云服务器文件加密系统

开发报告

2017 年 3 月

目录

一、项目概述

1.1 项目背景

随着信息技术的不断发展和应用，人们在享受到越来越丰富的信息资源的同时，也受到了越来越严重的安全威胁，信息安全的重要性与日俱增。在目前信息安全的应用领域中，各类加密算法得到了广泛应用。然而计算机上使用各类加密方法时，受各类因素影响，加密数据效率慢。在当下“云计算”日渐流行的趋势下，“云加密”的出现可以使加密的实现速率得到提高，从而逐渐引起大家的关注。

1.2 项目名称

云服务器文件加密系统

1.3 开发环境

操作系统：CentOS 6.5,Window 10,Android 22

开发平台：IntelliJ IDEA 2016,Android Stduio2.2

数据库：MySQL 6.3(innoDB 数据库引擎)

二、项目设计

2.1 需求分析

2.1.1 问题的提出

加密技术是最常用的安全保密手段，为了确保数据安全，将加密密钥与加密数据分开来极其重要。早先，加密的数据存储在服务器上，而如今许多应用程序托管在云端。计算机在进行数据加密时，对时间地点、设备人员等因素也有着要求。而“云加密”基于虚拟化技术，有着多元化的密钥管理方式，让用户更加方便、快捷、安全地对密钥进行操作。

2.1.2 需要的功能

总体功能：

用户能够注册，并且登录。登录成功后，可以上传文件，获取文件列表。选择具体文件后，可以获取所支持的加密方法及加密介绍，并进行服务器加密，下载到本地。下载成功后可以进行本地解密。

具体功能：

1.用户登录，注册。

2.文件加密，解密。包含 Md5，Des，RSA 算法。

3.文件上传，下载，压缩，解压。

4.针对用户安全性，需要对密码进行加密。

5.针对 URL 安全性，需要加入过滤器并保存用户 Cookie，登录成功后，才允许访问具体 URL，否则拒绝访问。

6.针对数据库负载，为减轻服务器查询数据库的压力，需要二级缓存。

7.针对用户体验，客户端所有耗时操作包含网络操作，图片加载等，均需要多线程异步操作。

8.针对数据查询量过大，服务器需要提供分页查询。

2.2 总体设计

2.2.1 概念模型设计

1. 用户程序流程：

用户能够注册，并且登录。登录成功后，获取文件列表，同时可以上传文件。选择具体文件后，获取加密方式列表及加密介绍，并进行服务器加密，下载到本地。下载成功后可以进行本地解密。基本程序流程如图 1

