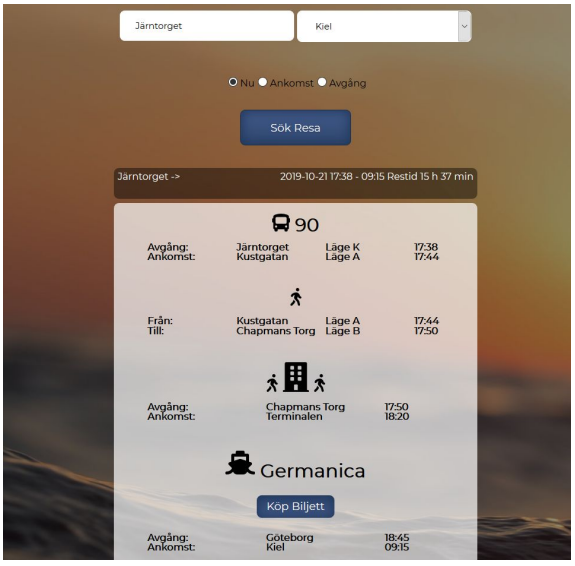
Grupp Jutlandica

Mattias Sikvall Källström, Daniel Danielsson, Eric Rylander, Carl Seeman, Johan Ericsson

2019-10-29



Sammanfattning

Gruppen har skapat en möjlighet för användare att söka efter en resa i realtid från en hållplats i Göteborg till en slutdestination i Danmark eller Tyskland. Tjänsten riktar sig till personer som vill resa kollektivt till danmark eller tyskland från göteborgsområdet. Utvecklingen har skett efter agila arbetsmetoder enligt kursens syfte och rapporten redogör för arbetet och utmaningarna som uppstått.

Förord

Denna rapport handlar om projektet i kursen “Agile software development”. Den går igenom arbetssätt, problem vi stött på och lösningar som framkommit gällande det agila arbetssättet, git, utvecklingsmiljöer och själva mjukvaru-utvecklandet.

Innehållsförteckning

Inledning

Syfte

Syftet med detta projekt har i första hand varit att skaffa oss erfarenheter av agil utveckling men även att utveckla en produkt som kan skapa värde för färje och linjetrafiken, antingen direkt genom att vara en färdig produkt, eller genom att utforska möjligheter och bereda väg för framtida utvecklare.

Metod

Projektet har genomförts med hjälp av agila metoder. Varje vecka har gruppen träffats, utvärderat user-stories, tasks, velocity och definition of done. Gruppen har träffats två till tre gånger i veckan där de gemensamt har jobbat för att lösa problem och utöver det jobbat individuellt där de har haft kontakt via messenger. För att effektivt kunna jobba agilt har vi haft stor användning av vårt git-repo på gitHub och vår trello-board.

För att genomföra projektet valdes webbutvecklings ramverket Google Web Toolkit (GWT). Detta val gjordes då det fanns en grundkunskap för GWT i gruppen och bedömningen gjordes att GWT skulle leda till att gruppen snabbare kunde skapa ett värdefullt resultat. GWT använder sig av java, CSS, HTML5 och XML för att producera webbapplikationer. Eftersom gruppen också kände sig bekväm med att att använda java var GWT ett bra alternativ då det använder java för både server-del och klient-del. Att det var möjligt att använda verktyg såsom GwtMobile för att porta applikationer till Android och iOS var också en starkt bidragande orsak till valet av GWT.

Gruppen har framgångsrikt hämtat xml-data från västtrafiks API för att kunna få tillgång till kollektivtrafiks data. Valet av att använda oss enbart av västtrafiks API begränsade vår kollektivtrafiks data till enbart västra götalands, men gruppen såg detta som en nödvändig begränsning av sitt scope. För att få data som representerade färje avgångarna ansågs simulering av dummy-data som den enda gångbara metoden. Dummy-datan ska existera i NeTEx format och förseningar som skapas med hjälp av SIRI och vara så verklighetstrogen som möjligt. För att visuellt kunna visa en resenärs färdväg valdes openstreetmap. Detta dels för att openstreetmap hade den funktionalitet som gruppen ansåg sig vara i behov av, men också för att det är gratis och lättillgängligt.

För att demonstrera applikationen använde sig gruppen av en raspberry-pi. Denna dator fick agera värd för en Tomcat-server som i sin tur huserade applikationen och gjorde den tillgänglig på internet.

