物馆和展品厅已经逐渐普及二维码介绍系统。主要是结合图案和语音导览系统。国内的博物馆提供的主要功能如下：

（一）藏品数字展牌信息介绍。在藏品信息的展示上，无论是公立博物馆还是私人博物馆，往往都喜欢使用文字展牌的方式，观众对藏品文字展牌的阅读即是一个对藏品的背景和相关知识的过程，同时，当观众视线集中于藏品时，藏品则成为对文字展牌内容的印证，文字展牌的设立提升了观众对藏品的理解。通过文字展牌介绍文物的方式往往成本低成效快，观众只需要对文字信息进行阅读就可以了解到有关的藏品内容，一般情况下，这些文字展牌包括相关藏品的大多数信息，例如藏品的名称、发现或发明年代、藏品小故事等，但这些文字展牌的质量却良莠不齐，在文字展牌上的文字内容设置上就存在很多问题。随着我国博物馆数字化的不断发展，以二维码技术为核心的数字展牌开始出现在博物馆之中[4]，而它的出现使得文字展牌中出现的一些问题得到了有效地解决**。**

(二) 藏品语音导览辅助。语音导览是以声音为主要沟通的导览方式，现其已完成由人工到智能媒体的发展和转变，并成为博物馆教育职能的延伸，同时也成为博物馆公众服务不可或缺的一个重要组成部分。如今由于智能设备的出现以及二维码技术的不断发展，人们已经可以使用智能设备例如手机扫描博物馆内藏品附近的二维码图案进入到数字展牌界面[5]，来获取藏品语音导览以更好地对藏品进行了解。观众通过扫描二维码以链接到微信小程序上进行语音导览的这种方式以上海博物馆为佳。

## 1.3 研究内容

本文主要从以下三个方面开始研究：（1）以市面上已有的博物馆 App 或小程序为例，分析其产品的功能组成和特点，结合实际情况推出系统需求并以此划定出必要的功能模块。调研已有的跨平台方案以满足不同层次，不同手机，不同环境下的多功能多方面多角度的使用情形。（2）对大规模使用场景进行测试以验证系统的高并发性能。然而，系统的上限并非无法达到，需要针对服务器宕机的情况设计一主多备的策略来用于恢复数据和服务， 所以需要设计冗余备份以满足高可用性。（3）研究平台一致性问题，不同的客户端技术在不同的平台上进行交互时表现并非一致，需要设计出一套合理的接口进行组合以确保系统的健壮性。

# 第2章 相关技术简介

## 2.1 项目架构

为达到客户端多平台UI设计统一的效果，我们使用Flutter框架统一编译并部署到Android和iOS平台上，提高了一定的开发效率。服务端不是传统的LAMP或MVC架构而是一种基于微服务的架构。这是因为传统架构存在以下三个缺点：

（一）开发效率低。单体应用随着时间和业务量的增长，代码量会增多并且依赖复杂。 对于这样一个应用编译打包，部署测试一次需要的时间在小时级别。同时，也增加了团队协作的开发成本。

（二）系统高可用性差。系统中的许多服务都运行到一个进程中，一旦某个服务出现问题而崩溃，可能会影响整个进程对外提供服务的能力，如果该服务以集群的方式部署，甚至会出现雪崩效应导致一整片服务不可用。

（三）线上发布变慢。特别是对Java应用而言，代码膨胀带来的后果就是服务启动时间急剧拉长，如果服务器规模较大，这样的耗时是不可接受的，或者说当服务器宕机时，不能很快地重启服务会造成重大财产损失。

微服务化是指把传统的单机应用中不同功能模块抽象解耦出来变成一个个独立的小服务[6]，改造成通过RPC接口产生的远程方法调用。这对于项目的代码复用和开发人员的业务理解都有益处。微服务化有以下四个特点：

（一）服务粒度变细。微服务设施就是更细粒度的多个服务集合。划分粒度的标准是考察该功能涉及到的模块，只要该模块的依赖与其他模块没有耦合关系就可以称之为一个微服务。

（二）服务弹性部署。微服务中的服务单元部署各自独立，运行在同一宿主机环境下但互不影响。一个机柜中刀片服务器可以部署多个各不相同的容器服务，每个容器对外承接不同的服务需求。

（三）服务运维轻便。传统服务架构需要一个小型团队根据系统上游版本和各种依赖关系进行费时费力地维护。微服务的整个生命周期可交给个人来负责。

（四）服务管理中心化。微服务设施中的服务单元是业务分层之后的表征。必然存在着服务注册，管理与发现的平台来进行治理，维持服务的健康度。

具体来说，本系统使用Go语言实现服务接口，并且利用Redis做热点数据缓存，MySQL用来进行数据持久化，用MeiliSearch做全文搜索，对于App中的图片等其他静态资源，我们使用腾讯云对象存储提供的 SDK来进行访问。整体架构如图2-12-1所示。客户端用带有COS对象存储回源的CDN以HTTPS的形式访问App中的静态资源。通过gRPC协议规定的接口进行相应方法的调用，Service为接口的具体实现逻辑，与Redis和MeiliSearch一起组成三个容器构建服务框架。MySQL作为系统的数据持久化层，在Redis缓存失效时发挥重要作用。



