物业自助终端体系结构设计文档

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本/状态 | 作者 | 参与者 | 起止日期 | 备注 |
| 草稿 | Zrz |  | 2016-6-15 | 开始合并所有的设计文档 |
| 变更 | Zrz |  | 2016-6-16 | 版本1.1A、B，需求相关变更 |
| 变更 | Zrz |  | 2016-6-21 | 版本1.2A  与产品部门确定采用“自助终端+互联网服务平台方案”（方案A）；  取消物管系统电子票开票状态回写接口；  取消物管系统开票接口，物管系统（如极致软件）中不再集成电子票功能；  服务平台中暂不开发缴费/交费网页界面，统一由自助终端承接之；  服务平台一期暂不处理报修业务 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# 引言

## 目的

系统用于支持自助终端系统中，【服务缴费】这个主要的业务过程，系统属于带有行业性质的O2O服务性电子商务系统，包括O2O电子商务中常见的集束服务，在线支付，会员营销等内容，同时还包括物业相关的业务内容及税务相关的防伪税控工作流。本文档主要用于描述业务系统的组成，功能细节，以及实现细节需要调用的相关接口。接口定义和接口实现的逻辑部分需要参考 系统的接口设计文档。

## 定义、首字母缩写词和缩略语

|  |  |
| --- | --- |
| **定义、首字母缩写词和缩略语** | **描 述** |
| DEMO、原型 | 系统在设计之前有静态页面描述的原型系统，页面的展示效果和交互效果请以原型为准 |
| 规格说明 | 指代《航天物业电子发票\_产品需求文档》 |
| 智慧云社区 / 服务平台 | 规格说明中称“PC端后台系统” |
|  |  |

## 阅读对象

|  |  |
| --- | --- |
| **阅读对象** | **阅读建议** |
| 需求 | 概要设计：请重点阅读第5章节内容，核实与外部系统对接的前期准备及约定是否已完成  详细设计：请重点阅读第6,7,8章节内容，核实是否覆盖需求 |
| 编码 | 除了需要开发的功能点，请重点阅读第5章节中相关的内容 |
|  |  |
|  |  |

## 参考资料

|  |  |
| --- | --- |
| **名 称** | **描 述** |
| 《航天物业电子发票\_产品需求文档\_20160615更新》 |  |
|  |  |
|  |  |