Resultat

Git

Under projektets gång har Git används som verktyg för versionskontroll och gruppen har försökt att använda GitFlow strukturen där man utvecklar varje ny funktion i en ny “branch” utifrån en stor “develop-branch”. Detta för att underlätta merges och alltid har en fungerande applikation i master. Gruppen var familjär med git sedan innan, trots detta förekom det länge problem med att få det att fungera felfritt.

Första vecka fanns ingen kontroll på Git-strukturen och det hade lagts upp mycket filer varav många var helt onödiga. En task hade skapats i trello för att sätta upp en Git-struktur men den blev inte tillräckligt högt prioriterad. Git-strukturen blev kritiserad under handledningen och veckan därefter valdes GitFlow som den struktur Git-repot skulle följa. Detta bidrog till att strukturen och flödet blev bättre men problem kring merges existerade fortfarande.

Andra veckan rensades hela Git, och en ny “master-branch” laddades upp. Denna med ett nytt försök till en .gitignore-fil för att hjälpa med alla onödiga filer från GWT. Detta hjälpte ett tag men snabbt föll det igen och merge problem återuppstod.

Detta resulterade i ännu ett försök den tredje veckan efter, där en hel dag ägnades åt att försöka få Git merges att fungera. Där Git rensades upp för att sedan göra ett nytt försök med .gitignore. Detta slutade exakt som försöket föregående vecka, med en kort fungerade period innan problem kom tillbaka.

Tillslut under den fjärde och nästsista veckan lyckes vi med ett flertal merges utan konflikter, detta efter ytterligare ett försök med en .gitignore. Där vi hittade guider till hur en GWT .gitignore skulle skrivas. Efter att Git rensades en sista gång och den ny .gitignore lades till så fungerade merge under de sista två veckorna.

GWT

Stora delar av projektet utvecklades i Java. Det som sedan består i den färdiga produkten är det som utvecklats på server-sidan medans klient-delen översätts till Javascript.
God kunskap fanns inom gruppen av Java som utvecklings-språk sedan tidigare och den del av utvecklingen som innefattade logik skriven i Java fortskred i god takt.

Till en början utvecklades projektet mestadels i klientsidan, då denna skall översättas till JavaScript så fanns det många bibliotek som ej stöddes. Detta ledde till svårigheter med att implementera funktionalitet som var beroende av vissa bibliotek som ej stöddes.
Utvecklingen fortsatte en till två veckor på detta sätt innan det framkom att utveckling kunde ske i server-delen med godtyckliga bibliotek (då denna ej översätts till javascript). Detta ihop med information om hur kopplingen mellan server och klient kunde göras bidrog till att utvecklingen av "backend" gick framåt fort .

HTML/CSS

Modellerna som visas i applikationen är skrivna i Java och HTML. För att kunna skriva html kod i en javaklass så använder vi oss av en stringbuilder och lägger till HTML kod som en sträng för varje HTML tagg som behövs.

```
38              StringBuilder sb = new StringBuilder();
39
40              sb.append("<p class=\"buss\">Ankomst: ");
41              sb.append(trip.getEnd_station() + ": ");
42              sb.append(trip.getArrival_time());
43              sb.append("</p>");
44
```

Detta försvårade strukturen av html koden men det fungerade bara vi var noggranna.
CSS används för att styla sidan och det har fungerat bra även om vissa java objekt var svåra att styla med CSS. Koden kunde skrivas i en separat fil. Om det hade funnits lite mera tid så hade gruppen lagt lite mera fokus på att göra sidan responsiv och förbättrat utseendet/tydligheten ytterligare.