图 2-1系统架构图

## 2.2 C/S架构

本文研究的系统是一个典型的两层C/S架构，客户端为用户手机上的移动App,主要由Dart编写，并使用Flutter生态里的一些组件对UI进行修饰并增加动画，提高用户的使用体验。与服务端的通信基于HTTP/2的gRPC，这个协议工作在OSI七层网络模型中的应用层。使用Go编写有关的业务逻辑，并将其部署至容器中。

## 2.3 gRPC通信

gRPC[7]是Google公司开发的一套基于Protocol Buffer序列化协议的RPC通信框架，基于HTTP/2。该技术支持多种语言，并提供对网络层进行配置和纳管的方法。除用在典型的C/S架构的网络通信外，也可作为大型公司内部局域网负载均衡的方案之一。由于是开源框架，故使用者可对其二次开发。得益于Protocol Buffer序列化速度远超JSON，gRPC在有效网络信息的传输率和利用率上都远胜过去的RESTful风格的 API，因此可作为弱网通信技术和云计算[8]的基础。

## 2.4 Flutter框架

Flutter是Google公司开源的一套UI框架，支持iOS、Android、Web、Windows、MacOS、Linux。同时也是构建系统Fuchsia的默认套件。得益于其底层2D渲染引擎skia强大的跨平台性和性能，Flutter拥有无与伦比的跨平台性并且在多个平台的行为一致。在本文中，Flutter框架将被用于构建客户端App。

## 2.5 Redis缓存

Redis是一种K-V结构的NoSQL数据库[9]，包含各种各样的数据结构以满足不同的使用场景。Redis的读写性能非常强大，因为所有数据都放在内存中。同时，为了保证数据不会丢失，Redis会将数据以快照或日志的手段持久化到硬盘上。基于以上特性，Redis特别适合用来做大型关系型数据库系统如MySQL的缓存，缩短客户端响应时间，提高服务端性能。在本系统中，我们将使用Redis构建展品查询缓存。

## 2.6 MySQL数据库

MySQL现在为Oracle公司旗下的RDBMS，该公司提供的开源版本已成为世界上使用最广泛的数据库软件之一并占有重要市场份额。本文将用MySQL设计用户模型与展品详情模型。

## 2.7 MeiliSearch搜索引擎

MeiliSearch是一个开源并且高性能的轻量级搜索引擎，搜索结果的平均耗时低于50ms，支持全文搜索，能够理解错别字和拼写错误，支持同义词和汉字。可高度定制化搜索和索引。并且使用Actix-Web为前端提供RESTful API接口。该引擎使用Rust语言构建，具备安全，稳定和高性能几个特点。本文将用MeiliSearch搜索引擎为前端提供搜索展品的服务。

## 2.8 Docker容器化

Docker为云计算提供了一套轻量级的虚拟化方案[10]，包括一个容器运行时引擎和管理系统。Docker容器的好处主要有以下两点，环境隔离和迁移方便，环境隔离使得各个运行的实例之间互不干扰。迁移方便指的是Docker将应用所需内容全部打包到一个容器中，然后在虚拟机、服务器或云之间迁移该容器，而无需重构应用。本系统将用Docker编排Redis、MeiliSearch和微服务接口等服务。

# 第3章 二维码展品介绍系统需求分析

## 3.1 功能需求

展品介绍系统适用范围广，目标受众多，因此一个完善的系统功能众多，例如，用户登录，用户注册，扫描展品二维码，发表评论，展品录入和日志收集等，限于篇幅，本文着重于实现几个重点的核心功能。

## 3.2 性能需求

对于客户端来说，Flutter提供的渲染能力堪比原生。所以用户在使用时会觉得动画流畅、响应迅速，并且由于gRPC使用的是Protocol Buffer编码，交换的数据量小，较JSON有巨大优势，所以在网络不畅的地方体验也相对较好。

通过对以往的大型展会或展厅的日高峰人流量分析可以知道如何设计系统的负载，比如，上海世博会的日最高峰人流量约为60万，故宫博物馆的日最高峰人流量为8万。故暂定单机系统面临的RPS在10K至100K这个范围。如果遵循以往的传统架构，单机节点的MySQL的QPS最高就会达到100K，这需要极高的硬件配置并且一旦发生数据库故障，造成的损失极大。这种高风险低收益的传统方案已经不再适合于这样的高并发高可用的系统了。为此，我们使用Redis解决这一问题，Redis单点的QPS高达100K，同时Redis集群构成的分布式系统非常可控并且价格便宜。

## 3.3 安全需求

由于系统使用人数较多，因此对系统的健全性提出了要求。重点从两个方面考虑系统的安全性，数据隐私安全和应对系统宕机。

为了防止用户个人隐私泄露，gRPC提供类似于HTTPS的证书加密，能较好的保证传输过程中数据的安全。除此之外，在设计用户表时，不应该将密码明文存储，而是使用“加盐”哈希的方式存储其密码的哈希值。这样减少了密码暴露的可能性。