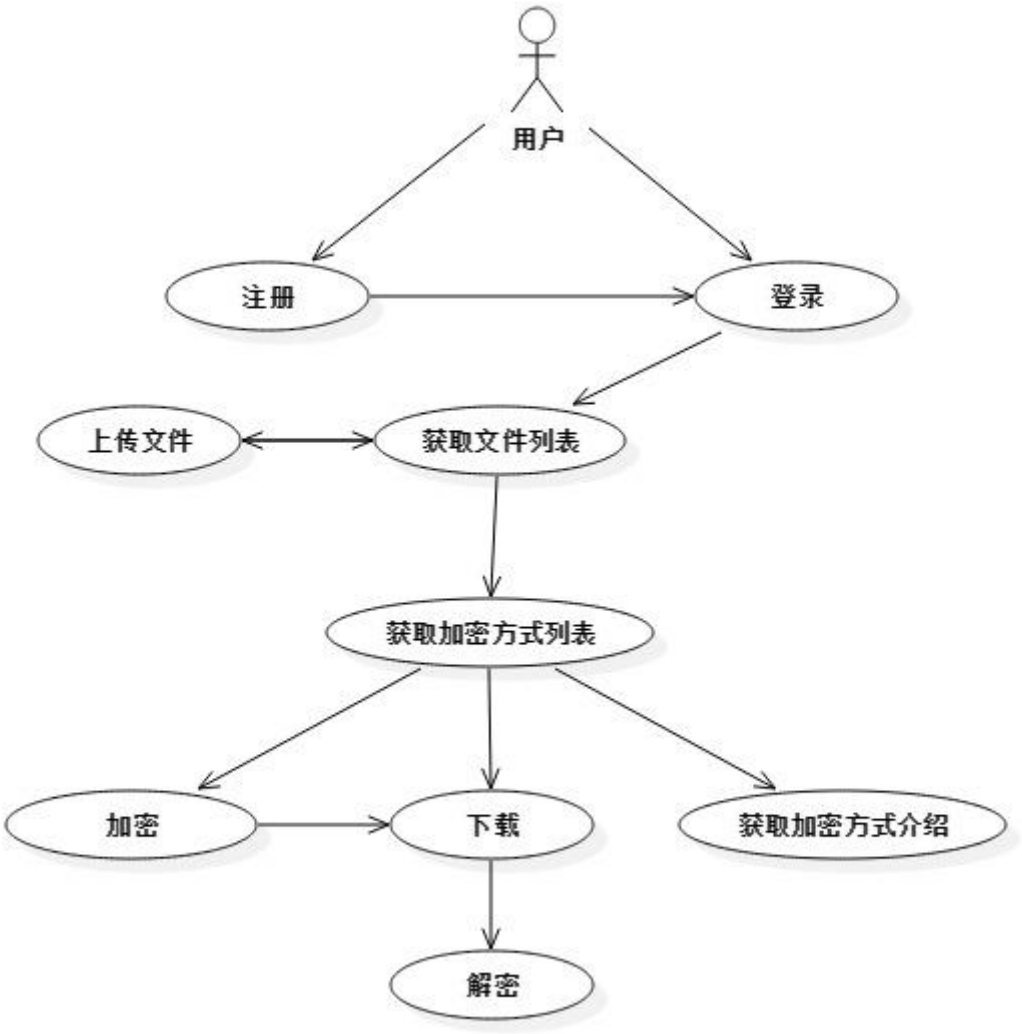

图 1

2.数据库 E-R 图：

将整体分为三部分，用户部分，文件部分，加密部分。E-R 图如图 2。

用户部分包含 User 用户库及 Session 会话库。用户库中保存用户基本信息，会话库中保存用户 Cookie 基本信息。

文件部分包含 File 文件库及 EncryptRelation 加密关系库。文件库保存文件基本信息，加密关系保存文件所对应加密类型信息。

加密部分包含 EncryptType 加密方式库，AES 算法库，DES 算法库，MD5 算法库，RSA 算法库。加密方式库保存加密方式的基本信息。其余算法库保存相关密钥，签名等。

