Planeringsverktyg för tågväxelbesiktning

– automatisering av en manuell arbetsmetod

LUNDS UNIVERSITET
Campus Helsingborg

LTH Ingenjörshögskolan vid Campus Helsingborg
Institutionen för datavetenskap

Examensarbete:
Max Noack
Ola Hansson

LTH Ingenjörshögskolan vid Campus Helsingborg
Lunds universitet
Box 882
251 08 Helsingborg

LTH School of Engineering
Lund University
Box 882
SE-251 08 Helsingborg
Sweden

Tryckt i Sverige
Lunds universitet
Lund 2015

Sammanfattning

I detta examensarbete har en första version av ett digitalt verktyg utformats. Det är unikt framtaget för företaget Tirawa Solutions AB som främst har i uppgift att planera besiktningar av olika järnvägsväxlar och spårdelar över hela Sverige. Syftet med examensarbetet är att undersöka vilka funktionaliteter som är intressanta för att på ett effektivt och samtidigt lättförståeligt sätt redovisa information om växlar och när de ska testas. Verktyget kan ersätta den nuvarande manuella planeringen - en planering som är tidskrävande, oöverskådlig och svår att hantera.

Ett av systemets huvudsyften är att generera en automatisk planering för när växlar ska besiktigas - en funktionalitet som har visat sig ge goda resultat vid användning. Algoritmen genererar en planering där majoriteten av de efterfrågade parametrarna beaktas. Planeringen av samtliga växlar i Sverige kan göras på ett par minuter, ett arbete som tidigare tog flera arbetsdagar.

En del fokus har lagts på att ge verktyget god användbarhet, men också på vad som ska finnas med för att ersätta det nuvarande arbetssättet på bästa sätt. Verktyget har utformats som en webbsida, detta främst för att få ökad tillgänglighet för anställda på Tirawa Solutions, Infranord och Sperry Rail. Tidigare arbetsformer har studerats och många av deras funktionaliteter har implementerats i någon form.

Nyckelord: tågväxelbesiktning, planering, effektivisering, algoritm, AngularJS.

Abstract

During this bachelor thesis a first version of a digital tool has been formed. The tool is uniquely developed for the company Tirawa Solutions AB whose main task is to plan inspections of different railway track parts all across Sweden. The main purpose of this thesis is to investigate what functionalities are interesting to achieve a system that can display information about railway switches and when to test them in an efficient and easy-to-grasp way. The developed tool can replace the manual planning **–** a planning that is time-consuming, unclear and hard to manage.

One of the system's main purposes is to generate an automatic planning for when railway switches should be inspected **–** a functionality that has proven to give good results. It generates a planning where a majority of the wanted parameters are taken into account. Planning all the switches in Sweden can be made in only a few minutes, something that used to take several days.

A lot of focus is put on usability but also what to include to replace the present way of work. The tool has taken the shape of a website, primarily to achieve an increased availability for the employees at Tirawa Solutions, Infranord and Sperry Rail. Former working methods has been studied and a lot of their functionality have been implemented in some way.

Keywords: railway switch inspection, planning, rationalization, algorithm, AngularJS.

Förord

Detta examensarbete utfördes på distans för Tirawa Solutions AB. Vi vill rikta ett stort tack till vår handledare på företaget, Mattias Gustafsson, för god vägledning samt tips och råd, och för att på ett pedagogiskt sätt gett oss kunskap om tågnätets kontrollsystem.

Vi vill även rikta ett tack till Christin Lindholm, examinator, och Christian Nyberg, handledare, på Lunds Tekniska Högskola Campus Helsingborg.

Helsingborg den 21 maj 2015
Max Noack och Ola Hansson

Innehållsförteckning

1 Inledning

För att garantera att Sveriges järnvägsnät är säkert utför Trafikverket regelbundna kontroller och underhåll på samtliga räler, växlar och andra rälskomponenter, se delkapitel 3.1 – Beskrivning av växelkontroller. I detta examensarbete ligger fokus på att effektivisera hanteringen och kontrollen av växlar, främst de som tillhör ett så kallat huvudspår då dessa används mest och därav har högst prioritet. Beroende på växelns besiktningsklass sker kontrollen med ett intervall från vart fjärde till varje år.

1.1 Bakgrund

För några år sedan ansvarade banverket för samtliga järnvägar i Sverige. I dagsläget ligger ansvaret hos Trafikverket som har en skyldighet att se till så att spåren underhålls. Trafikverkets uppgift är preciserad i verkets instruktion [3] enligt följande:

> "Trafikverket ska med utgångspunkt i ett trafikslagsövergripande perspektiv ansvara för den långsiktiga infrastrukturplaneringen för vägtrafik, järnvägstrafik, sjöfart och luftfart samt för byggande och drift av statliga vägar och järnvägar.
>
> Trafikverket ska verka för en grundläggande tillgänglighet i den interregionala kollektivtrafiken.
>
> Trafikverket ska med utgångspunkt i ett samhällsbyggnadsperspektiv skapa förutsättningar för ett samhällsekonomiskt effektivt, internationellt konkurrenskraftigt och långsiktigt hållbart transportsystem.
>
> Trafikverket ska verka för att de transportpolitiska målen uppnås."

Tirawa Solutions AB (TSAB hädanefter) är ett litet, relativt nystartat konsultbolag. Bolaget har sin bakgrund inom planeringsarbete för järnvägsindustrin samt kvalitets- och verksamhetsutvecklingsarbete inom privat samt offentlig verksamhet. Bolaget har i dagsläget två anställda och arbetar i nuläget mot ett par större kunder inom järnvägsbranschen. Den största kunden är det amerikanska bolaget Sperry Rail som är specialister inom oförstörande provningsmätningar av järnvägsräls. Sperry har under

många år haft kontrakt med Trafikverket där uppgiften är att säkerställa en säker anläggning där sprickbildningar i rälsen upptäcks och avlägsnas i tid. TSABs mål är att inga växlar ska testas för sent, något som kan ses som huvuduppgiften för detta examensarbete.

Utförandet av växelkontroller måste planeras manuellt – ett arbete som i skrivande stund görs för hand i Excel av anställda på TSAB. Av säkerhetsskäl ska besiktningarna göras +/- 2 månader från utsatt datum. Att manuellt sitta och pussla ihop en planering för ca 12000 växlar är ett tidskrävande arbete som hade kunnat göras enklare med hjälp av ett speciellt framtaget digitalt verktyg. Ett verktyg där man exempelvis kan sortera planerade kontroller per region, datum eller plats och som också kan genomföra automatiska planeringar utefter en specifik algoritm. Allt i syfte att ge en bättre översikt och underlätta planering för underhållsoperatörer.

1.2 Syfte och målsättning

Syftet med examensarbetet är att undersöka om det är möjligt att uppfylla de krav som efterfrågas från TSAB. I det ingår undersökning om huruvida man kan automatisera en relativt komplex planering med många parametrar och göra den så pass generell att den går att bryta ner i programkod. Resultatet ska redovisas i ett webbgränssnitt.

Syftet med systemet är att skapa ett verktyg för planering av olika typer av växelkontroller som på ett effektivt sätt strukturerar när de olika kontrollerna ska genomföras. Systemet ska vara lätt att förstå och ska ha en pålitlighet lika hög som tidigare använt verktyg.

Målet är att skapa ett planeringsverktyg som kan ersätta den manuella tidsplanering av växelkontroller som måste göras i dagsläget.

1.3 Problemformulering

För att uttrycka vad som ska undersökas följer här de frågeställningar som ligger till grund för examensarbetet och den produkt som utvecklats.

1. Hur ska algoritmen för planeringen utformas för att säkerställa att alla komponenter besiktigas och att det görs i tid?

2. Vilka programmeringsspråk är mest lämpliga vid utformandet av planeringsverktyget?
3. Hur ska man utforma gränssnittet så att arbetet blir effektivare och snabbare, samtidigt som det ersätter det befintliga systemet utan att försvåra arbetet?
4. Vilken databashanterare tillgodoser bäst systemets behov?
5. Hur ska systemet utformas så att operatörerna på ett enkelt sätt kan redovisa genomförd besiktning?

1.4 Avgränsningar

För att begränsa och koncentrera ovan nämnda frågeställningar avgränsas till en början följande punkter i problemformuleringen.

1. Prioritera funktionalitet över säkerhet.
   Fokus riktas först och främst mot att få en fungerande produkt som uppfyller de krav som problemformuleringarna beskriver. För att säkerhetsimplementationen anses tidskrävande får denna en lägre prioritetsnivå och sätts här som en avgränsning.

2. Endast implementera en webbversion av systemet.
   Systemets funktionaliteter prioriteras före införandet av stöd på flera plattformar, då det anses tidskrävande. Främst är stödet för mobila plattformar intressant.

2 Metod

Examensarbetets olika delar präglades av många olika arbetsformer. Kapitlet beskriver kort hur inläsning och analys av förutsättningar gjordes samt vilken arbetsmetodik som tillämpades i utvecklingsprocessen.

2.1 Inläsning och analys

Under inläsningsperioden söktes mycket information om programspråken på internet via officiella hemsidor och forum, då officiell information om de olika programspråk som användes finns lättillgängligt online. Några litterära källor användes också vad gäller information om databaser samt användbarhet. Tidigt upprättades även kontakt med TSAB för att få viktig bakgrundsinformation och diskutera användbarhet.

2.2 Utvecklingsprocess

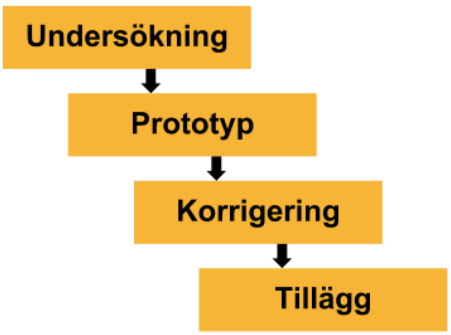
Arbetsmetoden som använts i detta examensarbete kan ses som en typ av hybridversion mellan den klassiska vattenfallsmodellen [19] och den mer moderna utvecklingsmodellen Scrum [18], där eftersökta egenskaper från båda håll tillämpades i arbetsmetodiken. Från vattenfallsmodellen användes i huvudsak fasuppdelning, där examensarbetet delats upp i olika delperioder. Egenskaper från Scrum användes inom de olika delperioderna, som främst baserades på iteration och återkoppling. Överst applicerades alltså vattenfallsmetodens tekniker med faser och struktur. Inom varje fas infördes Scrums mer agila och iterativa arbetssätt. Här under beskrivs dessa egenskaper mer utförligt och varför de valdes.

2.2.1 Vattenfallsmodellen

Vattenfallsmodellen är en systemutvecklingsprocess som baseras på ett sekventiellt flöde bestående av faser, där varje fas är en del i utvecklingen mot ett förutbestämt mål. Metoden används ofta i situationer där ändringar i ett sent skede är mycket kostsamma både tidsmässigt och ekonomiskt.

Vattenfallsmodellens syfte är att strukturera och motivera examensarbetets arbetsgång. Examensarbetet som helhet består av fyra faser; *Undersökning*, *Prototyp*, *Korrigering* och *Tillägg*. Mer utförlig beskrivning av de olika faserna görs i tabell 2.1.

Figur 2.1 nedan visar hur de olika faserna bygger på varandra. De oftast använda egenskaperna från vattenfallsmodellen som inte tillämpades är främst dokumentationen, en skriftlig kravspecifikation och införandet av rollfördelning. Anledningen till detta var att gruppen bestod av två personer som arbetade sida vid sida varje dag. Detta ledde till en bra förståelse för vem som gjorde vad. Dokumentation och kravspecifikation uteslöts då det inte fanns något önskemål för detta från TSAB.



Figur 2.1. Fasuppdelning enligt vattenfallsmodellen.

Enligt [19] s.36-40, nämns följande faser som vanligt förekommande i vattenfallsmodellen:

1. Feasibility study
2. Requirements analysis and specification
3. Design and specification
4. Coding and module testing
5. Integration and system testing
6. Delivery
7. Maintenance

Alla dessa punkter användes inte i examensarbetet, främst på grund av att systemet ses som en första version av ett system med stor utvecklingspotential. Det bör alltså påpekas att produkten som skapats inte är en leveransklar produkt. Därför är de områden som beskriver leverans och underhåll uteslutna helt från arbetsmodellen.

Tabell 2.1. Tabellen beskriver de olika faserna i examensarbetet.

Fas	Beskrivning
Fas 1 – Undersökning	1. Skapa förståelse för projektbeskrivningen. 2. Hålla möten med TSAB. 3. Prototyp av design/innehåll i systemet. 4. Analysera olika utvecklingsalternativ och deras prestanda. 5. Skapa ett UML-diagram över databasens relationer. 6. Skissa på systemets utseende och innehåll.
Fas 2 – Prototyp	1. Använd de metoder och tekniker som anses viktiga för att skapa en första prototyp - grundfunktioner som att visa motsvarande info från Excel-fil, filtrering samt statistik för totalt/testade/otestade. 2. Utveckla en planeringsalgoritm som genomför datumplaneringar efter specifika kriterier. 3. Inför funktioner för att skapa en personalplanering.
Fas 3 – Korrigering	1. Utvärdering och testning av prototypens funktionaliteter. 2. Funktionalitet för att exportera till Excel. 3. Lägga mycket fokus på användbarhet.
Fas 4 – Tillägg	1. Inför funktion för manuell planering och redigering. 2. Inför funktion för inloggning.

Faserna hänger ihop på följande sätt. Fas 1 är den fas som lägger grunden till examensarbetet. Här analyseras vilka frågeställningar som kan vara intressanta att arbeta mot. Det är också en inläsningsfas där de kunskaper och den bakomliggande information som är nödvändig läses in. Dessa kunskaper används sedan för att skapa en första modell över hur databasen ska se ut och vad systemet skulle kunna innehålla. Från detta leds projektet in i fas 2, som kan ses som den mest omfattande och betydelsefulla fasen, där resultat från fas 1 används för att skapa en första prototyp. I den

implementeras de egenskaper som prioriteras högst; där ibland sökning efter växlar, hantering av personalplanering och den automatiska planeringsalgoritmen. Fas 3 ger utrymme att utvärdera prototypen och se över hur väl den egentligen uppfyller kraven på systemet. De egenskaper som inte håller måttet korrigeras. Mer fokus läggs på att finslipa systemets användbarhet. I sista fasen, fas 4, förs de sista funktionaliteterna in som anses viktiga för att få ett fungerande system utifrån målbilden. Här läggs begränsningarna inloggning och manuell planering till, om tid finns. Om tiden inte räcker till kan fas 4 helt uteslutas.