## 文档概述

第2章节主要描述系统的整体结构

第3,4章节主要对需求进行分析，并得出设计的相关约束和主要的设计策略

第5章节主要描述系统架构相关的设计内容

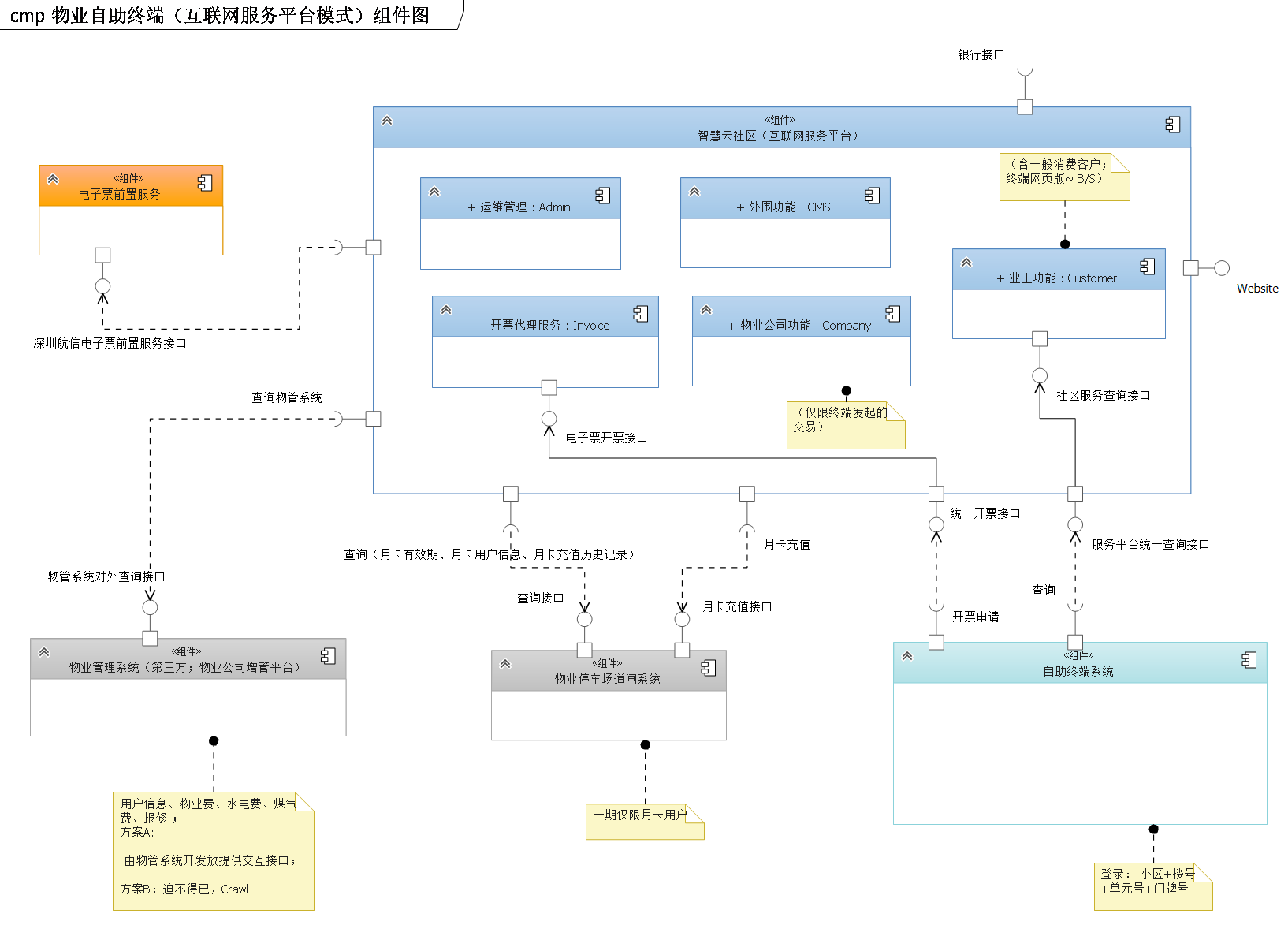
第6,7,8章节描述系统的各个组成部分的详细设计内容

第9章节描述用于支持系统的后台子系统部分的详细设计内容

第10章节描述系统的运行环境

# 系统概述

## 功能模块划分



1. 系统总共划分为五个组件：

A自助终端功能；

B社区服务云平台功能；

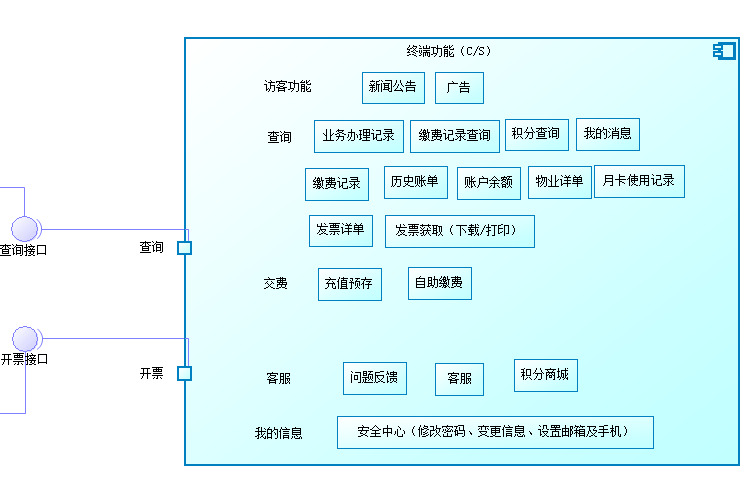
C电子票前置服务；

D物业管理软件；

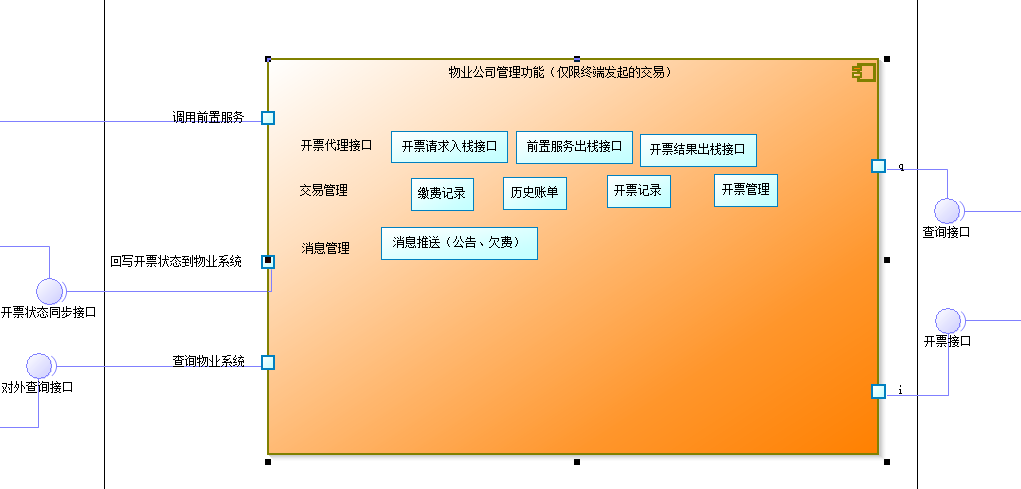
E物业停车场道闸管理软件。

其中D和E目前采用第三方实现；C 已由深圳航信电子票项目组实现。

1. 自助终端功能主要包括：
2. 访客功能
3. 查询
4. 交费
5. 客服
6. 用户中心

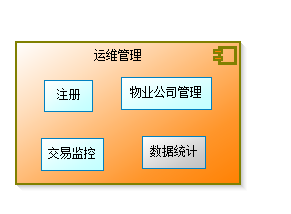


1. 社区服务云平台功能功能主要包括：
2. 开票代理接口
3. 交易管理
4. 消息管理

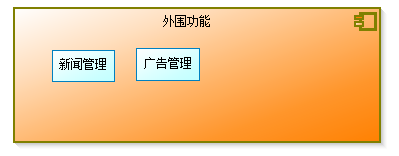


1. 运维管理功能主要包括支持系统运行相关的管理的监控功能。
2. 所有物业公司的基础字典，除了物业公司自己维护之外，客服需要权限进行操作。
3. 运维控制所有的用户注册和公司注册功能。
4. 运维还包括交易数据的监控以及异常处理功能。

系统上线阶段，可能的情况是，基础数据的初始化工作是由物业公司提供，或者从其他系统获取，通过运维的功能初始化到系统中。物业公司仅处理数据的核对和变更工作。



1. 外围功能包括的是网站结构中带有的功能，主要包括
2. 新闻动态，~~这部分内容预计使用深圳航天信息CMS，不再重新开发~~。
3. 论坛，暂不实现
4. 广告管理



1. 接口部分。
2. 物业公司需要维护的信息需要提供导入的接口，以方便物业公司对信息的维护工作；
3. 物业公司物管系统需提供常规业务查询接口，以便后台管理系统/自助终端获取业务信息；
4. 物业停车场道闸管理系统需提供查询和缴费接口；

# 设计约束

## 3.1需求约束

|  |  |
| --- | --- |
| **约束** | **描 述** |
| **功能约束** | 需要提供菜单、功能、按钮级权限控制 |
