|  |
| --- |
| XXX产品  **系统架构设计说明书** |

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 版本 | 说明 | 作者 |
|  |  |  |  |

目录

[1. 设计目标 2](#_Toc527733216)

[2. 系统集成设计 3](#_Toc527733217)

[2.1 系统关系图 3](#_Toc527733218)

[2.2 外部接口设计 4](#_Toc527733219)

[3. 领域设计 4](#_Toc527733220)

[3.1 领域模型 4](#_Toc527733221)

[3.2 核心业务流程设计 5](#_Toc527733222)

[4. 组件设计 6](#_Toc527733223)

[4.1 总体设计 6](#_Toc527733224)

[4.2 XXX组件 6](#_Toc527733225)

[5. 数据设计 6](#_Toc527733226)

[5.1 数据模型 6](#_Toc527733227)

[5.2 数据流 6](#_Toc527733228)

[6. 技术实现设计 7](#_Toc527733229)

[6.1 技术架构 7](#_Toc527733230)

[6.2 部署架构 7](#_Toc527733231)

[6.3 质量属性设计 7](#_Toc527733232)

*【文档编写说明】*

***本部分和文档内容示例部分，在正式文档中应删除***

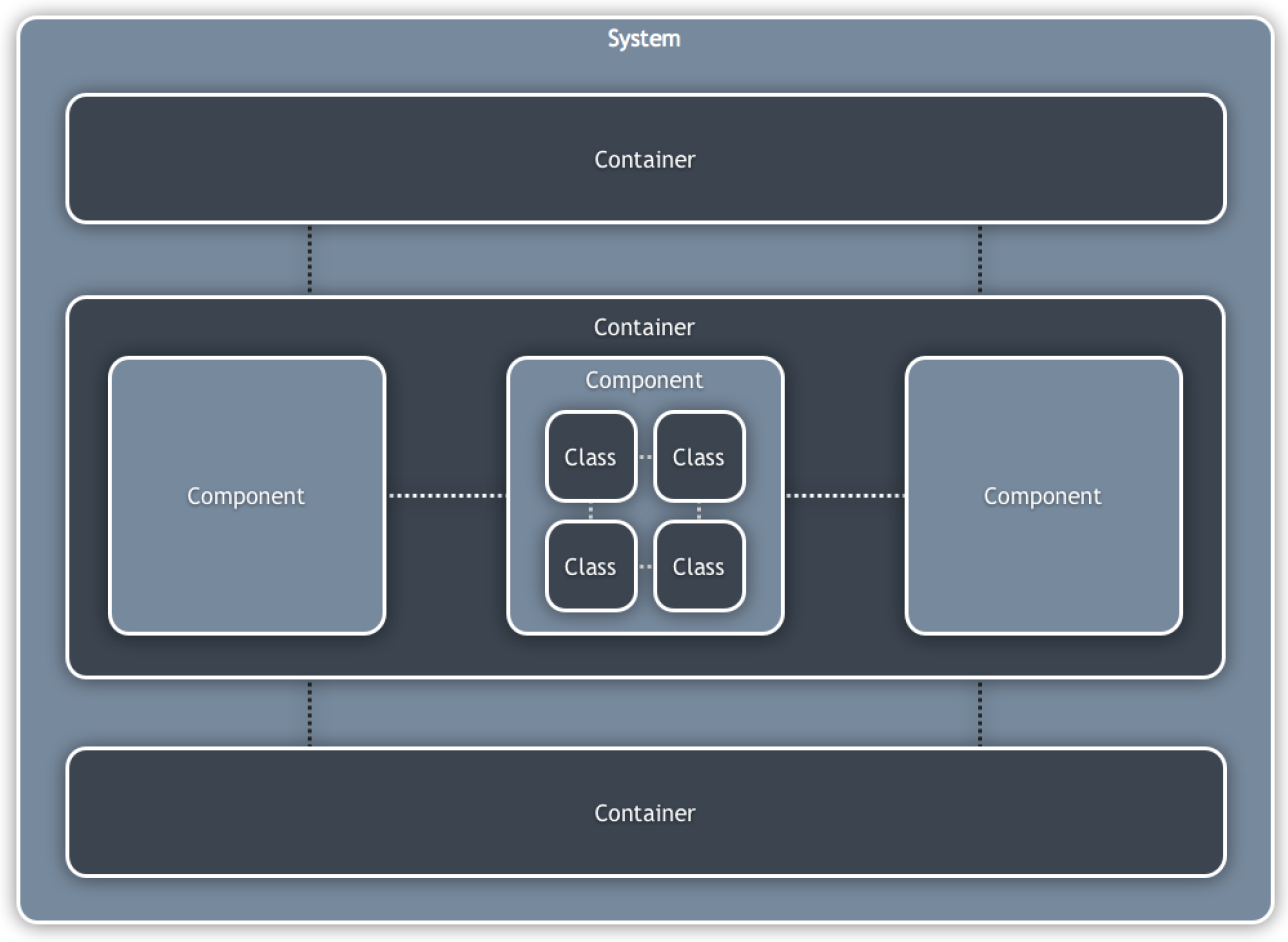
*系统架构设计的总体思路是：*

1. *明确设计目标，即需要设计一个什么样的系统，该系统在生产环境与其他系统和外部实体是如何交互的？功能需求和非功能需求到底是什么？*
2. *系统的概念设计，对业务进行分析，梳理出系统的业务领域（模块、子系统）、重要的概念（对象、数据实体）、以及概念之间的关系。并形成领域模型，具体可参考《领域驱动设计》；*
3. *进行逻辑设计，根据概念设计的领域模型，设计出系统的逻辑模型（组件或服务、以及组件内部的程序包、对象、数据库结构等逻辑对象），并按照组件概要说明基本结构。*
4. *进行部署设计，设计组件如何部署到运行环境。将运行环境抽象为一个个容器（节点），说明逻辑组件或服务如何部署到容器中，以及容器之间的通讯关系。*
5. *运维监控设计。说明未来生产环境如何进行运维和监控的方案*
6. *技术选型，列表说明所有使用的操作系统、数据库、第三方软件、SDK,开源程序包等。*