Då innehållet i varje fas inte är hugget i sten kan nya tillämpningar tillkomma och försvinna under arbetets gång. Denna egenskap härstammar från Scrum och bidrar till en friare utvecklingsprocess.

2.2.2 Scrum

Scrum innehåller iterativa egenskaper som bygger på återkoppling och utveckling av vad som gjorts tidigare. Inom varje enskild fas tillämpas dessa egenskaper då deras huvudsyfte grundas i att i första hand ge produktutvecklingen utrymme att växa åt rätt håll och på rätt sätt, men också för att låta andra områden som undersökning och analys formas efter produktens för stunden prioriterade behov. Denna metodik tillämpas främst i situationer där exempelvis funktionaliteter i programmet inte uppfyller de krav som TSAB ställer på slutprodukten, eller att de helt enkelt ogillas av användaren. Friheten att då kunna arbeta iterativt för att åtgärda och förebygga sådana situationer har här stort genomslag i syfte att effektivisera utvecklingsprocessen.

I Scrum används oftast något som kallas "sprint". En sprint är en specificerad tidsperiod som vanligtvis omfattar 3 till 30 dagar. Den inleds med en planeringssession där man bestämmer vad som ska göras under denna period. I detta examensarbete ses de olika faserna som enskilda sprintar på så sätt att de innehåller unika händelser som ska genomföras under fasen. Orsaken till varför Scrum valdes var främst på grund av den återkoppling som tillämpades samt att examensarbetets medlemmar sedan tidigare är bekanta med arbetsformen. Detta medför en rörligare arbetsgång och värdet av små resultat kan utvärderas och utvecklas iterativt.

Dock användes inte heller här en fullversion av Scrum. De egenskaper som inte tillämpades var främst dagliga möten och även här rollfördelning.

Dagliga möten var inte nödvändigt då examensarbetets medlemmar hela tiden arbetade tillsammans.

2.3 Källkritik

Mycket information och kunskap om räls och växlar, samt hur besiktning av dessa fungerar har fåtts från TSAB-anställda Mattias Gustafsson. Hans kunskap anses trovärdig och mycket av bakgrunden till examensarbetet baseras på samtal och möten med honom. Övrig information är hämtad från Banverket.se vilket av naturliga skäl får anses vara en god källa.

Några källor härstammar från litteratur, där ibland *Databasteknik* [2] för information om databaser och *Don't make me think* [1] för kunskaper om användbarhet. Båda dessa anses högst trovärdiga då de används som kurslitteratur på Lunds Tekniska Högskola.

Vid utveckling av prototypen och slutprodukten hämtades mycket information från internet. Vid implementation av funktionaliteter inom webbutveckling användes ett välkänt forum vid namn stackoverflow.com [20] som är ett stort forum för just utveckling i diverse språk och utvecklingsmiljöer. Eftersom det är just ett forum kan ingen av informationen här anses korrekt om den inte verifieras. Eftersom det handlar om eftersökta funktioner till systemet blev detta en enkel fråga. Mycket av den information som hittades här redovisade en ungefärlig eller allmän lösning på ett problem, som oftast behövde modifiering för att fungera som tänkt. Därför kan Stackoverflow i många fall ses som en källa till inspiration snarare än en källa till fakta. Att redovisa källor för samtliga trådar skulle bli överflödigt men då koden vid många tillfällen visat sig göra vad som efterfrågats kan det ses som en trovärdig källa.

Angående de källor som hänvisar till de språk som infördes vid utvecklingen av systemet användes oftast referensmanualer från utvecklarnas officiella hemsidor. Detta på grund av att de anses högst trovärdiga och för att väldigt lite information, framförallt om AngularJS, finns i tryckt form. Även här bör nämnas att informationen enkelt kunde verifieras genom testning i systemet.

3 Teknisk bakgrund

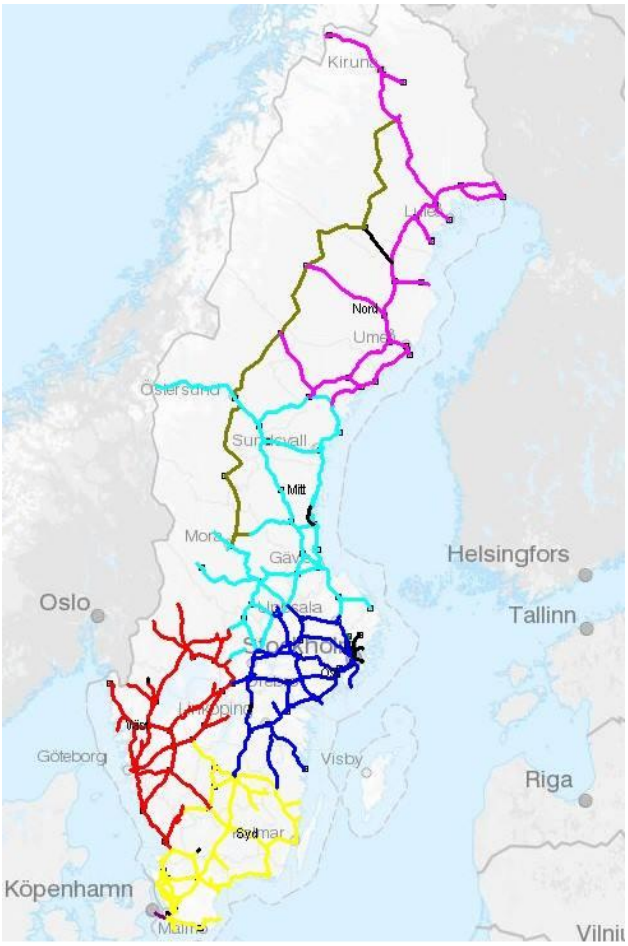
Det svenska järnvägsnätet är indelad i många olika delar. I detta kapitel beskrivs dessa delar och bakgrunden till växelkontrollerna. Dessutom ges en teknisk bakgrundsbeskrivning kring de programspråk och den databashantering som används i systemet.

3.1 Beskrivning av växelkontroller

Samtliga växlar i Sverige måste besiktigas minst vart fjärde år. Hur ofta beror på växelns besiktningsklass. I följande kapitel beskrivs hur dessa kontroller är strukturerade, hur planeringen av detta görs idag samt en allmän beskrivning om växlar.

3.1.1 Regioner

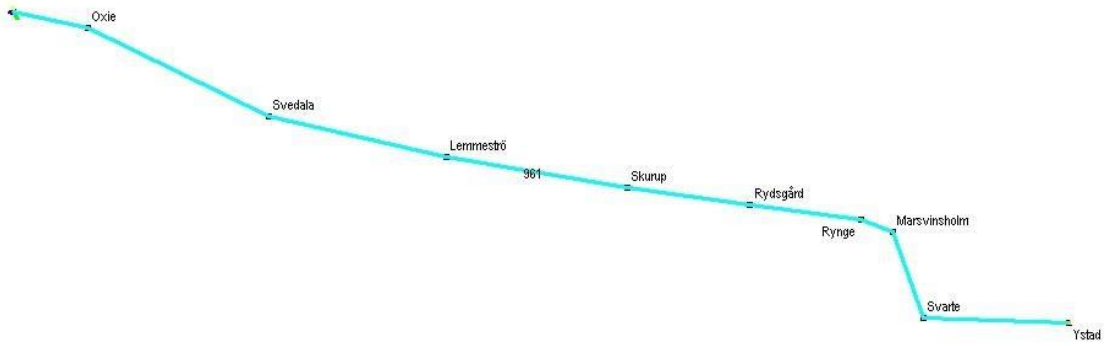
Kontroller av växlarna görs av operatörer inom de olika regionerna. Varje region har ett eget ansvar för sina växlar och bandelar, och därmed också det huvudsakliga ansvaret för att säkerhetskontrollerna genomförs i den regionen. I händelse att en region behöver stöd finns det möjlighet att omfördela personal mellan olika regioner. Detta är vanligt förekommande vintertid då tågspåren i norr är täckta med snö medan det kanske fortfarande är möjligt att arbeta i söder.



Figur 3.1. Bilden visar de olika regionerna i Sveriges järnvägsnät.

Figur 3.1 visar tydligt tågnätets regionsuppdelning. Nätet är uppdelat i sex olika regioner, Inlandsbanan(grön); Nord(lila); Mitt(turkos); Öst(blå); Väst(röd) och Syd(gul). Dock

inkluderas inte Inlandsbanan i systemet då trafikverket inte har underhåll över den regionen. Varje region består av flera olika bandelar. Vissa bandelar sträcker sig över flera regioner. En bandel kan ses som en specifik järnvägssträcka från punkt A till punkt B.



Figur 3.2. Bilden visar bandel 961 mellan Lockarp och Ystad.

I tågnätet benämns olika sträckor som bandelar. En bandel kan gå igenom flera orter, så kallade driftplatser. Figur 3.2 visar en bandel mellan driftplatserna Lockarp och Ystad. Sträckan mellan två driftplatser kallas linje.

3.1.2 Växlar

En växel (järnvägsväxel) är en förgrening av ett järnvägsspår. En växel består av rörliga delar och har till syfte att styra in tåg från ett spår till ett annat.

Varje räl, växel eller annan rälskomponent måste enligt trafikverkets regelverk kontrolleras regelbundet på grund av säkerhetsskäl. Då en kontroll genomförs letar man bland annat efter sprickor i rälsen, skador eller andra förslitningar som kan utgöra en säkerhetsrisk. I systemet hanteras endast växlar, men systemet kan komma att hantera samtliga rälskomponenter i en eventuell framtida vidareutveckling. Kontroll av rälsen görs idag av en så kallad mätvagn som körs på de bandelar som ska kontrolleras. Misstänksamma punkter rapporteras och ska verifieras inom tre veckor av samma personal som utför besiktningar på växlar. Växelkontroller och verifiering är de två största arbetsuppgifterna för operatörerna.

Varje växel har en del information kopplad till sig, bland annat information om när växeln senast besiktigades och var i tågnätet den finns. Samtliga parametrar nämns i tabell 3.1 nedan. Informationen är hämtad från banverket.se [4] och TSAB [21].

Tabell 3.1. Tabellen beskriver vilken information som kan kopplas till varje enskild växel.

Parameter	**Beskrivning**
Bandel	En specificerad spårsträcka. En region består av flera bandelar. En bandel kan sträcka sig över flera platser och flera regioner (se figur 3.1).
Plats	Ett specificerat område, även kallat driftplats.
Typ	Ett id-nummer som anger växelns modellnamn och typ.
Region	Anger vilken region växeln finns i. (Nord, Mitt, Öst, Väst eller Syd).
UNE	Betecknar om en växel tillhör ett huvudspår och i så fall om bandelen tar enkelriktat eller dubbelriktad trafik.
Besiktningsklass	Hur ofta växeln måste besiktigas. Följande klasser finns: • B1, vart fjärde år. • B2, vart tredje år. • B3, vart annat år. • B4 och B5, varje år.
Senast besiktigad	Det datum som växeln senast blev besiktigad.
Km + m fr	Avståndet från bandelens startpunkt i kilometer + meter till den punkt där växeln börjar.
Km + m ti	Avståndet från bandelens startpunkt i kilometer + meter till den punkt där växeln slutar.

3.2 Planering

All nämnd data som beskrivs i tabell 3.1 tillhörande de olika växlarna hanteras i dagsläget manuellt av TSAB med hjälp av verktyget Microsoft Excel i vilket all information sparas. Excel anses vara ett verktyg som inte är helt optimerat för arbetsuppgiften. Informationen om växlarna bör uppdateras minst en gång per år då växlar hela tiden byts ut, tas bort eller tillkommer. Kontroll av varje växel måste också planeras manuellt. Denna planering görs efter följande kriterier:

1. Inom det godkända intervallet av +/- 2 månader från föregående besiktningsdatum plus tidsintervall beroende på besiktningsklass.
2. När operatörerna har tid i sitt schema. Dessutom tar det 0.7h [21] i genomsnitt att genomföra en kontroll vilket också måste tas hänsyn till.
3. Om möjligt kombinera besiktningen med andra besiktningar som kan göras på platsen för att effektivisera operatörernas arbetstid.
4. I första hand prioritera besiktningar som blivit försenade.
5. Växlar som har planerad besiktning i år och är av besiktningsklass B1 och B2 ska kontrolleras i mån av tid.

För att användaren ska kunna effektivisera sitt arbete behövs ett mer översiktligt och lätthanterligt verktyg som kan hantera informationen om växlarna på ett mer specialanpassat vis.

3.3 Systemets krav

Implementering av företagets önskade funktionaliteter till någon form av digitalt verktyg kunde gjorts på många olika sätt. Antalet krav som efterfrågades var till en början lågt. De krav som ändå sattes på systemet, och som också användes för att motivera de tekniska val som gjorts, var huvudsakligen följande:

1. Systemet ska på ett effektivt sätt hämta, visa och lagra information från en lista med ca 12000 rader information, där varje rad i listan representerar en specifik växel.
2. Systemet ska vara översiktligt, lättanvänt, tillgängligt, dynamiskt och lätt att förstå.
3. Man ska enkelt kunna söka efter specifika växlar men man ska också kunna filtrera samlingen med växlar efter vissa specifika kriterier.
4. Systemet ska i ett senare skede ha funktionalitet för flera olika användare och ska vara enkelt att bygga ut.

Nedan beskrivs några av de tekniker som av examensarbetarna ansågs vara lämpliga kandidater att användas i systemet för att uppfylla dessa krav. Där ibland programspråk, scriptspråk och databashanterare.

3.4 Programspråk

Det togs tidigt ett beslut av examensarbetets medlemmar tillsammans med TSAB om att någon typ av webbgränssnitt var att föredra. Varför ett webbgränssnitt ansågs mest lämpligt tas upp under kapitlet 4.2 - "Varför webbaserat?". De programspråk som ansågs mest lämpliga och deras intressanta funktionaliteter beskrivs i denna del. Vilket programspråk som skulle användas var helt upp till examensarbetarna.

Inom webbutveckling finns det många olika utvecklingsmiljöer, programspråk och scriptspråk. Det är ett område som befinner sig i ständig utveckling. Efterfrågan på dynamiskt material och olika typer av mediehantering driver webbutvecklingen mot ett mer funktionsbaserat håll, där gamla språk som HTML och PHP inte håller måttet, och där exempelvis scriptspråk eller HTML5 får större inflytande. Scriptspråk kan användas för att få ett dynamiskt webbverktyg, det vill säga att sidan uppdateras utan att laddas om.

