**CRM客户管理系统概要设计**

# 1.目标

对公司的营销活动进行管理,实现利润的最大化，提高企业的竞争力，降低成本。

# 2.需求分析

## 2.1干系人分析

CRM客户管理系统的干系人总共有三类：公司运营决策者，系统开发团队，项目客户与用户。以这个分类进行分析：

### 2.1.1 公司运营决策者

公司运营决策者即公司总裁，各个事业部负责人；而相关的经营管理部门包含经营管理部，项目管理部。

（一）决策者的愿景：

1、高效

能够更好的管理客户关系，项目的成本预算以及分析可能存在的风险，提高工作效率。

2、好上手

开发平台的技术选型，要充分考虑到上手的容器度，降低学习成本。

4、技术选型要有激励性

开发平台的技术选型不能太生僻或者冷门，降低员工的学习热情，增加人员流动风险。

5、先进

所选技术不能太落后，应该是流行前沿的技术，保证能够在未来较长的时间内依然保持领先，能够在市场的竞争中保持技术优势和占有先机。

（二）经营管理者的愿景

1. 项目能够有序开发，里程碑清晰，按计划正常执行

2、项目能够控制成本

### 2.1.2系统开发团队

平台开发团队拥有固定的责任人，以项目经理和产品经理角色出现，此外还配备整套的非固定人员团队。细分为：责任人、研发人员、美工、测试人员。

从需求分析角度来说，平台研发团队的各个干系人是开发平台的研发需求提供者，在设计“开发架构”时有相应的愿景。

1、责任人（项目经理、产品经理、技术总监）

项目角度，希望项目能够有序的迭代，形成产品。各个组件拥有各自的版本，各自独立。

管理角度，希望能够以敏捷的方式把各个环节管理起来，做到有序。

规划角度，希望产品能够拥有持续可发展机制，路线图清晰，成本可控，资源稳定。

产品角度，希望能够满足各个干系人的要求，架构能够满足产品需要。

2、研发人员

希望，任务清晰；开发简单；环境清晰，简易；沟通渠道，信息渠道畅通明了。

3、美工

希望，前台框架充分独立；开发者能遵循统一的前台开发规范，尤其是样式开发。

1. 测试

希望，前台能够遵循测试规范进行开发；环境能够自动构建，版本清晰；建设期，设计稳定；沟通畅通，尽早参与设计，了解系统。

### 2.1.3项目客户与用户

客户：系统美观大方，功能稳定，满足需求。

用户：系统美观，易用。

## 2.2业务需求

将营销管理、实施管理、经营管理三个领域应用三位一体化，帮助企业提升整体营销管理竞争力、提高销售工作效率、提高服务满意度顾客忠诚度。

## 2.3非功能需求

### 2.3.1软硬件环境需求

系统应可运行于Windows7及其以上的版本

可通过浏览器访问，使用IE6.0或更高版本可顺利完成全部系统操作。

系统运行于局域网环境中。

系统数据库使用mysql。

### 2.3.2 性能需求

本系统在正常的网络环境下，应能够保证系统的及时响应

系统运行稳定。

功能模块相应响应时间不超过5秒

### 2.3.3 安全保密需求

本系统的系统架构，以及权限机制可以保证系统的安全性。

系统架构上保证使服务器数据源与客户端分离，保证数据的物理独立性； 系统的用户授权机制通过角色的定义管理实现，通过定义某些角色能进行的操作权限，和定义用户拥有的角色，限定用户的操作权限，实现对用户的授权。

### 2.3.4可维护性和扩展性

系统的设计应该保证系统今后能够随企业的发展进行扩展，同时有良好的可维护性。

## 2.4核心需求

因为工期较短，快速开发时要减少层，如果数据库没有变化，不该设计持久层。

# 3.设计思想

## 3.1 设计原则

架构设计将遵循3个原则，“自动化”、 “展示、行为分离”、 “保护应用”的原则。以下是对这3个原则的解释：

* 自动化

所谓自动化，主要是针对环境管理以及代码生成。首先，在搭建环境和部署实施过程中，有大量的操作行为。然而这些行为都要求严格的标准规范，操作相对繁琐，影响开发的进入。我们在这里可以使用脚本来解决，这一系列的繁琐的操作。其次，在一些简单的业务编写过程中，大量的重复逻辑分散在代码中，然而，代码生成可以解决这些重复劳动。通过代码生成，可以生成简单业务的快速编写。

* 分层隔离

应用层序，被理解成抽象的三层，UI层、应用层、服务层。是在考虑到服务层的稳定性以及UI层与应用层的不稳定性，要充分设计到分层隔离。对业务的修改严格关闭，对展现与数据源的变更进行开放。