| **功能约束** | 需要支持对企业管理员授权、收权、停用；企业管理员对企业操作人员的注册、授权； |
| **功能约束** | 需要支持对数据行按指定字段值范围，进行授权使用，包括查看、修改、删除、审批、确认等，常用的如公司信息、物业费明细管理、折扣率或优惠额等 |
| **功能约束** | 平台管理员只能管理、授权用户企业管理员、平台操作员权限，无业务操作权限 |
| **功能约束** | 系统必须提供完善的审计功能，对系统关键数据的每一次增加、修改和删除都能记录相应的修改时间、操作人和修改前的数据记录 |
| **功能约束** | 需要英语、汉语（简体 、繁体）支持，包括界面、内容、提示预警等所有信息；一期不支持除英、汉(简体、繁体)之外的语言，但需要能方便扩展更多语言的支持 |
| **功能约束** | 需要跨浏览器支持，包括IE8及以上，Chrome，360和腾讯浏览器；一期不支持火狐浏览器、Opera浏览器，但需方便扩展更多浏览器支持 |
| **功能约束** | 平台支持超级管理员与平台管理员角色，平台超极管理员只能管理、授权平台管理员，无业务操作权限 |
| **~~功能约束~~** | ~~支持根据不同语言使用不同的日期格式~~ |
| **用户界面约束** | 系统发现用户提交有误信息，必须以弹出窗口的形式明确提示用户错误的原因，并把界面控制焦点置于发生错误的控件对象上。 |
| **用户界面约束** | 系统支持操作员登录后，不超过三次鼠标点击，即可访问业务所需功能 |
| **安全性约束** | 系统在解决跨站攻击、注入攻击等系统性安全问题上，具体标准参照《OWASP Top10–2013》 |
| **安全性约束** | 用户名全系统不重复，字母数字组合，不区分大小写，长度不小于5位；  密码由字母数字组合，区分大小写，长度不小于5位，采用MD5或更高级别加密； |
| **安全性约束** | 3次密码输入错误，要求输入验证码；10次以上错误，锁定用户名； |
| **可扩展性约束** | 需要支持原有逻辑扩展，在接口不变的情况，能方便增加新的逻辑实现。  需要支持新增逻辑在不影响原有业务、代码情况下，能增加新业务、新逻辑的处理分支  需要支持原逻辑再封装，支持在原逻辑封装基础上，进行再封装、不同逻辑组合。 |
| **稳定性约束** | 系统可以通过运行更多的实例或者采用分布式处理来支持更多的用户。可以通过线性的增加硬件设备、实例个数或者分布式处理点来处理更多的数据量，能更好的在不增加响应时间的前提下支持更多的业务处理。 |

