Eine inoffizielle und kurze Einführung der Software wird hier gegeben. Das Ziel ist, die zugrundlegenden Ideen der Software sowie einige technischen Erklärungen zu beschreiben.

Wie Abbildung 1 gezeigt, ist das System in Server und Client verteilt und kann in vier Komponenten, ***locationParser***, ***server, client*** und ***springContext*** aufgeteilt werden. Die Beschreibungen der jeweiligen Komponenten sind wie folgt:

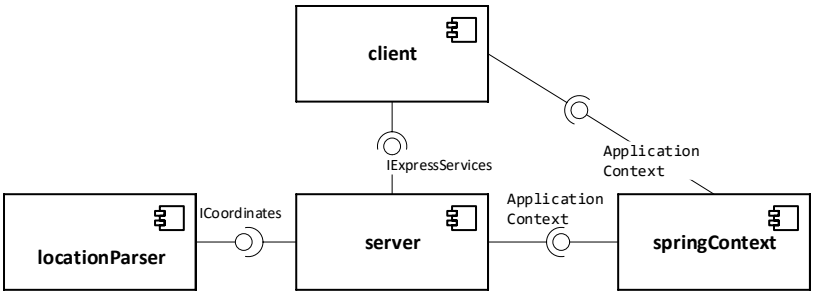

Abbildung 1: Komponentendiagramm

Das Interface ***ICoordinates*** der ***locationParser*** bietet eine geographische Koordinate mit einem gegebenen Lokationsnamen an. Beispielweise wird mit der Eingabe „*Metrostrasse 12, 40235 Düsseldorf*“ die Koordinaten *51.234806, 6.825593* geliefert. Auf Grund der zeitlichen Einschränkung verwendet die Software keine Google-Dienstleistungen. Stattdessen berechnet die korrekte Implementierung ***CoordinatesImp*** alle entsprechenden Koordinaten vorher.

Das System ist ein verteiltes System mit Hilfe von *RMI*. Alle Stores und Fleets werden zunächst in ***Server*** instanziiert. Da ihre Standorte im Vergleich mit *Customer* stabil sind, werden gleichzeitig ihre Distanzen berechnet und in der Klasse ***Fleet*** gespeichert.

Der ***Client*** simuliert die Route-Anfrage der Kunden. Das Interface ***IExpressServices*** liefert ein Objekt von Klasse ***Route***. Die optimale Route, Distanz sowie Zeit werden in dieser Klasse verkapselt. Der Algorithmus findet sich am Ende.

Da die Fleete, Stores und Customers zur Laufzeit dynamisch erstellt und gelöscht werden können, verwendet das System eine dynamische Arte und Weise, solche Objekte zu erstellen, indem das ***Spring-Framework*** im System eingeführt wird.

Der Algorithmus zur Berechnung der optimalen Route wird realisiert durch die Methode:
`public Route getRoute(String request)` in `public class ExpressMap`

Die Erklärung ist wie folgt:

1. Da die Standorte aus *real*-Seite (*fleets and stores*) im Vergleich mit den der Kunden stabil sind, werden die Distanzen zwischen Fleete und Lager während der Initialisierung vorberechnet und in den jeweiligen Fleet-Objekten gespeichert.
2. Für eine bestimmte Kunde-Anfrage muss nur die Distanz zwischen ihm und allen Lagern berechnen. Die gemeinsame Distanz kann mit Hilfe von Aufruf der in Fleet-Objekten gespeicherten Distanz-Caches berechnet werden.

Abbildung 2 zeigt ein Mini-Beispiel der Route-Berechnung. Bei der Initialisierungsphase werden die Variablen „DistanzToS*x*“ in F1 und F2 gespeichert. C1 fragt jetzt eine Route an. Die Distanzen zu allen Stores (S1 und S2) werden berechnet. Durch Vergleich findet C1 die optimale Route: F2-> S2->C1.

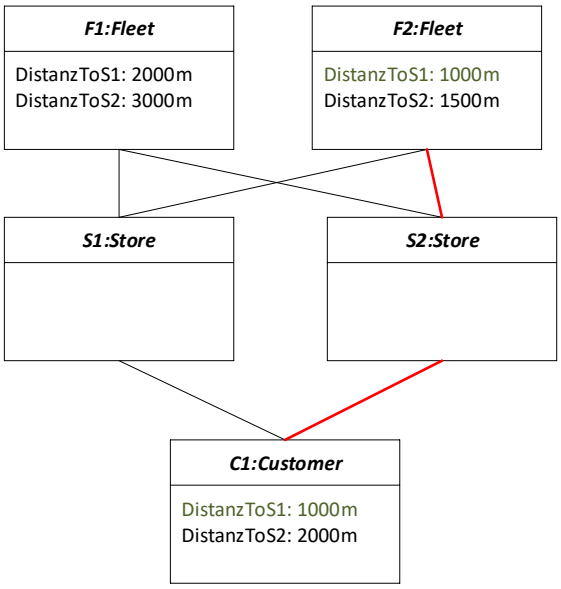

Abbildung 2: Ein Beispiel zur Berechnung der Route