|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| QQ截图20130904105420  **餐馆信息管理系统**  **系统概要设计报告**  设计人员： 2021112925-吴平凡  2021112923-袁旭阳  2021112937-李 品  指导教师 李 威   |  |  |  | | --- | --- | --- | | **文件状态：**  **[√ ] 草稿**  **[ ] 正式发布**  **[ ] 正在修改** | **文件标识** | **1.0** | | **当前版本** | **V1.0** | | **拟 稿 人** | 吴平凡 | | **拟稿日期** | 2024/4/22 | | **审 核 人** |  | | **审核日期** |  | |

编写说明

标题：系统概要设计报告

类别：文档

存放位置：项目文档\03、系统设计\系统概要设计报告.docx

编辑软件：Miscrosoft Word 2021 中文版

版本历史：

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 作者 | 日期 | 备注 |
| V1.0.0 | 吴平凡 | 2024/4/22 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

完成情况分工：

|  |  |  |  |
| --- | --- | --- | --- |
| 学号 | 姓名 | 工作量 | 完成工作内容 |
| 2021112923 | 吴平凡 | 60% | 撰写初稿，完成基本文档框架，格式统一化。 |
| 2021112925 | 袁旭阳 | 20% | 文档的审核与修改，各种图形的绘制。 |
| 2021112937 | 李品 | 20% | 软件需求规格说明书的审核与润色。 |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1. 引言    1. 编写目的       1. 明确系统架构和设计方案： 对系统的整体架构进行设计，包括系统的组成模块、模块之间的关系、以及各个模块的功能和实现方式等。这将为后续的详细设计和开发提供指导，并确保系统的各个部分能够协调一致地工作。       2. 规范开发过程和编码标准： 制定系统的开发规范和编码标准，包括代码风格、命名规则、接口定义等。这将有助于提高代码的质量和可维护性，并确保团队成员能够有效地协作开发。       3. 评估系统可行性和风险： 对系统的可行性和风险进行评估，包括技术可行性、经济可行性、以及潜在的风险和问题等。这将帮助项目管理者做出决策，并制定相应的风险应对措施。       4. 为后续开发阶段提供依据： 概要设计是详细设计和开发阶段的重要依据，它将为后续的开发工作提供详细的技术方案和指导。    2. 项目背景       1. 餐饮行业竞争激烈，需要提高运营效率：随着社会经济的发展，人们的生活水平不断提高，外出就餐已经成为一种普遍的生活方式。餐饮行业竞争日益激烈，餐馆需要不断提高运营效率才能保持竞争力。传统的餐馆管理方式依靠人工操作，效率低下，容易出错。餐馆管理系统可以有效地解决这些问题，帮助餐馆提高运营效率和服务质量。       2. 餐馆管理需求复杂，需要信息化手段进行管理：餐馆的运营涉及菜品管理、餐桌管理、订单管理、库存管理、财务管理等多个方面，管理需求复杂。传统的餐馆管理方式依靠人工管理，难以满足日益复杂的需求。餐馆管理系统可以将餐馆的各种管理工作进行信息化，提高管理效率和准确性。       3. 信息技术发展，为餐馆管理系统提供技术支撑：近年来，信息技术迅猛发展，为餐馆管理系统提供了强大的技术支撑。云计算、大数据、人工智能等技术的应用，使得餐馆管理系统更加智能化、人性化，能够更好地满足餐馆的管理需求。    3. 相关术语       1. 菜品管理：指对餐馆提供的菜品进行添加、修改、删除和查询的过程。       2. 餐桌管理：涉及到餐桌的分配、预订、清理和维护等功能。       3. 点餐员工作流程：服务员接收订单、服务和结账的整个操作流程。    4. 参考文献   [1]Object Management Group®, OMG® . Unified Modeling Language.2017-12.[<https://www.omg.org/spec/UML/2.5.1/>]  [2][Abraham Silberschatz](https://book.douban.com/search/Abraham%20Silberschatz) / [Henry F. Korth](https://book.douban.com/search/Henry%20F.%20Korth) / [S. Sudarshan](https://book.douban.com/search/S.%20Sudarshan).