飞掠设计规格说明

Ver:1.1

**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 9](#_Toc301301038)

[2.6 错误日志 9](#_Toc301301039)

[2.7 部署视图 9](#_Toc301301040)

[3 数据库设计 11](#_Toc301301041)

[3.1 逻辑模型 11](#_Toc301301042)

[3.2 物理模型 11](#_Toc301301043)

[4 质量及其他方面 12](#_Toc301301044)

[4.1 可维护性 12](#_Toc301301045)

[4.2 安全性 12](#_Toc301301046)

[4.3 可扩展性 12](#_Toc301301047)

[4.4 可靠性 12](#_Toc301301048)

[4.5 可用性 12](#_Toc301301049)

[4.6 性能设计 13](#_Toc301301050)

[5 附录 13](#_Toc301301051)

[5.1 附件 13](#_Toc301301052)

[5.2 修过记录 13](#_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《需求确认书》，，项目小组提出了这份软件设计说明书。

此概要设计说明书对飞掠APP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

react-native nodejs 冷启动

## 参考资料

部分内容参考了《飞掠需求确认书》

## 系统目标和约束

系统目标：项目需要完成查找，登入，修改个人信息，网上预订，登记，结账，旅客信息统计，收入支出统计等基本的管理。

系统的约束：因为时间有限，该系统只实现最基本的功能。

# 系统设计

## 系统架构概述

该系统的架构采用两层架构的模式：展示层（视图+业务逻辑）和数据访问层。采用面过程的设计方法。

## 对象模型

该系统采用非面向对象的设计方法。

## 接口

接口各大新闻门户网站，对获取的信息进行筛选，查重，去重，汇总接口每日各大航空公司的航班信息。

## 特性实现

1. 飞掠用户注册
   1. 用户注册页面
   2. 注册成功，可选择进入登录页面





1. 用户登录界面
   1. 用户登录页面
   2. 登录成功，可进入主页





1. 用户阅读相关新闻
   1. 点击首页



1. 查看新闻详情



1. 用户查看飞机分类页
   1. 点击飞机分类分页



1. 查看飞机的详情



## 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

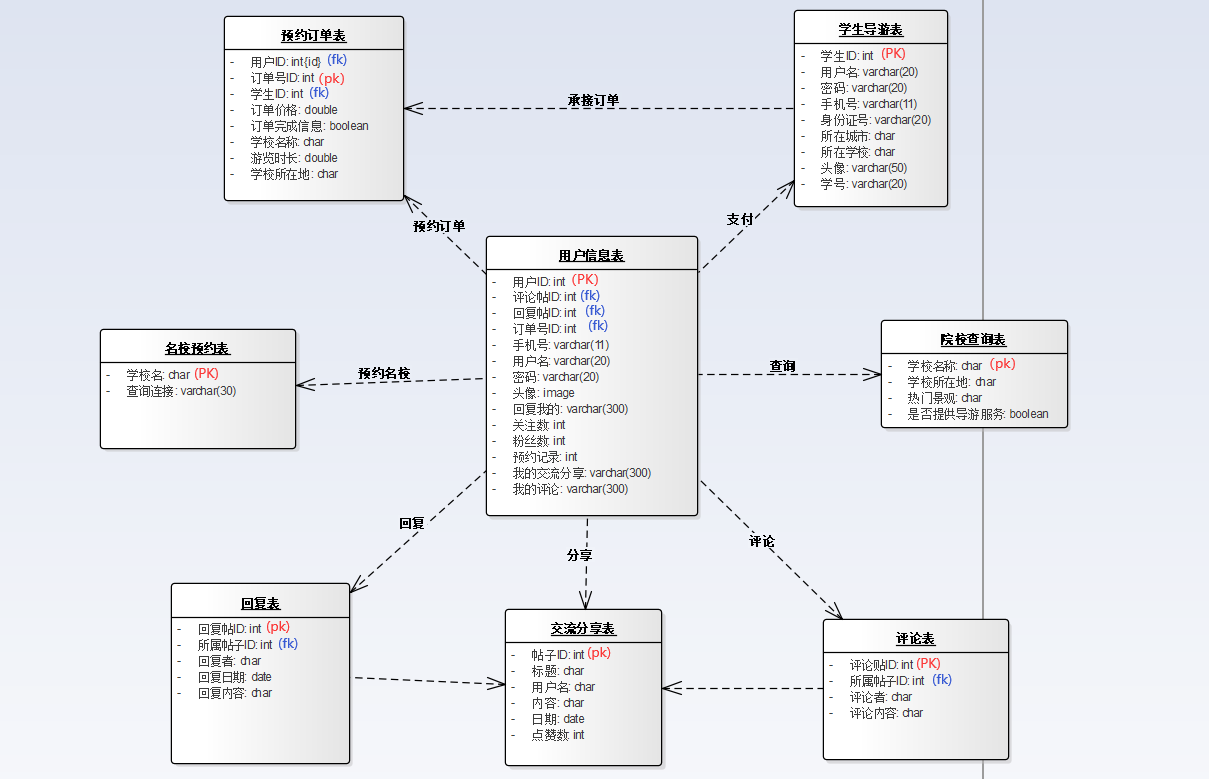
*[说明能够被处理和记录的错误类型和记录方式。]*

## 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. 一款数据库软件 MySQL 或SqlServer
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
4. 部署：
   1. 部署项目代码到服务器
   2. 部署数据库

# 数据库设计

## 逻辑模型



## 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

2. 推行试用时间及时找不足和可改进之处

3. 进行更新时不影响用户使用，更新过程时间缩短

4.上线后，如果后期需要更新迭代，可以在线及时配置和完成。运营者们将有更多的时间和精力花在运营和推广上面。