## 3.2隐含约束

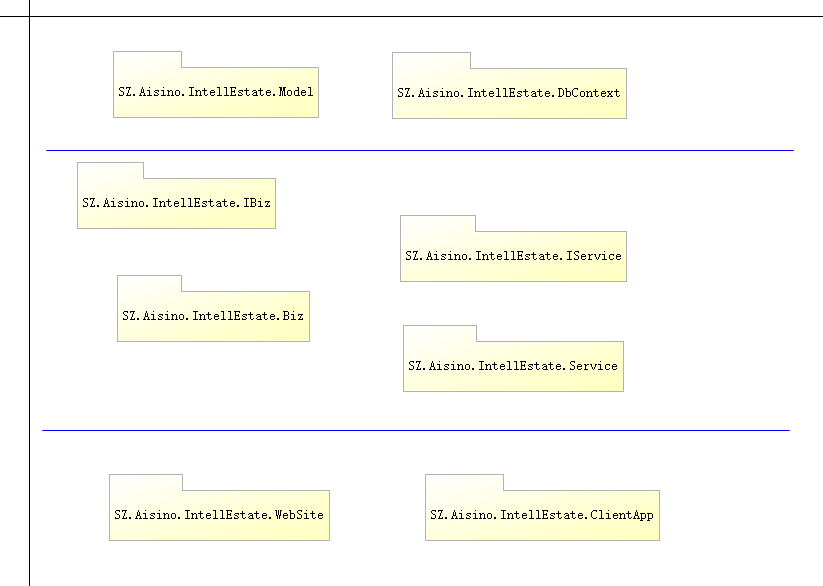
|  |  |
| --- | --- |
| **约束** | **描 述** |
| **功能约束** | 需求中提出的价格涉及多种货币，系统需要支持多币种（人民币、港币）的相关功能。 |
| **功能约束** | 需求中提出的报表部分，要求用户可自定义和扩展，因此系统需要支持自定义报表功能 |
| **功能约束** | 基础字典的管理需要同时支持通用字典和物业公司专用字典，物业公司在可以使用通用字典，专用字典，或者是专用字典+通用字典（专用字典中未定义的） |
| **功能约束** | 需求中未涉及运维相关的功能，但在某些逻辑中描述了运维的操作。因此系统需要有运维相关的功能。 |
| **性能约束** | 系统中使用的频度很高的数据需要有缓存机制，以增加页面的响应速度 |
|  |  |
|  |  |

# 设计策略

|  |  |
| --- | --- |
| **策 略** | **描 述** |
| **扩展策略** | 逻辑封装，面向接口编程  预计经常变化的内容采用XML配置文档将可变内容外置，扩展时不需要修改代码  Business层细化为两层，领域服务层（Biz）和服务facade层（Service）。其中Biz层基于领域模型划分为各个业务领域，封装具体的业务实现；Service层将Biz层的逻辑组织为大粒度的API，并有针对性地以 SOAP 或 Restful 的方式对外发布。 |
| **复用策略** | 与第三方系统对接信息采用XML配置，需要定义数据转换的处理接口，对接信息内容的生成要求可方便的进行扩展，适用于不同物业管理系统等其他需要交互的系统，可重用。 |
| **折衷策略** | 业务数据在遇到性能和功能冲突时，采用“分时”策略处理，将复杂操作拆分成多个简单操作  报表功能在遇到性能和功能冲突时，优先于功能，并通过SQL优化处理 |
| **数据关联策略** | 用户使用账户进行关联，不区分大小写  业务数据统一使用主键（Sql Server）或序列（Oracle）关联 |

