**详细设计文档**

|  |  |
| --- | --- |
| **课程名称：** | **企业软件项目实训** |
| **项目名称：** | **Petshop** |
| **学生姓名：** | **队长：邹鹏宇**  **队员：翁焕滨、吴金泽、卢越兴、秦华** |
| **学生专业：** | **软件工程** |
| **开课学期：** | **2018-2019第二学期** |

**软件学院**

**2019年6月**

1. **需求概述**

本项目以Fisco-bcos为底层平台，设计并实现一个虚拟宠物交易商店的区块链应用。

在这个虚拟宠物交易商店中，有普通用户和管理员两种角色。普通用户主要可以查看所有在售的宠物列表、上架并售卖自己的宠物、购买其他用户上架的宠物等。而管理员主要负责为普通用户开户、仲裁用户之间的交易纠纷等。该应用为用户提供了一个便利的宠物购买和转卖的分布式市场。

1. **需求分析**

**2.1 用户角色分析**

**2.1.1 管理员**

（1）登录：使用管理员密钥登录，进入管理员界面。

（2）处理退货：收到用户的退货请求后，选择接受或拒绝该请求，并强制进行相关的转账等操作。

（3）监测市场信息：查看市场的交易信息列表、在售宠物列表。

**2.1.2 普通用户**

（1）注册账户：发出注册请求，收到自己的密钥。

（2）登录：使用注册时返回的密钥登录。

（3）创建宠物：每个账户最多可创建一个宠物，可为该宠物添加相关信息，且价格不可超过5000。

（4）修改宠物信息：修改自己账户中拥有的宠物的在售状态和其他属性。

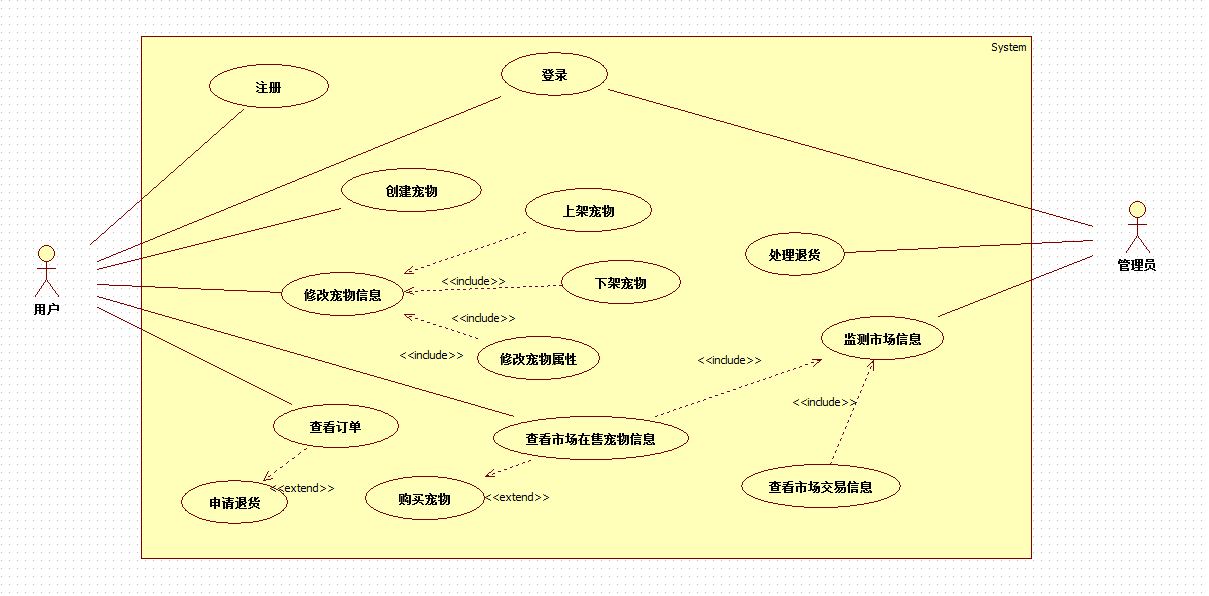
（5）查看市场在售宠物信息：查看市场中所有在售宠物的信息。

（6）购买宠物：购买市场中的在售宠物。

（7）申请退货：在购买交易发生后，向管理员发出退货申请。

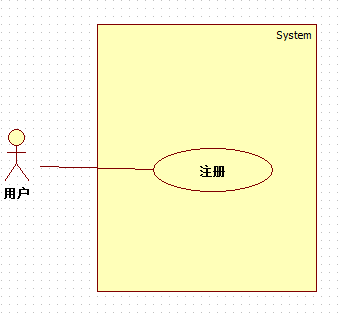
**2.2 功能性需求**

**2.2.1 系统功能架构**



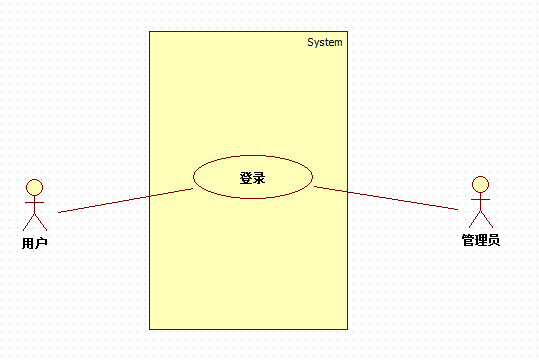
**2.2.2 具体功能**

**2.2.2.1 注册**

****

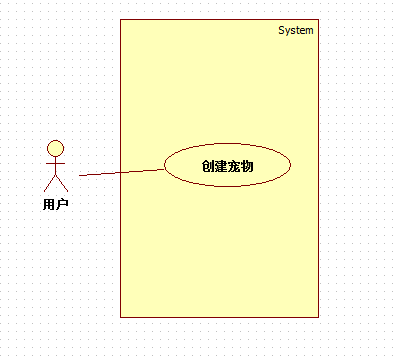