* 展示、行为分离

展示、行为分离，主要是指前台应用技术，通过前台页面的MVC模式，把展现形式与渲染逻辑以及前台业务逻辑分离。

* 保护应用

使用POJO的概念，保护业务实现，后台基础框架搭建的过程中，尽可能的符合合成/聚合原则。

## 3.2约束

系统具有（不限于）以下明显约束

1. 一期时间进度在5月中旬
2. 必须基于java
3. 必须学习成本低
4. 必须画面美观，不落俗
5. 技术要先进
6. 开发人员不熟悉Bootstrap，Gulpfile等先进技术，一定要简易
7. 要支持多媒体终端
8. 平台必须拥有权限管理业务功能组件
9. 开发平台会持续更新

# 4.设计思路

设计思路是在进行架构设计时的思维方式，分别从抽象到细致，从不同的角度进行分析和设计。

## 4.1四个架构思维角度

* 面向整体：概念架构；

概念架构的思考角度比较抽象，是从总体抽象结构上对架构进行分析，通过概念架构，可以直观的对总体架构有个直观的认识。

* 面向结构：逻辑架构；

业务，职责、协作，结构与关系。通过逻辑架构可以看清楚业务划分。在我们的开发平台中，客户是二次开发团队，所以，之类非常容易让人混淆技术与业务。与传统项目不同，我们提供的很多的技术服务就是业务。

* 面向节点：物理架构；

物理节点，拓扑结构。通过物理架构，我们可以知道系统的部署状况。

* 面向成果：技术架构；

充分考虑到自身的技术结构、协议，使用的第三方的技术、依赖、协议以及结构，通过技术架构可以直观的看出架构的技术选型。

## 4.2持续更新

产品的持续更新要考虑两个方面，一是系统获取需求，并在原有版本上修改产生新版本，即release；二是，新的版本更新二次开发所依赖的引用，即upgrade。

# 5.架构设计

## 5.1概念架构

概念架构的思考角度比较抽象，是从总体抽象结构上对架构进行分析，通过概念架构，可以直观的对总体架构有个直观的认识。从抽象角度上来看，系统分为4层：数据层、业务层、展示层、终端层:

图 5-1概念架构图

### 5.1.1数据层

数据层包含两种数据，一种是关系型数据，存储在数据库中。其中数据库选择MySQL。同时，要对文件有上传下载以及业务挂接的管理。

同时，关系型数据并不做分布式存储，集中管理在一台数据库服务容器中。

### 5.1.2业务层

业务层包含两个内容，业务部分，技术部分。

业务包含CRM、SYS。

技术包含权限管理、操作日志、字典管理、文件管理等。

### 5.1.3展示层

对应业务层管理自身的UI层逻辑，设计展示层。

### 5.1.4终端层

终端层支持PC浏览器和手机端浏览器。在多媒体终端中，以Native app-like UI（展示画面像本地app程序一样的运行）的方式展现。

## 5.2 逻辑架构

也叫业务逻辑架构，通过逻辑架构可以知道业务包含的功能以及功能之间的关系

图 5‑2逻辑架构图

CRM客户管理系统包含两个子系统:SYS管理系统和CRM管理系统。其中SYS管理系统又包含了字典管理、权限管理、角色管理、单位管理、用户组管理和用户管理六个模块。其中字典管理定义数据库中某些特殊字段的含义和用途，维护方便，减少工作量。权限管理是为了限制用户的行为，当用户拥有权限时才能进行权限允许的操作；系统中用户的操作权限是通过角色来控制的，通过定义不同的角色来赋予用户相应的权限；单位管理是对企业的部门进行管理，可以增加和删除部门；用户组管理是为了将几个角色形成一个组，每一个用户至少属于一个用户组，用户组包含了用户，那么这个用户就拥有了一系列的角色，而不用挨个把角色赋予给用户；用户管理正对平台使用者进行管理，常涉及权限管理、角色管理、和用户组管理。权限被赋予在角色上，角色可以赋予给某个用户，这样该用户就拥有了该角色所具备的权限。多个相似用户形成一个用户组（以便管理）。通常情况下，系统在开通之初，会生成一个拥有最大权限的用户，其他用户由该用户创建。

CRM管理系统包含了营销管理、实施管理、经营管理三个模块。营销管理模块包含客户管理、联系人管理、线索管理、商机管理、销售人员管理、活动管理、竞争对手管理、合同管理和方案管理。其中客户管理管理销售已知客户信息；联系人管理管理所有对外联系人；线索管理管理由客户获知的线索；商机管理将已知的线索转换为商机；销售人员管理管理公司内的销售人员；活动管理管理销售活动相关信息；竞争对手管理管理竞争对手信息，包括公司规模、主要市场、主要产品、方案、案例等；合同管理管理公司内的合同文件；方案管理管理司内售前方案和项目方案。

实施管理模块包含项目管理、进度管理、文档管理、实施人员管理。其中项目管理用来管理项目；进度管理在项目启动后跟踪进度；文档管理管理项目产生的重要文档；实施人员管理对负责项目实施的人员进行管理。

