易筹私募股权融资系统
技术说明文档

1 系统设计原则 2
1.1 实用性原则 2
1.2 先进性原则 2
1.3 可扩展性和可维护性原则 2
1.4 系统设计安全、可靠性原则 2
1.5 高可用性原则 3
1.5.1 缓存 3
1.5.2 数据模型 3
1.5.3 算法 3
1.5.4 架构 4
1.6 模块化 4
1.7 松耦合 5
2 技术参数说明 5
2.1 开发工具 5
2.2 数据库 5
2.3 第三方控件 5
2.4 系统架构 6
2.5 代码规范 6
2.5.1 数据设计规范 6
2.5.2 代码规范 6
3 系统安全设计 6
3.1 基于 SSL 证书数据传输加密 6
3.2 前后台分离部署 6
3.3 权限管理和密码管理 7
3.4 缓冲区溢出攻击 7
3.5 SQL 注入 7
3.6 限制 URL 访问 7
3.7 跨站请求 7
3.8SESSION、CACHE 安全 7
3.9 业务安全 8
3.9.1 身份验证安全 8
3.9.2 权限管理 8
3.9.3 资金安全 8
3.9.4 交易操作安全 8
3.10 数据安全 8
3.10.1 数据安全 8
3.10.2 数据校验 9
3.10.3 数据备份与恢复 9
3.10.4 接口安全 9
4 交互设计理念 10
4.1 视觉交互设计 10
4.2 用户体验交互设计 10
5 软、硬件配置建议 10
5.1 服务器要求 10
5.2 软件应用 11

1　系统设计原则

1.1 实用性原则

系统力求最大限度的满足实际运营的需要，充分考虑个业务层次、各环节数据的实用性，把满足客户需求、用户体验最为第一要素进行考虑。用户操作界面设计尽可能考虑互联网操作习惯、人体结构特征及视觉特征，界面力求美观大方，操作力求简便实用。

1.2 先进性原则

采用目前先进、成熟的软件开发结束技术（如数据访问封装、实体封装、业务逻辑封装等多层体系结构），面向对象的设计方法，可视化的、面向对象的设计工具。

选用目前国际流行的 JAVA 开发语言，以及相关成熟框架。选用 MySQL 等成熟的数据库产品。

前端合理使用流行的 Ajax，JQuery 等技术，已实现更好地用户体验。

1.3 可扩展性和可维护性原则

为适应将来的发展，系统设计具有良好的可扩展性和维护性。系统设计尽可能模块化、组件化，并提供配置模块和客户化工具，使应用系统可灵活配置，适应不同的情况。数据库的设计尽可能考虑到将来的需要。

严格遵循给统一的代码规范：以 SVN 建立的 JAVA 代码规范为蓝本，我们整理了方丁易筹代码规范。整个系统严格按照代码规范书写。系统严格遵循 OOP 面向对象原则编写，所有代码、逻辑均采用类，继承，多台等封装，流程清晰，可维护性、可扩展性强。

1.4 系统设计安全、可靠性原则

系统与数据库系统的设计做到安全可靠，防止非法用户的入侵。系统采用多级认证措施，以及密码存储采用无可逆加密方法，管理员无法知道用户密码，非法获取到数据库记录，也不能逆向推算出用户的密码。

数据库的备份策略恰当 ,以防止灾难性故障丢失客户重要数据。在事故发生后也能迅速恢复数据。

源代码安全：鉴于 JAVA 伪代码编译的不安全性，系统涉及到底层代、网络传输方面，均采用 C++书写公共函数。JAVA 代码经过加密设置。

接口安全：对于接口调用（如：WebService 的访问），为了减少每次调用传输用户 ID 和密码对系统的开销减少不安全因素，采用 MD5 摘要校验的方式开放式行访问。

防注入安全：使用方丁易筹独创的先进技术确保系统安全。

防暴力安全：系统在注册、登陆、忘记密码、修改个人资料、提交数据等敏感提交处，都设置了线程休眠，动态验证码（部分）杜绝暴力破解，恶意注册等。

Session、cache 安全：系统对 session，cache 的读写采用统一函数进行，对 key 加入自定义的前缀字符串，并对 value 增加校验，防止篡改客户端缓存或外部提交。

1.5 高可用性原则

互联网的告诉发展，用户对性能的要求越来越高，系统也越来越复杂，就这就造成了服务器、带宽等硬件投资等成本投入与运营受益之间的矛盾。在不增加硬件的基础上，我们从如下几个方面尽可能提供系统的高可用性：