如果峰值流量较高，系统可能一时无法承受进而宕机，为此我们需要做备份预案，比如为MySQL设置Master-Slave备份，为Redis设置AOF持久化等都是有效提高系统可用性的解决方案。

## 3.4 运维需求

传统应用的部署或运维面对的难题是开发人员进行开发的本地环境与测试服务器的环境不一致，整个持续集成或持续交付的流程所需的基础设施也与开发场景不一致。而容器由于使用了rootfs隔离技术就可使用COW机制构建一个分层的镜像文件，包含一个支撑其应用依赖和操作系统运行的最小单体结构。无论是对于开发者还是运维测试人员，容器技术的出现解决了传统场景下部署和测试困难的难题。

本系统使用docker-compose管理涉及到的一系列容器，即Service,Redis和MeiliSearch服务，能够在相同Linux内核下但不同的环境下的宿主机中随时迁移，测试，打包和发布，贯穿整个DevOps链路，同时也践行了敏捷开发的宗旨，提高了效率。

# 第4章 二维码展品介绍系统的功能描述与设计分析

## 4.1 系统设计目标

（一）客户端界面优美，响应迅速，动画流畅，运行稳定。

（二）文档健全，版本易于管理，提交历史清晰。

（三）整个系统能够承受一定程度的并发。

## 4.2 系统结构

移动端是一个跨Android平台和iOS平台的手机App，名称为博物馆，由三个子系统组成：登陆系统，设置系统，展品系统。其系统功能结构如图3-1所示：



图 3-1移动端功能结构图

（1）登录系统

用户可在此页面进行登录和注册。登录名为一个邮箱，并且限制了密码强度，对于可能出错的各种情况都给予了考虑：登录时可能会出现用户名不存在或者密码错误这两种情况；注册时可能会出现用户名已存在，两次输入密码不一致和密码强度过低这三种情况。认证的整个过程采用TLS加密，提高了安全性。

（2）用户系统

用户页面需要展示用户的基本信息，如头像和邮箱等。除此之外还可展示其预约情况，行程情况，消息通知，收藏的展品和发表的评论等等。在其之下的设置系统可让用户更改用户名，更改邮箱，更改密码或注销账号。这些信息在系统内部都进行了加密存储。

（3）展品系统

展品页面展示出了当前博物馆的部分藏品，用户可以通过扫描给定的二维码显示出该藏品的详细信息，比如名称，朝代，描述，发现位置和历史介绍等。同时每个藏品页面提供收藏功能，用户可以将感兴趣的藏品收藏归纳起来方便整理查看。除此之外，用户可以通过搜索关键字来查找特定展品。

## 4.3 服务端结构

（1）Service接口

服务端提供的微服务接口由proto格式的文件描述，该文件描述和规范了RPC的请求和返回信息范式。由Protocol Buffer Compiler编译后生成对应的代码，屏蔽掉了网络和序列化过程的细节，使得开发者只需关注如何实现方法本身即可。

本系统提供的服务接口分别是用户登录和展品获取。

用户登录服务中提供两个RPC远程过程调用方法，分别是登录和注册，这两种RPC方法共享同一请求参数和返回结果，即AuthRequest和AuthReply，AuthRequest表示登录的用户名和密码。AuthReply表示刚刚的登录或注册请求对应的返回结果，包含成功的标志位和可选的失败原因。

展品获取服务中提供三个RPC远程过程调用方法，分别是二维码查询，获取部分藏品帖子和搜索藏品。每个藏品用一个唯一的UUID标识，用户扫描二维码后，将该UUID通过onQuery方法调用获得返回结果PostReply，其中包含了成功的标志位，可选的内容和失败原因。用户进入App底部栏的藏品页面时，会触发获取全部藏品帖子的RPC方法，请求参数为空，返回结果一个PostReply数组。用户点击搜索时会将其输入的关键字作为搜索请求的参数并同样返回PostReply数组。

本系统涉及到的RPC方法定义如表4-1所示：

表 4-1 系统 RPC 参数

|  |  |  |  |
| --- | --- | --- | --- |
| **Service** | **RPC** | **Parameter** | **Return** |
| Authenticate | OnLogin | AuthRequest | AuthReply |
| Authenticate | OnSignUp | AuthRequest | AuthReply |
| Post  Post  Post | OnQuery  OnFetchAll  OnSearch | PostRequest  Empty  Token | PostRely  Posts  Posts |

Authenticate涉及到的消息结构如表4-2所示：

表 4-2 Authenticate 消息结构

|  |  |  |  |
| --- | --- | --- | --- |
| **Schema** | **Type** | **Name** | **Encode** |
| AuthRequest | string | name | 1 |
| AuthRequest | string | password | 2 |
| AuthReply | string | success | 1 |
| AuthReply | AuthError | error[optional] | 2 |
| AuthError | int | MISMATCHED\_PASSWORD | 0 |
| AuthError | int | NONEXISTENT\_USER | 1 |
| AuthError | int | DUPLICATE\_NAME | 2 |
| AuthError | int | UNKNOWN | 3 |