*整体设计方法可抽象为”C4”方法。*

*即 系统(System)的架构= 系统上下文(Context) + 容器)Container) + 组件(Component) + 类(Class)*

*如下图所示：*

**

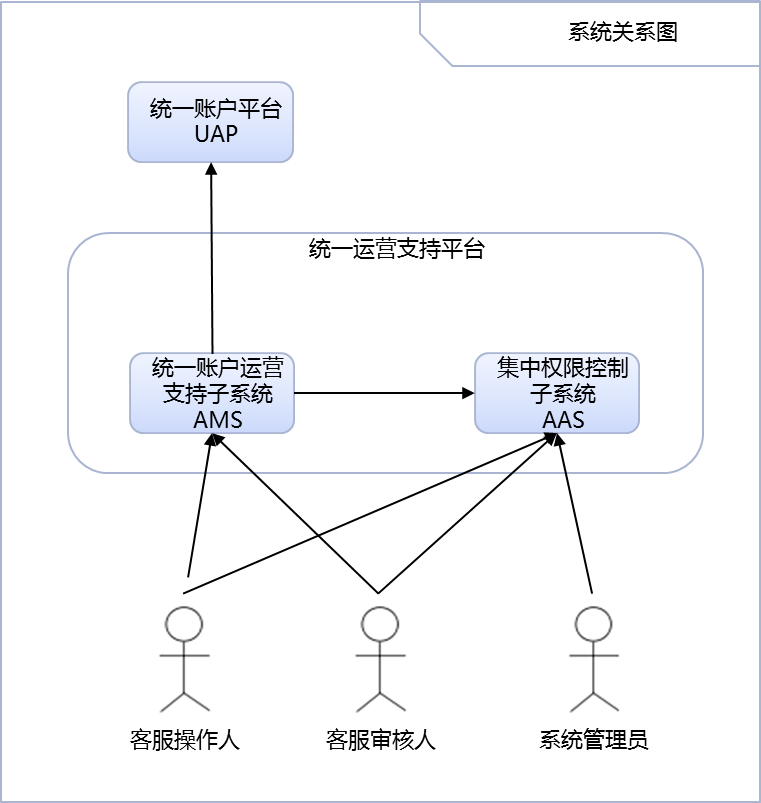
# 设计目标

|  |  |  |
| --- | --- | --- |
| 分类 | 要求 | 实现说明 |
| 技术层级 | 基础平台/技术平台/应用产品/大数据产品 |  |
| 性能 | * 系统高峰TPS要求 * 关键功能响应时间要求 |  |
| 容量 | * 用户容量：目标支持用户数、高峰用户数、预期增长率； * 线上数据规模容量，容量增长率 |  |
| 高可用性 | * 高可用指标(例如99.9%） |  |
| 安全 | * 使用环境（内/外网） * 安全合规要求 * 数据加密要求等 |  |
| 兼容性 | * 需要支持的浏览器版本， * 需要兼容的特殊软件版本 |  |
| 业务连续性 | * 系统BIA等级 * 数据备份恢复要求 * 系统容灾要求 |  |

# 系统集成设计

## 系统关系图

*系统需求的概要说明，以及系统外部关系的简单说明。通过系统关系图说明系统的主要外部集成关系。*



## 外部接口设计

* *说明系统对外提供的接口,此处仅说明接口大致设计。详细接口定义在《接口API规格书》中说明。*
* *说明系统依赖的外部系统接口。*

# 领域设计

## 领域模型

*说明系统的整体概念设计方案，说明根据需求设想的领域模型，对核心的概念进行说明。说明系统的核心（顶级）业务流程和核心领域类(Class)以及模块划分设计。即系统拆分为哪些模块、类或服务，以及核心概念之间的关系。具体需要说明*

1. *核心的概念，通过类图进行简要说明*
2. *核心的流程，基于概念类图中的类，说明如何进行交互来实现核心业务流程*
3. *基于上述设计说明如何将系统拆分为结构化的模块和组件。*

*概念设计示例（为一个HR系统的员工管理模块的示例）*

*员工管理分为员工信息管理、员工配置管理、劳动合同管理和核心人员管理四个子模块，主要包括三个方面的工作：基本的人事信息管理，为高管提供决策支持，以及提供有关员工信息的报表。核心的概念模型如下：*



*系统提供****员工（核心概念）****入职，配置、离岗、离职、离退休、返聘等全生命周期的管理功能。*

***员工****入职后，由相关操作员进行员工信息录入，分配工号，员工可以有多种的扩展信息。信息管理维护分层进行：完全属于员工 个人情况而与工作无关，且该信息变动不影响任何业务的，可由员工自行维护；属于员工个人情况且与工作业务关系不大的，员工本人维护，人力资源部审核后生效；其他信息均由人力资源部进行管理维护。*

*员工入职后，可将员工分配到工作结构（机构，岗位、工作地域）上。一个员工可以有多种****分配方案（核心概念）****，但是至少有一个****主分配方案****。比如一个员工可以分配到某个机构的某个岗位作为主分配方案，同时还可兼职到其他机构。根据员工分配类型的不同，系统提供不同的操作流程，主要有岗位聘用、借聘、交流聘用、返聘等几种。*

*可以将任意多个员工组合起来建立****员工组（核心概念）****，方便操作员的查询统计和其他业务操作。员工组是员工的集合，各个员工组之间相互独立。比如可以为核心人员创建一个员工组；将参加某次培训的人员筛选出来创建一个员工组。*