数据库系统概念(7th ed.).([杨冬青](https://book.douban.com/search/%E6%9D%A8%E5%86%AC%E9%9D%92)/[李红燕](https://book.douban.com/search/%E6%9D%8E%E7%BA%A2%E7%87%95)/[张金波](https://book.douban.com/search/%E5%BC%A0%E9%87%91%E6%B3%A2%20%E7%AD%89)译).机械工业出版社,2021-6   1. 系统体系结构设计    1. 系统特点分析       1. 功能完整性: 系统涵盖了餐馆运营所需的全部功能，包括菜品管理、餐桌管理、点餐管理、库存管理、财务管理、数据分析等。       2. 易用性: 系统界面简洁易懂，操作简单，方便用户使用。       3. 性能: 系统运行速度快，稳定性好，能够满足餐馆日常运营的需求。       4. 安全性: 系统采用安全可靠的技术方案，能够保护用户数据安全。       5. 可扩展性: 系统架构设计合理，支持根据业务需求进行扩展。    2. 系统体系结构设计       1. 系统体系结构模式   采用MVC（模型-视图-控制器）模式，以分离内部业务逻辑和用户界面。   * + 1. 系统体系结构设计        1. 逻辑视图设计   图2-1  说明：  Dish（菜品）  Table（餐桌）  Order（订单）  Waiter（服务员）  关系描述：Waiter与Order关联，Order与Dish和Table关联。   * + - 1. 开发及运行视图设计   开发视图：  图2-2  说明：  1. 用户界面层（Presentation Layer）   1. 管理员界面：提供菜品和餐桌信息管理功能，包括添加、删除、修改菜品和餐桌信息的界面。 2. 点餐员界面：用于点菜和排队信息的记录，以及餐桌状态的管理，包括查看菜品信息、记录点菜和排队信息、修改餐桌状态等功能。   2. 应用层（Application Layer）   1. 菜品管理模块：负责处理管理员对菜品信息的增删改查操作。 2. 餐桌管理模块：处理管理员对餐桌信息的增删改查操作。 3. 点餐服务模块：处理点餐员记录点菜，以及修改餐桌状态等操作。 4. 排队服务模块：处理排队排队信息，以及修改队伍状态等操作。   3. 领域层（Domain Layer）   1. 菜品对象：定义菜品的属性和行为，如菜名、价格、描述等。 2. 餐桌对象：定义餐桌的属性和行为，如编号、容纳人数、状态等。 3. 订单对象：记录顾客点菜信息，包括菜品、数量、餐桌等信息。 4. 队列对象：记录顾客排队信息，包括排队中，入座等信息。   4. 数据访问层（Data Access Layer）   1. MySQL数据库：存储菜品信息、餐桌信息和订单信息等数据。   5. 技术选型   1. 前端框架：使用Qt框架实现用户界面，提供跨平台的图形界面开发能力。 2. 后端数据库：采用MySQL数据库存储系统数据，提供数据持久化支持。    * + 1. 运行视图：     图2-3  文字描述：详细说明各组件的功能和责任：  1. 管理员操作流程   1. 管理员登录系统。 2. 进入管理员界面，可以进行菜品管理和餐桌管理操作。 3. 在菜品管理界面，管理员可以进行菜品信息的查看、添加、删除和修改操作。 4. 在餐桌管理界面，管理员可以进行餐桌信息的查看、添加、删除和修改操作。   2. 点餐员操作流程   1. 点餐员登录系统。 2. 进入点餐员界面，可以查看菜品信息和处理点餐订单。 3. 点餐员可以查看菜品列表，选择顾客点菜。 4. 点餐员记录顾客点菜信息，并将订单信息提交。 5. 点餐员可以管理餐桌状态，标记餐桌为空闲、就餐中或需要清洁等状态。   3. 系统运行流程   1. 用户通过客户端（管理员界面或点餐员界面）登录系统。 2. 客户端发送登录请求到服务器端。 3. 服务器端验证用户身份，返回登录结果。 4. 登录成功后，客户端获取相应的权限和菜单信息。 5. 用户根据权限进行相应的操作，如管理员进行菜品和餐桌管理，点餐员进行点餐和餐桌状态管理。 6. 客户端将用户操作发送到服务器端进行处理。 7. 服务器端更新数据库中的相关数据，如菜品信息、餐桌状态等。 8. 客户端接收服务器端的响应，并更新界面显示。   4. 数据库交互   1. 客户端发送数据请求到服务器端。 2. 服务器端接收请求，进行相应的数据库操作。 3. 服务器端将数据库操作结果返回给客户端。   5. 实时监控与数据分析   * + 1. 系统实时监控餐桌状态和排队情况。     2. 系统收集并分析顾客点菜数据，生成报表和统计信息。     3. 管理员和点餐员可以查看实时监控和数据分析结果，进行相应的调整和优化。        1. 部署视图设计   部署图：展示系统的物理部署在服务器和客户端设备上。  图2-4  文字说明：以下是硬件要求和网络配置描述：  硬件要求：  1. 