Post涉及到的消息结构定义如表4-3和表4-4所示：

表 4-3 Post 消息结构-1

| **Schema** | **Type** | **Name** | **Encode** |
| --- | --- | --- | --- |
| PostRequest | string | id | 1 |
| Empty | N/A | N/A | N/A |
| Token | String | Value | 1 |
| Posts | PostReply | posts[repeated] | 1 |
| PostReply | bool | success | 1 |
| PostReply | PostContent | content[optional] | 2 |
| PostReply | PostError | error[optional] | 3 |
| PostContent | string | id | 1 |
| PostContent | string | name | 2 |
| PostContent | string | dynasty | 3 |
| PostContent | string | descr | 4 |
| PostContent | string | intro | 5 |

表 4-4 Post 消息结构-2

| **Schema** | **Type** | **Name** | **Encode** |
| --- | --- | --- | --- |
| PostContent | string | pic | 6 |
| PostError | int | NONEXISTENT\_POST | 0 |
| PostError | int | INVALID\_POST\_ID | 1 |
| PostError | int | UNKNOWN | 2 |

（2）MySQL表的设计

用户表的结构如表4-5所示：

表 4-5 用户表

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **含义属性** | **类型** | **备注** |
| id | 主键 | INTEGER | AUTO\_INCREMENT |
| name | 账号 | VARCHAR(20) | NOT NULL |
| pswdhash | 密码哈希值 | VARCHAR(64) | NOT NULL |

后端将用户密码以“加盐”哈希的方式存储，这是为了防止用户密码过于薄弱或容易被字典攻击破解，服务初始化时后端利用给定的随机字符串与密码相加，并用SHA256哈希算法计算哈希值，以逆向不可破解性保证了用户信息的安全。

展品表的结构如表4-6所示：

表 4-6 展品表

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **含义属性** | **类型** | **备注** |
| id | 主键 | BINARY(16) | UUID v4 |
| name | 名称 | VARCHAR(25) | NOT NULL |
| dynasty | 朝代 | VARCHAR(32) | NOT NULL |
| descr | 描述 | VARCHAR(1024) | NOT NULL |
| intro | 介绍 | LONGTEXT | NOT NULL |
| pic | COS-URL | TEXT | NOT NULL |

因为MySQL索引基于B+树，主键字段或UNIQUE约束字段须为有序结构。将展品表的主键设置为UUID而表中无其他唯一约束带来的缺点是会引起MySQL索引失效。但优点有二：

（一）分布式系统中不会出现因为将主键设置为自增整数造成的重复冲突，保证在同一时间内所有展品id的唯一性。

（二）如果用自增整数来标识帖子，很容易被爬虫获取数据，增加了系统负载。

（3）Redis缓存的设计

在初始化数据阶段，每录入一个展品，就将其对应的UUID值放入pid数组中，并且将这个UUID值映射到展品自身的字段数组中。这样，可以通过Redis中的lrange方法直接获取全部的pid而不需要通过keys \*这种耗时的办法，提高了性能[11]。客户端并不直接连接Redis服务，而且通过RPC调用以后，后端通过Redis客户端向Redis-server发出命令以获取相应的数据。与展品相关的三个RPC调用的方法逻辑如下：

1. OnQuery(in) -> PostReply: 后端将in参数中的id值解析为UUID，若解析失败，将返回INVALID\_POST\_ID。若解析成功后在Redis缓存未命中则会请求MySQL中的数据，若依然没有相同的UUID则返回NONEXISTENT\_POST。只有Redis缓存命中或者未命中时能从MySQL获取到数据的情况下才会返回要查询的帖子。
2. OnFetchAll() -> Posts: 当用户滑动到展品栏时，会触发OnFetchAll RPC 方法，该方法请求后端一次性发送所有的藏品信息。当系统初始化成功后该方法总是调用成功。
3. OnSearch(token) -> Posts:用户将搜索的关键字通过OnSearch RPC方法发送到后端，后端与MeiliSearch通信，MeiliSearch代理了后端的搜索服务并返回命中关键字的所有藏品信息。

（4）MeiliSearch引擎

MeiliSearch引擎要求将数据编排成能反序列化到JSON的格式。后端在初始化时与MeiliSearch进行通信并完成对藏品各个字段的索引构建。

（5）Docker-compose编排容器[12]

museum是主服务，其启动依赖于它将本地的8080端口映射到容器中的8080端口，启动时依赖于redis和meili这两个服务。并且它还将当前工作目录映射到/code目录下，方便了即时部署。通过增加额外的extra\_hosts来与宿主机的MySQL服务通信。

docker-compose.yml结构如表4-7所示：

表4-7 docker-compose.yml属性

|  |  |  |  |
| --- | --- | --- | --- |
| **Service** | **Image** | **Volumes** | **Ports** |
| museum | Dockerfile | .:/code | 8080:8080 |
| redis | redis:latest | N/A | N/A |
| meili | getmeili/meilisearch | N/A | N/A |