|  |  |
| --- | --- |
| 用例标识 | 1 |
| 用例名称 | 注册 |
| 参与者 | 用户 |
| 简要说明 | 用户可通过此功能注册一个新的账户 |
| 基本事件流 | 1.用户进入“登录”页面  2.点击【立即注册】按钮  3.跳转到 “注册”页面  4.点击【注册】按钮  5.系统显示“注册成功”，并返回一个私钥 |
| 备选事件流 | 1.用户进入“登录”页面  2.点击【立即注册】按钮  3.跳转到 “注册”页面  4.点击【取消】按钮  5.返回“登录”页面 |
|  |  |
| 异常事件流 | 1.用户进入“登录”页面  2.点击【立即注册】按钮  3.跳转到 “注册”页面  4.点击【注册】按钮  5.系统显示“注册失败” |
| 前置条件 | 网络状态良好，用户已进入登录页面 |
| 后置条件 | 显示注册成功，并返回用户私钥 |
| 非功能性需求 |  |
| 相关业务数据 | 用户私钥 |
| 用例相关人 | 卢越兴、秦华、邹鹏宇 |
| 注释 | 用户需要将自己的私钥记录，用于以后登录 |

**2.2.2.2 登录**

****

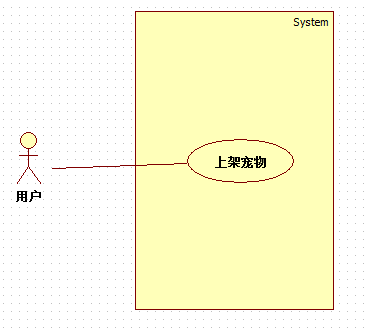
|  |  |
| --- | --- |
| 用例标识 | 2 |
| 用例名称 | 登录 |
| 参与者 | 用户、管理员 |
| 简要说明 | 用户和管理员可通过此功能登录进入各自的页面 |
| 基本事件流 | 1.用户进入“登录”页面  2.在输入框中输入自己的私钥  3.点击【立即登录】按钮  4. 跳转到用户主页 |
| 备选事件流 | 1.管理员进入“登录”页面  2.在密钥输入框中输入自己的私钥  3.点击【立即登录】按钮  4. 跳转到管理员主页 |
|  |  |
| 异常事件流 | 1.进入“登录”页面  2.在密钥输入框中输入自己的密钥  3.点击【立即登录】按钮  4. 系统显示“用户未注册” |
| 前置条件 | 网络状态良好，已进入登录页面 |
| 后置条件 | 跳转到用户或管理员首页 |
| 非功能性需求 |  |
| 相关业务数据 | 用户或管理员私钥 |
| 用例相关人 | 秦华 |
| 注释 |  |