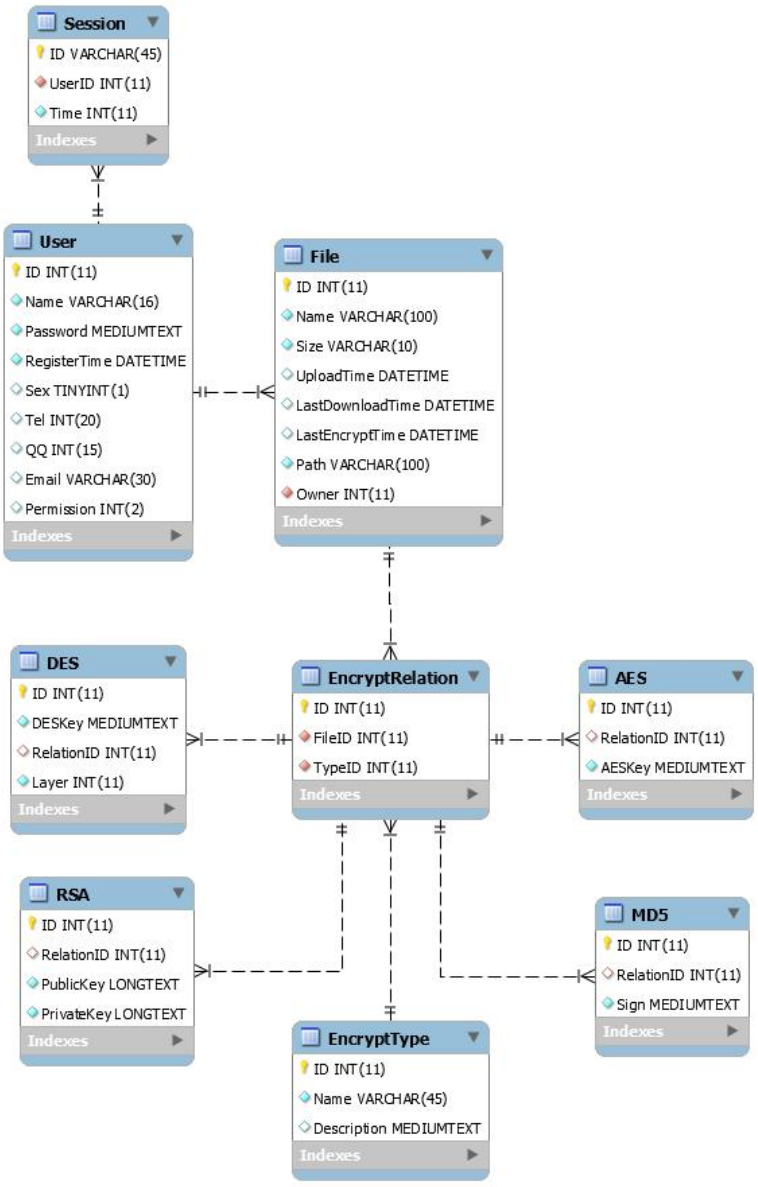

图 2

2.2.2 逻辑结构设计

逻辑结构具体设计图如图 3。说明如下：

用户库通过会话控制与会话库形成一对 N 的关系。

用户库通过传收文件与文件库形成一对 N 的关系。

文件库通关加密文件与加密关系库形成一对 N 的关系。

加密方式库通过选择加密方式与加密关系库形成 1 对 N 的关系

加密关系库通过数据加密与 DES 库形成一对 N 的关系。

加密关系库通过消息保护与 MD5 库形成一对 N 的关系。

加密关系库通过公钥加密与 RSA 库形成一对 N 的关系。

加密关系库通过区块加密与 AES 库形成一对 N 的关系。

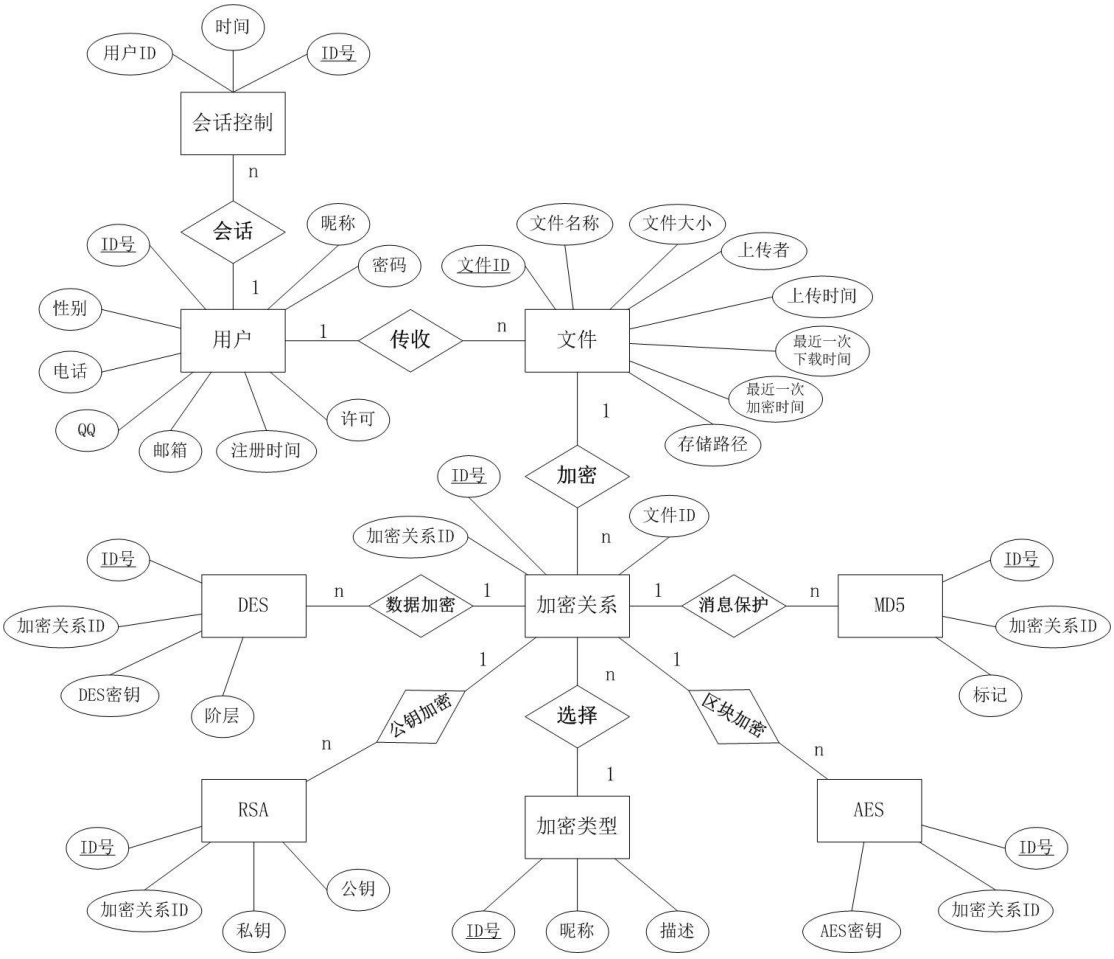

图 3

2.2.3 物理结构设计

1.用户信息表：

保存用户 ID，昵称，密码，注册时间，性别，电话，QQ 号，邮箱，权限。其中用户 ID 为主键，自增长，注册时间服务器自生成。具体设计表如表 1。

字段名	数据名	数据类型	描述
用户 ID 号	ID	Int（11）	主键
昵称	Name	Varchar(16)	
密码	Password	Mediumtext	
注册时间	RegisterTime	Datetime	
性别	Sex	TinyInt(1)	
电话	Tel	Int(20)	
QQ 号	QQ	Int(15)	
邮箱	Email	Varchar(30)	
权限	Permission	Int(2)	

表 1

2.文件传收表：

保存文件 ID，文件名称，文件大小，上传时间，上次下载时间，上次加密时间，存储路径，上传者。其中文件 ID 为主键，自增长，上传者为外键，关联用户信息表的用户 ID。具体设计表如表 2。