# 第5章 二维码展品介绍系统

## 5.1 移动端实现

移动端共用Flutter绘制了22个页面，所有页面都使用的是弹性布局，以确保在不同屏幕和分辨率大小的手机上都能表现出统一的视觉效果。

启动App后首先看到的是开屏页，开屏页的壁纸来源于中国国家博物馆官方App启动的开屏页，在停留一秒后进入登录页面。开屏页的实现基于flutter\_native\_splash。

登录页面的中心是一张卡片风格的表单。在其上方是App的英文名称museum，使用Vladimir字体进行渲染，线条优美简洁。对于登录和注册遇到的不同情况，在底部都为其实现了对应的错误信息提示。无论是登录还是注册，都首先检查用户的输入是否符合标准。登录名应为一个有效的邮箱，而密码不得少于6位。

检查完毕后，若用户进行的是注册操作，则后端判断用户名已存在，则提示“用户名已存在，请直接登录”。若用户进行的是登录操作，则针对用户名不存在和密码错误这两种情况分别给出“用户名不存在”和“密码错误”的信息提示。登录界面的非线性动画较多，比较错误信息的弹出，登录按钮会变成进度指示条。登录成功后，上方的英文标题会向上淡出，而中心的卡片则旋转变小成为一个菱形，随后弹出主界面。登录页的实现基于flutter\_login。应用开屏页与登录页面如图5-1所示。



图5-1 App启动开屏页与登录页面

App主界面如图5-2所示，由底部栏分为四部分，分别是首页，展览，藏品和我的。首页的顶部是AppBar，左边为通知按钮，用于显示用户收到的通知，中间为博物馆的logo，右侧是扫码和搜索两个核心功能的按钮。在其之下的是一个Swiper组件，用于显示最近博物馆举办的展览，可以左右滑动和自动播放，组件的实现基于card\_swiper。在swiper下面的是八个功能区，分别是全景国博，参观预约，讲解导览，国博好客，餐饮服务，国博文创，便民服务和国博课堂。由于本系统主要关注的是将二维码技术和信息服务结合起来的工程实践，故这个八个按钮暂且未实现。

继续向下滑动看到的是一个新闻推送板块“儿童语音导览”，下面就是展览推荐，点击按钮“全部”即可让页面滑动到展览。点击其中任意一个就可滑动到相应的展览详情页。在首页底部的是另一个推送板块。

展览页的AppBar左侧为标题，右侧为搜索按钮。顶部是一个虚拟展厅的图片板块，用于增加美观性。主体是用格子间布局排布并且中心对称的展览卡片。

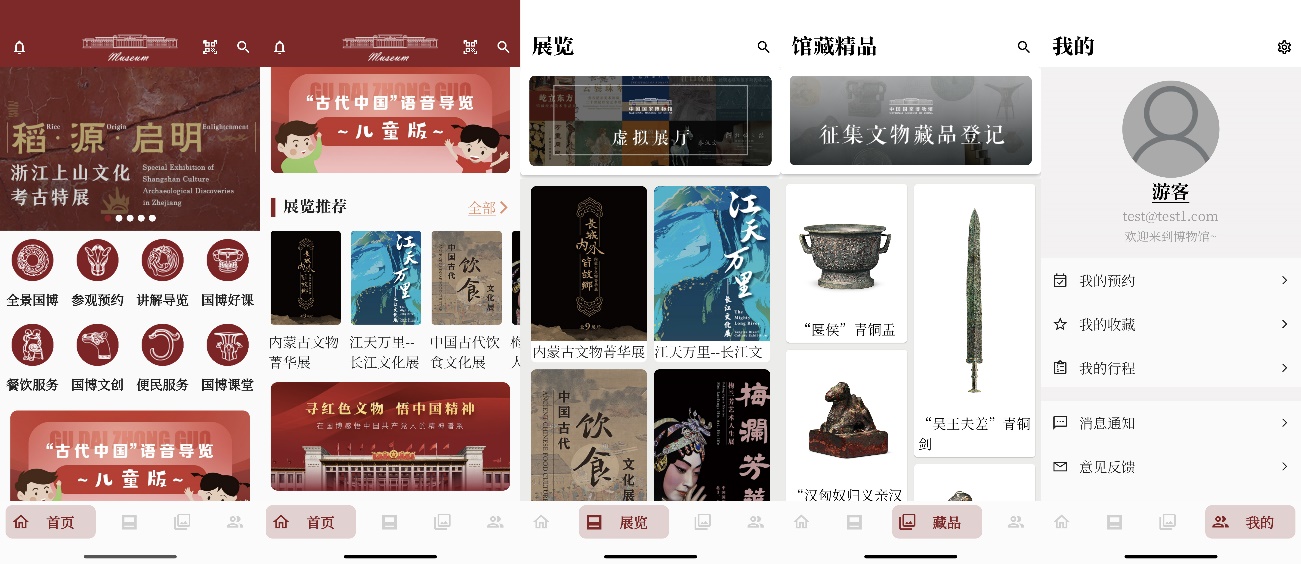


图 5-2 App主界面

展览页的详情页面的AppBar左侧是一个返回按钮，用于返回上一级页面，中间是标题。顶部为该次展览所用的宣传页。而主体部分则是对展览的介绍。



图5-3 展览详情

扫描页如图5-4所示，该页面即为App的核心功能，通过扫描在后端服务初始化过程中生成的二维码得到一个UUID值，通过OnQuery(in)这个RPC方法返回展品详情并渲染。扫描框居于屏幕上方2/5处，为正方形，周围是灰黑色透明的玻璃样式。