服务器端硬件要求：租用微软的专用服务器  2. 数据库服务器硬件要求：  与服务器端相同：数据库服务器与应用服务器部署在同一台物理服务器上，因此硬件要求与服务器端相同。  3. 客户端硬件要求：   1. 处理器：客户端windows系统。 2. 内存：客户端应用通常不需要大量内存，但至少需要足够的内存来运行操作系统和应用程序。 3. 存储：足够的存储空间来存储操作系统、应用程序和用户数据。 4. 网络接口：适配设备的网络接口，通常为Wi-Fi或移动数据连接。 5. 接口设计    1. 外部接口       1. 顾客端接口:          1. 接口名称: 查询菜品信息          2. 接口参数: 菜品名称、分类等          3. 接口返回值: 菜品信息列表  |  | | --- | | {  "接口名称": "查询菜品信息",  "接口参数": {  "菜品名称": "string",  "分类": "string"  },  "接口返回值": {  "菜品信息列表": [  {  "菜品ID": "string",  "菜品名称": "string",  "菜品价格": "number",  "菜品描述": "string",  "菜品图片URL": "string",  "分类": "string"  },  // 更多菜品信息对象  ]  }  } |  * + 1. 供应商端接口:        1. 接口名称: 采购食材        2. 接口参数: 食材名称、数量、价格等        3. 接口返回值: 采购成功/失败信息  |  | | --- | | {  "接口名称": "采购食材",  "接口参数": {  "食材名称": "string",  "数量": "number",  "价格": "number"  },  "接口返回值": {  "采购结果": "string",  "成功信息": {  "采购订单ID": "string",  "采购总额": "number"  },  "失败信息": {  "错误码": "string",  "错误信息": "string"  }  }  } |  * + 1. 第三方支付平台接口:        1. 接口名称: 发起支付        2. 接口参数: 订单金额、支付方式等        3. 接口返回值: 支付成功/失败信息  |  | | --- | | {  "接口名称": "发起支付",  "接口参数": {  "订单金额": "number",  "支付方式": "信用卡/借记卡/微信支付/支付宝"  },  "接口返回值": {  "支付状态": "string",  "信息": "支付已成功处理/支付未成功处理"  }  } |  * 1. 内部接口      1. 菜品管理接口:         1. 接口名称: 新增菜品         2. 接口参数: 菜品名称、价格、库存数量、所属分类等         3. 接口返回值: 新增菜品成功/失败信息      2. 订单管理接口:         1. 接口名称: 创建订单         2. 接口参数: 顾客信息、菜品信息、订单金额等         3. 接口返回值: 订单号  1. 系统数据库设计    1. 概念数据库设计       1. ER图：展示实体Dish、Table、Order、Waiter及其联系。   图2-5   * 1. 逻辑数据库设计      1. 使用Navicat进行数据库逻辑建模。   图2-6   * + 1. 导出SQL脚本，创建数据库。  |  | | --- | | /\*  Source Server : test  Source Server Type : MySQL  Source Server Version : 80036  Source Host : mycgck.mysql.database.azure.com:3306  Source Schema : 餐馆信息管理系统  Target Server Type : MySQL  Target Server Version : 80036  File Encoding : 65001  Date: 10/05/2024 14:04:09  \*/  SET NAMES utf8mb4;  SET FOREIGN\_KEY\_CHECKS = 0;  -- ----------------------------  -- Table structure for 收银记录  -- ----------------------------  DROP TABLE IF EXISTS `收银记录`;  CREATE TABLE `收银记录` (  `来客\_日期` date NOT NULL,  `来客\_编号` int NOT NULL,  `时间` time NOT NULL,  `金额` float NOT