XML, SIRI och NeTEx

Visionen för applikation var att den skulle kunna läsa in SIRI-data för att simulera färjorna, detta var något som började jobba med tidigt. Mycket tid spenderades i början för att få kunskap om hur XML fungerade och hur det det interagera med Java. Först skapades ett program för att testa hur man läser XML-filer och skriver ut data i ett Java-program. Sedan när detta skulle implementeras till GWT insågs det att många av standard biblioteken i Java inte stöds i GWT. Detta gjorde att gruppen var tvungna att lära oss ett nytt sätt att läsa XML i GWT, sen insågs det att läsa in filer som GWT-klient inte var speciellt enkelt. Utan det ägnades mycket tid till att lista ut hur en XML-fil skulle läsas in till klienten. I slutet av vecka två insåg gruppen att GWT hade en klient och server och att tanken som gruppen arbetat efter var fel , istället för att läsa XML i klienten valdes det att göra detta på server där standard biblioteken fanns. Detta underlättade rätt mycket och utveckling därefter fick rätt fort framåt. Tills gruppen under sista veckan pratade med Eddie från RealTimesFerries, då han förklarade att SIRI enbart används för att skicka realtidsdata och NeTEx för de planerade avgångarna. Något gruppen hade helt missförstått, vilket resulterade i att den XML-hanterare som skrivits nästintill blev meningslös. Men den skulle kunna användas i framtiden när man implementerar support för SIRI och NeTEx.

Diskussion

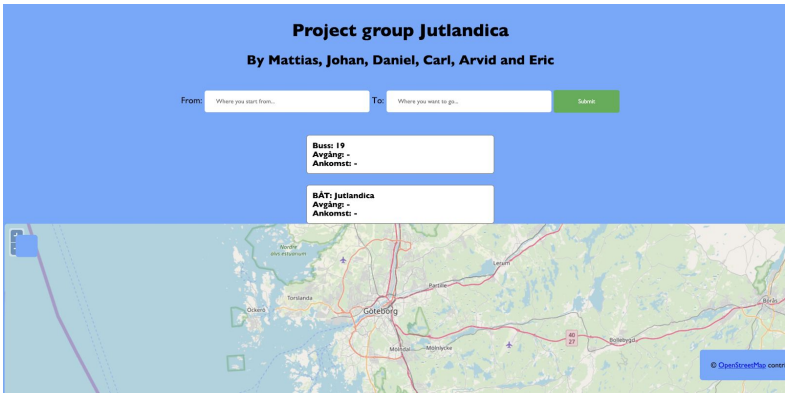
Är vi nöjda med resultatet?

Vi är överlag nöjda med vår produkt. Vi känner att vi har skapat något vi står för och som kan bli värdefull för framtida utveckling, såväl för andra som för oss själva.

Är resultatet det vi förväntade oss vid start?

Vi hade en vision vid uppstart av projektet om att göra en webb applikation som skulle kombinera västtrafik och stena-lines linjetrafik och visa detta som en resa. Detta har vi uppnått.
Vi hade även visioner om att använda “open-street map” och porta vår applikation till iOS och android samt bygga ut sökfunktionerna till att inkludera danmarks kollektivtrafik.
Dessa visioner har vi ej uppnått. Det är inte något som vi är missnöjda över (mer än att det hade varit kul att få till det) då vi redan från start var medvetna om att det skulle bli besvärligt att uppnå alla våra visioner.

Mockup:



Idag:

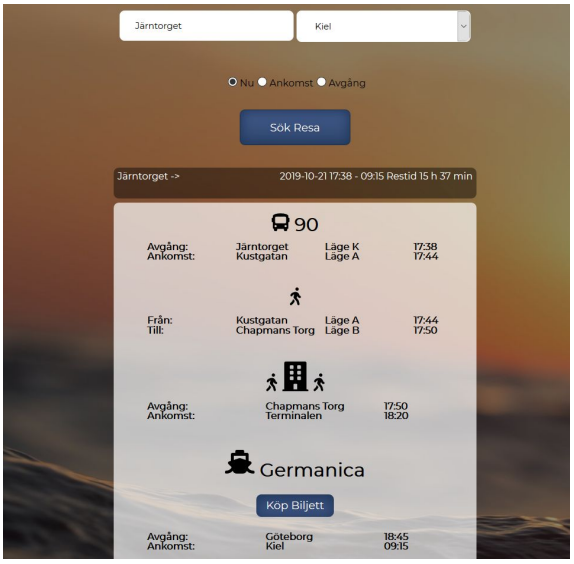


fig1.

Vår slutgiltiga produkt skiljer sig i vissa avseende från vår mockup från första veckan. Båda är reseplanerare där det går att söka efter resor vars destination och utgångspunkt ges av användaren. I dagens version har vi inte implementerat någon karttjänst. Dock finns det möjlighet att söka på ankomst eller avgångs samt så har dagens version ett mer estetiskt tilltalande utseende.