字段名	数据名	数据类型	描述
文件 ID 号	ID	Int（11）	主键
文件名称	Name	Varchar(16)	
文件大小	Size	Mediumtext	
上传时间	UploadTime	Datetime	
上次下载时间	LastDownloadTime	Datetime	
上次加密时间	LastEncryptTime	Datetime	
存储路径	Path	Varchar(100)	
上传者	Owner	Int(11)	外键

表 2

3.会话控制表：

保存会话 ID，用户 ID，时间。其中会话 ID 为主键，为服务器 SessionID，用户 ID 为外键，关联用户信息表的用户 ID。具体设计表如表 3。

字段名	数据名	数据类型	描述
会话 ID 号	ID	Varchar(45)	主键
用户 ID	UserID	Int(11)	外键
时间	Time	Int(11)	

表 3

4.加密关系表：

保存 ID 号，文件 ID，加密类型 ID。其中 ID 号为主键，自增长，文件 ID 为外键，关联文件传收表的文件 ID 号，加密类型 ID 为外键，关联加密类型表的 ID 号。具体设计表如表 4。

字段名	数据名	数据类型	描述
ID 号	ID	Int(11)	主键
文件 ID	FileID	Int(11)	外键
加密类型 ID	TypeID	Int(11)	外键

表 4

5.加密类型表：

保存 ID 号，名称，描述。其中 ID 号为主键，自增长。具体设计表如表 5。

字段名	数据名	数据类型	描述
ID 号	ID	Int(11)	主键
名称	FileID	Varchar(45)	
描述	TypeID	Mediutext	

表 5

6.MD5 表（消息摘要算法）：

保存 ID 号，关系 ID，摘要。其中 ID 号为主键，自增长，关系 ID 为外键，关联加密关系表的 ID 号。具体设计表如表 6。

字段名	数据名	数据类型	描述
ID 号	ID	Int(11)	主键
关系 ID	RelationID	Int(11)	外键
摘要	Sign	Mediutext	

表 6

7.AES 表（高级加密标准）：

保存 ID 号，关系 ID，AES 密钥。其中 ID 号为主键，自增长，关系 ID 为外键，关联加密关系表的 ID 号。具体设计表如表 7。

字段名	数据名	数据类型	描述
ID 号	ID	Int(11)	主键
关系 ID	RelationID	Int(11)	外键
AES 密钥	AESkey	Mediutext	

表 7

8.DES 表(数据加密算法)：

保存 ID 号，关系 ID，DES 密钥，加密阶层。其中 ID 号为主键，自增长，关系 ID 为外键，关联加密关系表的 ID 号。具体设计表如表 8。

字段名	数据名	数据类型	描述
ID 号	ID	Int(11)	主键
关系 ID	RelationID	Int(11)	外键
DES 密钥	DESkey	Mediutext	
阶层	Layer	Int(11)	

表 8

9.RSA 表(数据加密算法)：

保存 ID 号，关系 ID，公钥，私钥。其中 ID 号为主键，自增长，关系 ID 为外键，关联加密关系表的 ID 号。具体设计表如表 9。

字段名	数据名	数据类型	描述
ID 号	ID	Int(11)	主键
关系 ID	RelationID	Int(11)	外键
公钥	PublicKey	Longtext	
密钥	PrivateKey	Longtext	

表 9

2.3 详细设计

2.3.1 系统功能分析

针对项目需求，服务器采用 Spring+Struts 2+Hibernate 4 框架。客户端采用 MVP+Dagger 2+Retrofit 2+RxJava 框架。客户端与服务器交互，采用 Json 及表单格式传输。

针对服务器依赖包过多，版本不一致导致的冲突问题。采用 Maven 管理服务器代码。

针对客户端多线程异步问题，采用 RxJava。

针对服务器二级缓存问题，采用 ehcache。

针对密码加密，在客户端与服务器通信前进行一次 MD5，在服务器存入数据库前进行一次 MD5。即二次加密。

针对数据传输量问题，提供分页查询，以防查询量过大。

针对 URL 安全性问题，加入 CookieFilter 过滤器。

最后将整体分为，用户模块，文件模块，加密方式模块，加密关系模块。

2.3.2 技术说明

1.Spring：

Spring 框架是开源的 Java 平台的控制容器转化的应用程序框架。框架的核心功能可以由任何 Java 应用程序使用，并且在 Java EE 平台之上可以构建很多 Web 应用程序的扩展。 尽管该框架并没有强加任何特定的编程模型，但它已经在 Java 社区中变得流行起来，作为企业 JavaBeans（EJB）模型的替代，替代或甚至添加。

2.Struts 2：

Apache Struts 2 是开发 Java EE Web 应用程序的开源 Web 应用程序框架。它使用并扩展了 Java Servlet API，以鼓励开发人员采用模型视图控制器（MVC）架构。

