El primer paso para el desarrollo de la aplicación distribuida será identificar los agentes involucrados y los mensajes que estos deben intercambiar.

En cuanto a los agentes, estos se pueden diferencian por:

* Transportista + dispositivo con aplicación embebida.
* Servidor que recibe las peticiones de los dispositivos.

De forma que la aplicación estará dividida en dos partes: un servidor que recibe peticiones, y una serie de clientes (transportistas) que envía peticiones a dicho servidor a través de una aplicación distribuida embebida en su dispositivo móvil.

Por tanto, los mensajes se pueden clasificar en:

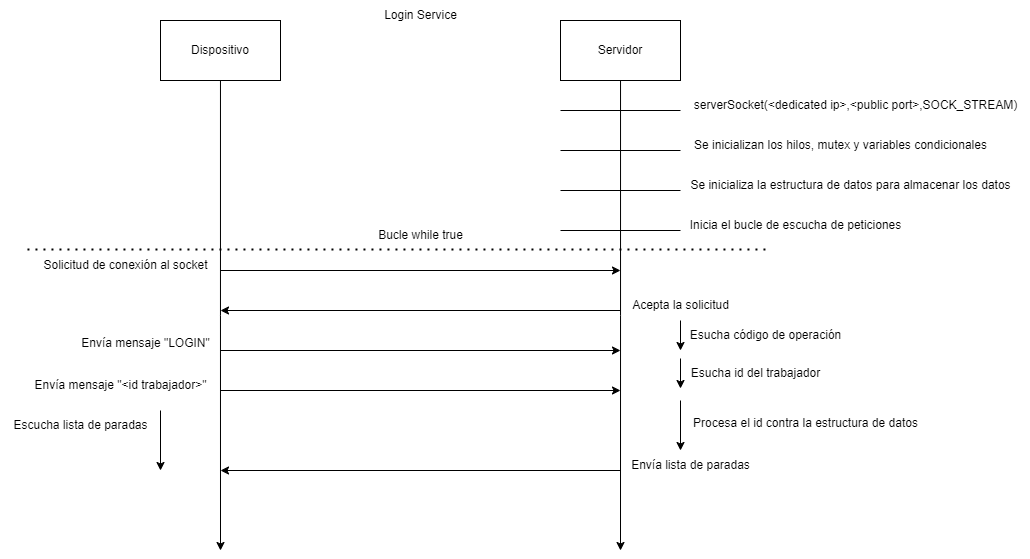
* Enviados desde el transportista hacia el servidor:
  + Transportista inicia la jornada
  + Paquete/s entregado (id paquete, id transportista, firma cliente, dni cliente)
  + Transportista termina jornada (lista de paquetes no entregados)
* O enviados desde el servidor hacia el transportista:
  + Lista de paquetes a entregar
  + Mensajes en respuesta a paquete marcado como entregado en el sistema.

El protocolo de mensajes elegido será TCP sobre IP, ya que priorizaremos la fiabilidad en la entrega de los mensajes. Esto es crucial para evitar que los paquetes no se marquen correctamente en el sistema, lo que podría generar fallas de seguridad y responsabilidades que podrían recaer sobre el trabajador cuyo mensaje no fue recibido en el servidor su hubiéramos elegido UDP.

La función de cada agente en la aplicación cliente-servidor que vamos a definir será:  
El cliente (transportista) es el elemento activo en la comunicación que inicia las transacciones, mientras que el servidor es el componente pasivo que espera a recibir las peticiones para responder en base a los datos que conoce y recibe.

El servidor ofrecerá tres servicios:

* LOGIN <id trabajador>
* DELIVERED [<id producto i>, <id producto i+1>, <id producto i+2>, …] <id transportista> <dni cliente> <firma cliente>
* LOGOUT <id trabajador> [<id producto i>, <id producto i+1>, …]

Cada uno hace la función descrita en las tres funcionalidades necesarias para la aplicación. La comunicación entre dispositivo y servidor utilizando socket TCP será:

En el servicio login se ha incluido el inicio de la ejecución del servidor, en esta parte el servidor crea un servidor socket en una dirección IP y puerto seleccionados, se inicializan tanto los hilos que se utilizará para atender a cada cliente como los mutex y variables condicionales para proteger las secciones críticas del código, por último se inicia la estructura de datos seleccionada para almacenar los datos sobre clientes, paquetes y trabajadores, esta podría ser, por ejemplo, árboles binarios de búsqueda, para optimizar la búsqueda de clientes/trabajadores/paquetes por una id única. A continuación, empieza el bucle de escucha de peticiones.

Para comenzar una transacción, el cliente solicitará conectarse al socket del servidor.

Una vez conectado, enviará una cadena de caracteres con el código de la operación que desea realizar, en este caso, la cadena LOGIN.

El servidor procesará la cadena e internamente procesará el socket que se ha creado para atender a este nuevo cliente en un hilo nuevo, haciendo una copia segura del descriptor de dicho socket utilizando mutex y variables condicionales.

Una vez asegurada la petición LOGIN del cliente en un hilo nuevo, el servidor continuará escuchando peticiones mientras el hilo procesa el resto de los argumentos.

Al ser una operación LOGIN, esperará a recibir del cliente la id del trabajador que desea iniciar su jornada, una vez recibido, realizará una serie de comprobaciones internas para asegurar la consistencia del sistema y la seguridad de los trabajadores, como comprobar si dicho id ya tenía una sesión iniciada, o si el id no existiese en el almacén de datos. Este tipo de incidencias se reportarían directamente al dispositivo cliente o a un servicio externo encargado de procesarlas.

Una vez realizadas dichas comprobaciones, se utilizaría algún tipo de algoritmo de optimización para obtener la mejor combinación de paquetes para el dispositivo teniendo en cuenta su localización, las horas que dura su jornada y el destino y peso de los paquetes y la lista resultante se mandaría a través del socket hacia el dispositivo y se cerraría el socket de ambos lados y se cerraría el hilo encargado de la petición.

Gráfico de cajas y bigotes

Descripción generada automáticamente con confianza media

En la funcionalidad de entrega de paquete, lo primero que hace el cliente es solicitar conectarse al socket del servidor. Una vez conectado envía la cadena de caracteres con el código de operación DELIVERED.

El servidor procesa la cadena, procesa el socket y hace una copia segura del descriptor del socket utilizando mutex y variables condicionales. una vez se a asegurado la petición en un hilo nuevo, el servidor continúa escuchando peticiones mientras el hilo procesa el resto de argumentos

La operación DELIVERED el cliente envía al servidor la lista de paquetes entregados, el id del transportista, el dni del cliente y la firma del cliente. Una vez recibido inicia comprobaciones para asegurar la consistencia del sistema.

Imagen que contiene Gráfico de cajas y bigotes

Descripción generada automáticamente

El cliente solicita conectarse al socket del servidor, una vez conectado envía una cadena de caracteres con el código de operación LOGOUT.

El servidor procesa dicha cadena de caracteres y procesa el socket creado para atender al cliente.

Al ser la operación LOGOUT el servidor espera a recibir el id del transportista y la lista de paquetes no entregados del dispositivo. Una vez le ha llegado toda la información la procesa. Una vez procesado y terminada la comunicación de cierran los sockets y los hilos,