QR 代码

描述已自动生成

图5-4 扫描二维码界面

藏品页如图5-5所示，其AppBar右侧搜索与首页搜索功能一致。主题是采用瀑布式布局的藏品卡片，这样就解决了不同高度的藏品在普通的格子间布局的情况下视觉效果不统一的现象。详情页的AppBar右侧有一个分享功能按钮，顶部为展品的图片显示，下面的版块里，左上是藏品名称，左下为藏品介绍，右上为收藏按钮，右下为朝代。

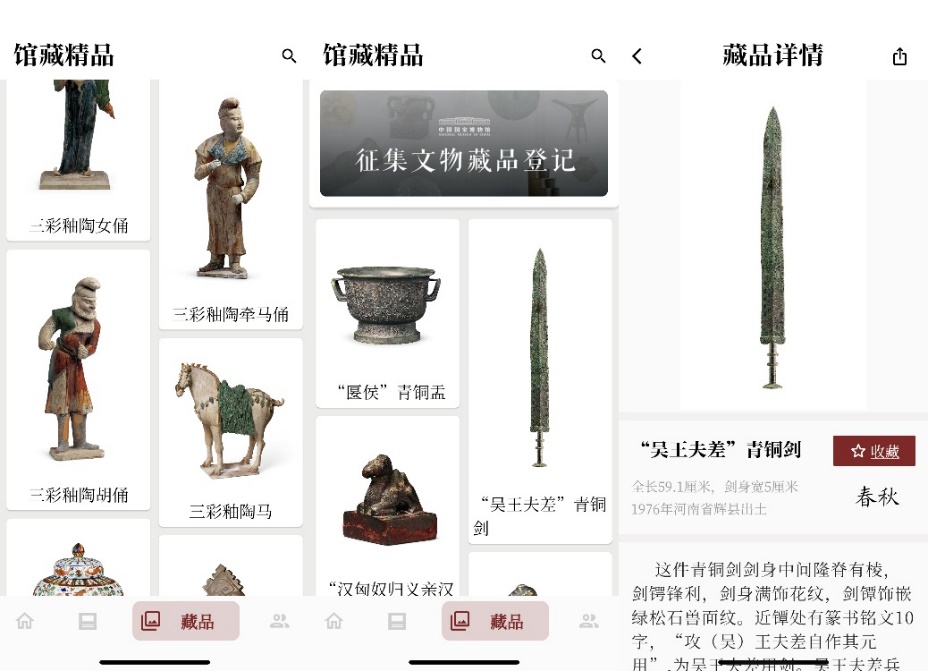


图5-5 藏品页面

通过点击图16顶部灰色中间个人信息的区域或者AppBar右上角的设置按钮即可进入用户设置界面。菜单栏包含六大板块，分别是个人资料，修改密码，账号管理，隐私设置，清除缓存和当前版本。

个人资料如图5-6所示，其中包含了设置昵称，性别和生日等选项。修改密码需要输入正确的旧密码和符合强度的新密码。账号注销需要阅读并同意所有条款并确认。



图 5-6 用户个人页面

用户设置详情如图5-7所示，输入昵称和输入邮箱均为一个单独的圆角输入框，输入完成后会对其进行格式检查，用户通过保存按钮即可验证。修改性别和生日使用的iOS的原生控件，确保了系统风格的统一。



图5-7 用户设置详情

## 5.2 服务端实现

（1）数据库初始化

通过爬虫获取了中国国家博物馆网站的40个藏品，完成了藏品信息的初始化。在将信息整理成Post结构体之后，通过腾讯云提供的COS Go SDK将藏品对应的图片和利用其UUID值生成的二维码上传至COS对象存储服务器中，并开启了CDN加速。此时我们将所有结构体插入数据库中。建立Redis缓存时，我们将所有Post都从MySQL中取出，将id字段作为键，全部字段放入列表中。并用一个键为pid的数组维护所有Post的ID。在这之后将Post数组给MeiliSearch搜索引擎建立索引。

（2）gRPC鉴权

通过openssl生成自验证的公钥和证书，首先虚拟一个CA机构，生成CA认证机构的证书密钥key，然后去除密钥里的密码。用私钥生成CA认证机构的证书ca.crt。即用私钥生成公钥，再把公钥包装成证书。生成的证书称为根证书，可以用来认证其他证书。

生成CA机构的证书后需要生成对应实体的私钥，还要一个生成关于该实体的证书请求文件CSR。私钥已经包含在这个请求文件中，接着用刚刚虚拟出来的CA认证机构把请求文件生成对应实体的公钥并包装成一个证书。这样就完成了鉴权所需的私钥和证书。在gRPC Server初始化时为其注册即可。

