



**系统设计报告**

**项目名称： clipboard同步剪切板**

**小组成员：赵文 201721130272**

**姚镇东 201721130243**

**孙睿 201721130223**

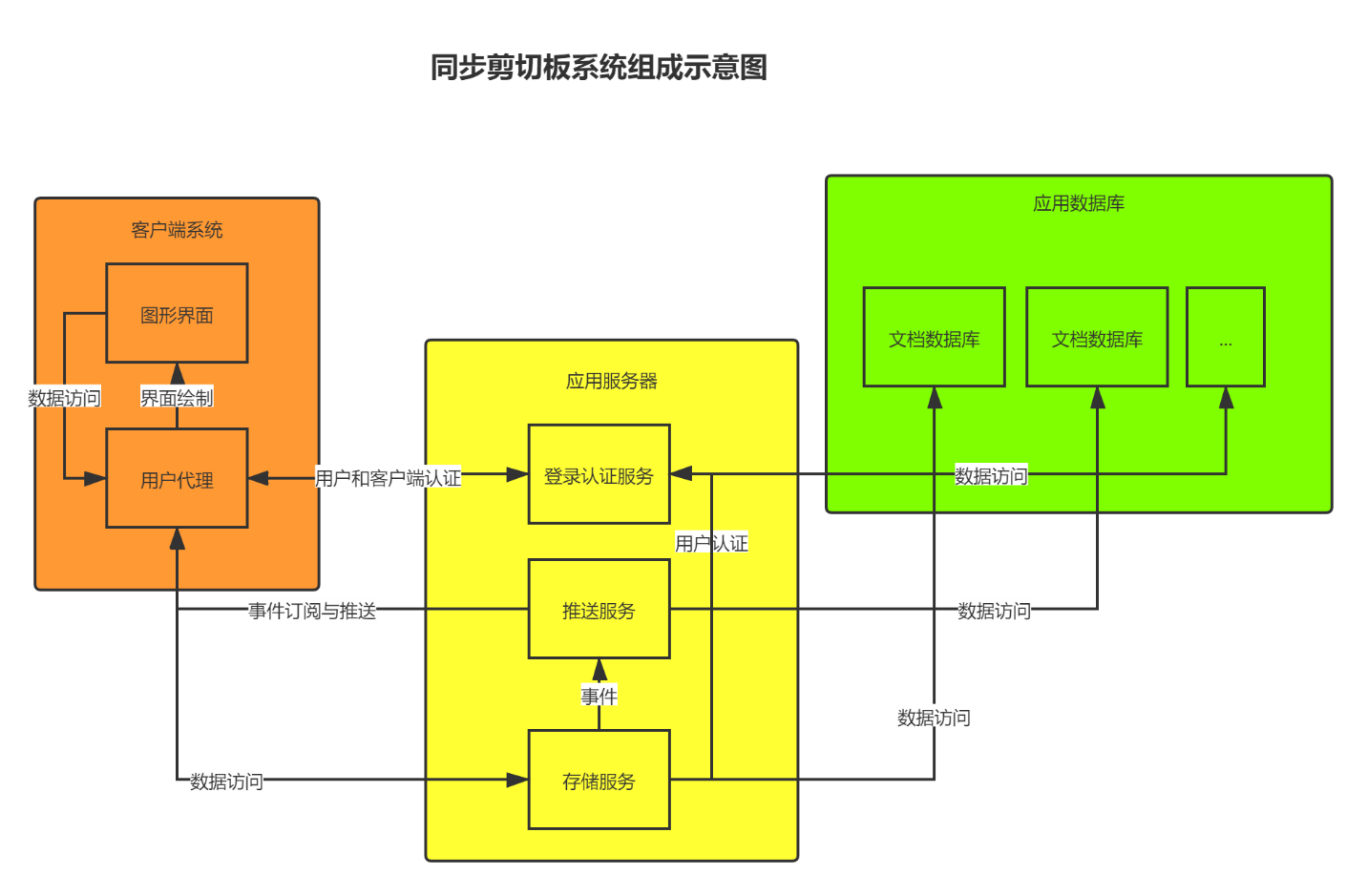
**魏杨政 201721130262**

**指导教师： 李胜老师**

**二〇二〇年三月**

1. 系统概览

同步剪切板的目的是同步用户在多台计算机上的剪切板内容。系统第一阶段被设计运行在桌面平台上，下一阶段进入移动平台。系统由五个主要部分组成，分别为：应用数据库，应用服务器，登录认证服务器，用户客户端。应用数据库存储用户数据以及系统运行数据，应用服务器对客户端提供数据访问和功能接口，登录认证服务器负责用户的登录认证，客户端程序为用户提供图形界面和基础功能。系统的全局组成如下图所示。



用户通过图形界面访问系统提供的同步剪切板服务，用户的一系列操作通过客户端的用户代理发送给应用服务器处理。用户代理需要并处理本地数据缓存，维持与服务器的连接，处理来自本地系统的事件，处理来自应用服务器的事件，监听本地操作系统事件。按照操作性质的不同，应用服务器被划分为三个服务——登录认证服务、推送服务和存储服务。登录认证服务处理用户登录，注册和用户认证，认证授权类似于OAutho2协议，用户代理访问服务前首先需要通过登录获取访问令牌，获取到访问令牌之后用户代理依照此令牌访问推送服务和存储服务。推送服务负责将应用服务器中产生的事件推送到客户端，用户代理通过订阅服务器事件来接受来自服务器的事件，之后更具事件来完成业务逻辑。存储服务提供用户数据的存储，包括账户信息，同步的剪切板数据。应用数据库是所有应用数据存储的地方，用户数据，系统数据被存放在此处。