3.Hibernate 4：

Hibernate ORM（简称 Hibernate）是 Java 编程语言的对象关系映射工具。它提供了将面向对象的域模型映射到关系数据库的框架。 Hibernate 通过用高级对象处理函数替换直接，持久的数据库访问来解决对象关系阻抗不匹配问题。主要功能是从 Java 类映射到数据库表，以及从 Java 数据类型映射到 SQL 数据类型。 Hibernate 还提供数据查询和检索功能。它生成 SQL 调用，并缓解开发人员对结果集的手动处理和对象转换。

4.Model-View-Presenter：

模型视图呈现器（MVP）是模型 - 视图 - 控制器（MVC）架构模式的推导，主要用于构建用户界面。在 MVP 中，主持人承担“中间人”的功能。 在 MVP 中，所有演示逻辑被推送给演示者。

5.Retrofit 2：

类型安全的 HTTP 客户端的安卓框架。是 OkHttp 的再次封装。它可以通过注解来描述 HTTP 请求，URL 参数，查询参数，同时还支持多个请求体和文件上传下载等。

6.Dagger 2：

Dagger 是针对 Java 和 Android 的完全静态的编译时依赖注入框架。旨在解决基于反思的解决方案的开发和性能问题的许多困扰。

7.RxJava：

RxJava 是由 Java VM 实现的反应性延伸。用于通过观察者序列来构建异步和基于事件的程序的库。它扩展观察者模式以支持数据/事件序列，并添加允许您以声明方式组合序列的运算符，同时提取对低级线程，同步，线程安全性和并发数据结构等问题的隐藏。

8.EhCache：

Ehcache 是广泛使用的开源 Java 分布式缓存，用于通用缓存，Java EE 和轻量级容器。 它具有内存和磁盘存储，复制和无效，监听器，缓存加载器，缓存扩展，缓存异常处理程序，gzip 缓存 servlet 过滤器，RESTful 和 SOAP API。Ehcache 是根据 Apache 开源许可证提供的，并且得到了积极的支持。

9.MD5:

MD5 算法是广泛使用的哈希函数，产生 128 位哈希值。 虽然 MD5 最初被设计为用作加密哈希函数，但是已经发现它具有广泛的漏洞。 它仍然可以用作校验和来验证数据的完整性，但只能用于无意的损坏。

10.DES:

DES 全称为数据加密标准，即数据加密标准，是一种使用密钥加密的块算法，1977 年被美国联邦政府的国家标准局确定为联邦资源处理标准（FIPS），并授权在非密级政府通信 中使用，随后该算法在国际上广泛流传开来。

11.AES:

高级加密标准（英语：Advanced Encryption Standard，缩写：AES），在密码学中又称 Rijndael 加密法，是美国联邦政府采用的一种区块加密标准。这个标准用来替代原始的 DES，已经被多方 分析，广为全世界所用。经过五年的甄选流程，高级加密标准由美国国家标准与技术研究院（NIST）于 2001 年 11 月 26 日发布于 FIPS PUB 197，并在 2002 年 5 月 26 日 日成为有效的标准。2006 年，高级加密标准已成为对称密钥加密中最流行的算法之一。

12.RSA:

RSA 公钥加密算法是 1977 年由罗纳德·李维斯特（Ron Rivest），阿迪·萨莫尔（Adi Shamir）和伦纳德·阿德曼（Leonard Adleman）一起提出的。RSA 是目前最有影响力的公钥加密算法，它能够抵抗到目前为止已知的绝大多数密码攻击，已被 ISO 推荐为公钥数据加密标准。

2.3.3 系统功能模块设计及实现

将整体分为，用户模块，文件模块，加密方式模块，加密关系模块，每个模块所对应的基本功能如下：

1.用户模块：登录，注册

2.文件模块：上传，下载，获取文件列表

3.加密方式模块：获取加密方式

4.加密关系模块：加密，解密

针对用户模块，分别给出服务器与客户端实现图，服务器如图 4，客户端如图 5。其余模块类似。具体解释如下：

1.服务器为 MVC 模式。

Model 中包含 BaseDaoSupport，数据库操作方法基类。BaseDaoSupportImpl，实现 BaseDaoSupport，通过 Resource 注解获取 SessionFactory 实例后，通过 Transactional 注解开事务管理的数据库操作实现类。UserEntity，包含全部关系的 Hibernate 逆向工程类。UserBaseEntity，不包含外键关系，用于 Json 传输及解析的 Hibernate 逆向工程类。

Controller 中包含 BaseUserService，继承 BaseDaoSupport 的 User 相关操作数据库方法的接口定义。UserServiceImpl，实现 BaseUserService，继承 BaseDaoSupportImpl，通过 Service 注解注入及 Transactional 注解开事务管理的可操作数据库的 User 相关操作数据库方法实现类。BaseAction，通过 Resource 注解获取 Service 接口实例的控制器抽象基类。UserAction，继承 BaseAction，通过 Controller 注解管理，包含具体操作方法对实例进行操作的实际控制器类。CookieFilter，负责管理 Cookie，过滤 URL 请求的过滤器。