NULL,  `支付方式` varchar(16) CHARACTER SET utf8mb3 COLLATE utf8mb3\_general\_ci NOT NULL,  PRIMARY KEY (`来客\_日期`, `来客\_编号`, `时间`) USING BTREE,  CONSTRAINT `收银记录\_ibfk\_1` FOREIGN KEY (`来客\_日期`, `来客\_编号`) REFERENCES `来客` (`日期`, `编号`) ON DELETE RESTRICT ON UPDATE RESTRICT  ) ENGINE = InnoDB CHARACTER SET = utf8mb3 COLLATE = utf8mb3\_general\_ci ROW\_FORMAT = DYNAMIC;  -- ----------------------------  -- Table structure for 来客  -- ----------------------------  DROP TABLE IF EXISTS `来客`;  CREATE TABLE `来客` (  `日期` date NOT NULL,  `编号` int NOT NULL,  `人数` tinyint NOT NULL,  `座位数要求` tinyint NOT NULL,  `排队时间` time NULL DEFAULT NULL,  `入座时间` time NULL DEFAULT NULL,  `桌号` char(5) CHARACTER SET utf8mb3 COLLATE utf8mb3\_general\_ci NULL DEFAULT NULL,  PRIMARY KEY (`日期`, `编号`) USING BTREE,  INDEX `桌号`(`桌号`) USING BTREE,  CONSTRAINT `来客\_ibfk\_1` FOREIGN KEY (`桌号`) REFERENCES `餐桌` (`桌号`) ON DELETE RESTRICT ON UPDATE RESTRICT  ) ENGINE = InnoDB CHARACTER SET = utf8mb3 COLLATE = utf8mb3\_general\_ci ROW\_FORMAT = DYNAMIC;  -- ----------------------------  -- Table structure for 点菜记录  -- ----------------------------  DROP TABLE IF EXISTS `点菜记录`;  CREATE TABLE `点菜记录` (  `来客\_日期` date NOT NULL,  `桌号` char(5) CHARACTER SET utf8mb3 COLLATE utf8mb3\_general\_ci NOT NULL,  `客户点菜编号` int NOT NULL,  `来客\_编号` int NOT NULL,  `菜名` varchar(32) CHARACTER SET utf8mb3 COLLATE utf8mb3\_general\_ci NOT NULL,  `时间` time NOT NULL,  `状态` varchar(5) CHARACTER SET utf8mb3 COLLATE utf8mb3\_general\_ci NOT NULL,  PRIMARY KEY (`来客\_日期`, `客户点菜编号`) USING BTREE,  INDEX `来客\_日期`(`来客\_日期`, `来客\_编号`) USING BTREE,  INDEX `菜名`(`菜名`) USING BTREE,  CONSTRAINT `点菜记录\_ibfk\_1` FOREIGN KEY (`来客\_日期`, `来客\_编号`) REFERENCES `来客` (`日期`, `编号`) ON DELETE RESTRICT ON UPDATE CASCADE,  CONSTRAINT `点菜记录\_ibfk\_2` FOREIGN KEY (`菜名`) REFERENCES `菜品` (`菜名`) ON DELETE RESTRICT ON UPDATE CASCADE  ) ENGINE = InnoDB CHARACTER SET = utf8mb3 COLLATE = utf8mb3\_general\_ci ROW\_FORMAT = DYNAMIC;  -- ----------------------------  -- Table structure for 菜品  -- ----------------------------  DROP TABLE IF EXISTS `菜品`;  CREATE TABLE `菜品` (  `菜名` varchar(32) CHARACTER SET utf8mb3 COLLATE