1. 客户端图形界面设计

客户端图形界面较为简单，由一个工具栏和一个列表组成。工具栏提供本地内容检索，列表显示剪切板内容。通过对话框来设置应用、登录和注册等交互。

图形界面的交互方式不局限于下面的实例，在实际编写过程中更加情况添加组件和元素。

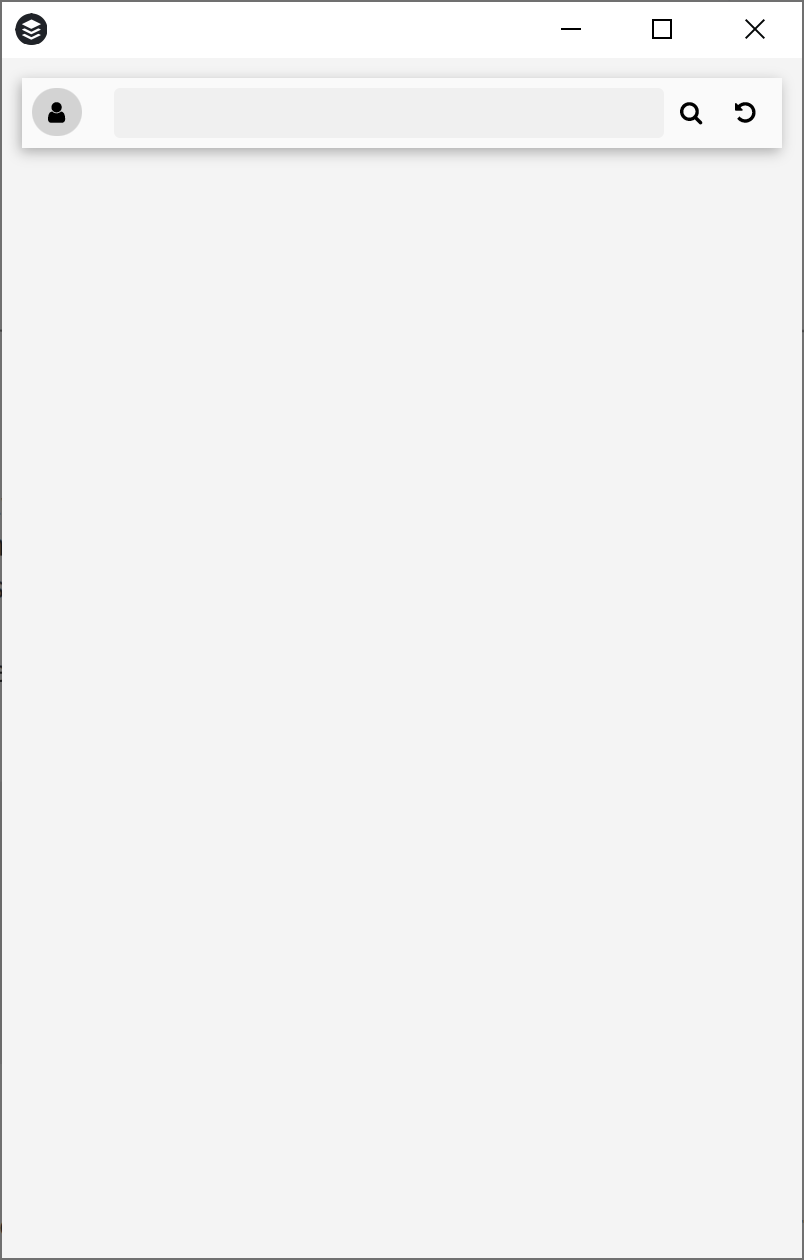


图 1主界面

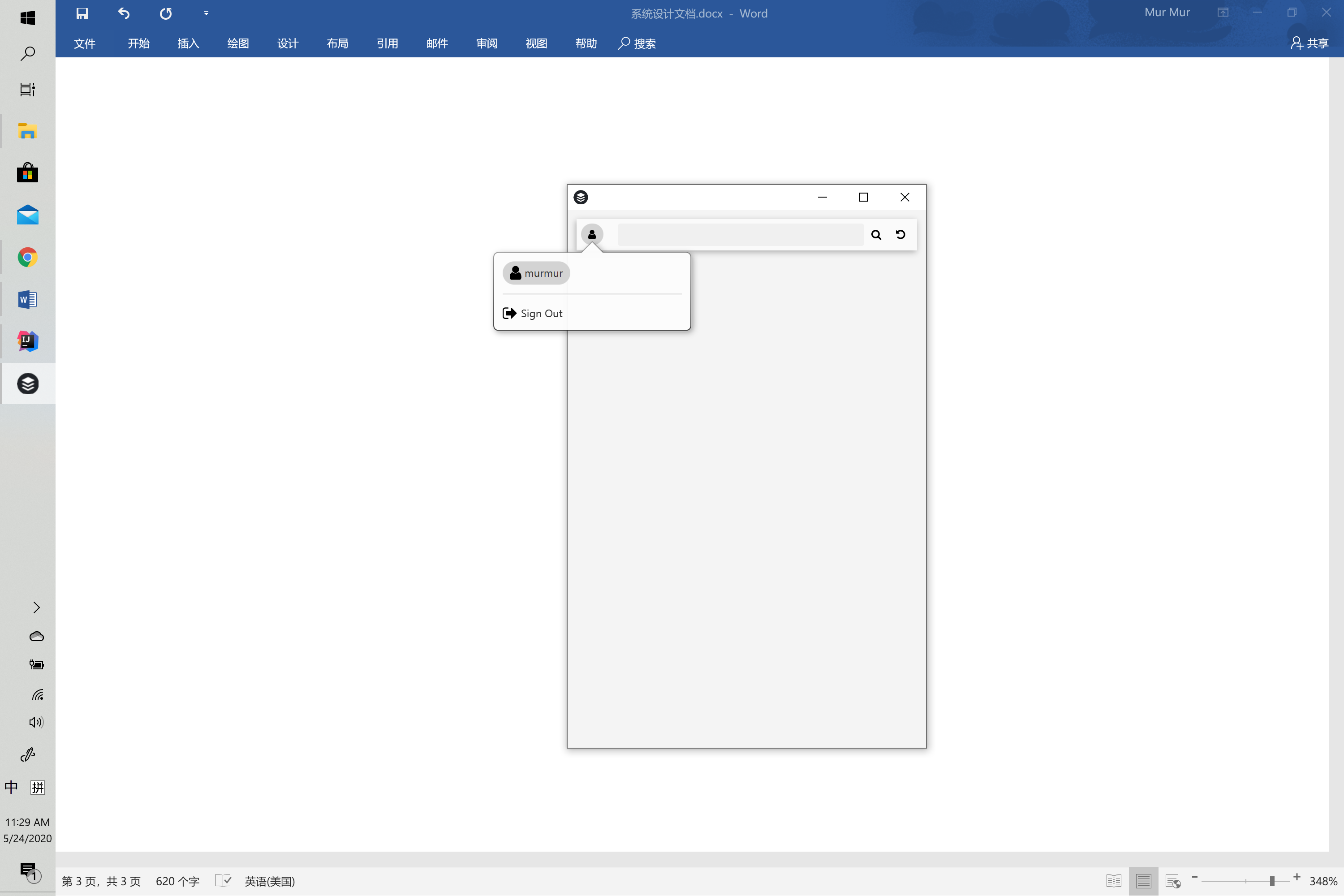