# 总体设计

## 封装规范



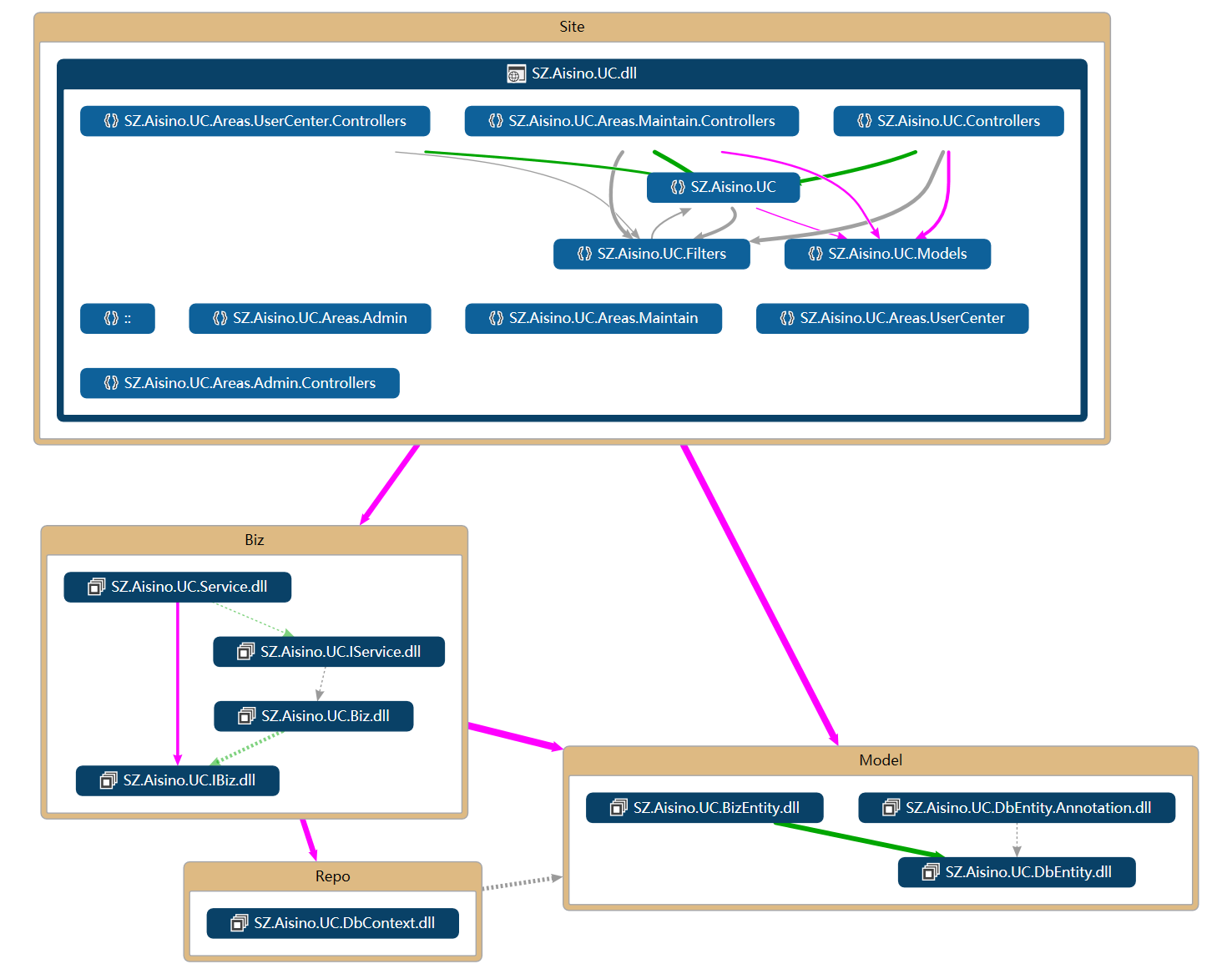
封装方案主体参考重构逻辑，三层架构。Model层和DbContext处理数据的映射和增删改查操作；Biz层和Service实现领域模型，Biz层封装可重用的逻辑代码，根据逻辑实现的不同分别封装不同的DLL，Service层对应大粒度的业务过程的封装。最终的应用层UI和逻辑分离。

### 接口定义

|  |  |  |
| --- | --- | --- |
| 所属系统 | 接口 | 说明 |
| 智慧云社区（互联网服务平台） | SZ.Aisino.IntellEstate.IQueryService. | 智慧云社区查询接口：互联网服务平台对外部系统有限开放的查询接口 |
| SZ.Aisino.IntellEstate.IEinvoiceService. | 统一开票接口：互联网服务平台对外部系统有限开放的电子票开票代理服务接口 |
| 电子票前置服务 |  | 深圳航信电子票前置服务接口 |
| 物业停车场道闸系统（第三方 ） | 三方待协商 | 查询接口（月卡有效期、月卡用户信息、月卡充值历史记录） |
| 三方待协商 | 月卡充值缴费接口  （月卡基准费率配置在“公司基础信息维护模块”中实现） |
|  |  |
| 物业管理系统（第三方 ） | 三方待协商 | 物管系统对外查询接口 |
| ~~三方待协商~~ | ~~开票状态回写接口~~ |

### 逻辑层级关系

#### SZ.Aisino.IntellEstate的层级关系



此处以服务平台-权限子系统为例说明：

1. 在FAÇADE层增加业务对象模型定义，根据业务对象打包前台接口，有利于前台接口的使用和管理。
2. DAL（数据操作层）和MODEL层（表字段映射）继续保持分离。
3. 业务逻辑分开打包，并用接口支持扩展。通过依赖注入，动态加载需要使用的操作实例。

### 类库命名规范

1. 类库命名格式定义为“SzAisino.<项目代码>.<类库分层>.<类库子分类>”。最大层级不超5 层。

项目代码暂定：IntellEstate

1. 程序集/包名称采用“驼峰”格式。
2. 类名称，方法名称和属性名称采用“驼峰”格式，类名称，属性、方法名称首字大写，字段名称首字小写。
3. 接口名称固定大写“I”字头；但如过领域模型的核心类首字母为I，则接口前缀“I\_”,如 I\_InformationSyncService
4. 函数/方法命名时采用“动词+名词”组合方式。