经营管理模块包含预算管理、回款管理、成本管理和经营分析。其中预算管理对项目的经营和销售进行预算；回款管理及时对销售款项进行回收，最大限度地降低经营风险；成本管理对销售、实施和差旅工作中产生的成本进行管理；经营分析对公司经营、产品的销售以及项目中可能存在的风险进行分析，最大限度的降低风险。

## 5.3技术架构

技术架构是面向成果的，它关心部署成果物时的技术应用。其视角是自身的技术结构、协议，使用的第三方的技术、依赖、协议以及结构，通过技术架构可以直观的看出架构的技术选型，以及层次结构下技术的结构与协议关系。以下是技术架构图：

图 5‑3技术架构图

### 5.3.1数据层

#### 5.3.1.1文件管理

文件采用集中分布式管理技术实现，不论是部署在本地还是部署在远端，都被分装实现，初步设计采用FTP技术作为文件存储管理协议。至于在应用调用时或者前台引用时，由Controller封装并跳转并转向文件服务，当然这里要用接口封装，以便替换。其所拥有的数据存储在关系型数据库中。

#### 5.3.1.2数据库管理

数据库管理存在两种方案，一个是集中式部署，一个是分布式部署。

（一）集中式部署

数据库管理，目前采用集中式部署，伪分布式实现的方式进行管理，所采用的分布式协议为JTA。

（二）分布部署

如果项目中有分布式需求，可以采用数据库分布式部署结构。其中对于交叉数据依赖和引用可以使用：

MySQL

使用FEDERATED和Replication技术来模拟远端表格为本地表格数据。

### 5.3.2公共层

所谓公共层，是指平台搭建和运行过程中后台应用程序使用的支撑工具。比如日志，工作流，测试，容器，ORM等等。

在平台搭建和运行过程中，我们的后台应用程序使用以下支撑工具：

表 5‑3-2公共层技术选型

|  |  |  |
| --- | --- | --- |
| 序号 | 技术名称 | 用途 |
| 1 | Logback | 运行日志 |
| 2 | Spring | 容器管理，负责IOC和AOP |
| 3 | Hibernate | 负责ORM持久化管理 |
| 4 | Activity | 工作流 |
| 5 | Jackson | JSON转换工具 |
| 6 | pjl-comp-filter | 请求响应压缩工具，压缩率35% |
| 7 | jansi | 让日志以多种颜色在控制台输出，方便调试 |

### 5.3.3领域层

领域层使用Interface的方式对controller层提供服务，同时内部也互相以Interface的方式提供服务。换句话说，就是领域层以面向接口编程的方式编写。

事务，在此层配置。保护业务逻辑的稳定性。

### 5.3.4应用层

应用层使用集中分发的处理，让所有展示层请求都从同一个入口进入。能够保证异常、上下文等的有效支持。

与展示层沟通时，使用DTO概念，通过JSON的格式进行传输。

### 5.3.5展示层

展示层使用Bootstrap框架。通过AngularJS技术实现展示、行为分离。通过JQuery实现前台验证。

## 5.4 物理架构

物理框架是面向部署的，部署起来十分简单，是一个非常单一的web容器加上一个数据库服务器。至于未来分布式，无非加几台数据库服务器，应用服务仍然集中部署。以下是其拓扑图：



图 5‑4交付时拓扑图

# 6.具体设计

## 6.1 字典管理

定义数据库中某些特殊字段的含义和用途

## 6.2 权限管理

限制用户的行为，当用户拥有权限时才能进行权限允许的操作

## 6.3 角色管理

通过定义不同的角色来赋予用户相应的权限

## 6.4 单位管理

对企业的部门进行管理

## 6.5 用户组管理

方便为用户赋权

## 6.6 用户管理

对平台使用者进行管理

## CRM管理

### 6.7.1 营销管理

#### 6.7.1.1 客户管理

管理销售已知客户信息

#### 6.7.1.2 联系人管理

管理所有对外联系人

#### 6.7.1.3 线索管理

管理由客户获知的线索

#### 6.7.1.4 商机管理

将已知的线索转换为商机

#### 6.7.1.5 销售人员管理

管理公司内的销售人员

#### 6.7.1.6 活动管理

管理销售活动相关信息

#### 6.7.1.7 竞争对手管理

管理竞争对手信息，包括公司规模、主要产品、方案、案例等

#### 6.7.1.8 合同管理

管理公司内的合同文件

#### 6.7.1.9 方案管理

管理司内售前方案和项目方案

### 6.7.2 实施管理

#### 6.7.2.1 预算管理

对项目的经营和销售进行预算

#### 6.7.2.2 回款管理

及时对销售款项进行回收，最大限度地降低经营风险

#### 6.7.2.3 成本管理

对销售、实施和差旅工作中产生的成本进行管理

#### 6.7.2.4 经营分析

对公司经营、产品的销售以及项目中可能存在的风险进行分析，最大限度的降低风险

# 7.实现