图 2主界面菜单

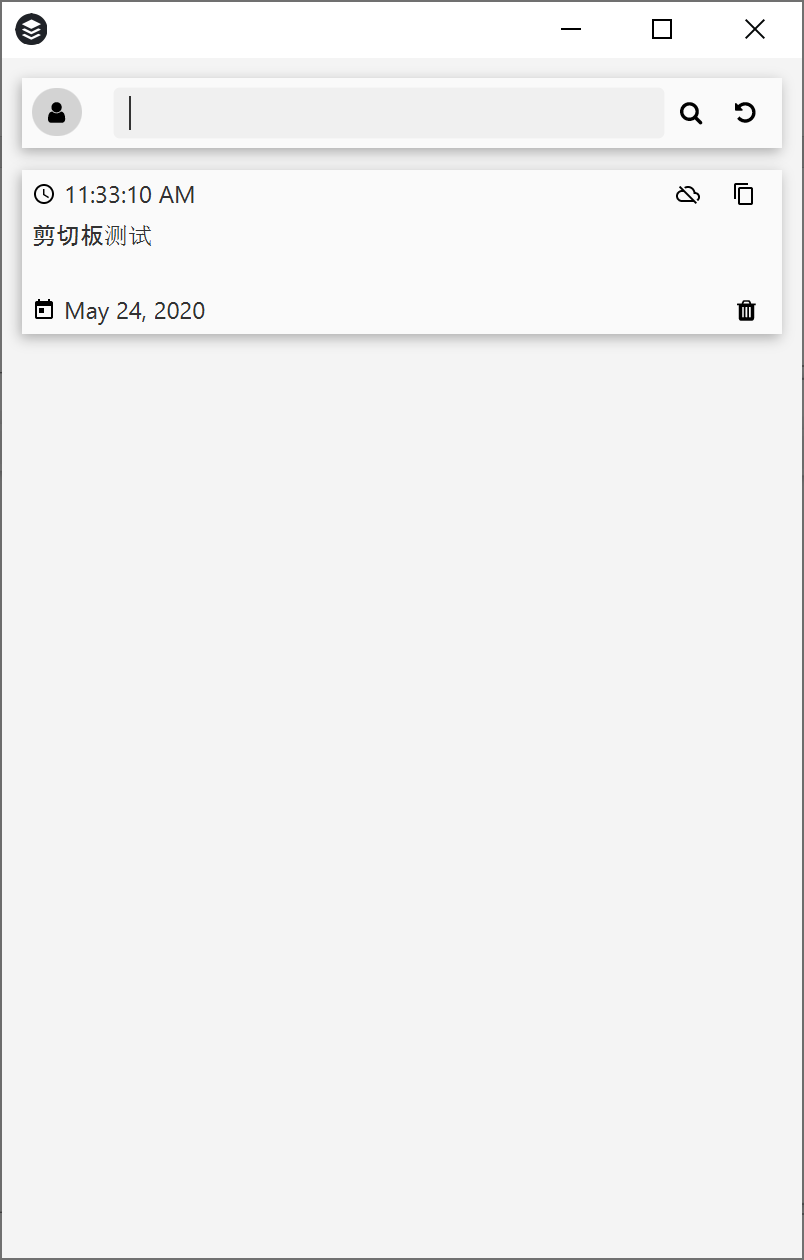


图 3剪切板条目

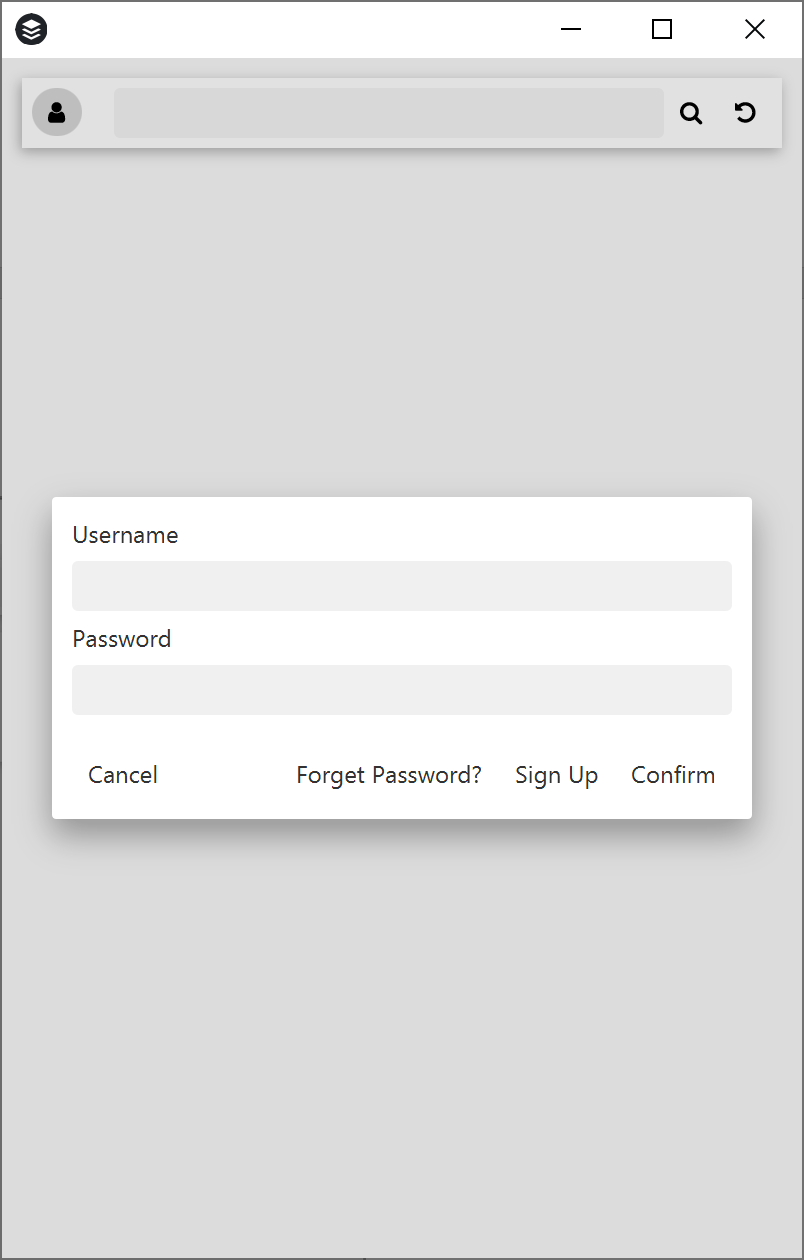


图 4用户登录对话框

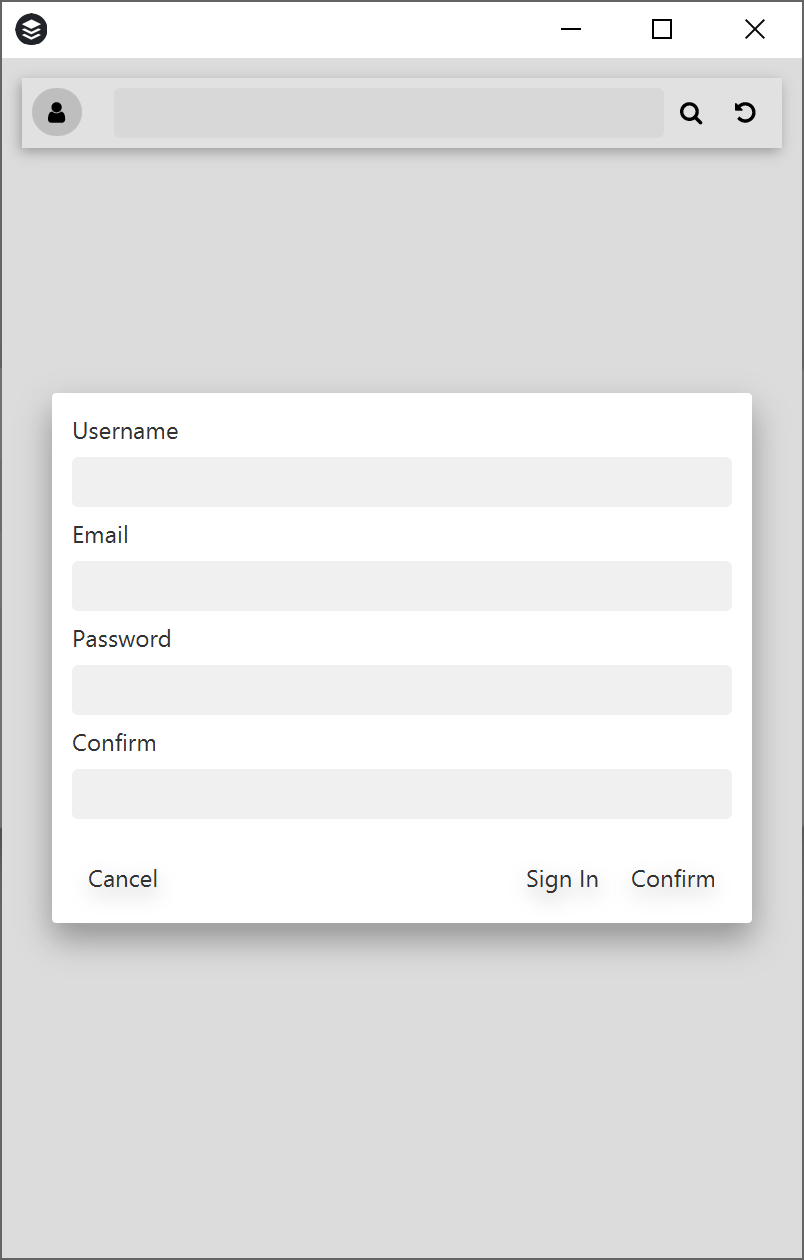


图 5用户注册对话框