**2.2.2.3 创建宠物**

****

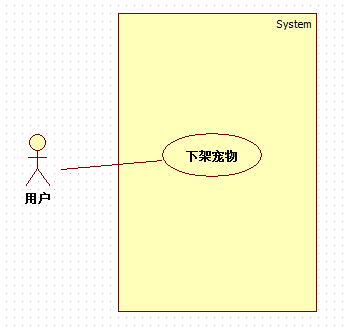
|  |  |
| --- | --- |
| 用例标识 | 3 |
| 用例名称 | 创建宠物 |
| 参与者 | 用户 |
| 简要说明 | 每个用户有且仅有一次创建宠物的资格 |
| 基本事件流 | 1.用户点击【我的信息】按钮  2.进入“用户信息”页面  3.用户点击【创建宠物】按钮  4.弹出“创建宠物框”  5.输入宠物相关信息  6.点击【创建】按钮  7.显示创建成功并关闭创建宠物框  8.用户的宠物列表中增加新建的宠物 |
| 备选事件流 | 1.用户点击【我的信息】按钮  2.进入“用户信息”页面  3.用户点击【创建宠物】按钮  4.弹出“创建宠物框”  5.输入宠物相关信息  6.点击【取消】按钮  7.关闭创建宠物框  1.用户点击【我的信息】按钮  2.进入“用户信息”页面  3.用户点击【创建宠物】按钮  4.显示 “已创建过宠物” |
| 异常事件流 | 1.用户点击【我的信息】按钮  2.进入“用户信息”页面  3.用户点击【创建宠物】按钮  4.弹出“创建宠物框”  5.输入宠物相关信息  6.点击【创建】按钮  7.显示创建失败 |
| 前置条件 | 网络状态良好，用户已登录，且未创建过宠物 |
| 后置条件 | 显示创建成功，用户宠物列表中新增创建的宠物 |
| 非功能性需求 |  |
| 相关业务数据 | 用户宠物列表、宠物信息 |
| 用例相关人 | 邹鹏宇、秦华 |
| 注释 |  |

**2.2.2.4 上架宠物**

****

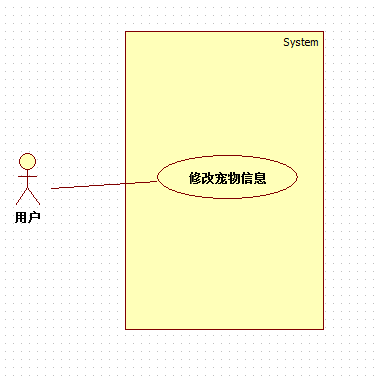
|  |  |
| --- | --- |
| 用例标识 | 4 |
| 用例名称 | 上架宠物 |
| 参与者 | 用户 |
| 简要说明 | 用户可将自己账户中拥有的宠物上架 |
| 基本事件流 | 1.用户点击【我的信息】按钮  2.进入“用户信息”页面  3.用户将鼠标移至宠物列表中的一个不在售的宠物上  4.显示【上架】按钮  5.点击【上架】按钮  6.弹出“上架确认框”  7.点击【确定】按钮  8.显示“上架成功”，且该宠物显示为在售 |
| 备选事件流 | 1.用户点击【我的信息】按钮  2.进入“用户信息”页面  3.用户将鼠标移至宠物列表中的一个不在售的宠物上  4.显示【上架】按钮  5.点击【上架】按钮  6.弹出“上架确认框”  7.点击【取消】按钮  8.弹出框关闭 |
| 异常事件流 | 1.用户点击【我的信息】按钮  2.进入“用户信息”页面  3.用户将鼠标移至宠物列表中的一个不在售的宠物上  4.显示【上架】按钮  5.点击【上架】按钮  6.弹出“上架确认框”  7.点击【确定】按钮  8.显示“上架失败” |
| 前置条件 | 网络状态良好，用户已登录，且被选中的宠物是不在售状态 |
| 后置条件 | 显示上架成功，该宠物的状态变为在售 |
| 非功能性需求 |  |
| 相关业务数据 | 宠物信息 |
| 用例相关人 | 邹鹏宇、秦华 |
| 注释 |  |