Vad har vi lärt oss?

Detta projekt var vårt första möte med det agila arbetssättet. Vi har lärt oss grunderna i hur man arbetar agilt med scrum. Denna erfarenhet ser vi alla som berikande då vi fått klart för oss att det agila arbetssättet är vanligt förekommande på arbetsmarknaden.
Vi har även lärt oss hur man arbetar med GWT, detta kan vara värdefullt då det är en snabb väg att få fram både klient- och server-del till en webbapplikation. Vi har även fått reda på att liknande utvecklingsmiljöer tillkommit t.ex vaadin, vilket gör att vi kanske kommer stöta på liknande sätt att arbeta i framtiden.

Hur har det agila arbetssättet fungerat?

Det har varit många problem genom projektets gång vilket har fått det agila sättet till utveckling att visa sin kraft. Det är framförallt samordningen av att göra ett större projekt med flera inblandade parter som ska lösa en outforskad uppgift som ställde till det.
Innan vi började denna kurs hade vi inte någon ide till ett projekt så allt är skapat inom denna period. En stor uppgift för fem personer att ta sig ann och till vår hjälp använde vi oss av agil mjukvaruutveckling. Uppgiften är en lysande demonstration för denna agila metod styrka. Det hade varit mycket svårt att först planera för hur projektet skulle se ut och sedan nå samma resultat som vi har nått nu.
Inför varje sprint har vi i möten styrt vilka moment vi ska lägga högst vikt på, anledning att så snabbt som möjligt få en fungerande produkt. Efter någon vecka hade vi nått så pass långt att vi kunde se framför oss vad vi skapat och efter det blev våra möten mycket mer tydliga då vi kunde bestämma mer konkret vilka funktioner vi vill implementera. Med tiden blev vi dessutom bättre på att bestämma vår velocity samt hur lång tid de olika uppgifterna kunde ta att göra.
I början hade vi tagit fram en ganska diffus “definition of done” och antecknade inte några acceptance criteria på de uppgifter vi satte upp i vår backlog. Det gjorde det svårt att veta hur långt varje task var kommen utan det blev var och ens ansvar att avgöra när man var klar. Det blev då svårt att hjälpa till på de uppgifter som inte var helt klara när sprintens slut närmade sig.

Vi tog sedan tag i att alla uppgifter som läggs till i sprint backloggen ska ha tydliga acceptance criteria och det ledde till att det både blev enklare att bestämma tid på den uppgiften och att alla i gruppen var med på när den kan anses vara klar.
Vi tappade en gruppmedlem under projektets gång, vi har även projektmedlemmar med barn som har missat tillfällen på grund av VAB. Detta har påverkat vår produktivitet negativt, men alla har haft bra kommunikation med gruppen och vi har sänkte vår velocity efter behov.

Hur vi skulle gjort om vi gjorde projektet igen?

Valet av ramverket GWT baserades på att vi var överens om att vi ville göra en webbapplikation men fortfarande programmera vår backend i java. Vi hade även grundläggande kunskaper inom webbutveckling och GWT inom gruppen innan projektets start. På så sätt var tanken att vi skulle komma igång i vårt arbete och skapa värde för produktägaren snabbare. Hade vi istället börjat med helt okända ramverk och tekniker för att exempelvis skapa en mobilapplikation så tror vi att det hade tagit en längre tid att skapa värde. Utvecklingen i GWT har däremot varit allt annat än problemfri, men vi hade troligtvis inte minskat mängden problem i och med val av andra ramverk/tekniker. Under presentationen av projektet rekommenderade Eddie på RISE att många av de problem vi haft med GWT skulle kunnat ha löst om vi istället använde Vaadin som bygger på GWT.

Hur vi arbetade som grupp?

Vi har programmerat som team, parprogrammerat och utvecklat projektet enskilt om vartannat. Vi har även haft möten där endast planering och reflektion skett.
Till en start var allt nytt med scrumboard och det agila sättet att jobba, vilket gjorde att vi var tvungna att lägga mycket tid på att skriva user stories, tasks, lägga upp sprinten för veckan mm. Med tiden blev vi alltmer effektiva på att hålla dessa möten korta och sakliga.