### 数据库设计规范

Oracle：

1. 数据表名称和字段名称采用全大写，“名词+(动词)”组合，单词之间使用下划线分割。例如订单主表可命名为“ORDER\_HEAD”，对应的跟踪表可命名为“ORDER\_HEAD\_TRACE”.
2. 数据表命名不要带有 HIS，TEMP，TMP，BAK，BACK等容易混淆的关键字。
3. 所有的数据表都需要带有唯一的SEQ关键字，并作为主键处理，除非特殊需要。SEQ的命名规则为“表名\_SEQ”。
4. 所有的数据表都需要带有时间戳字段和操作人字段。
5. 不允许使用触发器。
6. 一个数据表不要使用超过一个的BOLB（CLOB）类型字段。
7. 字符型的字段统一使用“VARCHAR2”格式，长度设置为预估长度的2-5倍。尽量不超过2000长度。
8. 所有的公司名称，堆场名称等涉及公司名称的字段，统一长度为VARCHAR2(2000)

### 数据库核心实体关系图



## 多语种支持解决方案

### 需求说明

需求提出需要对系统支持多语种方案，包括UI界面，提示信息和内容三个部分。

### 解决方案

#### UI界面的多语种支持

UI界面的多语种支持有多个解决方案：1）采用.net本身的语言包功能，在编码时绑定资源文件； 2）采用分站点开发，不同的语言单独部署不同的站点； 3）采用数据库的编码对照方式处理。

三种方案都能实现UI界面的多语种支持，相比较而言，使用语言包的方案和本地文件有关，在程序发布时不利于针对集群的部署。而采用不同站点的方案,UI维护的工作量会成倍增加，不利于功能的扩展和维护。因此采用数据库编码对照方式处理。

#### 提示信息的多语种支持

提示信息的多语种支持使用数据库编码方式处理，需要提示的信息先维护到数据库并编码，前台通过编码获取信息。

#### 内容的多语种支持

内容的多语种支持，采用多语种数据维护，显示时根据语种区分的方式进行。需要根据需求给出的“需要多语种的业务信息”，在表结构设计时根据语种维护多个内容，信息在显示的时候根据语种加载内容。这部分内容在具体的功能设计时描述，不做整体解决方案。

#### 相关问题

多语种的页面布局有一定的影响，开发界面时需要测试多个语种的显示效果，避免语种切换导致的版面问题。

### 数据库设计



1. “SYSTEM\_UI\_LANGUAGE<界面语言对照>”用于保存语言的多语种对照信息。
2. “PAGE\_NAME<页面名称>”表示允许对不同页面中相同内容进行不同的语言定义。
3. **使用方法建议**： 建议在开发时首先开发中文语言的界面，元素内容统一使用相同的前缀区分；之后使用工具将页面的内容扫描入库，写入“页面名称，中文”，自动编码并替换。之后在增加对应的语言加载代码即可。 语言翻译建议导出表结构之后进行翻译，之后在导回数据库中。 最后在调整不同语言的显示效果。



1. “SYSTEM\_MESSAGE”用于保存系统的消息。
2. 使用时需要根据语系获取不同的字段，格式化之后显示到界面。格式化内容的语言切换在具体的功能中描述。

## 多币种支持解决方案

### 需求说明

系统需要支持币种处理，包括本位币和统计报表时的币种转换。

### 方案说明

1. 在所有使用价格的位置增加币种参数，“币种 + 价格”合并体现完整的价格信息。
2. 币种来源自每家物业公司的“本位币”全局参数配置。
3. 所有显示价格，金额的页面同时显示金额和币种，格式为“200USD”。
4. 每家物业公司独立维护自己的汇率转换关系，在需要使用的环节根据每家物业公司的汇率和设定进行货币转换。

### 数据库设计



1. “CURRENCY\_RATE<汇率转换表>”用于保存货币转换的汇率信息，每家物业公司维护自己使用的汇率转换关系。
2. 汇率带有有效起至时间控制，在一个时间点应该只存在一套转换关系（输入数据时控制）。实际使用汇率时根据业务的时间查找汇率表获取对应汇率进行处理。
3. 汇率数据采用标记删除，避免已经使用的汇率找不到数据根源。