**2.2.2.5 下架宠物**

****

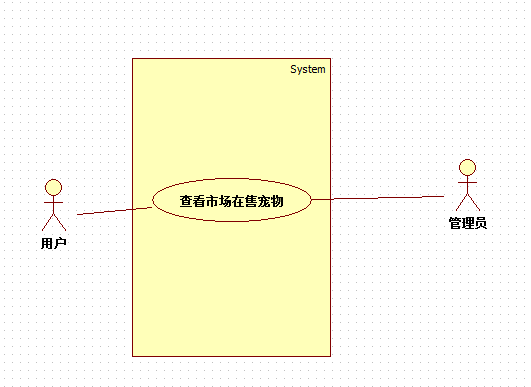
|  |  |
| --- | --- |
| 用例标识 | 5 |
| 用例名称 | 下架宠物 |
| 参与者 | 用户 |
| 简要说明 | 用户可将自己正在售卖的宠物下架 |
| 基本事件流 | 1.用户点击【我的信息】按钮  2.进入“用户信息”页面  3.用户将鼠标移至宠物列表中的一个在售的宠物上  4.显示【下架】按钮  5.点击【下架】按钮  6.弹出“下架确认框”  7.点击【确定】按钮  8.显示“下架成功”，且该宠物显示为不在售 |
| 备选事件流 | 1.用户点击【我的信息】按钮  2.进入“用户信息”页面  3.用户将鼠标移至宠物列表中的一个在售的宠物上  4.显示【下架】按钮  5.点击【下架】按钮  6.弹出“下架确认框”  7.点击【取消】按钮  8.弹出框关闭 |
| 异常事件流 | 1.用户点击【我的信息】按钮  2.进入“用户信息”页面  3.用户将鼠标移至宠物列表中的一个在售的宠物上  4.显示【下架】按钮  5.点击【下架】按钮  6.弹出“下架确认框”  7.点击【确定】按钮  8.显示“下架失败” |
| 前置条件 | 网络状态良好，用户已登录，选中的宠物是在售状态 |
| 后置条件 | 显示下架成功，该宠物的状态变为不在售 |
| 非功能性需求 |  |
| 相关业务数据 | 宠物信息 |
| 用例相关人 | 邹鹏宇、秦华 |
| 注释 |  |

**2.2.2.6 修改宠物信息**

****

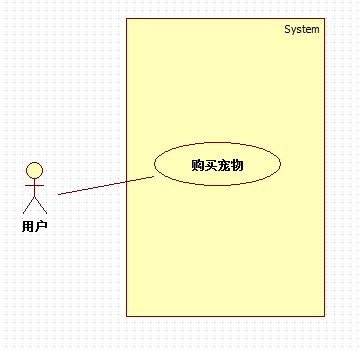
|  |  |
| --- | --- |
| 用例标识 | 6 |
| 用例名称 | 修改宠物属性 |
| 参与者 | 用户 |
| 简要说明 | 用户可通过此操作修改自己拥有的宠物除在售状态之外的其他信息 |
| 基本事件流 | 1.用户点击【我的信息】按钮  2.进入“用户信息”页面  3.用户将鼠标移至宠物列表中的一个在售的宠物上  4.显示【修改信息】按钮  5.点击【修改信息】按钮  6.弹出“修改信息框”  7.输入想要修改的信息  8.点击【保存编辑】按钮  9.“修改信息框”关闭，宠物的信息更新 |
| 备选事件流 | 1.用户点击【我的信息】按钮  2.进入“用户信息”页面  3.用户将鼠标移至宠物列表中的一个在售的宠物上  4.显示【修改信息】按钮  5.点击【修改信息】按钮  6.弹出“修改信息框”  7.输入想要修改的信息  8.点击【取消】按钮  9.“修改信息框”关闭 |
| 异常事件流 | 1.用户点击【我的信息】按钮  2.进入“用户信息”页面  3.用户将鼠标移至宠物列表中的一个在售的宠物上  4.显示【修改信息】按钮  5.点击【修改信息】按钮  6.弹出“修改信息框”  7.输入想要修改的信息  8.点击【保存编辑】按钮  9.显示“修改失败” |
| 前置条件 | 网络状态良好，用户已登录 |
| 后置条件 | 显示修改成功，宠物信息已更新 |
| 非功能性需求 |  |
| 相关业务数据 | 宠物信息 |
| 用例相关人 | 邹鹏宇、秦华 |
| 注释 |  |

**2.2.2.7 查看市场在售宠物**

****

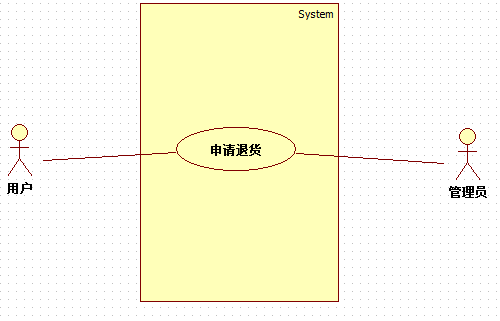
|  |  |
| --- | --- |
| 用例标识 | 7 |
| 用例名称 | 查看市场在售宠物信息 |
| 参与者 | 用户、管理员 |
| 简要说明 | 用户、管理员可通过此功能查看市场所有在售宠物的详细信息 |
| 基本事件流 | 1.用户点击【宠物市场】按钮  2.显示市场在售宠物列表 |
| 备选事件流 | 1.管理员点击【宠物市场】按钮  2.显示市场在售宠物列表 |
|  |  |
| 异常事件流 |  |
| 前置条件 | 网络状态良好，用户或管理员已登录，市场上有宠物在售 |
| 后置条件 | 显示市场在售宠物列表 |
| 非功能性需求 |  |
| 相关业务数据 | 宠物信息 |
| 用例相关人 | 秦华 |
| 注释 |  |

