每妆App软件架构设计规格说明

Ver:1.0.0

**目 录**

[1 引言 3](#_Toc301301027)

[1.1 目标 3](#_Toc301301028)

[1.2 文档范围 3](#_Toc301301029)

[1.3 术语和缩略语 3](#_Toc301301030)

[1.4 参考资料 3](#_Toc301301031)

[1.5 系统目标和约束 3](#_Toc301301032)

[2 系统设计 4](#_Toc301301033)

[2.1 系统架构概述 4](#_Toc301301034)

[2.2 对象模型 4](#_Toc301301035)

[2.3 接口 4](#_Toc301301036)

[2.4 特性实现 4](#_Toc301301037)

[2.5 错误代码 4](#_Toc301301038)

[2.6 错误日志 4](#_Toc301301039)

[2.7 部署视图 5](#_Toc301301040)

[3 数据库设计 6](#_Toc301301041)

[3.1 逻辑模型 6](#_Toc301301042)

[3.2 物理模型 6](#_Toc301301043)

[4 质量及其他方面 6](#_Toc301301044)

[4.1 可维护性 6](#_Toc301301045)

[4.2 安全性 7](#_Toc301301046)

[4.3 可扩展性 7](#_Toc301301047)

[4.4 可靠性 7](#_Toc301301048)

[4.5 可用性 7](#_Toc301301049)

[4.6 性能设计 7](#_Toc301301050)

[5 附录 8](#_Toc301301051)

[5.1 附件 8](#_Toc301301052)

[5.2 修过记录 8](#_Toc301301053)

# 引言

## 目标

在完成软件开发前期的准备工作如项目需求等，结合《每妆App需求规格说明书》项目小组提出了这份软件架构设计说明书。

此概要设计说明书对每妆App前后台的功能分配，模块划分，程序的总体结构，输入输出和接口设计，运行设计，数据结构设计及出错设计等方面作了全面的概括性的说明，为软件详细设计奠定了基础，同时作为系统分析员工作的阶段性总结和程序员进行开发及未来测试的重要文档资料。

## 文档范围

本文档包括系统的架构设计、数据库设计、安全性、可靠性、可用性等方面的规划和设计。

## 术语和缩略语

## 参考资料

部分内容参考了《每妆App需求规格说明书》

## 系统目标和约束

系统目标：实现项目规划中的大多数功能。

系统的约束：因为时间有限，不确保所有功能都能实现。

# 系统设计

## 系统架构概述

该系统的架构采用三层架构（MVC）的模式：展示层、逻辑层和数据层。采用面向对象的设计方法。

## 对象模型

该系统采用面向对象的设计方法。

## 接口

*[详细说明本系统内部每个接口的每个方法的定义。*

*注意：如果本系统需要和其他系统交互，则应该将交互的接口协议单独成册，而不是写入本节中。]*

## 特性实现

详见界面设计说明书。

## 错误代码

*[详细列出错误处理机制所能处理的所有错误代码及其含义]*

## 错误日志

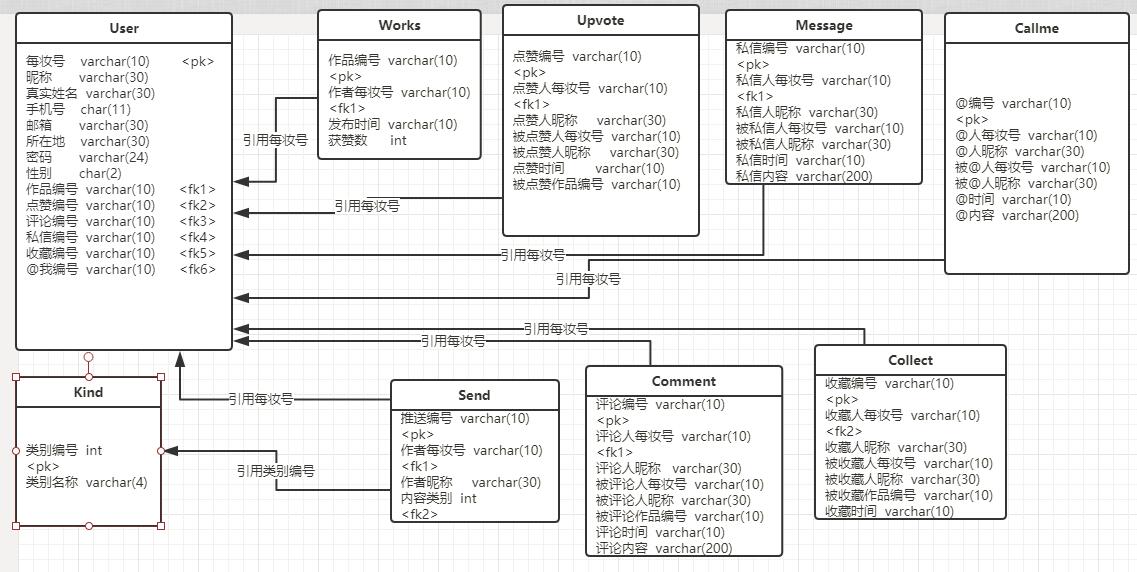
*[说明能够被处理和记录的错误类型和记录方式。]*

## 部署视图

1. 硬件环境：
   1. 需要互联网（采用云服务器，所以无须部署物理服务器）
2. 软件：
   1. 需部署一台Linux操作系统的服务器，计划采用CentOS 6.5 64位系统。
   2. 使用node.js编写WEB服务器。
   3. 数据库软件 MySQL
3. 配置：
   1. 配置Linux环境
   2. 配置node服务器环境
   3. 配置数据库环境
   4. 配置云服务器
   5. 配置域名端口
   6. 安装mocha
4. 部署：
   1. 配置mocha虚拟机,关闭错误提示。
   2. 配置虚拟机连接xshell的文件，修改IP地址、数据库的用户名和密码。
   3. 正确使用nodejs

# 数据库设计

## 概念模型



## 物理模型

1. 为每个表初始设置300M的存储空间，以10%的大小扩展。
2. 为上传的图片预留空间。

# 质量及其他方面

## 可维护性

1. 通过程序注释等方式增加代码的可读性和可维护性。
2. 将数据访问层分离，做成一个个函数，由其它层调用，以增加代码的可维护性。
3. 数据库有日志记录，系统一旦出现故障有恢复到故障之前的信息和数据的能力。

## 安全性

1. 密码使用md5加密
2. 对用户的输入进行验证
3. 对用户的输入的特殊字符进行转义，防止SQL注入攻击

## 可扩展性

1. 可以通过增加硬件资源的方式提高系统的响应速度。
2. 可以通过修改代码，扩充系统的功能。

## 可靠性

1. 数据库按时备份，有日志记录

## 可用性

1、做好充分的前期工作，系统一旦投入使用，尽量减少宕机的次数和时间

## 性能设计

*[系统性能通常使用事务处理能力或资源利用率来度量。确定系统需求的方法：*

*识别约束：包括预算、时间、基础结构、可选的开发工具和技术；*

*确定功能特性：要符合使用场景和用例*

*确定负载：使用本系统的客户数量]*

# 附录

## 附件

见文件列表

## 修改记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本 | 修改人 | 修改时间 | 修改内容 | 核准 |
| 1.0.0 | 实训小组 | 2018-11-28 |  |  |