Hur uppfyllde vi det socialkontraktet?
Våra ambitioner var att jobba runt 20 timmar per person och vecka. Då timmarna ej räknades exakt var detta mer en målsättning som vi uppfyllde vissa veckor men ej alla. Överlag har alla jobbat på bra och tillfört värde i form av kod, tankat och ideér till projektet.

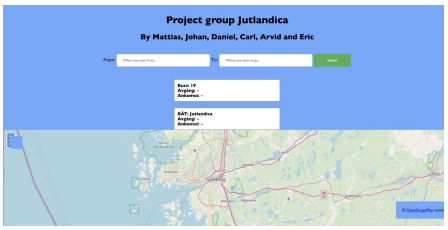
Vi hade som målsättning att klara kursen och göra en produkt som vi kunde stå för och använda i vår portfolio. Även om det fortfarande finns utvecklingspotential är vi alla nöjda med vår produkt.
Möten, beslut och ideér har även detta gått i linje med vårt sociala kontrakt.
Tekniska resurser likaså och vi skrev även om kommunikation på distans att vi skulle använda messenger i första hand och telefoner i andra hand främst kontorstider. Detta har fungerat bra och alla har varit tillgängliga och svarat inom acceptabel tid.
Vår scrum master har haft kontakt med projektägare och andra grupper för att undersöka samarbetsmöjligheter och även tagit ansvar för att researcha gitflow strukturen.

Hur skulle vi gjort om vi gjorde projektet igen

Vi är alla överens om vikten i att få git och utvecklingsmiljöer att fungera så fort som möjligt. Detta var något som drog ut på tiden och vid ett andra försök skulle detta hamna som absolut högsta prioritet.
Under projektets gång har vi fått fram info vid vissa tillfällen som vi önskade att vi haft tidigare (hur västtrafiks api fungerar, server/klient - del i gwt, hur NeTex och SIRI är tänkt att användas, gitignore), det är ju alltid enkelt att vara efterklok, men aningens mer research skulle kunna förhindrat missförstånd som på sikt skulle ha sparat mycket tid.