**2.2.2.8 购买宠物**

****

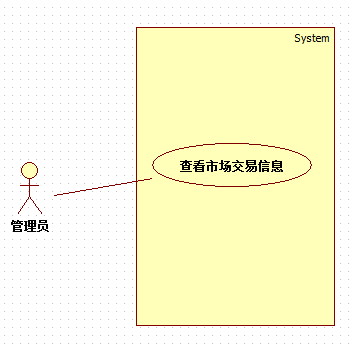
|  |  |
| --- | --- |
| 用例标识 | 8 |
| 用例名称 | 购买宠物 |
| 参与者 | 用户 |
| 简要说明 | 用户可在市场内在售的宠物列表内购买宠物 |
| 基本事件流 | 1.用户点击【宠物市场】按钮  2.显示市场在售宠物列表  3.将鼠标移至列表中的一个宠物上  4.显示【购买】按钮  5.点击【购买】按钮  6.弹出“购买确认框”  7.点击【确定】按钮  8.显示购买成功 |
| 备选事件流 | 1.用户点击【宠物市场】按钮  2.显示市场在售宠物列表  3.将鼠标移至列表中的一个宠物上  4.显示【购买】按钮  5.点击【购买】按钮  6.弹出“购买确认框”  7.点击【取消】按钮  8.购买确认框关闭  1.用户点击【宠物市场】按钮  2.显示市场在售宠物列表  3.将鼠标移至列表中的一个宠物上  4.显示【购买】按钮  5.点击【购买】按钮  6.弹出“购买确认框”  7.点击【确定】按钮  8.显示“该宠物已下架” |
|  |  |
| 异常事件流 |  |
| 前置条件 | 网络状态良好，用户或管理员已登录，用户余额充足 |
| 后置条件 | 显示购买成功 |
| 非功能性需求 |  |
| 相关业务数据 | 用户余额，宠物信息 |
| 用例相关人 | 邹鹏宇、秦华 |
| 注释 |  |

**2.2.2.9申请退货**

****

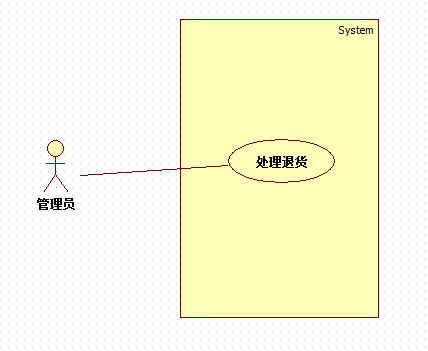
|  |  |
| --- | --- |
| 用例标识 | 9 |
| 用例名称 | 申请退货 |
| 参与者 | 用户 |
| 简要说明 | 用户可通过此操作向管理员提交退货申请 |
| 基本事件流 | 1.用户点击【我的信息】按钮  2.显示用户订单列表  3.在一个订单后点击【申请退货】按钮  4.在弹出框内填写退货理由  5.点击【提交】按钮  6.显示提交成功，订单后的【申请退款】变成【仲裁中】 |
| 备选事件流 | 1.用户点击【我的信息】按钮  2.显示用户订单列表  3.在一个订单后点击【申请退货】按钮  4.在弹出框内填写退货理由  5.点击【取消】按钮  6.弹出框关闭 |
|  |  |
| 异常事件流 |  |
| 前置条件 | 网络状态良好，用户或管理员已登录，有订单存在 |
| 后置条件 | 申请成功，返回订单界面 |
| 非功能性需求 |  |
| 相关业务数据 | 订单信息，订单状态 |
| 用例相关人 | 邹鹏宇、秦华 |
| 注释 |  |