3.4.1 PHP

PHP [8] är ett så kallat "server side"- scriptspråk vilket innebär att när en sida efterfrågas kompileras koden på servern som sedan returnerar en HTML-sida till användaren. Det används ofta för att skapa dynamiska hemsidor, till exempel en webbtidning. Istället för att skriva en HTML-fil per artikel kan man lagra innehållet i artiklarna i en databas och använda PHP för att hämta rätt artikel från databasen. Innehållet i artikeln skrivs sedan ut i en generell PHP-sida med plats avsatt för rubrik, ingress, brödtext och bilder.

3.4.2 AngularJS

AngularJS [5] är ett JavaScript framework utvecklat av Google som bygger på model-view-controller-konceptet. Modellen (model) representerar data som används i och visas av applikationen, vyn (view) är det användaren ser när hen använder applikationen och controllern är det som binder ihop de

två andra delarna. Controllern övervakar användarens handlingar och väntar på knapptryckningar eller liknande för att sedan vidarebefordra till antingen modellen eller vyn.

AngularJS är alltså ett ramverk med färdigskriven kod bland annat för att manipulera gränssnittet. I figur 3.3 nedan följer ett exempel på hur AngularJS tillåter utvecklaren att definiera utseendet för en rad i en tabell som sedan upprepas för samtliga objekt i en samling.

```
<tr ng-repeat="data in filtered = (list | filter:basicFilter | filter:detailedFilter | orderBy : predicate :reverse) |
startFrom:(currentPage-1)*entryLimit | limitTo:entryLimit">
    <td align="center">{{data.Bdl}}</td>
    <td>{{data.TplStr}}</td>
    <td>{{data.Benamning}}</td>
    <td>{{data.Typ}}</td>
    <td align="center">{{data.UNE}}</td>
    <td align="center">{{data.Agare}}</td>
    <td align="center">{{data.Bklass}}</td>
    <td align="center">{{data.Recentdate}}</td>
    <td align="center">{{data.Nextdate}}</td>
    <td align="center">{{data.Lastweek | testThisYear}}</td>
    <td class="editable" align="center">{{data.Plannedweek | plannedWeek}}</td>
    <td class="editable" align="center">{{data.Tested}}</td>
</tr>
```

Figur 3.3. Skärmbilden visar ett exempel på hur utvecklare kan använda AngularJS för att skapa en mall som upprepas över en samling objekt.

Koden i figur 3.3 används för att skriva ut tabellen med resultatet från filtreringen på *Sök*-sidan. Attributet *ng-repeat* anger att *tr*-elementet (eller tabellraden) ska upprepas för varje objekt i samlingen *filtered*. Förutom *ng-repeat* finns till exempel *ng-click* som lyssnar efter att användaren klickar på något, *ng-class* som ändrar CSS-klass på ett element beroende på exempelvis värdet i en cell eller liknande och *ng-disabled* som anger att ett element inte går att interagera med, se figur 3.4.

```
<div class="singleResult" ng-click="addTrackpart(data.Bdl)" ng-class="{'no-extras': editable}"
     ng-repeat="data in filteredTracks = (trackParts | filter:trackpartFilter | filter:bandel)">
    <input type='checkbox' ng-model='isChecked' ng-disabled="editable" ng-class="{'no-extras': editable}"
           ng-checked="checkTrackpart(data.Bdl)"> {{data.Bdl}} <br>
    <hr>
</div>
```

Figur 3.4. Skärmbilden visar ett exempel på ett HTML-element som modifieras med hjälp av AngularJS då användaren interagerar med applikationen.

I figur 3.4 syns hur *ng-checked*, vars uppgift är att bestämma ifall en checkbox ska vara ikryssad eller ej, anropar metoden *checkTrackpart* som

kollar ifall en bandel är vald eller inte – se figur 3.5. Funktionen är skriven i JavaScript och använder variabeln *$scope* som representerar den tidigare nämnda modellen vars uppgift är att rymma data som används av applikationen.

```
$scope.checkTrackpart = function(trackpart) {
    for(var t = 0; t < $scope.selectedTrackparts.length; t++) {
        if(trackpart == $scope.selectedTrackparts[t]) {
            return true;
        }
    }
    return false;
};
```

Figur 3.5. En skärmbild på funktionen *checkTrackpart* som kollar ifall användaren har valt en bandel eller ej.

3.4.3 SQL

Att välja en relationsdatabas där SQL används som programspråk var för detta system självklart, främst på grund av dess enkelhet och då det har många efterfrågade egenskaper. Mer om varför SQL valdes nämns senare i delkapitel 4.6 – Val av databashanterare.

Några av de mest använda databashanterare som har stöd för SQL är främst DB2 från IBM, Microsoft Access, Microsoft SQL Server, MySQL, Oracle Database [2], s. 9. Alla har samma grundläggande databasstruktur, där information som hör ihop sparas i tabeller där varje entitet representerar ett objekt, i det här fallet en spårväxel. För att ändå skilja de nämnda databashanterarna åt tas deras huvudsakliga egenskaper, målgrupp och grundtanke upp i tabell 3.2. Det bör också nämnas att jämförelsen sker mellan de senaste utgivna versionerna av samtliga hanterare.

Tabell 3.2. Här beskrivs de olika databashanterarnas grundläggande egenskaper.

Namn (tillverkare)	Senaste uppdaterad	Kräver licens	Beskrivning
DB2 (IBM)	2013	Ja	Skapat av IBM som är en av grundarna till SQL. Senaste stabila versionen är från 2007.
Microsoft Access (Microsoft)	2010	Ja	En klientbaserad databashanterare som använder sig av databasmotorn JET.
Microsoft SQL Server 2014 (Microsoft)	2014	Ja	En annan klient från Microsoft. Huvudsakliga SQL-språk som används är T-SQL och ANSI SQL.
MySQL 5.6 (Oracle Corporation)	2014	Nej	Grundat av ett svenskt företag. Är känd för att hantera små till medelstora databaser bra. Programmet är helt avgiftsfritt.
Oracle Database (Oracle Corporation)	2013	Ja	En klient skapat av företaget Oracle.

4 Analys

Detta kapitel kommer behandla problemformuleringarna i delkapitel 1.3 – Problemformulering och besvara följande frågor som ställdes där.

1. *Hur ska algoritmen för planeringen utformas för att säkerställa att alla komponenter besiktigas och att det görs i tid?*
2. *Vilka programmeringsspråk är mest lämpliga vid utformandet av planeringsverktyget?*
3. *Hur ska man utforma gränssnittet så att arbetet blir effektivare och snabbare, samtidigt som det ersätter det befintliga systemet utan att försvåra arbetet?*
4. *Vilken databashanterare tillgodoser bäst systemets behov?*
5. *Hur ska systemet utformas så att operatörerna på ett enkelt sätt kan redovisa genomförd besiktning?*

Det kommer bland annat redogöras för hur den automatiska planeringen utformades, vilka scriptspråk som valdes samt en analys kring systemets utseende och användbarhet.

Analysen av punkt 1 och 3 grundade sig i samtal med TSAB för att få en så bra förståelse för problemen som möjligt, samt för att säkerställa att rätt information om hur planeringen genomförs togs i beaktning. Då användaren i det här fallet var TSAB var det enkelt att verifiera de krav och idéer som förts fram.

Mycket av implementationen av punkt 3 och 5 krävde motivation inom användbarhet. Denna information hämtades främst från Krugs bok [1], samt genom mindre möten med TSAB där användbarheten diskuterades.

Analysen av punkt 2 och 4 har skett genom att läsa på om olika alternativ och väga fördelar mot nackdelar. Fokus låg här främst på systemets behov.

För att generellt beskriva hur olika områden och funktionaliteter prioriterades redovisas dessa i tabell 4.1 där områden överst på listan prioriterades först.

Tabell 4.1. Tabellen beskriver hur olika arbetsområden prioriterades.

Område	Prioritet
Sökfunktion	Hög
Automatisk planering	Hög
Användbarhet	Hög
Översikt förfalloveckor	Hög
Manuell planering	Medel
Översikt planerade veckor	Medel
Exportering av sökdata	Medel
Säkerhet	Låg
Stöd för mobila plattformar	Låg

4.1 Automatisk planering

Utformandet av den automatiska planering som har till syfte att underlätta arbetet för den som gör planeringen av när de olika växlarna ska besiktigas visade sig vara mer komplex än väntat. När TSAB gör sina planeringar tar man i dagsläget hänsyn till många parametrar. För att få en uppfattning om dess komplexitet redovisas dessa parametrar med en kort beskrivning i tabell 4.2. Rimlighetsfaktorn redogör för vilka av dessa som planeringsalgoritmen på något sätt kommer kunna ta hänsyn till vid en automatisk planering. 5 ansågs enligt examensarbetarna mycket rimligt och 1 ansågs helt orimligt.

Tabell 4.2. I tabellen beskrivs de parametrar som är intressanta vid en planering av en växel.

(id) Parameter	Beskrivning	Rimlighets-faktor (1-5)
(1) Kontrollera så det finns tid i schemat.	Undersök om operatörerna i regionen har tid i sitt schema för den valda veckan.	5
(2) Är veckan inom tillåtet intervall för besiktning av växeln?	Kontrollera om veckan ligger inom det godkända intervallet om +/- två månader utifrån nästa, optimala besiktningsdatum.	5
(3) Ta hänsyn till besiktningar i närheten.	Om möjligt kombinera besiktningen med andra besiktningar som kan göras på platsen för att effektivisera arbetet.	4
(4) Prioritera efter utgångsdatum.	Prioritera besiktningar efter närmsta utgångsdatum.	5
(5) Prioritera besiktningsklass.	Prioritera växlarna efter besiktningsklass. I första hand kommer B5. Därefter i fallande ordning B4, B3, B2 och slutligen B1.	3
(6) Finns det snö på spåren?	Då det ligger snö på spåren är det omöjligt för operatörerna att genomföra fullständiga kontroller.	1

För att få en utförligare analys över parametrarnas rimlighet, se delkapitel 5.4 – Planeringsalgoritmen.

4.2 Varför webbaserat?

För att få ett system som var åtkomligt på så många enheter som möjligt bedömde examensarbetets medlemmar att ett webbgränssnitt vara det mest optimala. Man kan då med säkerhet definiera vilken webbläsare och operativsystem som systemet kommer fungera på och på så sätt slippa acceptanstester och installation hos kund. Dessutom vet man att det kommer fungera och se likadant ut på alla enheter som uppfyller kraven. En annan fördel är tillgängligheten. Är systemet upplagt som en hemsida är den tillgänglig var som helst ifrån. Detta är dock också nackdelen. Att ha

systemet nåbart för alla som har tillgång till internet utsätter också systemet för risker. För att uppnå god säkerhet ska, om tid finns, krypterad inloggning implementeras i systemet vilket hindrar utomstående att få tillgång till det. En annan ofrånkomlig nackdel är kravet att användaren har tillgång till internet, dock ansågs inte denna nackdel vara ett problem.

4.3 Val av scriptspråk

Då en stor del av uppgiften för systemet är att organisera och filtrera data beroende på flera parametrar kändes det nödvändigt att använda något sorts scriptspråk. Att ständigt behöva bekräfta sin sökning, vilket är fallet om man väljer att använda PHP, och eventuellt justera sökparametrarna och genomföra en ny sökning kan ur användarvänlighetsperspektiv anses irriterande. Av denna anledning valdes AngularJS som grund för den dynamiska delen av systemet. För exempel på hur AngularJS kan användas tillsammans med HTML för att modifiera utseendet av en webbsida se avsnitt 3.3.1 – Programspråk.

När det kommer till back-end och kommunikation mellan databas och användargränssnitt valdes däremot PHP. Två stora fördelar med PHP är dels dess inbyggda stöd för diverse databashanterare samt det faktum att examensarbetets medlemmar hade erfarenhet av just PHP/MySQL-kombinationen sedan tidigare.

4.4 Utseende

Vad gäller verktygets utseende fanns det inga specifika krav från TSAB, vilket gav utvecklarna förhållandevis fria händer. Som mycket annat på internet bestäms det visuella av CSS-dokument, i det här fallet användes CSS som hämtats från ett öppet bibliotek kallat Bootstrap [7]. Bootstrap bidrog till att få en mer dynamisk och strukturerad sida som är lätt för utvecklare att ändra i. Mycket fokus lades på funktionalitet, enkelhet och kvalitet, då TSABs behov av korrekthet prioriterades före överflöd av funktionalitet.

4.5 Användbarhet

Fokus lades också på användbarhet. Användbarhet, eller *usability* inom programvaruutveckling, är idag ett välanvänt begrepp som tillämpas, eller

borde tillämpas, i alla produkter som utvecklas i professionellt syfte. I boken ”Don’t make me think, Revisited" skriven av Steve Krug [1] beskrivs användbarhet som följande.

> "...usability really just means making sure that something works well: that a person of average (or even below average) ability and experience can use the thing - whether it's a Web site, a fighter jet, or a revolving door - for its intended purpose without getting hopelessly frustrated." [1], s. 5.

Fokus på användbarhet är en kritisk del i utvecklingen för att få ett dynamiskt system som är enkelt att förstå sig på och som samtidigt kan tillfredsställa användarens behov på ett logiskt sätt, där användaren själv inte behöver tänka. Ju större system som utvecklas, desto mer tid måste spenderas på att få en god användbarhet.

Man måste vara medveten om att användaren ofta gör fel, eller inte beter sig som skaparen av ett gränssnitt hade tänkt sig. Då ska ansvaret alltid ligga på att systemet ska hantera detta så att användaren inte straffas för sitt fel utan istället hjälps av systemet. Ett gränssnitt har alltid skyldighet att återge information till användaren på ett sätt som är tillfredsställande. Systemet som beskrivs i denna rapport är förhållandevis litet. Dock är närvaron av dessa tillämpningar minst lika viktiga som andra delar av systemet. Dålig användbarhet leder till ett icke funktionellt gränssnitt. Mycket av informationen kring användbarhet hämtades från boken ”Don’t make me think, Revisited” [1]. Här nämner Krug mycket om hur man kan förvänta sig att en användare beter sig när hen interagerar med någon form av webbsida. Nedan följer några egenskaper som måste tas i beaktning då en webbapplikation utformas.

1. ”Most Web use involves trying to get something done, and usually quickly […] Web users tend to act like sharks: They have to keep moving, or they’ll die.” [1], s. 22. Som utvecklare måste man räkna med att användaren är ute efter specifik information, gärna också så snabbt som möjligt.
2. ”…we’re really only interested in a fraction of what’s on the page […] Scanning is how we find the relevant bits.” [1], s. 22. Sökandet efter information på en sida göras inte systematiskt. Användaren är

nästan alltid ute efter relevant information som matchar det som eftersöks.