View 为 Android 客户端。

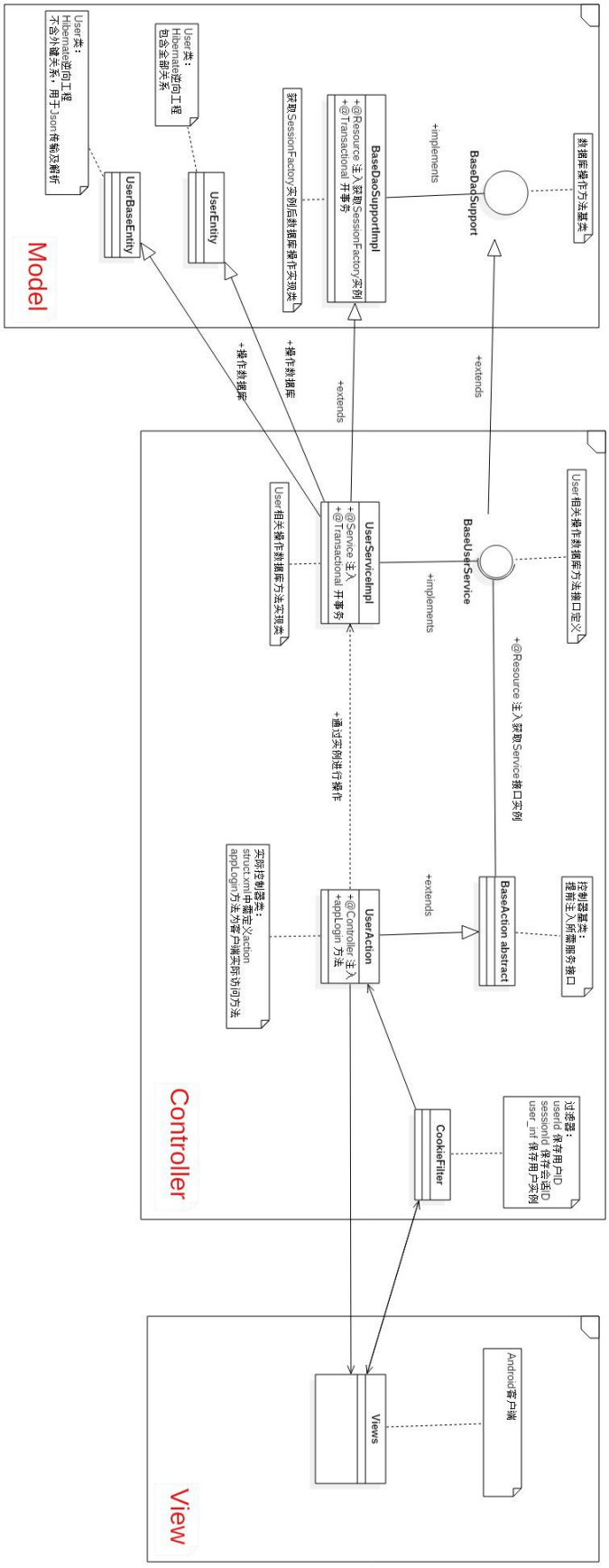
Model
User类：
Hibernate逆向工程
不含外键关系，用于Json传输及解析
User类：
Hibernate逆向工程
包含全部关系
UserBaseEntity
UserEntity
获取SessionFactory实例后数据库操作实现类
BaseDaoSupportImpl
+@Resource 注入获取SessionFactory实例
+@Transactional 开事务
+implements
BaseDaoSupport
数据库操作方法基类
+操作数据库
+操作数据库
+extends
+extends
Controller
User相关操作数据库方法实现类
UserServiceImpl
+@Service 注入
+@Transactional 开事务
BaseUserService
User相关操作数据库方法接口定义
+implements
+@Resource 注入获取Service接口实例
+通过实例进行操作
实例控制器类：
struct.xml中配置action
applogin方法为客户端实际访问方法
UserAction
+@Controller 注入
+applogin 方法
+extends
BaseAction abstract
控制器基类：
提前注入所需服务接口
CookieFilter
过滤器：保存用户ID
userId 保存用户ID
sessionId 保存会话ID
user_inf 保存用户实例
View
Views
Android客户端

图 4

2.客户端为 MVP 模式。

View 中包含 BaseActivity 基类。Contract.View，Contract 接口中定义基本交互方法的接口。Activity，继承 BaseActivity，实现 Contract.View，负责用户交互的实现类。

Presenter 中包含 Component，被 Activity 中 injectDependences 方法初始化，通过 Component 注解定义的组件接口。BaseRepository，网络操作相关方法的 RxJava 格式的方法定义接口。Repository，实现 BaseRepository，通过 ApiService 初始化，网络操作相关方法具体实现类。BaseUseCase，基于该模块模型，网络操作相关的方法定义接口。UseCase，实现 BaseUseCase，通过 BaseRepository 初始化，可以获取模型，设定模型的网络操作相关方法的具体实现。Contract.Presenter，Contract 接口中定义基本逻辑操作方法的接口。Presenter，实现 Contract.Presenter，通过 RxJava 进行主要异步耗时逻辑操作的主持人类。Module，通过 BaseRepository 初始化，通过 Provides 注解注入 UseCase 及 Presenter，通过 Module 注解定义的模块类。