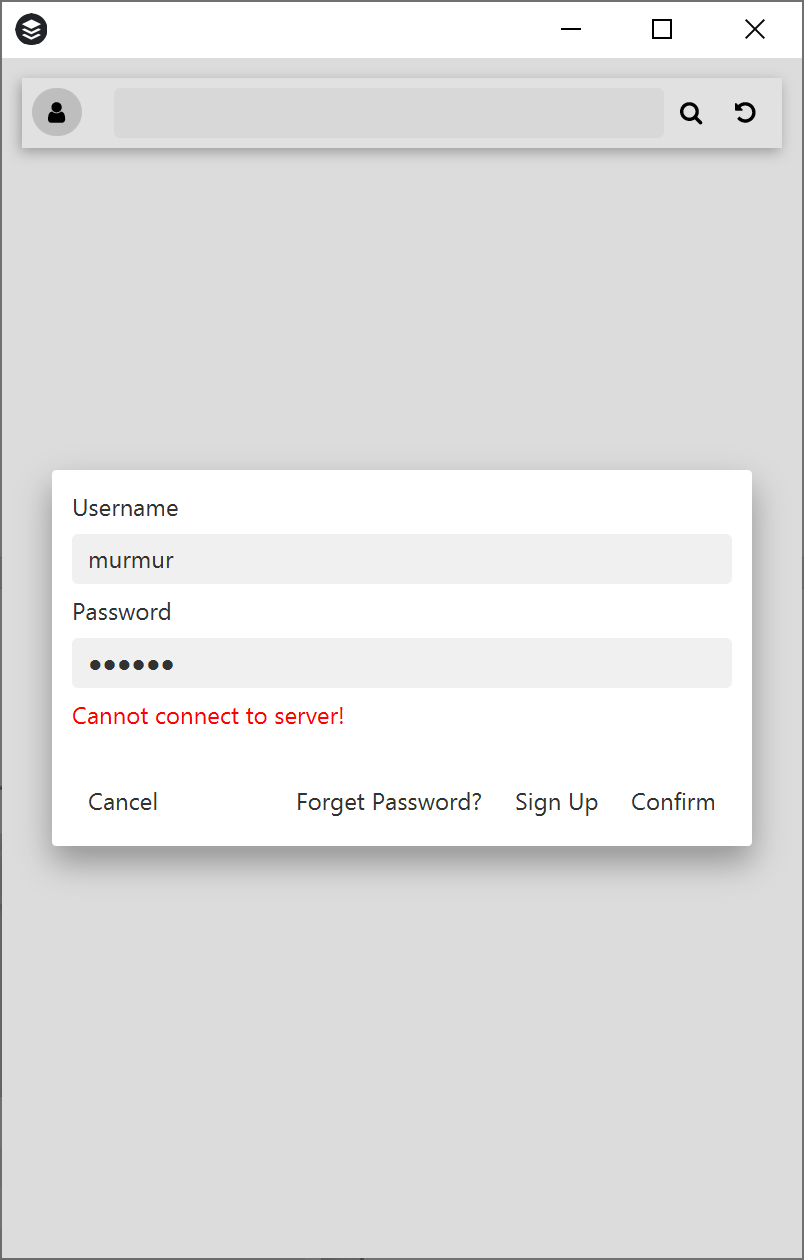


图 6用户登录错误提示

1. 客户端代理

客户端代理的作用是接受来自图形界面的事件和来自操作系统的事件以及缓存数据到本地和与服务器同步数据。用户代理需要维持至少三个个线程的持续运行，一个是剪切板监听线程，另一个是服务器事件监听线程，最后是守护线程，负责控制剪切板线程、服务器事件线程的生命周期控制，以及处理来自图形界面的数据访问和操作指令。可以将界面事件处理和数据访问从守护线程中抽取开来，交给图形界面的主线程处理，也可独立为客户端代理线程。

下图描述了剪贴板线程和服务器事件监听线程的基础状态模型。应用程序启动时，在启动本地监听事件的触发下，启动监听线程，进入空闲等待，一旦有新的事件发送，将事件发送给注册的处理器处理。处理或者监听过程中的异常将会导致线程进入异常终止状态，最终一条线程终止事件会被发送给守护线程。