*员工之间可以根据****关系类型****的不同建立多种****员工关系（核心概念）****，比如师徒关系等。*

## 核心业务流程设计

说明系统的核心业务流程是如何实现的，可通过活动图、时序图等方面进行概要说明；

# 组件设计

## 总体设计

*1、通过组件静态视图（一张或者多张）说明系统中包括的所有组件及相互依赖关系。*

*组件是系统粗粒度的结构单元，通过组件之间的交互调用完成系统的业务逻辑处理。根据系统技术架构不同他可以是一个微服务，一个SOA的Service，一个EJB或者一个数据库存储过程*

### 组件清单

|  |  |  |  |
| --- | --- | --- | --- |
| 模块 | 组件 | 职责 | 技术实现方式 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## XXX组件

*针对领域模型中说明的每个模块，以及模块下的组件，进行概要的说明。*

*根据架构设计方法不同，需要分解到不同的粒度的组件，比如基于微服务的架构，每个微服务就是一个组件。基于传统结构化的设计，每个最小的模块就是一个组件。*

### 概要设计

1. *针对重要的关键组件，说明其实现方式、程序结构以及核心交互逻辑。通过类图、活动图等多种方式说明*

### 接口设计

*说明组件对外提供的接口，概要说明。详细的接口定义在其他文档体现；*

# 数据设计

## 数据模型

*1、通过E-R图等方式对数据模型设计进行说明*

*2、如果用到*Redis或*其他No-SQL数据库，也要对其进行说明*

## 数据流

*描述系统的核心数据流，通过数据流图等方式进行说明*

# 技术实现设计

## 技术架构

*1、描述系统总体技术架构图。重要的技术组件都应在途中进行说明*

*2、说明程序的技术分层结构及实现技术。例如接口层、数据访问层等*

*3、用到的关键技术服务也需要在图中说明，例如数据库、中间件、redis缓存等*

*4、通过列表说明各个部署容器和组件的选型决策：*

1. *操作系统、数据库、JVM等基础软件；*
2. *第三方SDK和开源组件库*
3. *第三方工具软件*
4. *如选择特殊硬件，也需说明*

*；*

## 部署架构

1. *通过部署图的方式说明系统各部署容器之间的结构和通讯关系。*
2. *说明部署容器与部署组件的装配关系。*
3. *此处仅说明简要架构图，详细说明在《部署方案》。*
4. *列表说明每个部署容器单元所需的服务器的资源与配置要求。*

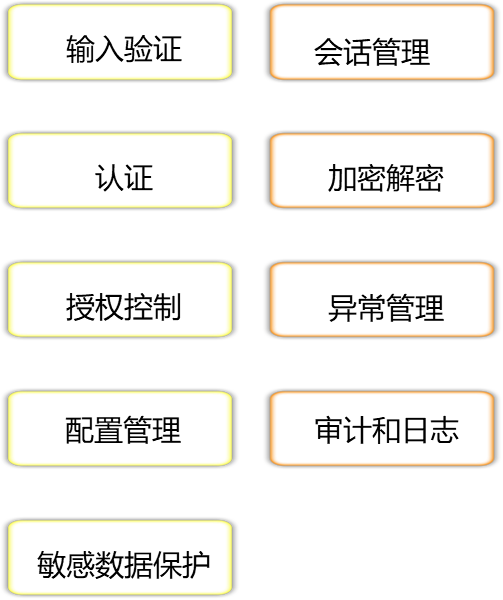
## 质量属性设计

### 容量与性能设计

### 可靠性设计

### 安全性设计

1. *通过Thread Model分析安全威胁及对应的安全方案。*
2. *说明针对安全问题的具体解决方案。*
3. *从应用层面说明以下重要的安全解决方案：*

**

### 可监控性设计

*说明系统层面、业务层面和应用程序层面的监控方案。*

1. *具体监控技术实现方案*
2. *业务指标，需要监控的业务指标及告警方案*
3. *应用指标，需要监控的应用程序指标及告警方案*

### 可维护性设计

*如有特殊的运维操作要求，在此进行方案说明*