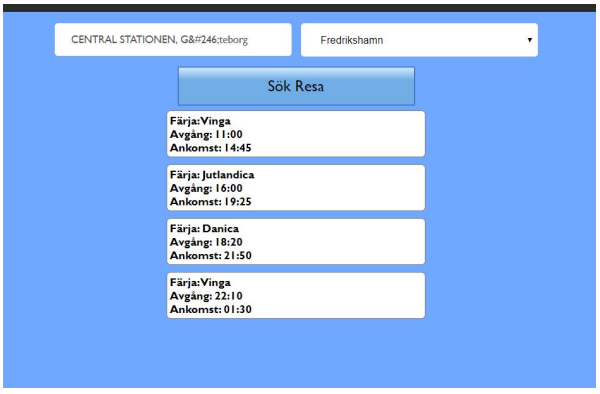
Bilagor

Bilaga 1



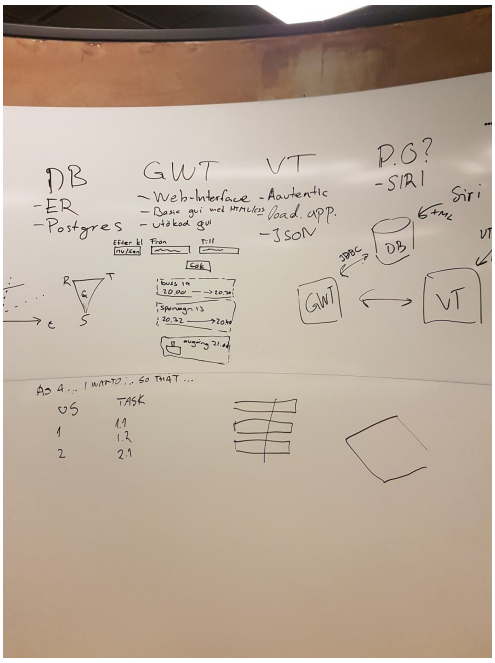
Mockup

Bilaga 2

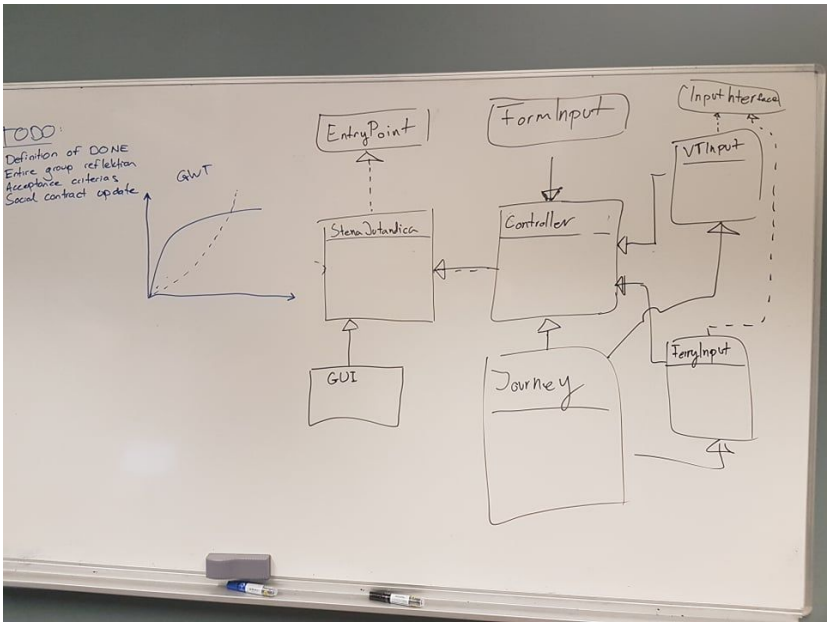


Bilaga 3

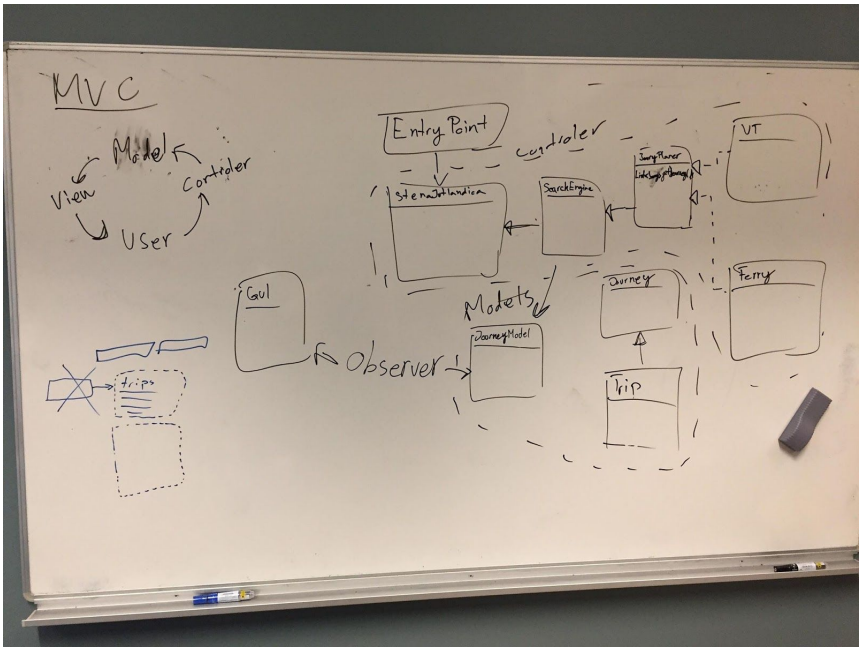
Första fungerande testet


DB
GWT
VT
P.G?
-ER
-Postgres
- Web-Interface - Autentic
- Basic gui med HTML/CSS
- Utökad gui
-JSON
Siri
DB
GWT
VT
As a ... I WANT TO ... SO THAT ...
US
TASK
1
1.1
1.2
2
2.1