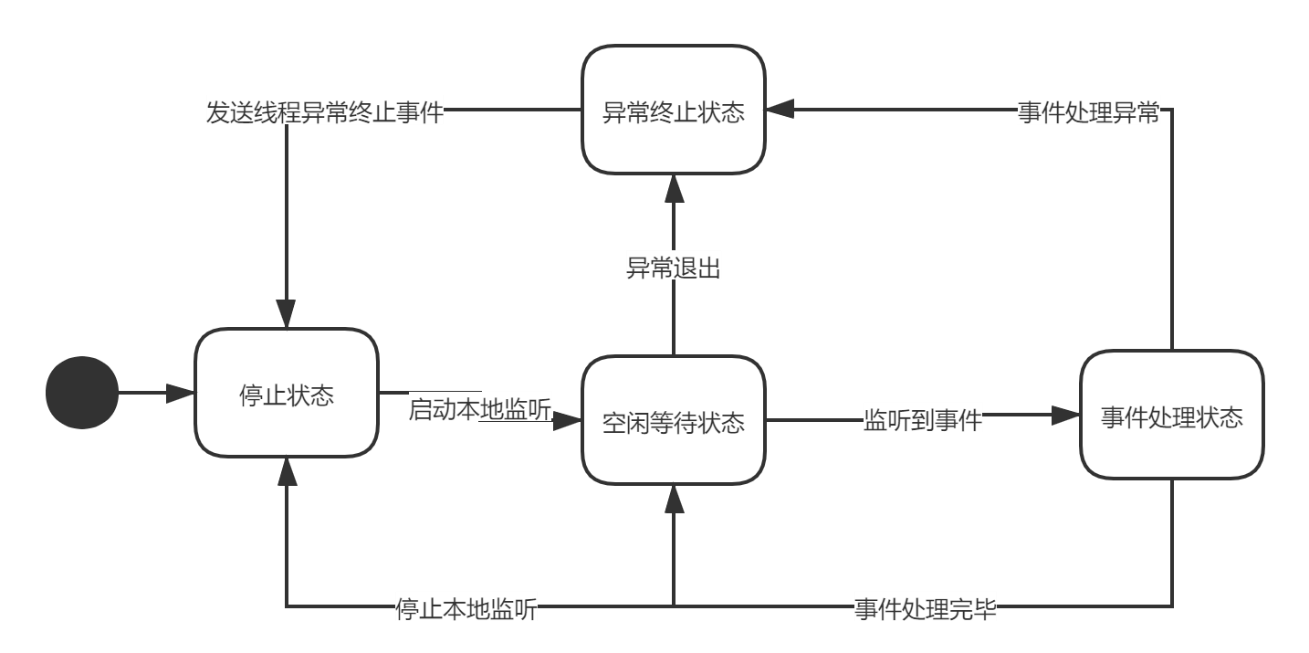


图 7监听线程模型

守护线程收到监听线程异常终止后，将检查应用上下文AppContext决定重启策略。

根据异常原因， 对网络异常按照重试的策略处理，重试时间按照retry = min(retry \* k, max)计算，初始时retry = 0.02, k = 1.5，max = 