**2.2.2.10 查看市场交易信息**

****

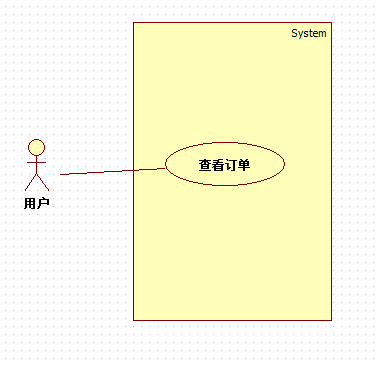
|  |  |
| --- | --- |
| 用例标识 |  |
| 用例名称 | 查看市场交易信息 |
| 参与者 | 管理员 |
| 简要说明 | 管理员可通过此功能查看市场中发生的所有交易信息 |
| 基本事件流 | 1.管理员点击【所有交易】按钮  2.显示市场中所有交易的列表 |
| 备选事件流 |  |
|  |  |
| 异常事件流 |  |
| 前置条件 | 网络状态良好，管理员已登录并进入交易信息页面 |
| 后置条件 | 展示交易列表 |
| 非功能性需求 |  |
| 相关业务数据 | 交易列表、交易详情 |
| 用例相关人 | 秦华 |
| 注释 |  |

**2.2.2.11 处理退货**

****

|  |  |
| --- | --- |
| 用例标识 | 11 |
| 用例名称 | 处理退货 |
| 参与者 | 管理员 |
| 简要说明 | 管理员可通过此功能处理用户的退货请求 |
| 基本事件流 | 1.管理员点击【退货信息】按钮  2.显示退货信息列表  3.点击【同意】按钮  4.弹出确认框  5.点击【确定】按钮  6.显示“已同意退货” |
| 备选事件流 | 1.管理员点击【退货信息】按钮  2.显示退货信息列表  3.点击【同意】按钮  4.弹出确认框  5.点击【取消】按钮  6.弹出框关闭  1.管理员点击【退货信息】按钮  2.显示退货信息列表  3.点击【拒绝】按钮  4.弹出确认框  5.点击【确认】按钮  6.显示“已拒绝退货”  1.管理员点击【退货信息】按钮  2.显示退货信息列表  3.点击【拒绝】按钮  4.弹出确认框  5.点击【取消】按钮  6.弹出框关闭 |
|  |  |
| 异常事件流 |  |
| 前置条件 | 网络状态良好，管理员已登录，有未处理的退货请求 |
| 后置条件 | 显示退货处理结果 |
| 非功能性需求 |  |
| 相关业务数据 | 退货信息列表、用户账户余额、用户宠物列表 |
| 用例相关人 | 秦华 |
| 注释 |  |

**2.2.2.12 查看订单**

****

|  |  |
| --- | --- |
| 用例标识 | 12 |
| 用例名称 | 查看订单 |
| 参与者 | 用户 |
| 简要说明 | 用户可通过此功能查看自己的订单 |
| 基本事件流 | 1.用户点击【我的信息】按钮  2.展示用户订单列表 |
| 备选事件流 |  |
|  |  |
| 异常事件流 |  |
| 前置条件 | 网络状态良好，用户已登录 |
| 后置条件 | 展示用户订单 |
| 非功能性需求 |  |
| 相关业务数据 | 用户订单信息 |
| 用例相关人 | 秦华 |
| 注释 |  |

**2.3 非功能性需求**

**2.3.1 安全需求**

1）严格权限访问控制，用户在经过身份认证后，只能访问其权限范围内的数据，只能进行其权限范围内的操作。

（2）不同的用户具有不同的身份和权限，需要在用户身份真实可信的前提下，提供可信的授权管理服务，保护数据不被非法/越权访问和篡改，要确保数据的机密性和完整性。

（3）能经受来自互联网的一般性恶意攻击。如病毒（包括木马）攻击、口令猜测攻击、黑客入侵等。

**2.3.2 可靠性**

系统的可靠性也是很重要的一方面。本系统的可靠性主要体现在两个方面:

1.数据可靠性：可靠性主要体现在数据的可靠存储和防止数据丢失。

2.操作可靠性：通过功能的正确实现和硬件的合理配置，保证系统能顺利和准确的完成用户所要求的特定任务。

具体体现在：

（1）因软件系统的失效而造成不能完成业务的概率要小于5%

（2）在1,000,000次交易中，最多出现1次需要重新启动系统的情况

（3）对输入有提示，数据有检查，防止数据异常。

**2.3.3 性能**

在95％的情况下，一般时段响应时间不超过5秒，高峰时段不超过8秒。

在推荐配置环境下：登录响应时间在5秒内，刷新市场列表响应时间在5秒内，购买操作或退款操作响应时间秒5内，修改宠物信息响应时间3秒内。

在非高峰时间管理员查看整个市场的交易记录，可以在5秒内得到搜索结果。

**2.3.4 可用性**

系统的可操作性要求：系统具有友好的用户交互方式和高效合理的功能配给及组织形式。本系统对可操作性的设计要求主要可概括为以下几个方面：

1.功能定义明确，区分清晰：对系统的主要功能有明确的定义，并且在界面及系统内部有着明确的区分，使得用户可以非常便捷的识别和使用相关功能等。

2.操作直接，关联性充分：在用户的功能实现成本上，要尽量降低用户实现特定功能所需的点击量和复杂程度。主要通过两点来实现，一方面对特定功能，要尽量减少功能实现的层深，扩展功能树的广度而降低深度。另一方面，要使得相关功能的关联性尽量充分，使得用户可以通过功能之间的关联链接来便捷的从一种功能链接到相关的功能。

1. **系统设计**

**3.1 智能合约设计**

**3.1.1 DataProcess.sol**

这是一个用于数据处理的合约，包括一下两个函数：

（1）strConcat：字符串连接

（2）getIntToString：将整数转换为字符串

**3.1.2 Market.sol**

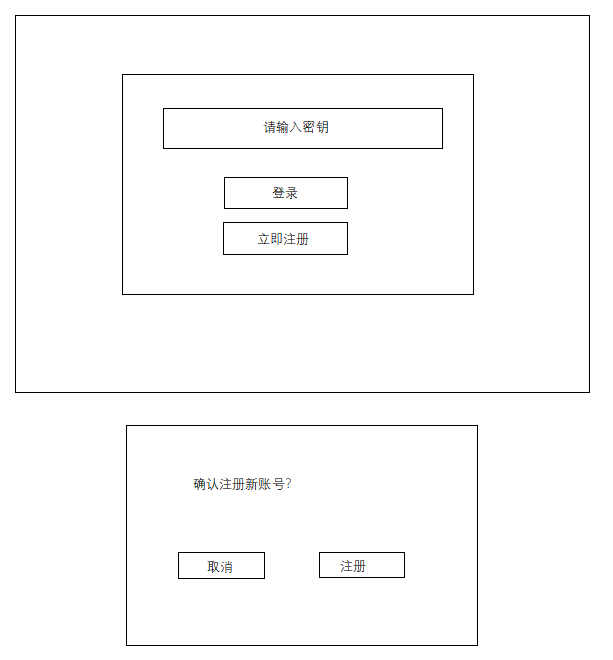
|  |  |
| --- | --- |
| 定义宠物结构体 | struct Pet |
| 全局变量声明 | 管理员地址，address private adminAddress;  order合约地址，OrderContract OD;  在售宠物，uint petIdNum=1;  订单列表，Pet[] petList; |
| 映射 | 判断是否已创建过宠物，mapping (address => uint8) createdPet;  地址与余额映射，mapping (address => uint) Balance;  判断是否老用户，mapping (address => uint8) oldUser;  宠物id到用户地址的映射，mapping (string => address) petIdToOwner;  用户地址与身份的映射，mapping(address => uint8) userIden; |
| 构造函数 | 设置合约部署人为管理员，constructor() public |
| 修饰符 | 判断身份是不是管理员，modifier isAdmin() |
| 内部调用函数及只有管理员可调用的函数 | 初始化用户余额以及是否创建过宠物参数，initUser()  设置宠物主人，setPetOwner()  设置order合约地址，用于调用其中的函数，setOrderAddress()  支付对应金额，内部调用，pay()  管理员使用的转账函数，用于退货方面的操作，payByAdmin()  转交宠物所有权，用于购买宠物和退货时使用，changePetOwner() |
| 用户相关函数 | 创建用户函数，初始化余额和是否创建过宠物参数，createUser()  获取用户身份，getUserIden()  获取调用者的余额，getBalanceOfMe()  通过地址获取余额，getBalance() |
| 宠物相关函数 | 用户查看自己的宠物，getPetListFromAddress()  展示市场在售宠物，showPetOnSell()  创建宠物，若没有创建过就先创建宠物列表再创建宠物，createPet()  购买宠物，buyPet()  出售宠物，sellPet()  取消出售宠物，cancelSellPet()  修改宠物信息，changePetInfo() |