## 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]*

**预算：**

1.工作场所使用预算（自建/租用场地）

2.硬件费用 (电脑，服务器)

3.软件费用（操作系统，办公软件）

4.软件开发支出（人员报酬，出差外出调研费用）

以上四项由软件项目规模和开发周期决定（工作量核算，人员与时间）

后期：系统维护预算，营销计划广告推广预算，风险预算

**时间：**

前期设计——开发环节——测试环节 ——运营推广环节

**可选的开发工具和技术：**

工具：vscode 墨刀 EA

数据库sql

Node.Js

React

**确定功能特性：要符合使用场景和用例**

注册登录

发布商品

查找商品

后台可接受访问数据（浏览量 活跃度 帖子动态 发布审核）

**确定负载：**使用本系统的客户数量可能在某短时间内呈现集中（但不至于激增），即高并发。此时间段可能网络负载过大，应采用相应技术解决（需要使用负载均衡、[数据同步技术](https://www.baidu.com/s?wd=%E6%95%B0%E6%8D%AE%E5%90%8C%E6%AD%A5%E6%8A%80%E6%9C%AF&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)：  
①通过负载均衡服务可以集合多台主机的[服务能力](https://www.baidu.com/s?wd=%E6%9C%8D%E5%8A%A1%E8%83%BD%E5%8A%9B&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)，并可以随时实现水平扩展、系统扩容，增加对外业务[服务能力](https://www.baidu.com/s?wd=%E6%9C%8D%E5%8A%A1%E8%83%BD%E5%8A%9B&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)。 配合应用[架构设计](https://www.baidu.com/s?wd=%E6%9E%B6%E6%9E%84%E8%AE%BE%E8%AE%A1&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)，有效避免单点故障，最大限度的保持业务在线状态。②通过某种负载分担技术，将外部发送来的请求均匀分配到对称结构中的某一台服务器上，而接收到请求的服务器独立地回应客户的请求。[均衡负载](https://www.baidu.com/s?wd=%E5%9D%87%E8%A1%A1%E8%B4%9F%E8%BD%BD&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)能够平均分配客户请求到服务器列阵，籍此提供快速获取重要数据，解决大量并发访问服务问题。）

高并发的处理：

① 配置多个Linux外部服务器，Nginx反向代理。

② 增加缓存服务器，将数据放入内存里面，增加读取速度。

③ 搭建Redis集群。

④ 做数据分离（建立历史表，用的技术是Quartz（每天定时定点的执行任务））。

⑤ 将逻辑处理涉及到多处数据库连接操作，优化成一个存储过程，只调用一次数据库，从而缩短数据库的访问时间。

# 附录

## 附件

见文件列表

## 修过记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本 | 修改人 | 修改时间 | 修改内容 | 核准 |
| 1.0 | 黄天润 | 2020/4/19 |  |  |