## 多浏览器支持解决方案

### 需求说明

系统要求支持浏览器兼容，支持IE8-10，Chrome，360和QQ浏览器。浏览器兼容问题主要集中在CSS样式的处理和JS脚本的跨浏览器兼容。

### 方案说明

1. HTML原型在制作时需要注意处理样式相关内容的浏览器兼容问题。
2. 页面的样式修饰全部要求使用CSS样式表控制，不要在元素上直接修饰。以方便在出现问题时统一修改。
3. JS脚本的兼容问题，需要注意对以下JS脚本的浏览器兼容：
4. 弹出窗口，系统的弹出窗口需要使用层弹出模式，不使用原有的Show Module方式。主要目的是避免浏览器对弹出窗口的拦截而导致功能不可用。因此建议在系统框架搭建的过程中，优先编写或者改造开源的弹出窗口控件。
5. 日期控件，系统需要使用到日期的录入控件，建议修改原有的日期控件或者测试开源的日期控件，在框架搭建阶段确定下能支持浏览器兼容的日期控件。
6. Message窗口，系统的Message消息不在使用JS的Alert函数处理，需要使用弹出层的方式，建议在框架阶段预先封装好消息提示的相关控件。
7. 菜单的显示效果，系统的菜单定义为下拉层的方式显示，相关的JS脚本需要预先进行浏览器兼容处理。建议在框架阶段确定一套系统使用的JS程序包，其他组件可JS脚本的编写基于这个包进行开发，避免不同的同时使用不同的程序包带来的浏览器兼容问题。

特别需要使用到的其他JS脚本和控件，在实际编码过程中需要进行浏览器兼容检查。

## 权限控制方案

### 方案说明

* 数据权限控制体系，以“User\_Data\_Func<用户数据权限>”数据表为核心，所有的权限控制都最终绑定到一个用户账户上。通过SQL语法实现控制逻辑。
* 某些设置在生成语法时，需要同时生成对多个功能点，多个用户账户的配置。这部分内容通过模板配置的方式实现批量生成。
* 权限控制分为三类，查询类，操作类和录入控制类
* 所有的查询操作，在查询数据时，都需要询问“User\_Data\_Func <用户数据权限>”获取到附加的SQL语法，以实现各种控制逻辑需要的效果。
* 所有的操作（Action）在执行时需要将操作的业务数据送到接口，与“User\_Data\_Func <用户数据权限>”中的条件进行验证，以确定是否允许操作。
* 录入控制的权限，设置内容保存在“User\_Data\_Func <用户数据权限>”中，实现部分由于涉及到的响应方式不同，需要在各个具体的功能中编码实现。
* 权限控制的重点在于所有的功能点都接受“User\_Data\_Func <用户数据权限>”的控制，无论现阶段是否有对应的权限设置对应到这个功能点。之后如果有新的权限控制需求，或者是原有的权限控制需求有改变的，仅需要变更规则写入“User\_Data\_Func <用户数据权限>”的方式，即可扩展对应的功能。

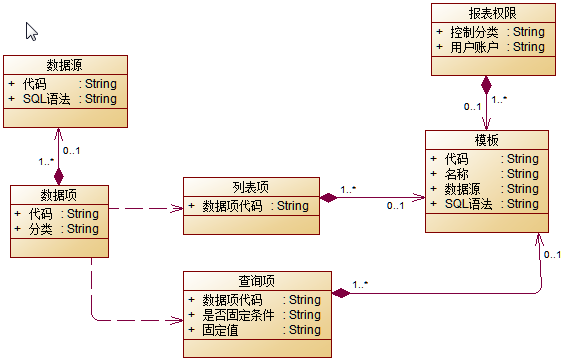
## 自定义报表解决方案

### 需求说明

用户要求有自定义报表功能，需要满足以下要求

1. 可自由增加新的报表模板
2. 报表模板需要支持自定义查询条件，并且支持固定的查询条件和可变更的查询条件两种不同的设置
3. 用户可管理报表模板的修改权限和使用权限
4. 报表模板需要支持分组汇总报表
5. 报表设置和使用需要支持双语：中文和英文
6. 报表需要导出EXCEL功能

### 解决方案



1. 自定义报表主要有两部分构成：“数据源和数据项”定义报表可以使用的数据范围；“模板、列表项和查询项”定义用户是如何统计和使用数据。
2. 模板的定义时需要指定一个，且仅能指定一个数据源。
3. 数据源的范围不限定于单表，

对于Oracle：

请采用ORACAL的“With … As”语法组织数据源。数据源的组织依赖以下规则： 1）不得出现同名字段，同名字段采用别名方式重新命名。 2）语法中需要出现显示的 “{REPORT\_WHERE}”占位符，此占位符在实际生成报表的SQL语法时会根据查询项中的设置，替换成实际的条件语法。

1. 模板中的“SQL语法”保存了最终报表执行时运行的SQL语法，此语法根据设置的“查询项，列表项”由逻辑生成。语法生成依赖以下规则： 1）固定条件直接生成语法。 2）可变条件生成参数方式的条件语法。 3）输出中存在“分组汇总项”时，其他项根据数据项分类的定义，采取不同的处理方式。例如：数字类型的处理为SUM，字符类型的处理成COUNT（\*）。

