Deusto

Facultad de Ingeniería
Ingeniaritza Fakultatea

DEUSZUM

Terres Escudero, Erik B.
Zarate Jayo, Gorka
Zugazaga Alonso, Aritz
Acha Aristegui, Amaia
Gago López,Leire
San Millan Langa, Lander

Curso 2019/2020

Ing. Informatica

Programacion III

Contents

1 INTRODUCCIÓN

DeusZum es una aplicación de gestión de dinero online para la gestión de pagos, proyectos o costes grupales. DeusZum ofrece un sistema seguro de transferencias de forma que el cliente no tenga que preocuparse de que este pasando con su dinero en todo momento.

DeusZum estará implementado en un servidor y daría servicio a los usuarios a través de un programa en Windows y una aplicación en Android. Cada una de ellas implementada para conectarse a un servidor central, en el cual se almacenarán los datos, procesarán las peticiones y se garantizará la seguridad de la información.

El servicio que ofrece DeusZum está dividido en diferentes sectores, algunos puramente económicos y otros de análisis de la conducta o de las transacciones. El servidor ejecutara todas las peticiones de los usuarios y devolverá la información que se haya solicitado. Toda la información mientras se enviá se encontrara encriptada para la seguridad del usuario.

2 FORMA DE TRABAJO

2.1 Division del trabajo

Para la realización del proyecto, hemos optado por la división del trabajo en grupos con temáticas separadas. Los dos grupos tendrán que trabajar en una categoría central entre IA y Seguridad, y, trabajar en una parte de la implementación y algoritmia. Las categorías y subcategorías serian las siguientes:

Implementacion	Uso de datos	Seguridad
Servidor	Data Analysis	Database
Windows	IA	Criptografia
Android	Big Data	Proteccion

Cada uno de los grupos tendra un conjunto de tareas las cuales se diviran en *tareas principales* y *tareas secundarias*. Las tareas principales son aquellas a las que se les daran prioridad y buscamos que esten terminadas a la hora de terminar el proyecto. Las tareas secundarias son aquellas que no tienen tanta importancia y que se empezaran una vez se vallan acabando las tareas principales.

A su vez, las tareas estaran numeradas segun importancia, dificultad y tiempo de trabajo con el fin de poder establecer un orden de prioridad para que su progreso haga que el proceso de construccion del proyecto sea relativamente optimo y de forma que su estructura no se vea afectada por software anterior.

Por ultimo, hay un sector no especializado que trata los temas de algoritmia sobre como handlear la forma de minimizar el total de pagos que hay realizar en forma de peticiones. Este algoritmo estara pensado en forma de algoritmo greedy al tener complejidad $O(n)$, pero en caso de que consigamos un algoritmo que no dependa de combinatorial algorithms o complejidades exponenciales para encontrar el minimo numero de pagos, tal vez aniadamos la opcion. Por ahora no sabemos como de rentable resultaria este algoritmo, ni tampoco sabemos ninguna upperbound del numero de movmientos que sea inferior a n aproximadamente. Esto se mostrara en la bibliografia final o en la documentacion sobre el software.

El apartado de algoritmia se dividira segun convenga, aunque se mostrara dentro de la documentacion quien ha hecho que tareas y como ha resuelto los problemas.

2.2 Error Handling

Para evitar errores que afecten al codigo a nivel global, hemos optado por utilizar github y hacer que cada miembro del grupo tenga su propia cuenta. Los errores, comentarios y todo se enviarian a traves de github excepto la informacion mas urgente que se enviaria a traves de un grupo de whatsapp o telegram/discord en si defecto.

2.3 Preparacion del temario

Para la preparacion del temario que queremos trabajar en el trabajo hemos optado por utilizar un conjunto de libros que consideramos basicos para las personas que no tienen mucha experiencia y utiles/importantes para las personas que tengan un cierto control en ciertos aspectos. Esta lista de libros se encontrara en el apendice de este documento categorizada.

3 PLANIFICACION

3.1 8 de octubre E1. Propuesta de proyecto y planificación

- Server
    1. Conseguir conectar un servidor SQL a el servidor.
    2. Hacer que el servidor lea archivos properties.
- Desktop
    1. Crear la ventana junto con sus componentes.
    2. Crear el metodo para enviar sockets.
- Android
    1. Leer un poco sobre Android
- Database
    1. Crear la estructura de la base de datos.
- Seguridad
    1. Haber leido un mínimo sobre RSA, Blowfish, MD5, SHA512, Cesar...
    2. Plantear los métodos de encriptación que se usarán para cada tarea dentro del servidor.
- IA
    1. Plantear que metodos usaremos en el proceso
    2. Plantear que tipo de bots queremos usar
- Miscelánea
    1. Entender como funciona la documentación del proyecto.
    2. Entender como funciona el sistema de comandos del servidor.
    3. Organizar el repositorio.
    4. Estudiar la posibilidad de cambiar la documentación a una wiki.