3. ”In reality, though, most of the time we *don’t* choose the best option – we choose the *first reasonable option*, a strategy known as satisficing.” [1], s. 24. Användaren kommer I de flesta fall inte lägga tid på att hitta information som bäst passar in på det som efterfrågas, utan istället välja det första alternativ som verkar kunna ge användaren det hen behöver.
4. ”Faced with any sort of technology, very few people take the time to read instructions […] it doesn’t matter to us whether we understand how things work, as long as we can use them.” [1], s. 25, 26. Vi kan inte räkna med att användaren kommer lägga onödig tid på att läsa instruktioner. Istället kommer hen att prova sig fram och hitta sin egen metod för att komma åt eftersökt information.

En användares tålamod avgör hur länge hen kommer att spendera på en webbplats innan vederbörande ger upp eller väljer en annan lösning. För att upprätthålla användarens motivation nämner Krug en rad olika egenskaper som gör att användaren är mer benägen att stanna kvar på en webbsida. Bland annat bör man tänka på följande:

1. “Know the main things that people want to do on your site and make them obvious and easy.” [1], s. 170. Att på förhand ta reda på vilken målgrupp, eller i det här fallet yrkesgrupp, som i huvudsak kommer att besöka webbplatsen samt dessa användares kompetens gör att man också kan förutse vad som kommer efterfrågas av dem.
2. ”Save me steps wherever you can.” [1], s. 170. Onödiga knapptryckningar, fält som måste fyllas i, eller att tidigare information så som användarnamn inte sparas kan vara frustrerande och tidskrävande. Underlätta alltid för användaren som mycket som möjligt.
3. ”Know what questions I’m likely to have, and answer them.” [1], s. 171. Återigen, försök alltid förutse vad användaren är ute efter.
4. ”Make it easy to recover from errors.” [1], s. 171. Systemet handlägger en stor mängd data, data som användaren själv ska kunna hantera i någon form. Det måste därför vara enkelt för användaren att ångra fel.

Med alla dessa personlighetsdrag hos den tänkta användaren eliminerades tidigt många användbarhetsproblem. Resultatet av denna analys redovisas i kapitel 5 – Resultat, där systemets utseende och funktionalitet översiktligt sammanfattas tillsammans med vilka val som gjordes med tanke på användbarheten.

4.6 Val av databashanterare

SQL [2] är ett välanvänt språk för att hämta och modifiera data i en relationsdatabas. Anledningen till varför en databashanterare som använder SQL valdes till systemet grundas främst i dess stöd inom webbutveckling. Scriptspråket PHP har inbyggd funktionalitet för att skapa sessioner, hämta ut och skriva in data in en SQL-databas. Dessutom var examensarbetets medlemmar mest bekanta med SQL vilket bidrog till en lägre inlärningskurva.

God databashantering var för denna slutprodukt avgörande för att få ett så tidseffektivt, hållbart och tillgängligt system som möjligt. Systemets databassystem hanterar förhållandevis små datamängder men behovet av en databashanterare som uppfyllde TSABs krav på responsivitet och korrekthet var stor. MySQL valdes därför som databashanterare eftersom den mötte dessa krav. Dessutom är det gratis och kräver ingen licens för att användas. En annan anledning till att MySQL valdes var på grund av att den ansågs av examensarbetets medlemmar vara en välanvänd databashanterare vilket bidrog till enkelhet att felsöka på internet.

4.7 Redovisning av besiktning

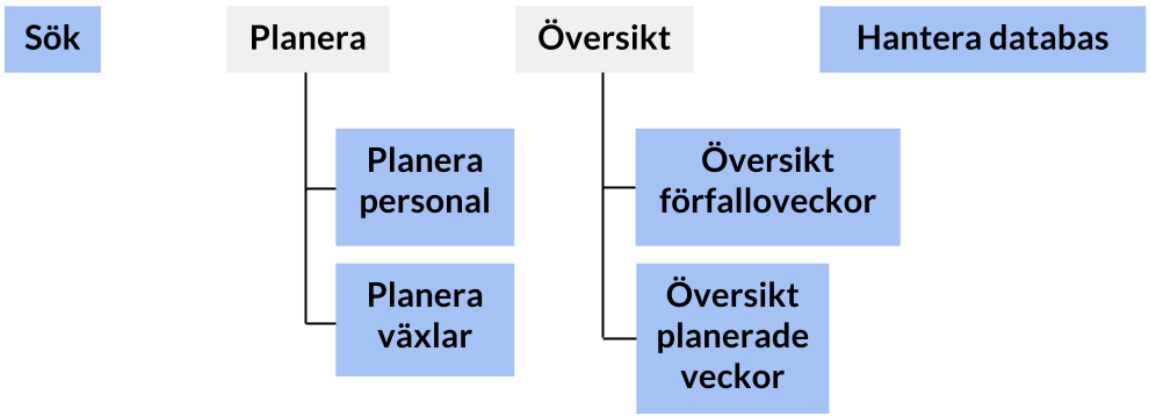
Det fanns behov av att en operatör, var hen än befinner sig, skulle kunna nå systemet för att så snabbt som möjligt efter genomförd kontroll rapportera in vilken växel som besiktigats. Kravet på tillgänglighet var därför mycket viktigt. Detta var ytterligare en anledning till varför ett webbaserat gränssnitt valdes.

I systemet ska en operatör enkelt kunna ange exakt vilken växel som besiktigats och när. I dagsläget skriver operatörerna in att de har besiktigat en växel dels i en Excel-fil som delas i deras intranät men också i Trafikverkets system. Mattias på TSAB menar därför att operatörerna istället för att skriva in i den delade Excel-filen hade kunnat logga in på en

webbsida och föra in samma information. Detta skulle innebära ett system som bättre återspeglar verkligheten och ger mer översikt än ifall informationen finns uppdelad på flera Excel-filer.

5 Resultat

Stora delar av detta kapitel beskriver den prototyp och efterföljande slutprodukt som utvecklades i examensarbetet. Många av de funktionaliteter och användbarhetegenskaper som motiverades i kapitel 4 – Analys, och deras implementation i prototypen respektive slutprodukten redovisas i tillhörande delkapitel nedan. Figur 5.1 visar en översikt på de sidor som finns i systemet.



Figur 5.1. Figuren visar en sitemap över slutprodukten.

I tabell 5.1 ges en kort summering av dessa sidor. Här visas också vilka sidor som tillkom efter utveckling av prototypen samt vilka som ändrades på något sätt. Dessa ändringar redovisas under delkapitel 5.5 – Slutprodukt.

Tabell 5.1. Tabellen redovisar innehållet i slutprodukten.

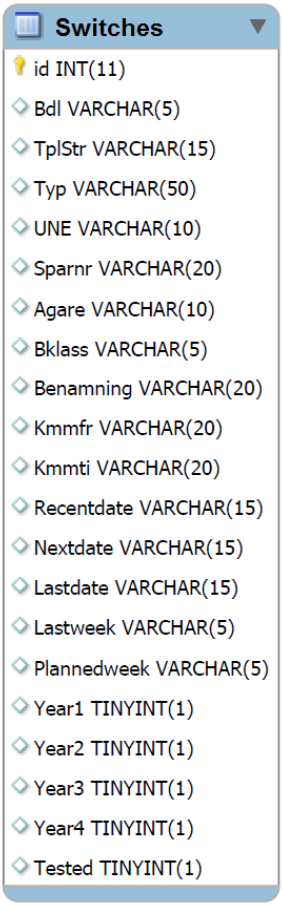
Namn på sida	**Kort beskrivning**	**Hur ändrades den efter prototypen?**
Sök	Innehåller filtret som gör att användaren kan söka efter växlar.	Ändrat utseende. Tillägg av exportering och manuell redigering.
Planera personal	Här kan användaren planera personalens arbetstimmar.	Inga ändringar.
Planera växlar	Sida för att utföra den automatiska planeringen.	Ändrat utseende och till viss del även funktionalitet.
Översikt förfalloveckor	Ger en översikt över antal växlar som förfaller varje vecka och hur många av dessa som hunnit testats.	Ändrat utseende.
Översikt planerade veckor	Ger en översikt över planerade växlar i förhållande till personalplaneringen.	Finns inte med i prototypen. Tillagd först i slutprodukten.
Hantera databas	Importerar Excel-filen till databasen.	Inga ändringar.

5.1 Databashantering

För att flytta data från en Excel-fil till databasen används funktioner på sidan *hantera databas* som läser in information från en CSV-fil. Varje rad är information tillhörande en entitet, och varje cell separeras med ett semikolon. En CSV-fil kan beskrivas som en rådatafil innehållande exporterad data från en Excel-fil. Detta underlättar hantering och inläsning av data för systemet.

En rad i CSV-filen för en växel i syd-regionen kan se ut såhär:

522;Å;Spårväxel - EV-SJ43-11-1:9;;4;Syd;B1;Ja;3;210+ 201;210+ 213;2013-10-29



Figur 5.2. Visar databastabellen *Switches* som beskriver växlarna.

För att hantera data tillhörande växlarna samt information om personalens arbetsschema upprättades en SQL-databas. Databasen innehåller två tabeller; *Switches* (växellista) och *Staffplanning* (personalplanering). Databasen har låg komplexitetsnivå och det finns ingen direkt koppling mellan de två tabellerna. Följande två avsnitt beskriver de båda tabellernas innehåll och syfte.

5.1.1 Växellistan

Växellistan eller *Switches* består av 22 kolumner. Varje rad representerar data tillhörande en växel, data som bland annat importerats från Excel-filen innehållande alla växlar. Figur 5.2 visar tabellens innehåll. Tabell 5.2 beskriver kort vad varje tabellkolumn i växellistan innehåller. Den redogör också för vilka parametrar som importeras från Excel-filen och vilka som skapas av systemet vid importering till databasen. Syftet med de genererade kolumnerna är bland annat att underlätta vid hämtning av specifik data.

Tabell 5.2. Tabellen beskriver vilka parametrar som finns i växellistan i databasen.

Parameter	**Beskrivning**	**Från Excel-filen**
id	Då det krävs tre parametrar för att unikt identifiera en växel infördes en id-parameter som ökar med 1 för varje växel som lagras i databasen.	Nej
Bdl	Vilken bandel växeln finns på.	Ja
TplStr	Vilken driftplats växeln finns på.	Ja
Typ	Växelns typ.	Ja
UNE	Anger om växeln tillhör ett huvudtågspår.	Ja
Sparnr	Spåret som växeln befinner sig på.	Ja
Agare	Vilken region växeln tillhör.	Ja
Bklass	Växelns besiktningsklass.	Ja
Benamning	Växelns nummer – används vid identifiering.	Ja
Kmmfr	Kilometer + meter från en given punkt till start av växeln.	Ja
Kmmti	Kilometer + meter från en given punkt till slut av växeln.	Ja
Recentdate	Datum för senaste besiktning.	Ja
Nextdate	Nästa optimala besiktningsdatum. Utgår ifrån ”Recentdate” och ”Bklass”.	Nej
Lastdate	Sista tillåtna besiktningsdatum. Utgår ifrån ”Recentdate” och ”Bklass”.	Nej
Lastweek	Vilken vecka som ”Lastdate” befinner sig i.	Nej
Plannedweek	Planerad vecka. Är tom vid importering till databasen. Sätts när växeln planeras av planeringsalgoritmen eller av användaren.	Nej
Year1	Är 1 om växeln ska testas detta år.	Nej
Year2	Är 1 om växeln ska testas om ett år.	Nej
Year3	Är 1 om växeln ska testas om två år.	Nej
Year4	Är 1 om växeln ska testas om tre år.	Nej
Tested	Anger vilket datum en växel är testad. Fältet är tomt vid inläsning till databasen men kan ändras av användaren via systemet.	Nej
Testedweek	Anger vilken vecka en växel är testad. Fältet är tomt vid inläsning till databasen. Ändras till veckan tillhörande "Tested" då det värdet ändras.	Nej

Vad gäller motivering till val av databashanterare är växellistan mest betydande. I tabell 5.3 nedan redovisas tiden det tar att läsa in ett visst antal växlar till
Sök-sidan från databasen på två olika enheter. En rimlig inläsningstid ligger kring 2-3 sekunder, en tid som är jämförbar med att öppna ett Excel-dokument.

Tabell 5.3. Tabellen visar resultaten då växellistan hämtas från databasen till *Sök*-sidan.

Enhet	**Operativsystem**	**Webbläsare**	**Antal växlar**	**Tid (s)**
Asus ZenBook ux31a	Windows 8.1	Chrome Version 42.0.2311.152 m	10699	6,86
Samsung ATIV Book 9 Plus	Windows 8.1	Chrome Version 42.0.2311.152 m	10699	7,18

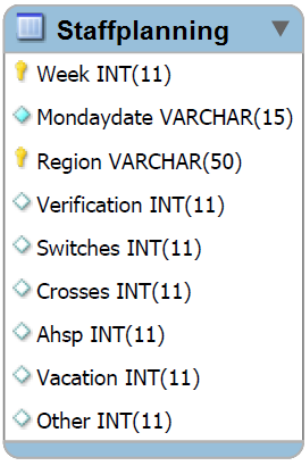
Det bör också nämnas att webbsidans server endast hade tillgång till 10Mbit/s uppladdningshastighet och det är först och främst den som är flaskhalsen vid inläsningen. En uppladdningshastighet på 10Mbit/s är på gränsen till för långsamt, ett rimligt antagande är att framtida server har mycket högre kapacitet än så. Vid test på en localhost-server, som är mer jämförbar med en framtida server, mer än halverades hämtningstiderna. I förhållande till Excel anses resultaten vara rimliga.

5.1.2 Personalplanering

Staffplanning, figur 5.3, innehåller information om personalens planering. Varje entitet i tabellen representerar en vecka på året för regionen man hanterar. Tabell 5.4 innehåller en kort beskrivning på parametrarna i *Staffplanning.*

Tabell 5.4. Tabellen beskriver vilka parametrar som finns i tabellen som hanterar personalplaneringen.