1.5.1 缓存

合理使用页面缓存、页面局部缓存、页面布局不缓存、静态化、页面局部静态化、数据缓存、memcached 缓存等技术、提高访问相应性能，节省服务器、带宽开销。

1.5.2 数据模型

合理设计数据库表、数据库关系，适度冗余，提高数据库效率。

1.5.3 算法

系统采用方丁易筹的自有的高性能算法来解决一些复杂的计算问题，杜绝代码堆砌现象，提高数据的处理效率。

1.5.4 架构

采用方丁易筹自主研发的 SOA 架构方式，提高系统的并发性能。

SOA 是指为了解决在 Internet 环境中业务集成的需要，通过连接能完成特定任务的独立功能实体实现的一种软件系统架构。这种面向服务的构架是一种 IT 战略，它把包含在各种企业应用中的分散的功能组织为可互操作的、基于标准的服务 ,而这些服务可以被迅速的组合和重用以满足业务需求。一个服务就是一个代码模块，它可以由通过基于标准的接口访问的服务水平协议管理。每个服务是代表企业的一部分功能，它明确地映射到业务流程中的一个单一步骤。一个服务可以从头开始编写，也可以通过公开原有孤立的企业应用系统程序中现有服务功能模块来挖掘。经过一段的时间后，可以建立起服务目录，允许不同系统流畅地访问和重用业务功能。在支持战略变革的同时，能够消除数据冗余。

使用 SOA 架构,客户端和服务之间的通讯由消息的架构支配，只要消息符合协商的架构，则客户端或服务的实现就可以根据需要进行更改，而不必担心会破坏对方，即达到服务之间要求最小的依赖性，只要求它们之间能够相互知晓。SOA 的松耦合通讯机制提供了紧耦合机制所没有的许多优点，也有助于降低客户端和远程服务之间的依赖性。使应用程序环境更敏捷，能更快地适应更改，并且降低了风险。

1.6 模块化

作为一个庞大而复杂的软件系统，可按其功能或其他的特性，划分出许多的模块，每个模块具备自己的功能、逻辑和状态。系统被模块化后，每个模块分别地单独处理，而忽略其他模块的细节。处理所有模块的共同性和相互关系，并将之集成为完整的系统。

在本系统中将采用模块化的思路进行设计，这样可以降低系统的总体复杂度，使得开发人员易于理解和开发，并且各个组成模块具有一定的独立性后，如果某一或某些模块需要替代或修改时，只需保证与其他模块的接口不变 ,就可以保证其他模块能继续适用。这也将有利可于软件系统的日常维护。

不仅如此，使用模块化设计，还能够使得各模块可以分开部署，进而实现分布式部署方式。提高

系统性能和可用性。

1.7 松耦合

传统上，业务流程是在企业范围，甚至在企业的不同业务单元内开发。这些活动在详细的实时信息的帮助下进行管理。跨多个业务单元或跨企业的流程必须更加灵活，以应对各种各样的需求。在这种情况下，可能出现更多的不确定性：参与者及其角色不断变化，所需的交互类型也不断变化。在运营状况起伏不定的环境下，必须有一个松耦合架构，以降低整体复杂性和依赖性。松耦合使应用程序环境更敏捷，能更快地适应更改，并且降低了风险。

在系统设计中也将遵循松耦合的原则。客户端和服务之间的通讯由消息的架构支配，只要消息符合协商的架构，则客户端或服务的实现就可以根据需要进行更改，而不必担心会破坏对方，即达到服务之间要求最小的依赖性，只要求它们之间能够相互知晓。松耦合通讯机制提供了紧耦合机制所没有的许多优点，并且它们有助于降低客户端和远程服务之间的依赖性。使用 SOA 架构可以很好地实现系统的松耦合设计。