utf8mb3\_general\_ci NOT NULL,  `价格` float NOT NULL,  `折扣` float NOT NULL,  `材料` varchar(32) CHARACTER SET utf8mb3 COLLATE utf8mb3\_general\_ci NOT NULL,  `类型` varchar(32) CHARACTER SET utf8mb3 COLLATE utf8mb3\_general\_ci NOT NULL,  PRIMARY KEY (`菜名`) USING BTREE  ) ENGINE = InnoDB CHARACTER SET = utf8mb3 COLLATE = utf8mb3\_general\_ci ROW\_FORMAT = DYNAMIC;  -- ----------------------------  -- Table structure for 账户  -- ----------------------------  DROP TABLE IF EXISTS `账户`;  CREATE TABLE `账户` (  `账号` char(255) CHARACTER SET utf8mb3 COLLATE utf8mb3\_general\_ci NOT NULL,  `密码` char(255) CHARACTER SET utf8mb3 COLLATE utf8mb3\_general\_ci NULL DEFAULT NULL,  PRIMARY KEY (`账号`) USING BTREE  ) ENGINE = InnoDB CHARACTER SET = utf8mb3 COLLATE = utf8mb3\_general\_ci ROW\_FORMAT = DYNAMIC;  -- ----------------------------  -- Table structure for 餐桌  -- ----------------------------  DROP TABLE IF EXISTS `餐桌`;  CREATE TABLE `餐桌` (  `桌号` char(5) CHARACTER SET utf8mb3 COLLATE utf8mb3\_general\_ci NOT NULL,  `容量` int NOT NULL,  `状态` varchar(3) CHARACTER SET utf8mb3 COLLATE utf8mb3\_general\_ci NOT NULL,  PRIMARY KEY (`桌号`) USING BTREE  ) ENGINE = InnoDB CHARACTER SET = utf8mb3 COLLATE = utf8mb3\_general\_ci ROW\_FORMAT = DYNAMIC;  ……………省略…………… |  1. 系统出错处理设计    1. 出错信息       1. 一览表：列出可能的错误或故障，以及系统的响应信息和含义。  |  |  |  |  | | --- | --- | --- | --- | | 错误类型 | 错误描述 | 响应信息 | 含义 | | 登录错误 | 用户输入错误的用户名或密码，无法登录系统。 | "登录失败，请检查您的用户名和密码。" | 用户需要检查输入的用户名和密码是否正确，并重新登录。 | | 菜品信息错误 | 菜品信息不完整或错误，例如菜品名称、价格、库存数量等。 | "菜品信息错误，请检查并重新输入。" | 用户需要检查菜品信息是否完整和正确，并重新提交。 | | 点餐错误 | 顾客点餐时，选择的菜品已售罄或库存不足。 | "该菜品已售罄，请选择其他菜品。" | 用户需要选择其他菜品进行点餐。 | | 支付错误 | 支付过程中出现网络错误或支付系统故障。 | "支付失败，请重试或联系系统管理员。" | 用户需要重试支付或联系系统管理员寻求帮助。 | | 系统错误 | 系统出现未知错误，导致无法正常运行。 | "系统错误，请联系系统管理员。" | 用户需要联系系统管理员进行故障排除和修复。 |   表2-1   * + 1. 设计友好的出错提醒界面。        1. 例如“登陆错误”   图2-7   * 1. 补救措施  |  |  | | --- | --- | | 错误类型 | 补救 | | 登录错误 | 用户名和密码应区分大小写。密码应设置足够复杂，以提高安全性。 | | 菜品信息错误 | 菜品信息应由管理员统一维护，以确保准确性。系统应提供数据校验功能，防止错误信息的提交。 | | 点餐错误 | 系统应实时更新菜品库存信息，并及时提醒用户菜品售罄情况。可以提供菜品推荐功能，帮助用户选择其他菜品。 | | 支付错误 | 应选择可靠的支付系统，并定期进行维护。可以提供离线支付功能，以应对网络故障情况。 | | 系统错误 | 系统应记录错误日志，以便分析错误原因进行改进。可以提供系统监控功能，及时发现和解决系统问题。 |   表2-2 |