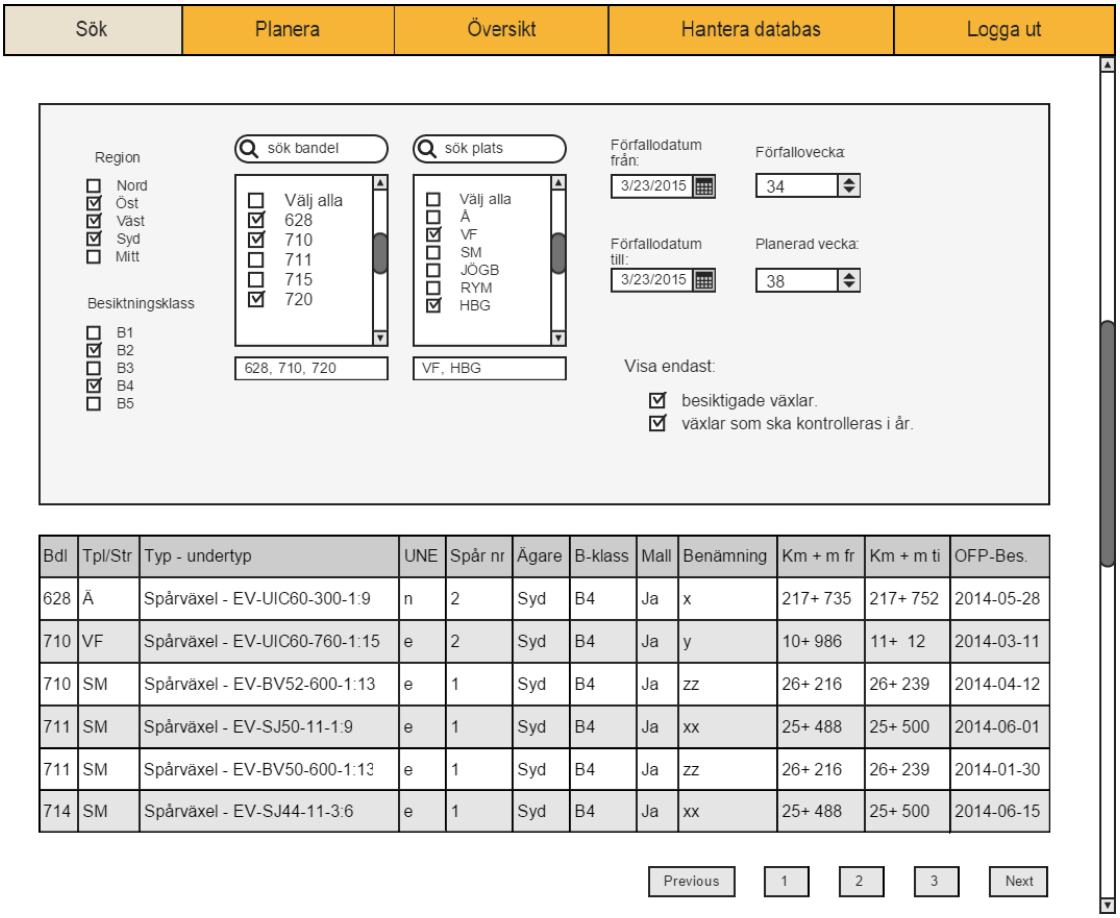
Parameter	**Beskrivning**
Week	Veckonummer.
Mondaydate	Måndagens datum för veckan.
Region	Vilken region planeringen tillhör.
Verification	Antal timmar verifikation.
Switches	Antal timmar växelkontroll.
Crosses	Antal timmar kontroll av kryss.
Ahsp	Antal timmar kontroll av avvikande huvudtågspår.
Vacation	Antal timmar semester.
Other	Annan inbokad tid.



Figur 5.3. Visar personalplaneringens tabell i databasen.

5.2 Utveckling

Första steget i utvecklingen var att skapa en bild över systemets uppbyggnad samt dess layout och funktionaliteter. Detta gjordes genom att skapa en modell, en så kallad mock-up. På så sätt kunde idéer över både användbarhet och funktionalitet samt deras samverkan diskuteras och utformas på ett kreativt sätt. Därmed kunde grunden till hela systemet bestämmas, både på front-end och back-end nivå. Figur 5.4 visar den modell som användes som referens vid utveckling av prototypen som beskrivs i delkapitel 5.3 – Prototyp.



Bdl	Tpl/Str	Typ - undertyp	UNE	Spår nr	Ägare	B-klass	Mall	Benämning	Km + m fr	Km + m ti	OFP-Bes.
628	Ä	Spårväxel - EV-UIC60-300-1:9	n	2	Syd	B4	Ja	x	217+ 735	217+ 752	2014-05-28
710	VF	Spårväxel - EV-UIC60-760-1:15	e	2	Syd	B4	Ja	y	10+ 986	11+ 12	2014-03-11
710	SM	Spårväxel - EV-BV52-600-1:13	e	1	Syd	B4	Ja	zz	26+ 216	26+ 239	2014-04-12
711	SM	Spårväxel - EV-SJ50-11-1:9	e	1	Syd	B4	Ja	xx	25+ 488	25+ 500	2014-06-01
711	SM	Spårväxel - EV-BV50-600-1:13	e	1	Syd	B4	Ja	zz	26+ 216	26+ 239	2014-01-30
714	SM	Spårväxel - EV-SJ44-11-3:6	e	1	Syd	B4	Ja	xx	25+ 488	25+ 500	2014-06-15

Figur 5.4. Visar en mock-up över sidan *Sök*s utseende i ett tidigt skede i utvecklingsprocessen.

5.2.1 Testning

Den testning som gjordes under utvecklingens gång är förhållandevis grundläggande och hade endast ett syfte; att säkerställa att den implementerade funktionen fungerade som det var tänkt och att utvecklingsprocessen gick åt rätt håll.

För att verifiera användbarhet och de funktionaliteter som infördes i systemet hölls kortare möten med Mattias på TSAB. Han fick då möjlighet att prova systemet varefter uppkomna tankar och kommentarer diskuterades. Denna information användes sedan för att förbättra *Sök*-sidan. Detta var en viktig del i den iterativa utvecklingsprocessen där nya implementationer i systemet godkändes av TSAB.

Vad gäller allmänna tester av systemet gjordes endast en officiell kvalitetstestning. Denna bekräftade om systemet var tillräckligt effektivitet

vid hämtning av information från databasen. Resultatet av denna har redovisats tidigare under delkapitel 5.1 – Databashantering i tabell 5.3.

Under utvecklingsprocessen genomfördes hela tiden mindre tester av införda funktionaliteter, där det helt enkelt undersöktes om de gjorde vad som var tänkt. Om inte, justerades koden tills ett eftertraktat resultat uppnåtts.

5.3 Prototyp

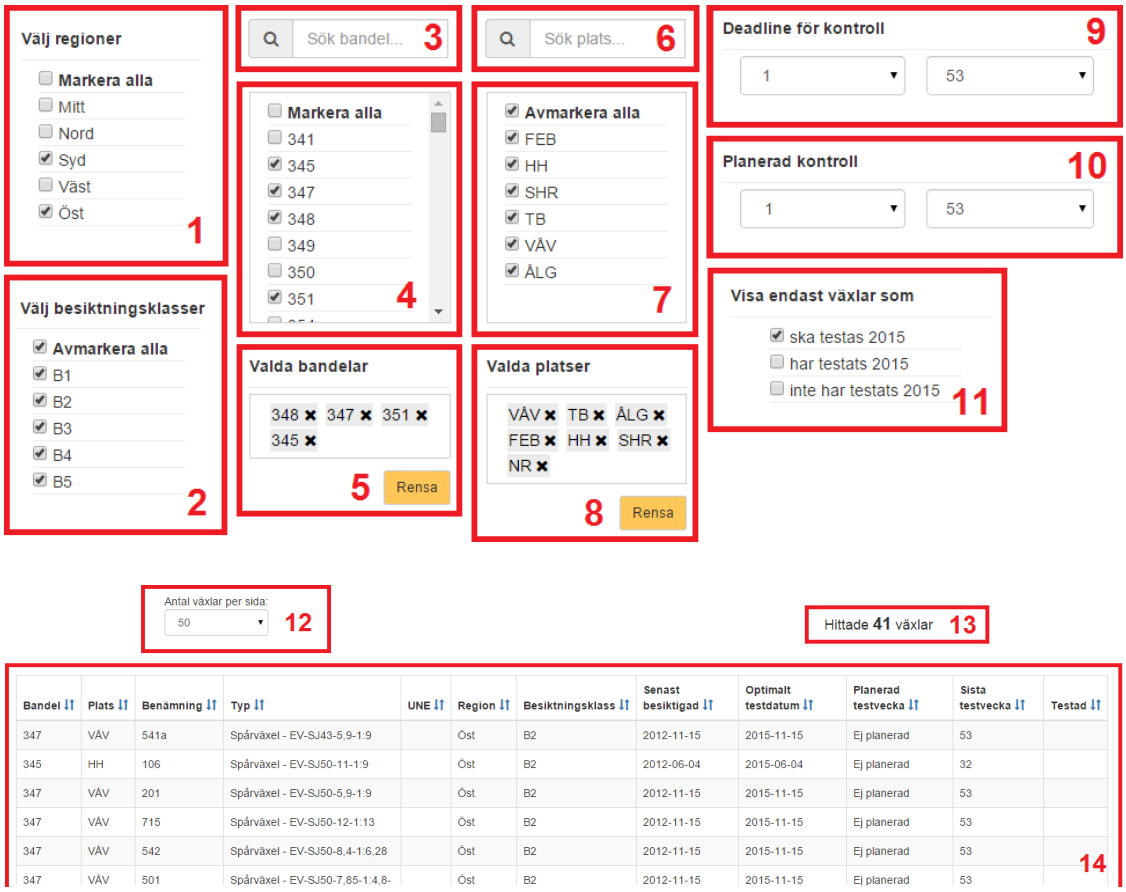
Enligt fasuppdelningen som beskrivs i avsnitt 2.2.1 - Vattenfallsmodellen skapades i fas 2 en prototyp av systemet. Den beskrivs i detta kapitel. Därefter, i fas 3 och 4, genomfördes lämpliga korrigeringar men också nya tillägg till prototypen. Detta redovisas i kommande delkapitel 5.5 – Slutprodukt och delkapitel 5.6 – Tillägg av manuell planering.

Prototypen skulle innehålla funktionalitet för att filtrera och presentera växlar på olika sätt, samt hantera personalplanering och någon form av automatisk planering. Den skulle även genomföra en filtrering av data som för användaren både upplevs enkel och logisk. Utöver filtrering skulle prototypen även ha funktioner för att skapa en planering av växlar och med det kom även en hantering av personalplanering.

Prototypen och senare också slutprodukten fokuserar på att hantera planering av växlar och personalplanering för ett specifikt år. Inga växlar eller personal kan planeras in senare än 31 december för det året man hanterar, och de växlar som blir försenade kommer därför att hanteras av nästa års planering. Orsaken till detta val är främst på grund av att informationen om växlarna uppdateras för varje år. Det kan ha tillkommit eller försvunnit växlar, eller växlar som bytt namn. Detta medför att systemet för varje nytt år måste uppdatera sin samling med växlar. Detta gör att det bara är intressant för systemet hantera ett specifikt år.

5.3.1 Filtrering och presentation av data

Sidan för att visa och filtrera data kallas *Sök*. Filtret består till stora delar av checkboxar då det är ett bra och lättbegripligt sätt att välja bland en mängd tillgängliga parametrar och på ett överskådligt sätt specificera filtreringen av växlarna. Figur 5.5 visar filtrets utseende och markerar de olika filterparametrarna med siffror.



Figur 5.5. Bilden visar en skärmbild av sökningsfiltret. Sidans olika segment är numrerade från 1 till 14.

Tanken är att användaren börjar från vänster och begränsar sin filtrering mer och mer ju längre till höger hen kommer, med mer grundläggande specifikationer till vänster, och mer detaljerade till höger. Då en användare ofta är bunden till en region (1) eller åtminstone bara intresserad av en region är det ett logiskt första steg i filtreringen. I besiktningsklasscheckboxarna (2) kan den eller de besiktningsklasserna som är intressanta att söka på väljas. Dessa boxar är alla ikryssade som default eftersom användarna oftast är intresserad av växlarna oavsett besiktningsklass, till exempel vid sökning på växlar som har planerats för en bestämd vecka.

Efter att användaren valt region(er) och besiktningsklass(er) visas samtliga bandelar som matchar kriterierna i en lista (4). Användaren kan där välja att markera enskilda bandelar, söka efter bandelar (3) eller välja alternativet markera alla som visas i listan. Beroende på användarens val gällande bandelar visas platser (7) som finns på dessa bandelar i valda regioner. Användaren får här samma alternativ gällande sökning (6) och val. Fälten (5) och (8) visar samtliga bandelar respektive platser som användaren har valt att filtrera efter. Varje bandel/plats visas tillsammans med ett kryss som kan användas för att ta bort vald parameter från filtreringen. Om man vill ta bort alla valda bandelar eller platser finns en knapp "Rensa" som enkelt tar bort samtliga valda parametrar från respektive grupp.

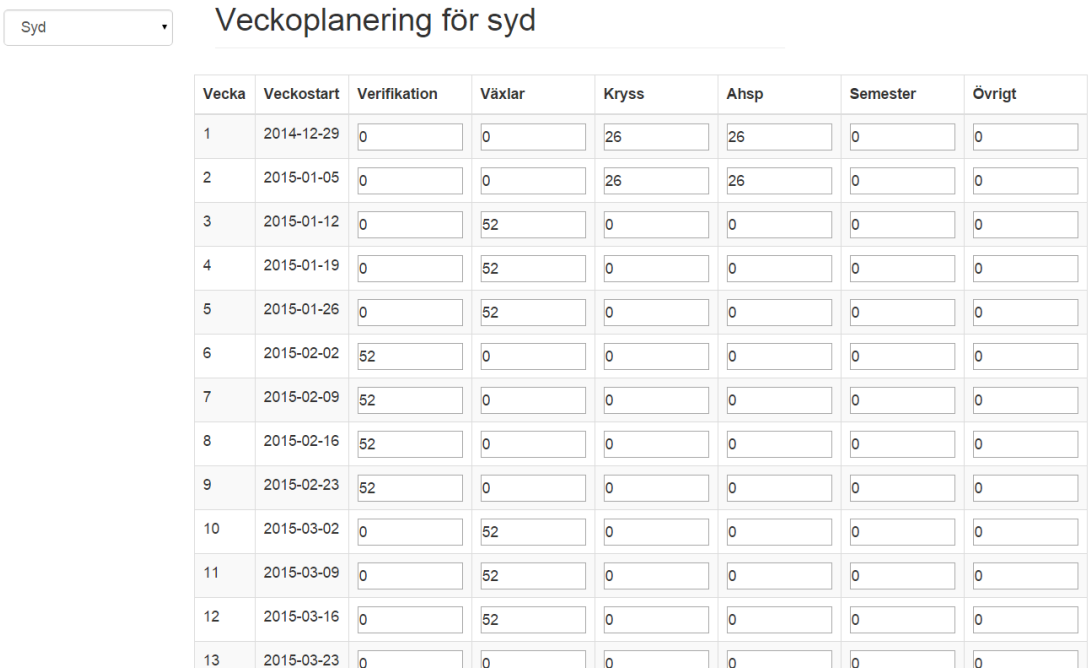
Nästa steg i filtreringen är att välja vilken del av året som är intressant. Användaren har här två alternativ; (9, "Deadline för kontroll") – där användaren filtrerar på vilken vecka som en växel senast ska testas i, samt (10, "Planerad kontroll") – där användaren filtrerar på vilken vecka en växel är planerad att testas i. Dessa fält representeras av dropdown-listor innehållande årets samtliga veckor där endast en vecka kan väljas.