## 异常和日志方案

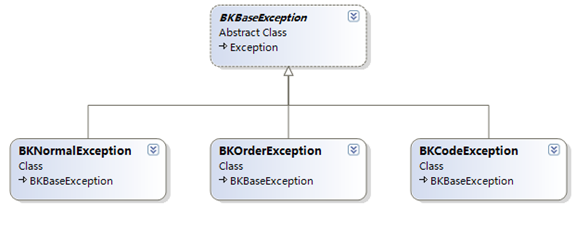
### 功能说明

异常和日志接口用于异常捕获，以及常规的业务日子记录。

### 数据库变更

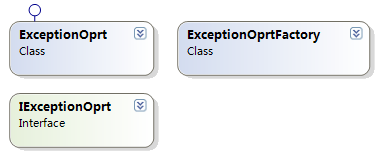
1. “OPERATION\_TYPE<操作类型>”日志记录的简略标题，用于区分一组相关的日志或者是为订阅日志提供分类。
2. “LEVELS<信息类型>”信息的分类，定义为4级“1-DEBUG 2-INFO 3-WARN 4-ERROR”。必要字段。
3. “INFORMATION<信息描述>”消息的主体，具体的内容在功能点的详细设计文档中进行描述。
4. “OBJECT\_TABLE<操作表名>，OBJECT\_SEQ<操作记录号>”用于特定数据的跟踪，目前仅在对“BUSINESS\_ORDER<订单表>”相关的日志中使用。后续可能扩展到其他需要关注的表。
5. “ORDER\_NO”在记录订单相关的日志时的必要信息，用于用户在追踪某一个订单时，作为查询条件使用。PS：此信息可能转换成Booking No，用于追踪订单的后续状态。
6. “OWNER\_COMPANY\_SEQ<物业公司>，OPRT\_CLIENT\_SEQ<操作终端>，PAY\_ CUSTOMER \_SEQ<业主>”用于登记日志的相关对象，这部分信息影响限定用户查询日志时的数据范围，后续根据实际的需求，可能作为索引使用。
7. “STATUS<数据状态>”取值为 “10-有效数据 -10-无效数据 -20-删除数据”。初始插入时为“10-有效数据”，其他两个状态值预留，目前暂不使用。

### 接口定义



在Exception对象基础上扩展了一层对象：

* “BKBaseException”类继承自“Exception”对象，抽象类，作为系统异常的基类。
* “BKNormalException”常规异常类，用于登记“非特定”的日志和异常。
* “BKOrderException”订单异常类，用于登记“订单”相关的日志和异常。和常规异常类的区别在于登记时会增加订单相关信息的采集。
* “BKCodeException”代码异常类，用于登记代码错误，常规用于Catch捕获中使用。



ExceprionOprt对象用于控制具体的日志写入操作，系统内部的异常或者是Exception对象都可以经过此类进行异常处理。

### 日志说明格式

在设计文档中的日志说明定义为以下格式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 标题 | LEVEL | 内容 | 接口 | 备注 |
| 订单撤销 | INFO  ERROR  DEBUG  WARNING |  | 常规异常  订单异常 | OWNER\_SEQ填写当前物业公司；  objName = “HTWY” |

## 缓存方案

### 功能说明

本章节用于描述缓存模块在系统中的使用场景，不包括缓存的具体设计，相关的设计请参考“IntellEstate缓存方案.docx”

### 缓存方案概要

1. 缓存的主体可以是dataTable，也可以是实体集合，请根据自身仓储机制的具体实现选择，通过system.RunTime.Caching实现。
2. 缓存的数据采用“首次使用加载”机制，即在缓存首次被请求时从数据库初始化。
3. 缓存通过XML配置表进行管理，“Configs\ DataTableCachingConfig.xml”，配置的主要内容包括Key值，需要获取的表和初始化的SQL语法。
4. 对于使用Oracle作为存储单元的方案，缓存的刷新推荐使用Oracle的通知机制实现，需要使用“ODP.NET”的指定版本配合。
5. 需要缓存的数据定义

|  |  |  |
| --- | --- | --- |
| 缓存数据表 | KEY值 | 说明 |
| DICT\_CODE | DICT\_CODE\_{OWNER\_SEQ} | 字典代码表 |
| SYSTEM\_UI\_LANGUAGE | SYSTEM\_UI\_LANGUAGE | 语言包 |
| DICT\_NATION\_AREA | DICT\_NATION\_AREA | 国家代码表 |
| DICT\_LOCATION\_AREA | DICT\_ LOCATION\_AREA | 地区代码表 |

PS：价格信息暂不列入缓存，未来可能需要支持多种的价格缓存，以支持界面价格的展示。

# 服务平台详细设计