Model 中包含通过 Expose 注解过滤相关属性，对应所属模块的序列化对象。

WebApi 中包含 ApiService，Retrofit2 注解定义的网络接口。Interceptor，Cache，Cookie，HttpLog 相关拦截器。

Server 为 J2EE 服务器。

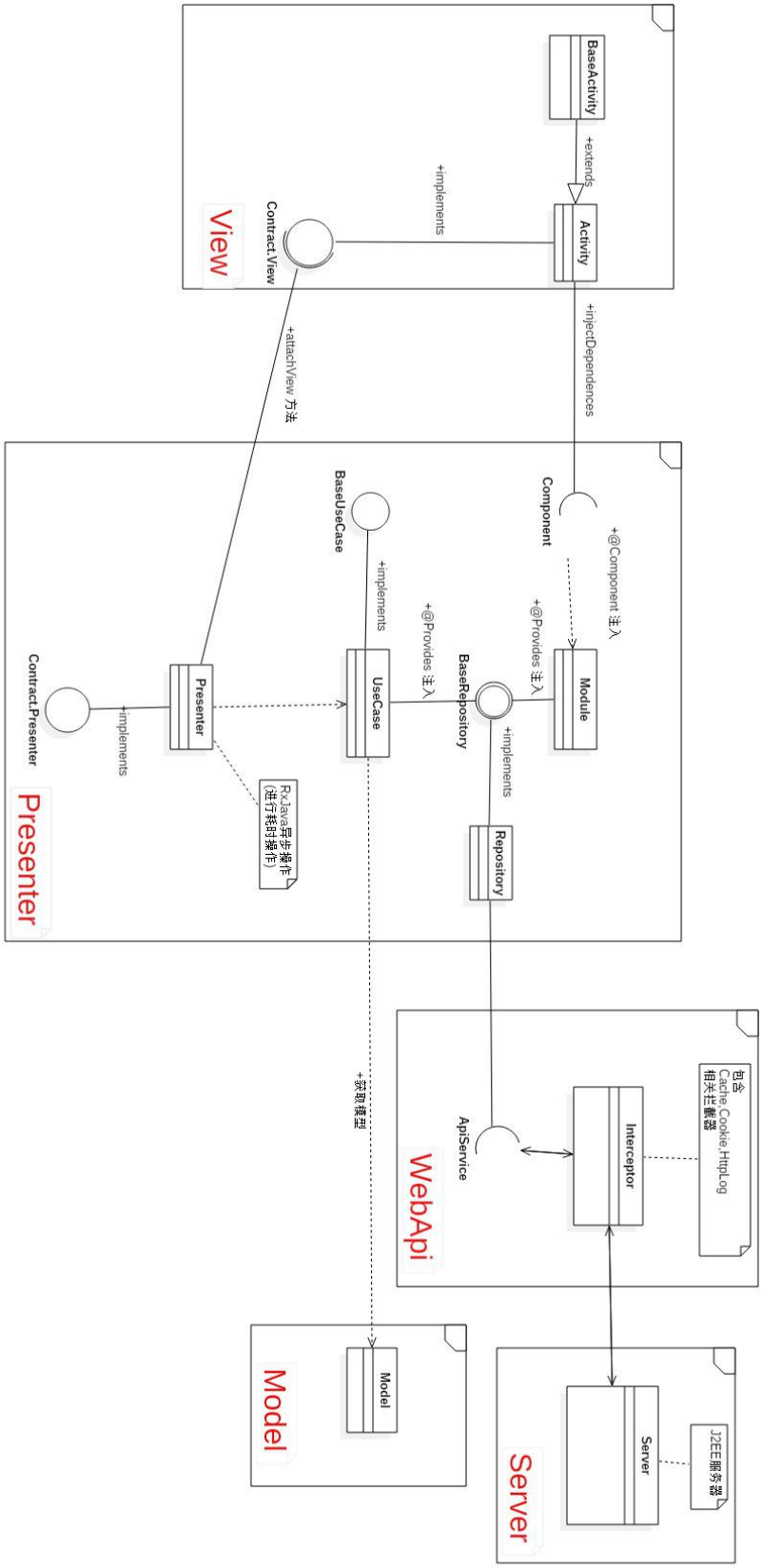
View
BaseActivity
+extends
Activity
+implements
Contract.View
+attachView 方法
+injectDependences
Presenter
BaseUseCase
+implements
Component
+@Component 注入
+@Provides 注入
Module
Contract.Presenter
+implements
Presenter
UseCase
+@Provides 注入
BaseRepository
+implements
Repository
RxJava异步操作
(进行耗时操作)
+获取数据
WebApi
ApiService
Interceptor
包含
Cache Cookie HttpLog
相关拦截器
Model
Model
Server
Server
J2EE服务器