(11) innehåller 3 checkboxar. Ifall den översta (11.1, "ska testas 2015") kryssas ur rensas (9) och (10) då (11.1) inte skulle ha någon effekt annars. Av samma anledning kryssas (11.1) i då något av fälten (9) eller (10) väljs. Ifall användaren vill begränsa sitt resultat till växlar som har testats eller inte har testats kryssar hen i (11.2, "har testats 2015") respektive (11.3, "inte har testats 2015"). Högst en av dessa kan vara ikryssade samtidigt.

När användaren har valt vilka parametraren hen vill filtrera efter visas en lista (14) med resultaten. Hur många växlar som matchar filtreringen visas i (13). Dropdown-lista (12) ger användaren alternativ på hur många träffar som ska visas per sida; 20, 50 eller 100.

5.3.2 Personalplanering

För att kunna planera när en växel ska besiktigas behövs en personalplanering. Som tidigare nämnts kan en växel endast planeras i en viss vecka om personalen ska arbeta med att besiktiga växlar den veckan. Då planeringen sker per region är det rimligt att även här arbeta efter det. Användaren kan enkelt välja region i en dropdown-lista och ändra antalet timmar per vecka och aktivitet. Gränssnittet för personalplaneringen syns i figur 5.6.



Figur 5.6. Bilden visar en skärmbild av personalplaneringen.

Gränssnittet är en kopia av databasens tabell *Staffplanning* (se delkapitel 5.1 – Databashantering). Här kan användaren själv skriva in hur timmarna i varje region ska fördelas för varje enskild vecka. Uppdateringar i gränssnittet sparas med hjälp av AngularJS direkt i databasen vilket gör att det inte behövs någon "Spara"-knapp eller liknande. Detta bidrar till den enklare och mer dynamiska sidan som var en av systemets efterfrågade kvaliteter.

5.3.3 Översikt förfalloveckor

Den sista implementeringen som gjordes i prototypen var en flik i gränssnittet som redovisar och sammanfattar data för att ge användaren en sammanfattning över antalet växlar per vecka. Figur 5.7 visar en skärmbild över sidan *översikt förfalloveckor*. Här kan man se översikt för samtliga regioner, samt trycka på kolumnen för varje region för att se information om den. Sidan visar en summering för varje vecka på året med start på första vecka det förfaller växlar i. Sidan visar det totala antalet växlar som förfaller i varje vecka, samt hur många av dessa som är testade respektive otestade. Dessutom redovisas antal växlar i varje region efter besiktningsklass kategoriserade i tre grupper; B1-B2, B3 och B4-B5. Dessutom visas antalet timmar som kommer behövas för att besiktiga alla

växlar som förfaller den specifika veckan i kolumnen längst till höger. Figur 5.7 visar tydligt hur samtliga växlar som förfaller vecka 17 har testats, men att inga av växlarna i vecka 16 testats.

Vecka	Veckostart	Syd	Totalt			Testade			Ej testade			Väst	Öst	Mitt	Nord	Totalt (n)	Totalt (h)
		Totalt	5-4	3	2-1	5-4	3	2-1	5-3	3	2-1	Totalt	Totalt	Totalt	Totalt		
10	2015-03-02	245	65	151	29	15	18		50	133	29			48	121	414	290 h
16	2015-04-13	53	53						53							53	37 h
17	2015-04-20	18	18			18										18	13 h
18	2015-04-27	11	6		5	5		2	1		3					11	8 h
19	2015-05-04	38	38						38					80		118	83 h
20	2015-05-11	6			6						6			94		100	70 h
21	2015-05-18	14	5		9	5					9			63	36	113	79 h
22	2015-05-25													48	13	61	43 h
23	2015-06-01													140	2	142	99 h
24	2015-06-08													127	12	139	97 h

Figur 5.7. Skärmbilden visar översiktsidans utseende i prototypen.

5.4 Planeringsalgoritmen

Planeringen av växlar är själva grunden till detta examensarbete och även det som främst efterfrågades av TSAB när den inledande kontakten togs. Algoritmen för planeringen är således en mycket stor del i frågan huruvida systemet kommer att kunna användas eller ej. En icke fungerande planeringsalgoritm bidrar direkt till ett dysfunktionellt system.

Sidan som genomför planeringen, *planera växlar* i gränssnittet, har i prototypen endast en knapp för varje region. Dock förbättrades dess utseende avsevärt till slutprodukten. För att se resultatet se delkapitel 5.5 – Slutprodukt.

Som nämnts i delkapitel 4.1 – Automatisk planering och redovisats i tabell 4.1 finns det sex specifika parametrar som den automatiska planeringsalgoritmen borde ta hänsyn till för att generera en optimal planering. Samtliga av dessa har i nedanstående beskrivning på något sätt kunnat implementeras i planeringsalgoritmen. Parametrarna (1) och (2) i tabell 4.1 är självklara för att överhuvudtaget kunna genomföra en grundläggande planering. (4) är också förhållandevis enkel att implementera. Vad gäller parameter (3) och (5) utgör de i sig inga hinder, prioriteringen för dessa görs utan problem. Däremot med parametrarna (1),

(2) och (4) inräknade blir prioriteringen av växlar inte längre tydligt definierad. För att kringgå detta problem prioriteras plats (3) i första hand, sedan besiktningsklass (5). Om då resterande parametrar (1), (2) och (4) uppfylls kan planeringen göras. Vad gäller parameter (6) anses den omöjlig att ta hänsyn till främst på grund av komplexiteten att på något sätt läsa av väderinformation som sedan används som parameter till algoritmen. Dock hanterar algoritmen i nedanstående beskrivna prototyp inhyrning av personal från andra regioner. Om personal i exempelvis nord-regionen är hindrad att utgöra sitt jobb på grund av snö, finns deras arbetskraft tillgänglig att bruka i andra regioner. På så sätt kan fler växlar besiktigas och förseningar undvikas.

Nedan beskrivs den algoritm som blev resultatet av många timmars resonerande och flertalet tester. Målet var att få en så korrekt planering som möjligt utefter de parametrar som var intressanta från varje enskild växel. Som tidigare nämnts görs planeringen enskilt för varje region, då en regions planering inte har någon direkt påverkan på de andra regionernas planering. För en bättre översikt över algoritmen se flödesdiagrammen som beskrivs i bilaga A1 och A2.

Algoritmen ser ut som följer:

1. **Inhämtning av data från databasen.** Växlar som ska testas detta år, inte är testade och som tillhör vald region hämtas och lagras i en variabel (*regionSwitches* hädanefter). Växlarna är sorterade efter besiktningsklass och sedan sista möjliga testvecka som senare kommer vara intressant vid planeringen. Även personalplaneringen för den valda regionen hämtas från databasen och lagras i en variabel (*regionTime* hädanefter). Det skapas även en tredje variabel, *plannedSwitches,* i vilken växlarnas id, geografiska plats samt planerade vecka kommer att lagras efter planering.

2. **Del ett av planeringsalgoritmen körs.** Målet med den första delen av planeringen är att planera så många växlar som möjligt och att gruppera ihop växlar som finns på samma plats/ort till samma vecka, samt att till viss mån prioritera växlarnas besiktningsklass. Växlar som inte kan planeras på grund av tidsbrist ligger kvar i *regionSwitches* när algoritmen har körts. Förfarandet nedan upprepas för varje växel *i* i *regionSwitches.*

1. För varje växel körs en funktion, *getFirstPlannableWeek*, som bestämmer vilken vecka som är den första möjliga att planera in besiktning i. Funktionen returnerar första vecka som ligger inom den specifika växelns testbara intervall (+/- 2 månader från optimalt besiktningsdatum) samt där det finns tid kvar i regionens schema. Om en sådan vecka finns subtraheras 0.7 timmar (schablontiden för en växelbesiktning) från den disponibla tiden för den framtagna veckan. Ifall ingen planeringsbar vecka hittas returnerar funktionen 0.
2. Veckan sparas i en temporär variabel, *week*.
3. Likaså sparas växelns plats undan i en temporär variabel, *place*.
4. *regionSwitches*, från position *i*+1, itereras. Kalla varje växel *j*.
    4.1. Ifall *j* befinner sig på *place* och *week* ligger inom *j*s godkända intervall körs *getFirstPlannableWeek* för *j*. Returvärdet sparas i en temporär variabel *tempWeek*.
    4.2. Då *tempWeek* är skiljt från 0 läggs id för växeln *j* in i *plannedSwitches* tillsammans med *tempWeek* samt växelns plats. Växeln tas bort från *regionSwitches*.
5. Om *week* är skiljt från 0 läggs id för *i* in i *plannedSwitches* tillsammans med *week* och *place*. Även här tas växeln bort ur *regionSwitches.*

3. **Del två av planeringsalgoritmen körs.** Denna del av algoritmen körs till dess att *regionSwitches* är tom eller högst fyra gånger. För varje varv då det fortfarande finns växlar kvar som inte är planerade ökas de veckor som, innan algoritmen kördes, hade inplanerad personal med 26 extra timmar (en persons arbetsvecka). På så sätt kan fler växlar planeras in på de veckorna som från tidigare varv var fullbokade. Anledningen till detta är främst möjligheten att hyra in ledig personal från andra regioner som då kan komma och hjälpa till en vecka då det finns mycket att göra i en viss region. I denna förhållandevis ovanliga situation kan algoritmen ge utrymme för de växlar som inte fick plats att planeras in och på sätt minimeras risken för förseningar. I de fall då det fortfarande finns växlar som inte har fått en vecka då de kan planeras efter totalt fem varv dras slutsatsen att personalplaneringen är alldeles för dåligt tilltagen och bör ses

över. Algoritmen för varv två till fem ser något annorlunda ut, förfarandet följer nedan.

1. Återigen itereras *regionSwitches* igenom, kalla påverkad växel *i*.
2. *i*s geografiska plats sparas undan i den temporära variabeln *place*.
3. *plannedSwitches* itereras igenom, kalla varje växel *j*. Gör följande:
    3.1. Kolla om *place* stämmer överens med *j*s plats och ifall *j*s planerade vecka ligger inom *i*s intervall.
    3.2. Om det finns tid i veckan – lägg till *i* i *plannedSwitches* med tillhörande plats och vecka, ta bort växeln ur *regionSwitches* och bryt iterationen.
4. Om ingen växel på samma geografiska plats var planerad inom godkänt intervall körs *getFirstPlannableWeek* för att se ifall *i* kan planeras in överhuvudtaget. Om så är fallet läggs den in i *plannedSwitches* och tas bort ur *regionSwitches*.
5. Om det fortfarande finns oplanerade växlar i *regionSwitches* efter fyra omgångar av överstående algoritm sparas dessa undan och returneras för att upplysa användaren om att alla växlar inte kunde planeras in. Användaren kan då välja att bekräfta planeringen på de växlar som gick att planera in, men ska också själv manuellt kunna skriva in veckor för de växlar som inte kunde planeras in av algoritmen. Den manuella planeringen ingick dock inte i prototypen utan lades till först i slutprodukten.

Funktionen *getFirstPlannableWeek* returnerar också 0 ifall sista möjliga testvecka för växeln ligger innan nuvarande vecka. Detta får till följd att växlar som är försenade inte planeras in av algoritmen. Av denna anledning får användaren möjlighet att manuellt planera dessa utan begränsningar på vilka veckor som tillåts.

5.5 Slutprodukt

Samtliga funktionaliteter som infördes i prototypen finns med i slutprodukten. Den iterativa processen med stark kundkontakt har direkt bidragit till att de funktioner som implementerats i systemet alltid har varit

relevanta för vidareutveckling av produkten. Slutprodukten innehåller en del tillägg
(5.5.2 – Andra tillämpningar samt 5.6 – Tillägg av manuell planering) och justeringar (5.5.1 – Användbarhet) jämfört med prototypen. Förändringarna har efterfrågats av TSAB efter demonstration av prototypen men även efter samtal som skett under vidareutvecklingen. Detta har varit till stor nytta för användbarheten och kan kanske ses som en typ av användbarhetstester.

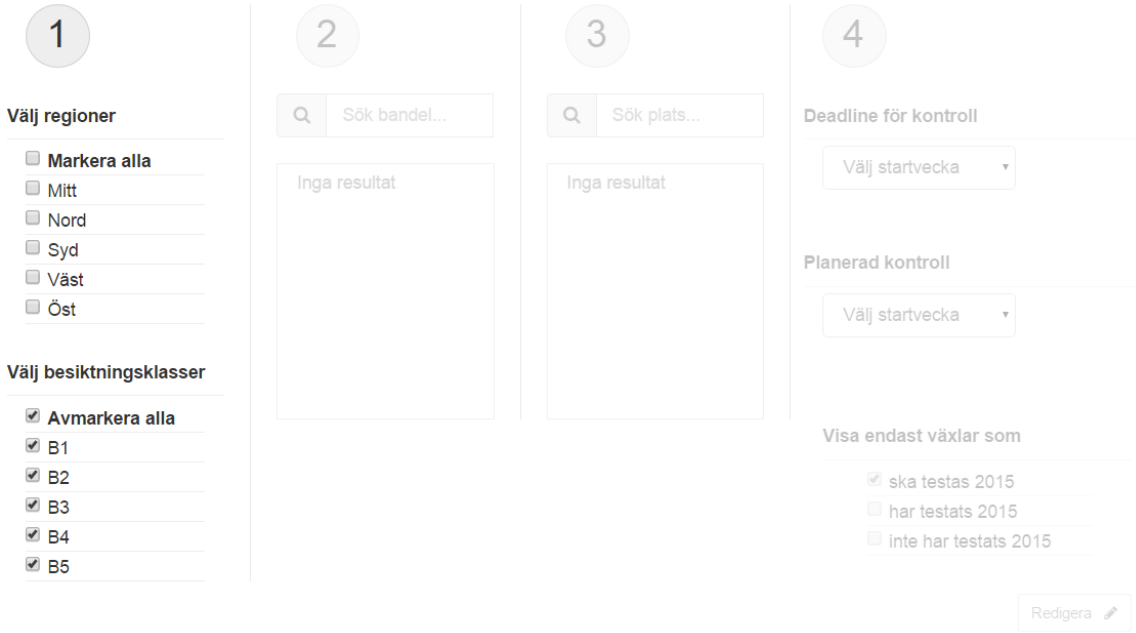
Dessutom fick systemet en omfattande uppdatering av användargränssnittet. Många av dessa ändringar baserades kring att förbättra användbarheten och att göra presentation av data och funktionaliteter mer överskådligt och lättare att förstå. Under avsnitt 5.5.1 – Användbarhet nedan nämns och visas de ändringar som skulle bidra till just det.