（3）登录和注册

与登录注册模块相关的RPC方法是OnLogin和OnSignUp。在收到参数AuthRequest后从中解析出用户名和密码，若是登录操作，其一判断用户名是否存在，不存在返回NONEXISTENT\_USER的错误，其二判断密码哈希值是否匹配，不匹配则返回MISMATCHED\_PASSWORD的错误。若是注册操作，其一判断用户名是否存在，若存在，返回DUPLICATED\_NAME的错误。

（4）扫描，查询和搜索

与扫描，查询和搜索相关的RPC方法是OnQuery，OnFetchAll和OnSearch。OnQuery方法会首先判断参数是否符合UUID规范，若不符合，返回非法请求参数的错误，则用Redis的lrange方法判断获取到的数组长度，若为0，说明不存在该帖子，返回不存在该帖子的错误。OnFetchAll方法不接收任何参数，仅为用户初次打开App时拉去藏品详情所用，调用应该总是成功。OnSearch方法接受一个参数Token，从该Token中获取用户要查询的关键字并将其交给MeiliSearch搜索引擎，随后将引擎的返回结果包装为Posts返回给客户。

## 5.3 性能测试

本设计对关键的核心方法OnQuery和OnSearch进行了测试。测试工具为开源的第三方工具ghz，专门用来测试gRPC框架的性能表现。测试环境为Windows11（客户端）和在其之上的WSL2（服务端）。CPU为8核8线程的i7-9700K，频率为4.5GHz。内存容量为32GB，其频率为3600Mhz。

测试结果如图5-8所示，左右分别代码OnSearch和OnQuery在面对C10K问题时的表现，即10000次查询或者搜索。

通过分析右边的图表可知，对OnQuery方法而言，每秒最多能处理10000多个请求，并且99%的请求时间都落在14ms内，其中，66%的请求都在4.3ms内完成，92%的请求都在8.6ms内完成，这是因为所有的查询都命中了Redis缓存，足见Redis的强大。

分析左边图表可知，对OnSearch方法而言，每秒最多处理2415个请求。这是因为系统的吞吐受限于MeiliSearch引擎，不过也是一个较为可观的表现。99%的请求都在58ms内返回，其中，83%的请求都在29ms内返回。