**3.1.3 OrderContract.sol**

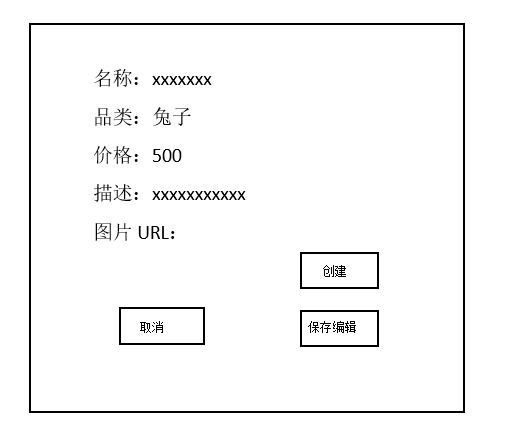
|  |  |
| --- | --- |
| 定义订单结构体 | struct Order |
| 全局变量声明 | market合约地址，Market MK;  管理员地址，address private adminAddress;  订单列表，Order[] public orderList;  订单id，uint orderIdNum=1; |
| 构造函数 | 设置管理员地址，constructor() public |
| 修饰符 | 判断身份是不是管理员，modifier isAdmin() |
| 内部管理员函数 | 设置market合约地址，setMarketAddress()  创建一个新订单，market管理员调用，createOrder() |
| 订单相关函数（管理员部分） | 返回所有订单，用于管理员查看，adminGetOrderList()  管理员获得请求仲裁的订单，adminGetReturnOrderList()  管理员接受退货申请，acceptReturn()  管理员拒绝退货申请，rejectReturn() |
| 订单相关函数（用户部分） | 用户获得自己的订单列表，userGetOrderList()  申请退货，applyForReturn() |

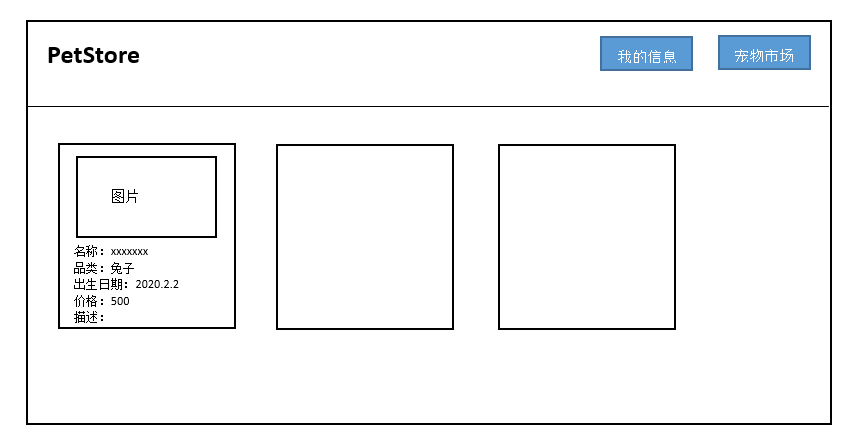
**3.2 界面设计**

**3.2.1 注册与登录页面**

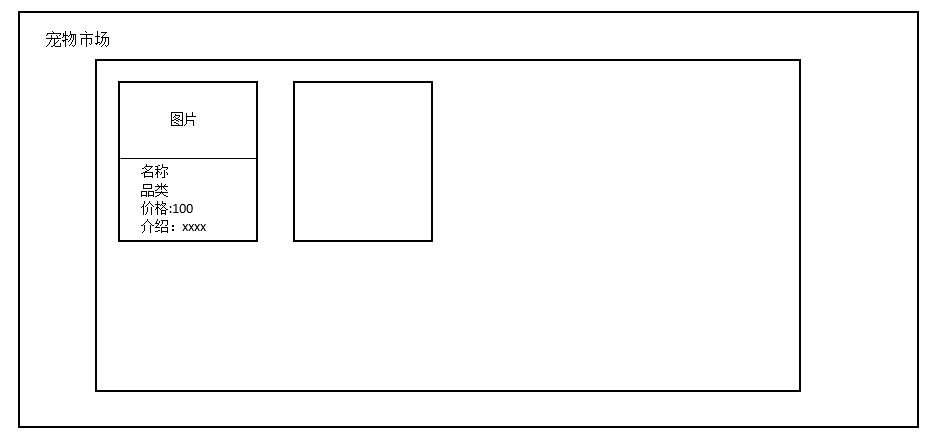


**3.2.2 用户页面（我的信息、宠物市场）**





**3.2.3 管理员页面（退货信息、宠物市场、所有交易）**



1. **逻辑部署方案**

Vue

前端

网络请求

Axios

路由管理

Vue Router

视图渲染

Vue Templete

状态管理

VueX

Web3sdk

SpringBoot

数据处理

后端路由

参数校验

Fisco-bcos

共识算法 加密算法 EVM

网络 存储 ...

底层

合约层

智能合约

通信接口

1. **物理部署方案**