5.5.1 Användbarhet

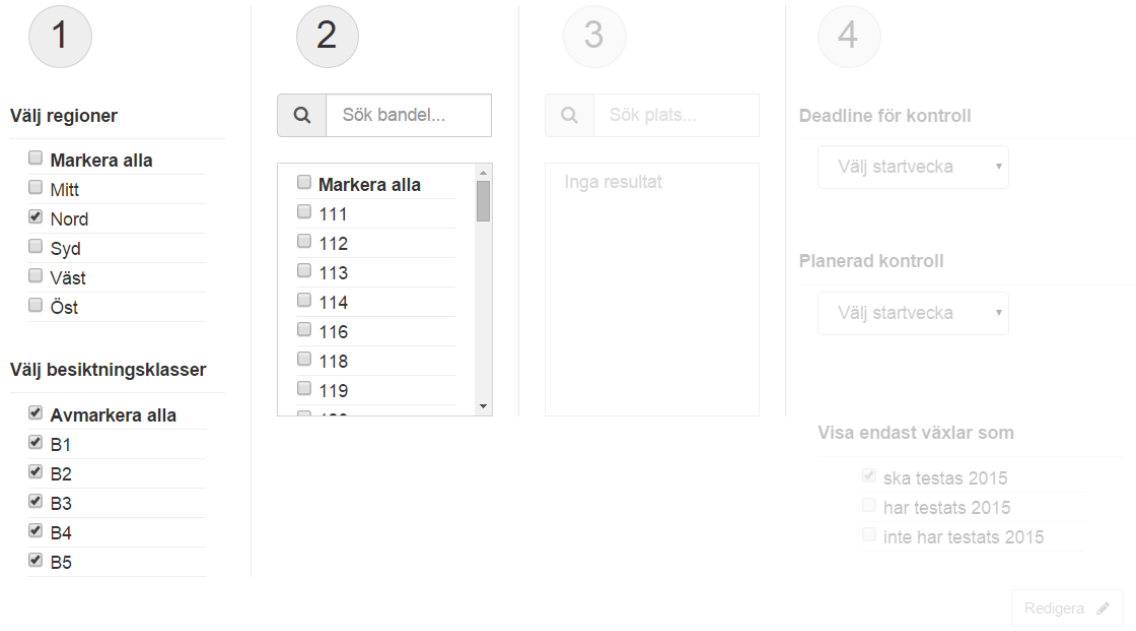
För att få ett användbart system skulle det bland annat vara lättförståeligt, tillgängligt, enkelt att använda och dynamiskt. Systemet skulle också ersätta den tidigare arbetsmetod som brukats av TSAB. AngularJS är ett typiskt exempel vad gäller lätthanterlig kod men det ses också som ett bra verktyg för att ge ett webbverktyg responsivitet och dynamik. Systemet innehåller så lite knappar och instruktioner som möjligt, till exempel innehåller gränssnittet inga ”Spara”-knappar eller liknande. Många av de implementeringar som nämns och nämnts om användbarhet baseras på inspiration från Krugs "Don't make me think" [1].

Sök

Här infördes ändringar som gav användaren tydligare instruktioner över hur filtreringen skulle genomföras. Genom att dela in filtret i fyra olika numrerade fält, där val i första fältet måsta göras för att det andra fältet ska synas, kunde flödet som nämns i prototypen tydliggöras. Figur 5.8 och 5.9 visar tydligt hur sidan ser ut när den hämtas och hur ett regionval i första fältet låser upp andra fältet.



Figur 5.8 Figuren visar det första steget i filtreringen på sidan *Sök*.



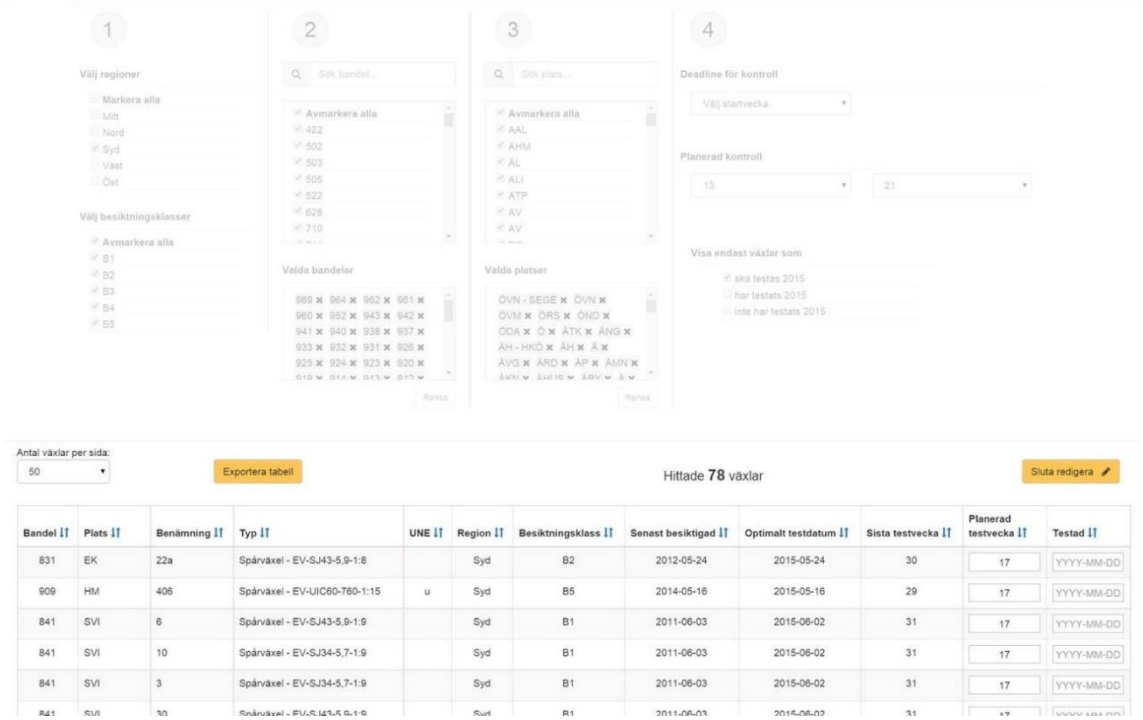
Figur 5.9. Figuren visar det andra steget i filtreringen på sidan *Sök* när en region valts.

Först när fält 1, 2 och 3 har valts kan ett resultat visas. Fält 4 visas sist och innehåller extra filtreringar som inte är nödvändiga för ett resultat.

Under sidan *Sök* lades även två nya funktionaliteter till. Två knappar visas nu när en filtrering genomförts. Den ena heter *Exportera tabell* och gör just det, exporterar den tabellen som användaren filtrerat fram. Användaren får då en fil av typen .xlsx (Excel-fil) innehållande den data som användaren filtrerat fram. Denna funktionalitet efterfrågades av TSAB för att kunna få ut en specifik växeltabell som enkelt skulle kunna vidarebefordras men också skrivas ut.

Prototypen fick aldrig funktionalitet för att manuellt sätta en växel som testad eller planera växlar, något som inte prioriterats tidigare. Sidan *Sök*s filter sågs som ett kraftfullt gränssnitt, inte bara för att söka upp specifika växlar utan också för att hitta större samlingar med liknande parametrar. Därför infördes här en till knapp. Knappen fick namnet *Redigera.* När användaren klickar på den ändras tabellens kolumner *Planerad testvecka* och *Testad* (se figur 5.10) längst till höger så att de går att redigera. Detta för att användaren enkelt ska kunna söka upp en specifik växel eller en samling växlar som hen vill planera eller som hen vill sätta som testade.

När användaren trycker på knappen *Redigera* hamnar tabellen i redigeringsläge. Detta betyder att hen kan skriva in och redigera de två kolumnerna till höger manuellt. Väl i redigeringsläge avaktiveras filtret. Detta låser också vilka växlar som visas i tabellen. När användaren gjort sina ändringar, kan hen trycka på samma knapp igen, som nu heter *Sluta redigera.* Filtret aktiveras igen och fälten *Planerad testvecka* och *Testad* går inte längre att redigera.



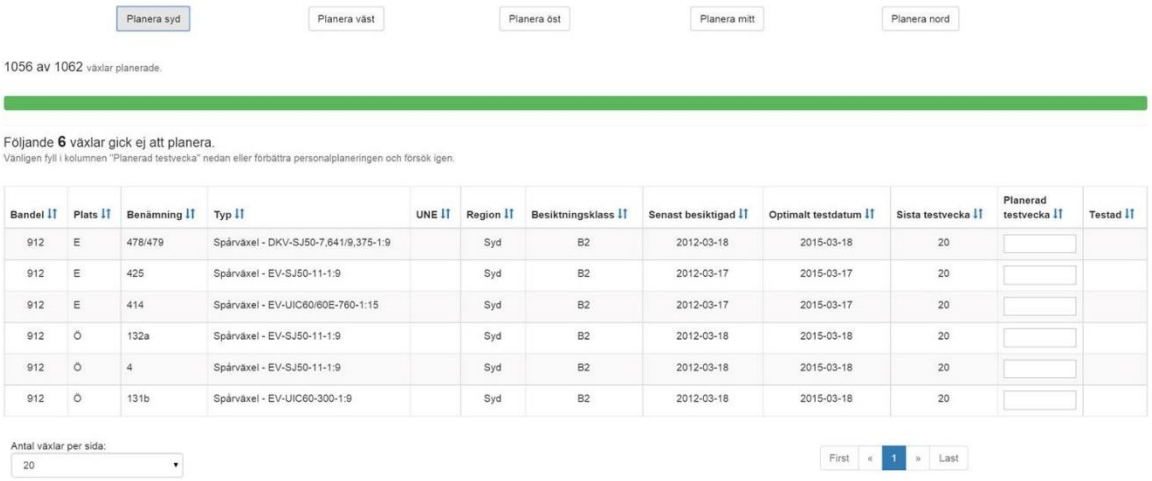
Figur 5.10. Visar sidan *Sök* i slutprodukten i redigeringsläge. Då redigeraknappen trycks in avaktiveras filtret och planerad vecka och testdatum kan ändras.

Då mycket fokus lades på användbarhet och att få användaren att känna sig bekväm med slutprodukten infördes en funktion för att enkelt fylla i flera celler. Användaren kan fylla i en cell för att sedan, likt i Excel, klicka på cellen och dra uppåt eller nedåt över andra celler för att fylla dessa med samma värde.

Planera växlar

Prototypens sida *Planera växlar* innehåller inte mer än fem knappar, en för varje region. När användaren trycker på knappen görs den automatiska planeringen. Den funktionaliteten finns fortfarande kvar i slutprodukten, dock infördes en visualisering för att ge användaren förståelse över vad som händer. En stapel laddas upp och antalet växlar som planeras räknas upp med tiden som algoritmen genomförs, se figur 5.11. Ifall algoritmen inte lyckas planera alla växlar får användaren möjlighet att skriva in de oplanerade växlarnas testvecka manuellt, se figur 5.12. Antal lyckade planeringar redovisas ovanför stapeln.

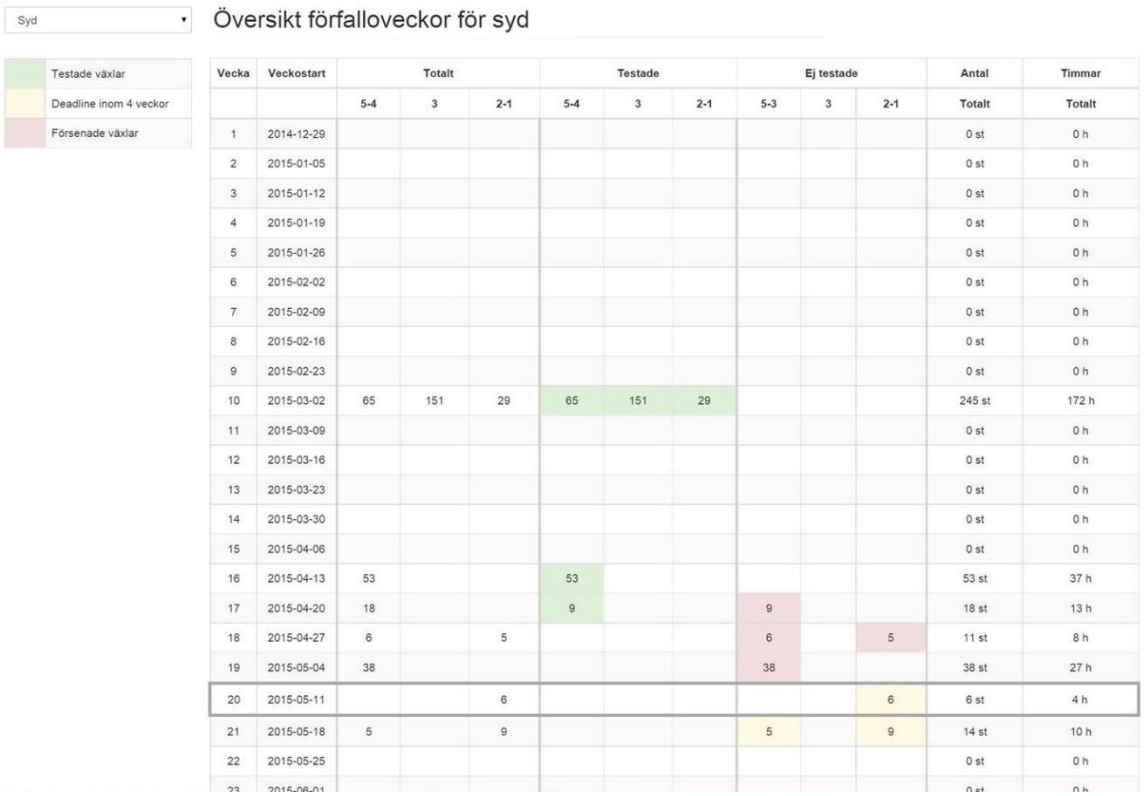
Figur 5.11. Skärmbilden visar hur den automatiska planeringen genomförs.



