Taakverdeling

**Sprint 1**

De doelstelling voor sprint 1 was de back end van de verkeersapplicatie realiseren.  
Hierbij was het de bedoeling de verkeer data op te halen bij de providers en deze in een databank op te slaan.  
Deze opdracht hebben we opgesplitst in 2 grote hoofdtaken.

Jarno en Simon hebben de eerste hoofdtaak, de data ophalen bij providers, voor hen rekening genomen. Dit gedeelte hield in dat verschillende AProviderConnector-klassen (één voor elke provider) de verkeer data van de verschillende trajecten moet ophalen en deze omvormen naar objecten die kunnen opgeslagen worden in de databank.  
Het ophalen van de data bij een provider gebeurt via de API die deze provider aanbiedt, waardoor de manier van data ophalen bij alle providers verschillend was.   
De klassen die zorgen voor de connectie met de providers Waze en Here heeft Simon geïmplementeerd, terwijl Jarno deze voor Google en Coyote heeft gemaakt. Omdat Perl in de applicatie moest verwerkt worden, is de coyote connector op deze manier gerealiseerd.  
Uiteindelijk is de TomTom connector er nog bij gekomen en hier heeft Robin voor gekeken.

Robin en Piet waren verantwoordelijk voor de 2e hoofdtaak, de opgehaalde data opslaan in een databank. De RDBMS van de databank is MySQL maar in sprint 1 was dit nog MariaDB, daarom draagt de DAO (Data Access Object) van de applicatie nog steeds de naam MariaDbConnector. Deze DAO zorgt dat Jarno en Simon de opgehaalde data via eenvoudige insert-commando’s in de databank konden opslaan. De andere taak van de DAO is een interface aanbieden van de databank.  
Robin had al heel wat ervaring met Linux en heeft de databank aangemaakt en gezorgd voor de nodige configuratie ervan (accounts ervoor aanmaken, tabellen aanmaken, …).

De klasse die het project runt en de klasse die instaat voor het constant pollen (Pollthread in sprint 1 en 2) van de data is ook door Robin ontworpen. Dit pollen zorgt dat om de 5 minuten deze providerconnectors worden aangewend om de data op te slaan. Een kleine taak die erbij kwam was het loggen van de fouten die voorkomen in deze verkeer-applicatie, deze was origineel ontworpen door Piet maar Robin had hier een beter alternatief voor gevonden door bibliotheken te gebruiken.

**Sprint 2**

In sprint 2 was het de bedoeling om deze opgehaalde data uit sprint 1 te visualiseren en beschikbaar te stellen voor gebruikers. De opdracht werd opnieuw ingedeeld in 2 grote taken.  
Enerzijds de data aanbieden aan gebruikers en derden via een REST API en anderzijds een dashboard ontwerpen voor de klant op basis van deze data.

Robin en Piet hebben de 1e hoofdtaak bij sprint 2, de data aanbieden via een REST API, verwezenlijkt. Hierbij moest op basis van JAX-RS jersey een REST API tot stand gebracht worden, waarbij de data aangeboden wordt in de vorm van URL’s. Robin deed grotendeels het ontwerp van de query’s, waarbij hij ook verbeteringen aanbracht aan de databank om deze query’s efficiënter te maken.  
Vervolgens heeft hij de meeste scripts ontworpen voor bijvoorbeeld dumps te halen van de productieomgeving.  
Piet heeft voor de methodes gezorgd die de requests aan de REST API afhandelen, Robin heeft deze dan helpen optimaliseren voor een betere output naar de gebruiker.

Jarno en Simon hebben zich over de 2e hoofdtaak ontfermd, het visualiseren van de data voor de klant via een eenvoudig te gebruiken dashboard. Het linkse paneel van het dashboard is vooral ontworpen door Simon maar Jarno heeft ook aan het visuele aspect ervan zijn bedrage geleverd. Dit paneel staat in voor de voorstelling van de data door verschillende functies aan te bieden (data over bepaald interval, live data, …) en deze te verwerken in grafieken.   
Jarno heeft zich naast dit paneel vooral toegespitst op het kaart-gedeelte. Hierbij heeft hij de waypoints op de kaart aangeduid en heeft hij het ook mogelijk gemaakt meer informatie te zien over routes via pop-ups.   
Simon heeft ook enkele REST API methodes zelf ontwikkeld om de dashboard-functies te realiseren.

**Sprint 3**

Dit was een korte sprint waardoor de backlogs beperkt zijn gebleven. Het debuggen van de applicatie en enkele verbeteringen aan het dashboard zijn functionaliteit stonden voorop.

Robin heeft de taak van de databank debuggen op zich genomen. Enkele bugs waren slechte performantie door een overmatig gebruik van thread’s (er werd 1 thread per route gebruikt) en problemen met API-KEYS van providers waarbij constant de request-limiet werd overschreden.  
Hiernaast heeft Robin ook gezorgd dat de globale performantie van het dashboard verbeterd werd door de data van de databank te crunchen. Dit wil zeggen dat de data nu geaggregeerd wordt per 5 minuten wat resulteert in minder data, dus rappere query’s.

Jarno heeft de kaart gedebugd. Er zaten nog enkele fouten in de waypoints van de kaart.  
Coyote zorgde ook voor problemen omdat ze 34 in plaats van 30 trajecten hebben.  
Ten tweede heeft hij ervoor gezorgd dat de dashboard instelling (kaartcoördinaten, opgeslagen intervallen, …) opgeslagen worden in de URL, waardoor delen van de instellingen vereenvoudigd.

Simon heeft de dag-functie toegevoegd aan het dashboard, zo kan je data ophalen voor een bepaalde dag en hoef je dit niet in de interval-functie op te vragen. Vervolgens heb je nu ook de mogelijkheid te kiezen of je de data die voor het dashboard wordt gebruikt niet iedere 5 minuten te vernieuwen. Hij heeft ook de javascript wat meer object-georiënteerd zodat je nu ook de functies op deze objecten moet uitvoeren.

Piet heeft enkele verbeteringen aan de REST API aangebracht; de REST API op een eenvoudige manier beveiligd met API-KEYS (deze kan uitgebreid worden naar een betere methode) en een goede communicatie met de gebruikers van de API indien er fouten zijn opgetreden.

Uiteindelijk heeft iedereen z’n code ook gedocumenteerd en deze meer leesbaar gemaakt.