Bilaga 4

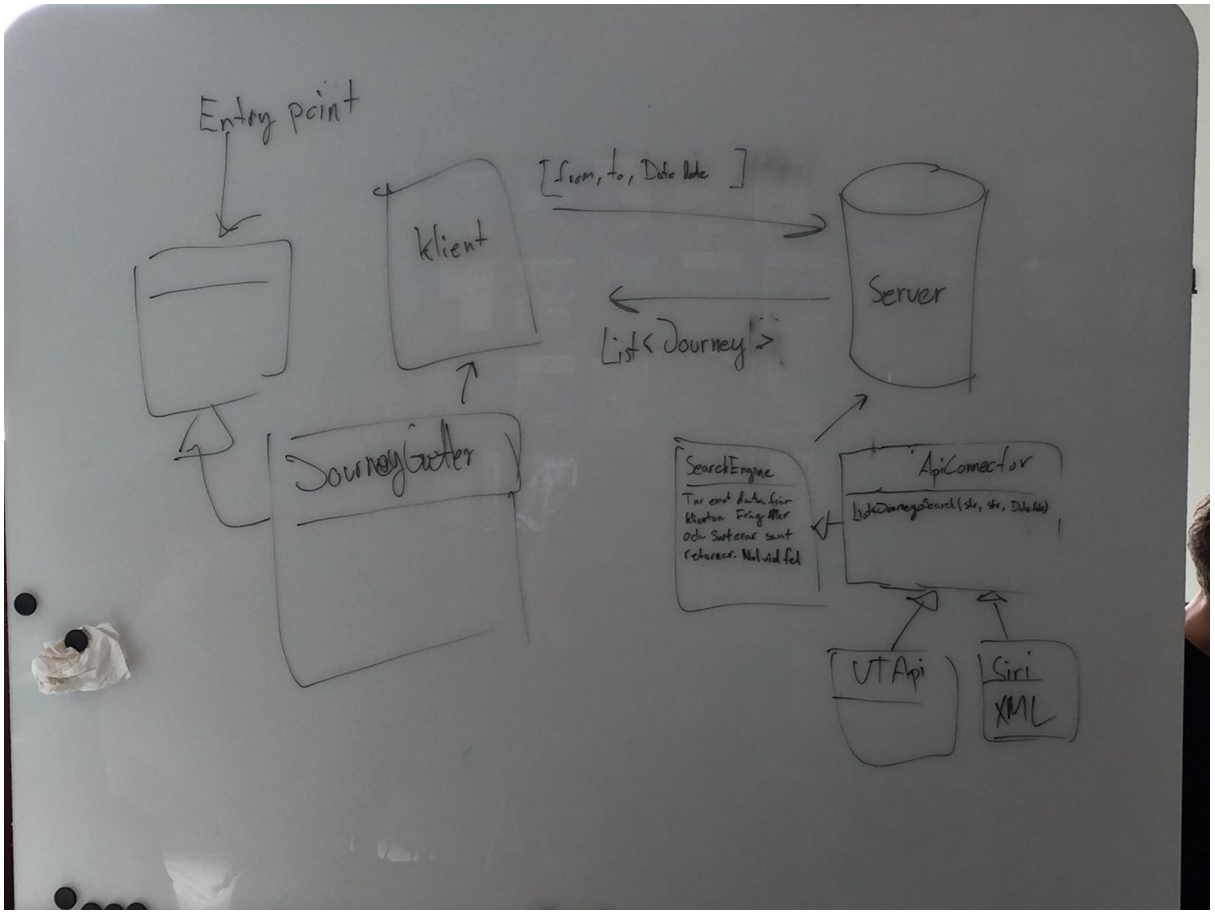

TODO
Definition of DONE
Entire group reflektion
Acceptance criterias
Social contract update
GWT
EntryPoint
FormInput
InputInterface
VTInput
StenaAutandica
Controller
FerryInput
GUI
Journey