30，单位为分钟；对授权错误发送错误信息到图形界面，提示用户重新登录等；对其它错误尝试重启若干次之后停止，返回错误信息给图形界面。

用户代理的本地缓存使用h2 database存储，存储内容为应用程序数据，剪切板缓存，登录用户信息，访问令牌。其组成结构可用下图来表示

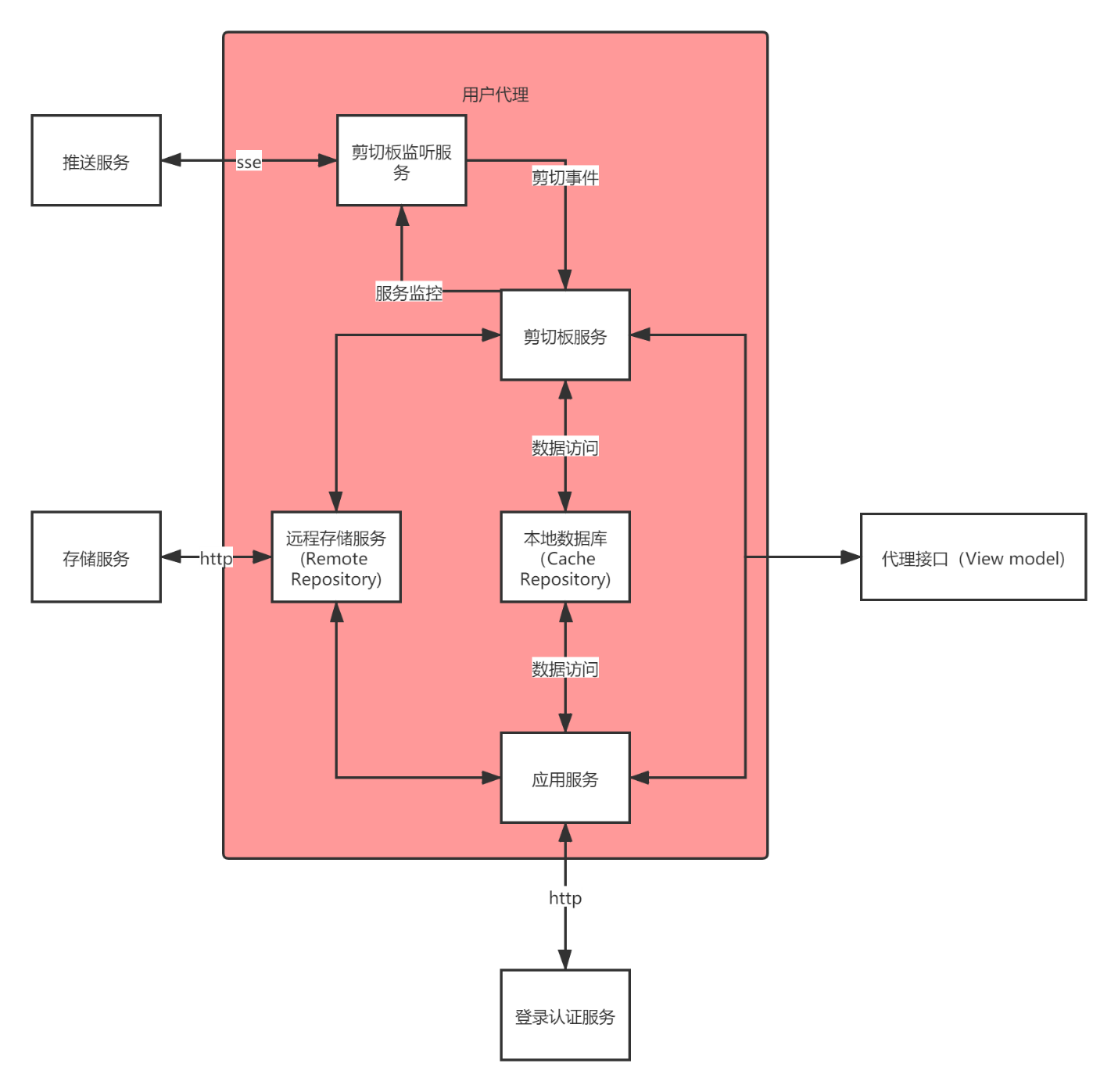
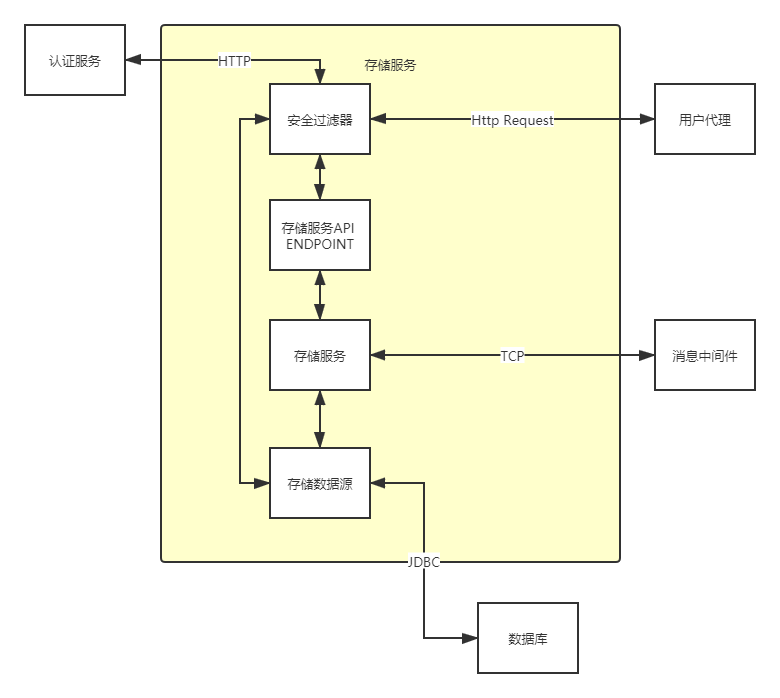