根据以上性能测试分析可知，系统较为出色的完成了前面提出的C10K问题。

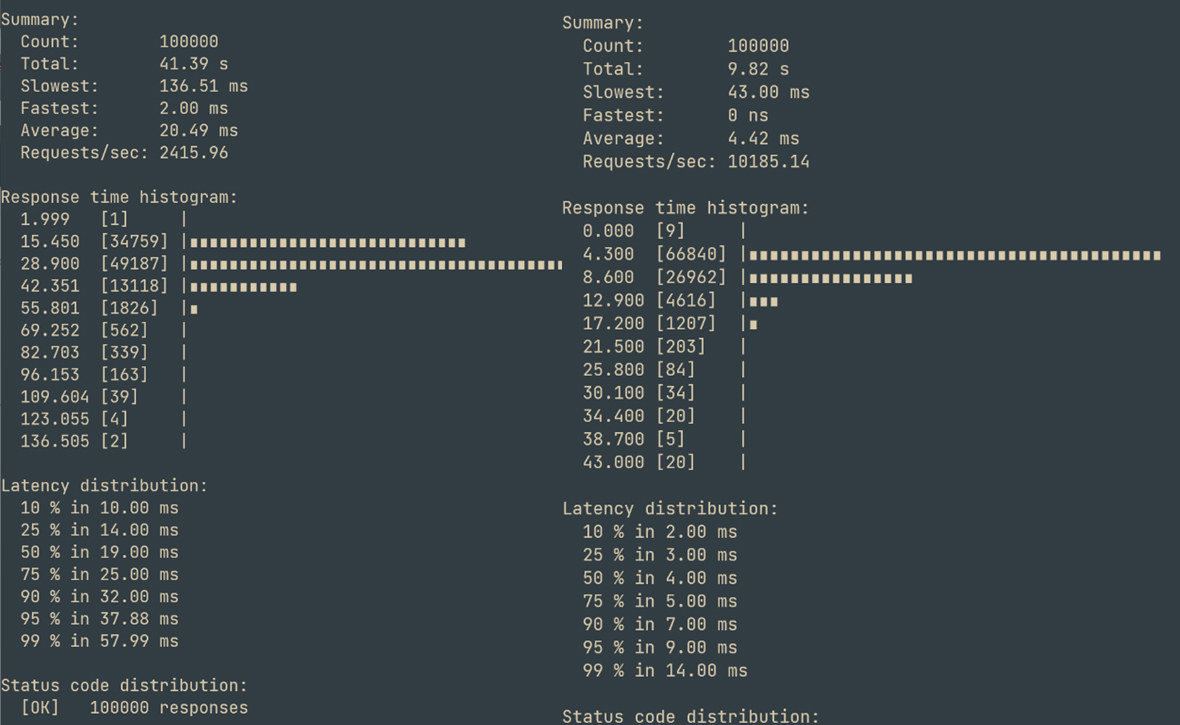


图5-8 性能测试

# 第6章 总结

本文将二维码与展品介绍系统进行了结合设计，并且调研了市面上已有的博物馆App，比如中国国家博物馆，在UI设计上一定程度地参考了它的实现。采用了多种混合技术开发整套系统，客户端为目前最流行跨平台UI解决方案Flutter,通过一套代码就可部署到多个不同的目标平台，并且UI美观，简洁流畅。服务端为云计算时代应用最广泛并且久经考验的容器和微服务技术，未来在分布式系统上，也极容易对服务端系统进行扩展。但是限于时间原因，本系统还存在以下不足：（1）没有开发一个Web管理界面以对藏品或者用户进行可视化管理。（2）部分UI功能没有实现，仅起到美化作用。（3）服务端代码的优化不足，比如没有建立SQL索引，没有考虑流量控制，部分设计冗余。

设计本系统也遇到了相当多的挑战并且收获颇丰，仅就客户端而言，由于对Flutter的布局不熟悉，编译出错的或者组件溢出的情况数不胜数，大多数情况都通过设置断点，打印消息，查看DevTools布局约束解决了，其余的通过StackOverflow网站解决了。在写服务端代码时也遇到了许多挑战，比如访问未经初始化的全局指针，由于在包初始化函数中重新定义并赋值了一个相同名字的指针造成访问全局指针时抛出空指针异常。再比如数据库的死锁问题，在注册时若遇到了MySQL表插入失败的情况就直接把当前函数返回而没有提交事务造成该表死锁无法再进行插入操作。在做并发测试时也遇到了许多意料之外的情况，比如测试1000名用户注册登录时MySQL报错太多连接，这是因为没有设置最大连接数的上限。类似的情况还有很多，限于篇幅就不一一列举了。

展望未来，优化的方向有很多，对于客户端来说，精炼UI的布局和各种细节，引入与Java原生开发安卓的混合编程，提高首屏加载时间和渲染性能都在计划之列。对服务端来说，可以考虑用手机验证码或者第三方登录的方式重新设计登录系统，考虑用OAuth2系统或者其他。就数据库而言，可以考虑对MySQL进行分库分表并进行主从冗余备份配置，对缓存而言，设置Redis集群且开启AOF持久化功能。在后端的服务入口处，可以设置网关集群并用Nginx部署以提高负载能力。若考虑亿级以上的并发量，系统还需引入类似于Kafka消息队列对流量进行削峰处理以保证系统高可用性。搜索引擎也可换为更强大的ElasticSearch并进行分布式部署。容器编排也可换位更为专业的Kubernetes（K8s）来进行处理以做到弹性扩缩容。

# 参考文献

1. Secure QR code system. Bani-Hani, R. M., Wahsheh, Y. A., & Al-Sarhan, M. B. (2014).
2. 曾子剑. 基于QR二维码编解码技术的研究与实现[D].电子科技大学,2010.
3. 于英政. QR二维码相关技术的研究[D].北京交通大学,2014.
4. 黄秋野. 博物馆中的数字化展览及展示技术研究[D].江南大学,2008.
5. 张美霞.新媒体技术支持下的场馆建设与场馆学习——以现代教育技术博物馆为例[J].中国电化教育,2017(02):20-24.
6. 马雄. 基于微服务架构的系统设计与开发[D].南京邮电大学,2017.
7. Wang, X., Zhao, H., & Zhu, J. (1993). GRPC. ACM SIGOPS Operating Systems Review, 27(3), 75–86.
8. 张建勋,古志民,郑超.云计算研究进展综述[J].计算机应用研究,2010,27(02):429-433.
9. 曾泉匀. 基于Redis的分布式消息服务的设计与实现[D].北京邮电大学,2014.
10. 王亚玲,李春阳,崔蔚,张晶.基于Docker的PaaS平台建设[J].计算机系统应用,2016,25(03):72-77.
11. 杨艳,李炜,王纯.内存数据库在高速缓存方面的应用[J].现代电信科技,2011,41(12):59-64.
12. 陈清金,陈存香,张岩.Docker技术实现分析[J].信息通信技术,2015,9(02):37-40.

# 致谢

首先要感谢的是我的论文指导导师丁新涛教授，丁老师在我论文写作期间时提出了宝贵而具有建设性的指导意见。值此成文之际，向丁老师致以崇高的敬意和衷心的感谢。

感谢安徽师范大学对我的教育之恩，无论走到哪里都不会忘记。

谢谢我的室友周志国，带给我许多欢乐，我们互相扶持，走过了一段难忘的岁月。谢谢我的朋友杨代领，无数个夜晚，无数次畅谈，无论是对于技术还是人生，感谢他的陪伴与倾听。感谢我的家人，永远那么支持我，包容我。

最后，对参加论文评审，答辩的老师表示衷心的感谢！