3.2 29 de octubre E2. Fase I del proyecto

- Server
    1. Estructurar bien el servidor.
    2. Todos los mensajes de los sockets deberán ir encriptados.
    3.
- Desktop
    1. Cliente funcional para enviar sockers
    2. Cliente visualmente no horrible
- Android
    1. Cliente funcional para enviar sockers
    2. Cliente visualmente no horrible
- Database
    1. La base de datos debe estar creada y con algunos datos de ejemplo
- Seguridad
    1. Las contraseñas de los usuarios deben estar encriptadas en algun tipo de hash.
    2. Los usuarios deben tener su clave privada para contactar con el servidor
- IA
    1. Tener un bot construido
- Data Mining
    1.
- Miscelánea
    1.

3.3 10 de diciembre E4. Fase II del proyecto

3.4 13 de diciembre E5. Informe de desarrollo

4 DIVISION DEL TRABAJO

La division del trabajo entre los grupos se hara de la siguiente forma:

Grupo 1	Grupo 2
Implementacion Desktop	Implementacion Server
Implementacion Android	IA
Proteccion	Data Mining
Database 2	Database 1

Los miembros de los grupos seran los siguientes:

Grupo 1	Grupo 2
Aritz	Erik
Leire	Lander
Gorka	Amaia

5 Implementacion de Server

La implementacion del Server sera una implementacion sencilla de un serversocket con la capacidad de extension de funcionalidades. Incluso tal vez no necesitar modificar el codigo del server per se para la ampliacion de su funcionalidad.

Los sockets del servidor tendrian una estructura tal que:

1. Conseguir acceder a la cuenta de un usuario.
2. Comprobar que el usurio existe y verificar que la contrasenia es correcta.
3. Verificar que no ha habido un abnormal comportation.
4. Devolverle un hash que representa su sesion actual en el servidor.
5. En caso de que quiera guardar la sesion, devolverle otro hash que represente su sesion y guardar en el servidor que esa sera su sesion activa.
6. Este hash debera ser modificado cada x tiempo.

Los sockets de funcionalidad estaran compuestos por un comando, luego el servidor devolvera un codigo hash de sesion activa y a continuacion el cliente enviara toda la informacion con el hash como cabecera.

Estos hash se enviaran a traves de la web mediante un sistema de criptografia implementado por nuestro equipo y, por otro lado, plantearemos un metodo mas seguro para mostrar la capacidad de implementar metodologias mas serias o profesionales.

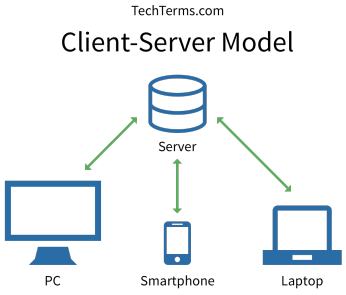

Figure 1: Modelo del Servicio.

El ServerHandler no debera ser modificado cada vez que se aniada un nuevo codigo de funcionalidad en el server. Para manejar esto, hemos pensado en hacer que el serverhandler funcione mediante un arbol Autobalanceado (AVL o Black Red por ejemplo) que se genera al inicio de la sesion del servidor y que despues, en cada peticion, el serverhandler simplemente busque cual de las hojas es y luego ejecute otra funcon que se encuentre en algun otro lado del codigo. Este arbol se generaria mediante una base de datos que contenga el conjunto de funcionalidades y la funcion asociada.

Los usuarios dentro del servidor deberían tener categorias de forma que un cliente no tenga acceso a el control de ciertos registros de la base de datos por ejemplo. Las categorias que se proponen son las siguientes:

Nombre	Permisos	Descripcion
Administrador	0	El administrador central de la aplicación tendría acceso a todos los comandos del server.
Programador	1	Los programadores tienen acceso a los comandos de edicion en el servidor, así como a algunas meta-funcionalidades. Pueden asignar más permisos (hasta el de programador) a otros usuarios.
Moderadores (?)		
Cliente	4	Los clientes deberían tener la capacidad de acceder a las acciones sobre el servidor que impliquen su informacion directa o informacion global publica. Asi mismo deben tener la capacidad de modificar cierta informacion suya.
Invitados		

5.1 Seguridad durante la transmision de los paquetes

Para asegurar la seguridad de los paquetes hemos establecido que se encontrarán encriptados por una key que tendrán tanto el servidor como el usuario. Para enviar esta key el servidor y el cliente se comunicaran de forma segura entre ellos.

Una persona que se registre por primera vez en el servidor(o se loguee) pasaría por el siguiente proceso:

1. El cliente crea un paquete con el usuario y contraseña, así como alguna información extra más.
2. Este paquete es encriptado por una key aleatoria de longitud aleatoria dentro de su dominio.
3. El cliente envia el paquete y el servidor lo recibe. El servidor encripta el paquete con una key aleatoria(la cual guarda de forma temporal) y lo envia de vuelta al cliente.
4. El cliente desencripta con su key el paquete, haciendo que su información de usuario solo esté encriptada ahora por la clave del servidor.
5. El cliente envia el paquete al servidor. Este lo desencripta con la clave que tenía guardada.