图 8用户代理结构

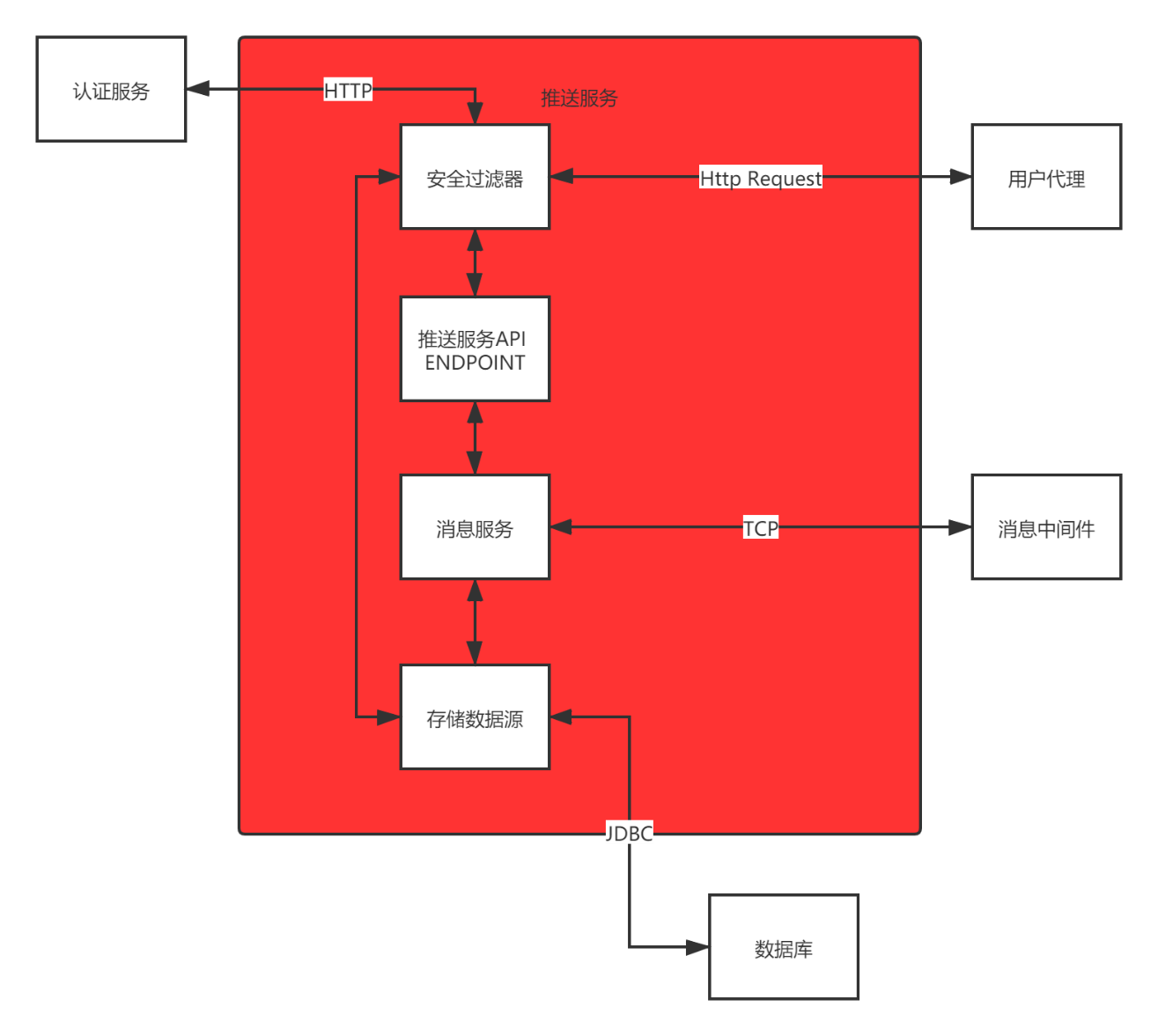
1. 存储服务

存储服务提供基本的CRUD操作，对外提供REST API。存储服务的结构如下图所示客户端代理通过HTTP 请求来访问存储服务API，请求首先经过安全滤器的过滤，去除掉非法请求，然后由API ENDPOINT路由到存储服务，来处理业务逻辑。

1. 推送服务

推送服务将来自服务器的事件推送给订阅的客户端，推送方式使用HTTP 中的SSE来实现，推送服务结构类似于存储服务。只是提供的端点API和中间业务层的逻辑处理不同。

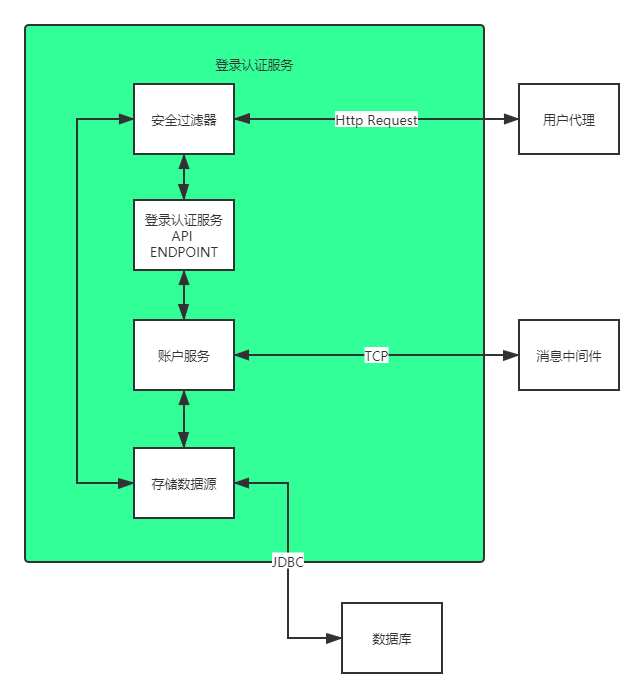
结构如下图所示：



推送服务需要记录每一个连接的客户端会话，将事件正确无误的从消息队列中取出发送给订阅的客户端，实现方式可以采用Kafaka。

1. 登录认证服务

登录认证服务管理用户的登录，用户访问令牌的校验，令牌的刷新等与账户安全相关的功能。结构依旧类似于前面所述的存储服务和推送服务。此处不多做描述。



1. 数据设计

从实际的需求出发，用户与用户之间没有关联，仅仅存在于用户与用户的剪切板内容。用户实际使用的时候会产生大量临时数据，适合于半结构化的数据库。因此选用MongoDB作为我们的数据存储。

* 1. 账户集合 Account Collection

@Document  
public class Account {  
 @Id  
 public String id;  
 @Indexed(unique = true)  
 public String username;  
 public String password;  
 @Indexed(unique = true)  
 public String email;  
 public String avatar;  
 public Date latestEmailChange;  
 public String oldEmail;  
 public Integer status;  
  
 public Date timestamp;  
  
 public Date create;  
 public Date update;  
  
 public enum AccountStatus {  
 *ACCOUNT\_STATUS\_ACTIVATE*(0),  
 *ACCOUNT\_STATUS\_REGISTERED*(1),  
 *ACCOUNT\_STATUS\_BLOCK*(2);  
  
 public Integer STATUS;  
 AccountStatus(int status) {  
 this.STATUS = status;  
 }  
 }  
  
 public static class AccountSupport {  
 public final static String *id* = "id";  
 public final static String *username* = "username";  
 public final static String *email* = "email";  
 public final static String *avatar* = "avatar";  
 public final static String *oldEmail* = "oldEmail";  
 public final static String *password* = "password";  
 public final static String *status* = "status";  
 public final static String *latestEmailChange* = "latestEmailChange";  
 public final static String *create* = "create";  
 public final static String *update* = "update";  
  
 }  
}

* 1. 剪切板内容集合Content Collection

public class Content {  
 public String id;  
 public String account;  
  
 public String content;  
 public String hash;  
 public Boolean star;  
 public Integer state;  
  
 public Date create;  
 public Date update;  
}

* 1. 令牌签名密钥集合 Access Token Sign Key Collection

@Document("access\_token\_sign\_key")  
public class AccessTokenSignKey {  
 public String id;  
 public Integer status;  
 public Binary publicKey;  
 public Binary privateKey;  
 public String publicKeyFormat;  
 public String privateKyeFormat;  
  
 public static enum AccessTokenSignKeyStatus {  
 *ACCESS\_TOKEN\_SIGN\_KEY\_STATUS\_MASTER*(0),  
 *ACCESS\_TOKEN\_SIGN\_KEY\_STATUS\_SECONDARY*(1);  
 public int STATUS;  
 AccessTokenSignKeyStatus(int status) {  
 this.STATUS = status;  
 }  
 }  
}

* 1. 令牌控制集合Access Token Control Collection

@Document("access\_token\_control")  
public class AccessTokenControl {  
 @Indexed(unique = true)  
 public String token;  
}

1. REST API接口

接口根据实际情况编写，原则上遵循RESTful风格。