Bilaga 5



Vi hade en tanke på att använda MVC mönstret

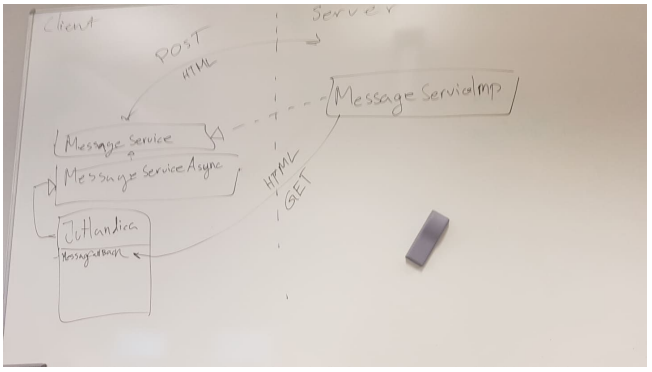
Bilaga 6

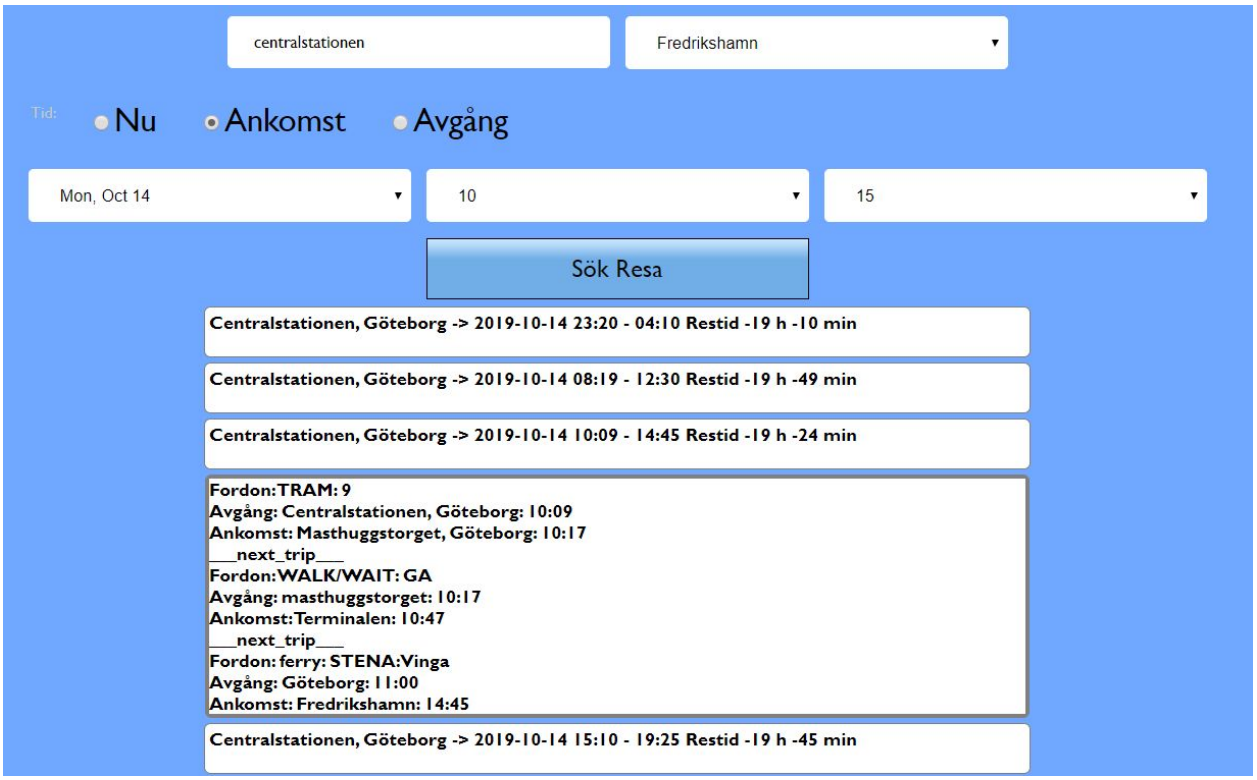


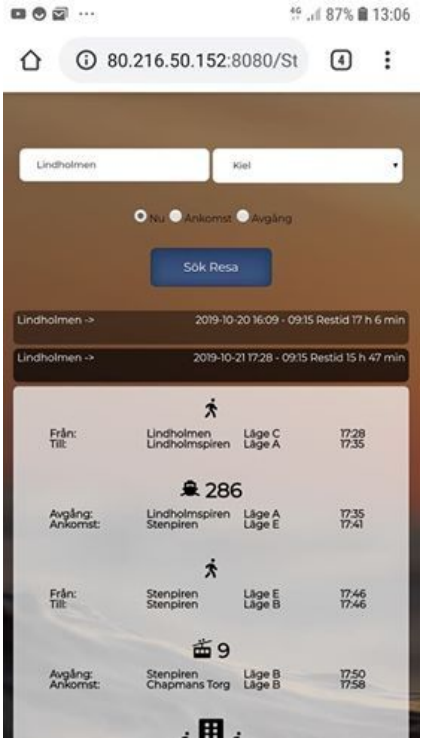
Bilaga 7

En fungerande version men med lite konstiga tecken.

Bilaga 8


Client
Server
POST
HTML
MessageServiceImp
MessageService
MessageServiceAsync
GetHandler
HTML
GET





Slutversion