Durante todo el proceso el paquete se ha enviado encriptado, manteniendo así la seguridad y evitando posibles ataques de Man in the Middle. Por otro lado, este método requiere que la función de encriptación (o las funciones) formen un grupo conmutativo sobre la composición.

Una vez se ha verificado la validez de las credenciales, el servidor usaría el mismo proceso para enviarle a el usuario el hash que usarán como key para encriptar paquetes a partir de ahora.

6 Implementacion de Desktop

La aplicacion de escritorio sera puramente una aplicacion que sirva como cliente para el servidor y que sirva para enviar sockets y representar de forma visual todos los comandos que se podran enviar al servidor.

La implementacion del software de Desktop se basará en tres valores principales: comodidad, sencilles y completitud. Buscamos crear una aplicacion que satisfaga todas las necesidades de los usuarios, manteniendo a su vez la sencillez que supone un software minimalista y con el apoyo de IA.

7 Implementacion de Android

8 IA

El apartado de IA de DeusZum se basara puramente en el analisis de la conducta de los usuarios dentro de la aplicacion con el unico fin de garantizar su seguridad.

Las tareas que queremos hacer en el apartado de IA son las siguientes:

- Analisis de los movimientos economicos de los clientes.
- Analisis del tracking de las direcciones de las transacciones de los clientes.
- Analisis conductual del cliente dentro de la apliacion.
- Prediccion de los clientes dentro de la aplicacion.

8.1 Analisis de los movimientos

8.2 Bots de Automatización sobre el servidor

Para tener en control el conjunto del servidor será necesario automatizar grandes partes del proceso, entre ellas, todos los temas que tengan que ver con analizar las posibles cuentas falsas, inactivas, errores de seguridad sobre las bases de datos, etc. Para ello necesitaremos programar un conjunto de bots los cuales actuaran segun arboles de decision o mediante funciones optimizadas previamente.

9 DATA ANALYSIS

- Clustering
- Analisis de mercado

10 Clustering de los clientes

Una utilidad que queremos que nuestro servidor tenga es la capacidad de analizar a los clientes y dividirlos en categorias similares. Esto nos seria util a la hora de plantear posibles promociones y a que clientes les pueden interesar. De esta forma, no saturariamos con spam a todos los clientes sino que escogeriamos los correos a enviar.

Para el clustering podriamos usar K nearest neighbour o algun tipo de sistema automatico que organice de forma autonoma todos los elemntos a base de analizar su informacion.

11 Analisis de las Promociones

Otra opcion importante es el analisis del mercado que tenemos. Cuanto tiempo pasan los clientes en nuestra app etc..

12 SEGURIDAD

- Encriptacion de todas las conexiones.
- Capacidad de trackeo del historial de conexiones

12.1 Encriptacion del protocolo de envio de hash y verificacion de usuario

Propiedades de esta encriptacion:

1. Ha de ser conmnutativa bajo composición.
2. Ha de ser inyectiva.
3. No puede ser roto por analisis de frecuencia.

12.2 Encriptacion de los codigos de los sockets

A la hora de enviar la informacion al servidor, nos parece importante que no pueda exitir un sniffer que robe la informacion de nuestros clientes como puede ser el hash de la sesion o como puede ser el usuario y contrasenia. Para evitar esto, necesitaremos que nuestra informacion se envia encriptada con algun tipo de codigo decriptable asi como con un conjunto de informacion basura para evitar que a base de fuerza bruta en codigos cortos se pueda adivinar lo que ciertos codigos encriptados hagan.

Para solucionar este problema, el grupo de seguridad se encargara de estudiar el funcionamiento de ciertos tipos de encriptacion inyectiva o biyectiva y analizaran algunos metodos para aumentar la seguridad del algoritmo de encriptacion que deberan crear.

Sea A el alfabeto, $f : A^n \to A^w$ la funcion de encriptacion con $n \in \mathbb{N}$.Las condiciones que establecemos ahora sobre el algoritmo de enriptacion seran las siguientes:

1. f sera inyectiva.
2. $\mathbb{P}(|f(x)| = |x|) \to 0$ cuando $n \to \infty$. siendo $x \in A^n$.
3. f no pueda ser roto por analisis de frecuencia.

12.3 Trackero de los pagos

13 DATABASES

Las tareas dentro de este subapartado se dividiran en 2, la de creacion y organizacion inicial de las bases de datos y la segunda, la de optimizacion de las bases de datos y su respectivo ajuste de seguridad con el fin de evitar ciertos tipos de ataque como las SQL Inyection o las code execution.

- Optimizacion de las bases de datos

14 CONCLUSIÓN

15 LIBRERIAS

1. **DeepLearning4J**: Usaremos esta libreria para la implementacion de redes neuronales y ejecutarla. A su vez, esta libreria permite el uso de tensores para la representacion de datos, lo cual nos puede ser util.

16 LIBROS

16.1 Java

16.2 Android

16.3 Server

16.4 IA

16.5 Data Analisys

16.6 Seguridad

1.

References

[1] Autor1, Autor2. (2002) *Nombre del articulo*, Lugar de publicación