2 技术参数说明

2.1 开发工具

Eclipse/MyEclipse

2.2 数据库

采用 MySQL 5.5.45 及以上版本作为数据库管理软件也可已选择 Oracle10g 等数据库软件

2.3 第三方控件

第三方支付 API、短信 API、ID5

2.4 系统架构

WEB 服务器：ngnix+apache 做负载均衡，采用 Jetty 作为 web 容器。

技术架构：SOA 面向服务的体系架构。

2.5 代码规范

2.5.1 数据设计规范

严格遵循数据设计规范到第 3 范式。杜绝不必要的冗余以及数据不一致性。遵循关系数据库逻辑要求。

2.5.2 代码规范

整个系统均按照严格的代码规范书写。系统严格遵循 OOP 面向对象的原则编写，所有代码、逻辑均采用类，继承、多态等封装，流程清晰，可维护性、可扩展性强。

3 系统安全设计

3.1 基于 SSL 证书数据传输加密

采用 SSL 证书家里传输（http 安全传输协议），信任等级强，需验证企业身份，审核严格，安全性高。

3.2 前后台分离部署

对应用进行前台用户端与后台管理端的独立部署，实现不同域名，不同 IP 的不部署。这样对管理后台网站暴露、管理后台为暴力猜码等黑客行为进行防范。

3.3 权限管理和密码管理

避免用户的越权操作。密码长度和规则的控制。密码在传输和存储方面都必须采用密文，密文不允许回显。

3.4 缓冲区溢出攻击

缓冲区溢出攻击（Buffer Overflow），可以限制查询结果的数量。通过代码走查，对系统的源码进行扫描，可以检查嵌套多层 while 循环、联表查询等。

3.5 SQL 注入

系统不存在在页面上编写 SQL 语句的代码，同时在执行 SQL 语句方面采用的是 PreparedStatement 处理。

过滤用户输入的特殊符号，用正则表达式，不安全字符屏蔽应对策略。

3.6 限制 URL 访问

系统采用过滤器对非法入侵的操作进行隔离，不允许通过猜测 URL 地址直接调用内容。

3.7 跨站请求

针对跨站请求的功能，系统会为每个业务请求方授权一个动态令牌，在提交数据的过程中，会校验该令牌，防止伪造该请求进行数据提交。

3.8session、cache 安全

系统对 session、cache 的读写采用统一的函数进行处理，对 KEY 加入自定义的前缀字符串，并对 Value 进行校验，防止篡改客户端缓存或外部提交。

3.9 业务安全

3.9.1 身份验证安全

系统登录采用用户名，验证码和密码验证方式，密码采用随机码和 MD5 加密的方式，防止在数据库中查询用户密码，确保身份验证安全。

系统针对需要验证手机短信的接口功能，增加图形验证码的功能，减少对短信接口的攻击。

3.9.2 权限管理

管理系统在角色上赋予功能操作权限，一旦权限设定之后，角色下的用户只能看到与自己工作相关的功能。并且在功能中只能访问本机构及向下机构的数据。

3.9.3 资金安全

资金数据是通过独立应用提供的接口服务，会对资金请求的操作、用户信息进行验证，防止非法篡改资金数据。

3.9.4 交易操作安全

在前端应用的业务请求是通过调用接口服务的方式获取或处理数据，需要验证数据权限的接口，服务会对请求访问的用户角色进行判断和验证，不同角色的用户对数据的处理有相应的校验规则。

用户在系统中的进入时间、操作时间、交易操作都将记录到交易日志表中，随时可查询到用户的操作信息。

3.10 数据安全

3.10.1 数据安全

由于数据主要以配置文件和数据库进行存储 ,数据的安全将主要依靠操作系统管理、数据库管理、机房管理等管理手段来解决，配合必要的系统安全手段来加强。

资金数据是通过独立应用提供的接口服务，会对资金请求的操作进行验证，防止篡改资金数据。

接口服务可以支持私密性图片单独目录进行保存。

3.10.2 数据校验

为了防止非法篡改数据库，我们在库表设计时对关键库表设置了单向散列加密校验字段。这种设计的基本思路是防止技术人员对业务数据的非法篡改，实现方法是对一个关键库表的关键字段进行加密运算得出校验码存放在校验字段中，一旦有人对数据库进行更改，则校验出错，系统视同无效数据。

3.10.3 数据备份与恢复

对于数据的安全管理主要考虑的是数据的容灾备份，完整、有效的备份是整个灾备体系能够发挥作用的基础。备份需要考虑如下内容：

- 能够制定管理上的策略，并将这些策略非常容易地部署下去；
- 应用数据库 7*24 小时在线备份；
- 对某些时间性较强的应用数据库定期进行历史数据归档；
- 实现分级存档管理（HSM）；
- 提供集中数据存储管理模式；
- 全自动备份；
- 同一版本数据备份多份（CLONE）；
- 对备份介质的有效管理；
- 对备份数据的及时恢复；
- 远程备份，系统的灾难恢复；

3.10.4 接口安全

与第三方系统的技术接口，其中对外开放接口采用身份验证技术，并对敏感数据进行加密处理。

4 交互设计理念

4.1 视觉交互设计

互联网时代用户对内容习惯是“扫 描”而非“阅读”。基于此我们更关注用户在不同硬件尺寸下的 3 秒视觉体验，强化按钮、图标、文案命明设计，强调互联网的 UI 要符合“扫描式”交互入口。

4.2 用户体验交互设计

以用户需求为开发核心。以需求变化为交互核心。这一“双心”机制是我们用户体验呢交互设计的要点，以此出发实现需求逻辑可视化、用户体验可视化、需求变化可视化的高品质产品设计与交互。

5 软、硬件配置建议

5.1 服务器要求

硬件服务器：

CPU：64 位至强 4 核，2.0G 主频以上

内存：4G 以上

硬盘：双 SCSI 硬盘镜像（Raid-0）,160G 剩余空间以上

操作系统：

主流 Linux 系统（不建议采用 Ubuntu 系统）

软件环境：

PHP5.3 及以上版本

JAVA 运行环境 jdk7.0 及以上版本

mysql-sever-5.5 及以上版本

Redis2.6 及以上版本

5.2 软件应用

开发	展示层：php,Ajax,jquery,css,自定义模板
	业务层：自主开发的配置化 xml 业务引擎
	持久层：自主开发的基于 jdbc 的 ORM 框架
运行环境	支持绝大多数 Linux、unix 系统，
技术框架	自主开发的框架，内置 jetty 容器