图 5

2.4 交互接口

2.4.1 方法接口

1.登录方法：

属于 userAction 的 login 方法。请求体为 Json 格式 UserBaseEntity 对象。返回 Json 格式 UserEntity 对象。具体定义表如表 10。

URL	/App/login	
Method	userAction.login	
Parameter		
Body	Json	UserEntity
Response	Json	UserBaseEntity

表 10

2.注册方法：

属于 userAction 的 register 方法。请求体为 Json 格式 UserBaseEntity 对象。返回 Json 格式 UserEntity 对象。具体定义表如表 11。

URL	/App/register	
Method	userAction.register	
Parameter		
Body	Json	UserEntity
Response	Json	UserBaseEntity

表 11

3.上传文件方法：

属于 fileUploadAction 的 execute 方法。请求体为 String 类型的 fileName，fileSize，owner 参数，List<File>类型的 file 参数，List<String>类型的 fileFileName，fileContentType 参数。返回 Json 格式 FileBaseEntity 对象。具体定义表如表 12。

URL	/App/uploadFile	
Method	fileUploadAction.execute	
Parameter		
Body	String	fileName
	String	fileSize
	String	owner
	List<File>	file
	List<String>	fileFileName
	List<String>	fileContentType
Response	Json	FileBaseEntity

表 12

4.获取文件列表方法：

属于 fileAction 的 getFileList 方法。请求参数为 owner 用户 ID。返回 Json 格式 List<FileBaseEntity>对象。具体定义表如表 13。

URL	/App/getFileList	
Method	fileAction.getFileList	
Parameter	owner	用户 ID
Body		
Response	Json	List<FileBaseEntity>

表 13

5.下载文件方法：

属于 fileDownloadAction 的 execute 方法。请求体为 Json 格式 FileEntity 对象。返回 InputStream 类型的 File。具体定义表如表 14。

URL	/App/downloadFile	
Method	fileDownloadAction.execute	
Parameter		
Body	Json	FileEntity
Response	InputStream	File

表 14

6.通过分页获取文件列表方法：

属于 fileAction 的 getFileListByPaper 方法。请求参数为 owner 用户 ID，rows 每页显示条数，paper 显示页码。返回 Json 格式 List<FileBaseEntity>对象。具体定义表如表 15。

URL	/App/getFileListByPaper	
Method	fileAction.getFileListByPaper	
Parameter	owner	用户 ID
	rows	每页显示条数
	paper	显示页码
Body		
Response	Json	List<FileBaseEntity>

表 15

7.获取加密类型方法：

属于 encryptTypeAction 的 getEncryptType 方法。返回 Json 格式 List<EncryptTypeBaseEntity>对象。具体定义表如表 16。

URL	/App/getEncryptType	
Method	encryptTypeAction.getEncryptType	
Parameter		
Body		
Response	Json	List<EncryptTypeBaseEntity>

表 16

8.加密文件方法：

属于 encryptRelationAction 的 encryptFile 方法。请求参数为 file_id 用户 ID，type_id 加密类型 ID，des_keyDES 密钥，des_layerDES 加密层数。返回 Json 格式 EncryptRelationBaseEntity 对象。具体定义表如表 17。

URL	/App/encryptFile	
Method	encryptRelationAction.encryptFile	
Parameter	file_id	文件 ID
	type_id	加密类型 ID
	des_key	DES 密钥
	des_layer	DES 加密层数
Body		
Response	Json	EncryptRelationBaseEntity

表 17

2.4.1 Json 接口

1.UserBaseEntity 对象：

包含 int 类型 id,String 类型 name,password,email,java.sql.Timestamp 类型的 registerTime，Byte 类型的 sex，Integer 类型的 tel，qq，permission。具体定义表如表 18。

Name	UserBaseEntity	
Column	int	id
	String	name
	String	password
	java.sql.Timestamp	registerTime
	Byte	sex
	Integer	tel
	Integer	qq
	String	email
	Integer	permission

表 18

2.FileBaseEntity 对象：

包含 int 类型 id，owner，String 类型 name，size，java.sql.Timestamp 类型的 uploadTime,lastDownloadTime,lastEncryptTime。具体定义表如表 19。

Name	FileBaseEntity	
Column	int	id
	String	name
	String	size
	java.sql.Timestamp	uploadTime
	java.sql.Timestamp	lastDownloadTime
	java.sql.Timestamp	lastEncryptTime
	int	owner
	String	path

表 19

3.EncryptTypeBaseEntity 对象：

包含 int 类型的 id，String 类型的 name，description。具体定义表如表 20。

Name	EncryptTypeBaseEntity	
Column	int	id
	String	name
	String	description

表 20

4.EncryptRelationBaseEntity 对象：

包含 int 类型的 id，fileId，typeId。具体定义表如表 21。

Name	EncryptRelationBaseEntity	
Column	int	id
	int	fileId
	int	typeId

表 21