Figur 5.12. Bilden visar resultatet av en planering med försenade växlar.

Översikt förfalloveckor

I slutprodukten ändrades sidan *Översikt förfalloveckor*s utformning omfattande. När sidan laddas in möts man av liknande resultat som figur 5.13 visar, där samtliga regioner summeras. När man vill se information för en specifik region väljs regionen i en dropdown-meny till vänster om tabellen. Då visas de bekanta kolumnerna *totalt, testade* och *ej testade* från prototypen. En sådan uppdelning gör även här det lättare att förstå, samtidigt som man fortfarande kan få upp en översikt för samtliga regioner.



Vecka	Veckostart	Totalt			Testade			Ej testade			Antal	Timmar
		5-4	3	2-1	5-4	3	2-1	5-3	3	2-1	Totalt	Totalt
1	2014-12-29										0 st	0 h
2	2015-01-05										0 st	0 h
3	2015-01-12										0 st	0 h
4	2015-01-19										0 st	0 h
5	2015-01-26										0 st	0 h
6	2015-02-02										0 st	0 h
7	2015-02-09										0 st	0 h
8	2015-02-16										0 st	0 h
9	2015-02-23										0 st	0 h
10	2015-03-02	65	151	29	65	151	29				245 st	172 h
11	2015-03-09										0 st	0 h
12	2015-03-16										0 st	0 h
13	2015-03-23										0 st	0 h
14	2015-03-30										0 st	0 h
15	2015-04-06										0 st	0 h
16	2015-04-13	53			53						53 st	37 h
17	2015-04-20	18			9			9			18 st	13 h
18	2015-04-27	6		5				6		5	11 st	8 h
19	2015-05-04	38						38			38 st	27 h
20	2015-05-11			6						6	6 st	4 h
21	2015-05-18	5		9				5		9	14 st	10 h
22	2015-05-25										0 st	0 h
23	2015-06-01										0 st	0 h

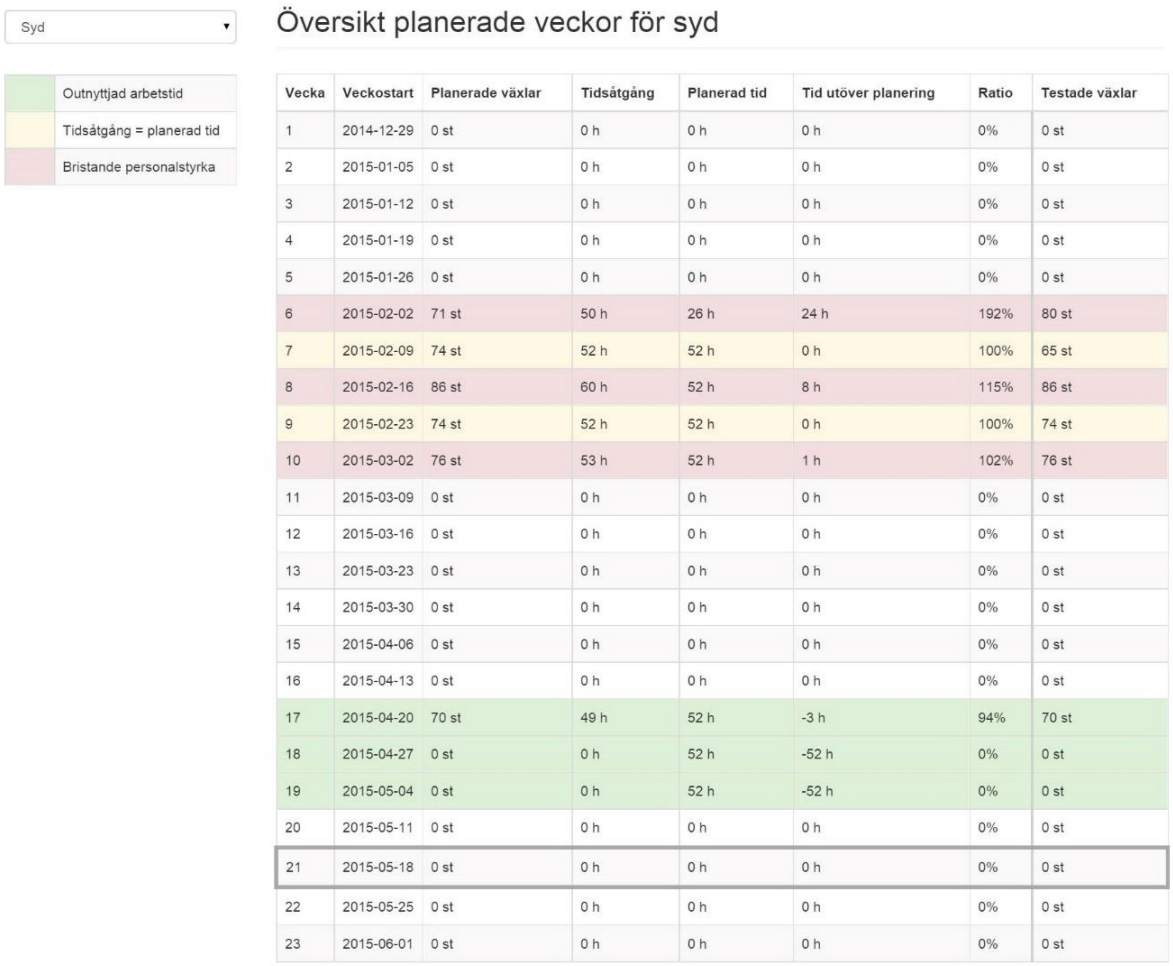
Figur 5.13. Figuren visar sidan *Översikt förfalloveckor* som den ser ut i slutprodukten.

5.5.2 Andra tillämpningar

Det var inte bara användbarheten som slipades till i slutprodukten. Några extra tillämpningar som tidigare inte hunnits med lades till i systemet. Dessa beskrivs nedan.

Översikt planerade veckor

En av de nya funktionerna som implementerades i slutprodukten är en översikt som kombinerar personalplaneringen med växelplaneringen, kallad *Översikt planerade veckor*. Här kan användaren se hur många växlar som är planerade, antalet timmar det skulle ta att utföra arbetet i teorin samt hur många timmar som är planerade för växelbesiktning i personalplaneringen. Med hjälp av dessa parametrar kan man även visa hur många timmar som fattas samt hur stor del av de planerade timmarna som faktiskt är uppbokade. Då de som utför kontrollerna har uppföljningsmöte kan det vara aktuellt att enkelt visa en summering över hur utfallet av planeringen faktiskt blev. Därför finns även antalet växlar som faktiskt blev testade per vecka med i tabellen som syns i figur 5.14.



Vecka	Veckostart	Planerade växlar	Tidsåtgång	Planerad tid	Tid utöver planering	Ratio	Testade växlar
1	2014-12-29	0 st	0 h	0 h	0 h	0%	0 st
2	2015-01-05	0 st	0 h	0 h	0 h	0%	0 st
3	2015-01-12	0 st	0 h	0 h	0 h	0%	0 st
4	2015-01-19	0 st	0 h	0 h	0 h	0%	0 st
5	2015-01-26	0 st	0 h	0 h	0 h	0%	0 st
6	2015-02-02	71 st	50 h	26 h	24 h	192%	80 st
7	2015-02-09	74 st	52 h	52 h	0 h	100%	65 st
8	2015-02-16	86 st	60 h	52 h	8 h	115%	86 st
9	2015-02-23	74 st	52 h	52 h	0 h	100%	74 st
10	2015-03-02	76 st	53 h	52 h	1 h	102%	76 st
11	2015-03-09	0 st	0 h	0 h	0 h	0%	0 st
12	2015-03-16	0 st	0 h	0 h	0 h	0%	0 st
13	2015-03-23	0 st	0 h	0 h	0 h	0%	0 st
14	2015-03-30	0 st	0 h	0 h	0 h	0%	0 st
15	2015-04-06	0 st	0 h	0 h	0 h	0%	0 st
16	2015-04-13	0 st	0 h	0 h	0 h	0%	0 st
17	2015-04-20	70 st	49 h	52 h	-3 h	94%	70 st
18	2015-04-27	0 st	0 h	52 h	-52 h	0%	0 st
19	2015-05-04	0 st	0 h	52 h	-52 h	0%	0 st
20	2015-05-11	0 st	0 h	0 h	0 h	0%	0 st
21	2015-05-18	0 st	0 h	0 h	0 h	0%	0 st
22	2015-05-25	0 st	0 h	0 h	0 h	0%	0 st
23	2015-06-01	0 st	0 h	0 h	0 h	0%	0 st

Figur 5.14. Bilden visar hur sidan *Översikt planerade veckor* kan se ut för syd-regionen.

Översikt planerade veckor, figur 5.14, fick likt *Översikt förfalloveckor* en tjockare kant runt den aktuella veckan för att visa var man befinner sig tidsenligt. Även här infördes färgkodning för att förtydliga vissa saker för användaren. Grön betyder att en vecka har outnyttjad arbetstid. Gul markerar att antal arbetstimmar går jämt upp med schablontiden för de inplanerade växelbesiktningarna. Röd betyder att antalet planerade växlar är fler än vad det finns arbetstimmar till.

Ändrad definition av testad växel i databasen

Det mesta av databasimplementationen förblev som den är i prototypen. En ändring som dock infördes var ändring av definitionen av kolumnen *Tested* (se delkapitel 5.1 – Databashantering). I prototypen används en etta för att

ange en växel som testad och nolla om den inte är testad. Ändringen, som tillkom efter samtal med TSAB, resulterade i att kolumnen håller ett datum när växeln är testad och anges tom när växeln inte är testad. Det infördes också en ny kolumn, *Testedweek*, som innehåller veckan för växelns testdatum om växeln är testad, annars tom. Detta för att lättare hantera veckor i systemet.

5.6 Tillägg av manuell planering

I fas 4 planerades funktionalitet som inte enligt examensarbetarna ansågs kritiska för att besvara problemformuleringarna och bevisa att det var möjligt att ersätta planeringen med en automatiserad algoritm. För att systemet skulle bli användbart ansågs dock den manuella planeringen viktig att implementera här då algoritmen bland annat inte tar hänsyn till avstånd mellan driftplatser eller försenade växlar. Algoritmen förlitar sig också på en god personalplanering och att personalen är flexibel vad gäller interregionalt arbete.

5.7 Kodens struktur

Slutligen bör det summeras när, var och till vad de olika programspråken användes. Figur 5.15 redovisar hur kodstrukturen i systemet hänger ihop.

Figur 5.15. Figuren summerar kodens generella struktur.

Systemet kan ses som tre större områden. Gränssnittet, det användaren ser, är uppbyggt av CSS och HTML. När användaren interagerar med systemet, till exempel vid filtrering av växlar eller vid val av region i en översikt, hanteras det av AngularJS och således JavaScript. I de fall information ska hämtas från databasen anropar AngularJS en PHP-fil innehållande SQL-syntax som svarar med ett objekt läsbart för AngularJS. Denna information visas sedan för användaren i gränssnittet. AngularJS kan alltså ses som en länk mellan gränssnittet och data som användaren hanterar.

5.7.1 HTML/CSS

Samtliga sidors gränssnitt är skrivna i HTML och CSS. Bootstrap används för att strukturera upp elementen och underlätta utvecklingen.

5.7.2 AngularJS

AngularJS tillsammans med HTML är grunden i koden. Genom att kombinera dessa skapas till exempel tabeller enkelt på ett dynamiskt och lättöverskådligt sett. Då de flesta av systemets hemsidor är uppbyggda kring just tabeller och att åskådliggöra relativt stora mängder data är AngularJS ett kraftfullt verktyg. Förutom att göra front-end-utvecklingen lättare används AngularJS för att anropa back-end-scripten i form av PHP-sidor, vars uppgift är att kommunicera med databasen. Anropen sker med den inbyggda komponenten $http [6].

5.7.3 JavaScript

Då AngularJS är skrivet i JavaScript används det oundvikligen vid samtliga funktioner skrivna i samband med AngularJS, där ibland planeringsalgoritmen.

5.7.4 PHP

PHP används som länken mellan användargränssnittet och databasen. Vid hämtning och uppdatering av databasen används PHP med databastillägget *MySQLi*. *MySQLi* innehåller bland annat funktionen *query* som utför en SQL-fråga. Då säkerhet inte har varit fokus under utvecklingen har inte prepared statements, ett sätt att skydda sig mot försök att modifiera databasen, införts. Denna säkerhetsaspekt, tillsammans med inloggning för att skydda obehöriga användare att nå systemet, bör implementeras vid eventuell vidareutveckling.

5.7.5 SQL

Som tidigare nämnts har SQL tillsammans med databashanteraren MySQL valts som databasalternativ och används således vid samtliga uppdateringar och hämtningar samt för lagring av data.

6 Slutsats

Sammantaget bevisar examensarbetets slutprodukt att det är fullt möjligt att uppfylla de krav som ställdes av TSAB. Samtliga krav som TSAB ställde på systemet uppfylls till så hög grad att examensarbetet och slutprodukten kan ses som lyckade. Systemet kan ersätta nuvarande arbetssätt för hantering av växlar vad gäller planering och översikt. Utförandet av planeringen har gått från att vara ett arbete som pågår under flera veckor till att ta ett fåtal minuter med hjälp av den automatiska planeringen och manuell korrigering.

För att summera resultaten och redovisa slutsatser besvaras samtliga frågeställningarna som ligger till grund för upplägget och utförandet av examensarbetet i delkapitel 6.1 – Utvärdering av problemformuleringar nedan.

6.1 Utvärdering av problemformuleringar

De problemformuleringar som i examensarbetets början ställdes är de som styrt hela examensarbetets arbetsgång. Det är på grund av dessa som olika implementationer i systemet har baserats. För varje frågeställning som angavs i delkapitel 1.3 – Problemformulering sammanfattas dess tillhörande resultat samt de val som gjordes kring dem och varför här nedan.

Hur ska algoritmen för planeringen utformas för att säkerställa att alla komponenter besiktigas och att det görs i tid?

> Algoritmen ger aningar om vad som är möjligt att åstadkomma på förhållandevis kort tid. Den visar också hur det är möjligt att definiera och automatisera komplexa arbetsuppgifter, åtminstone till en viss grad. Detta leder till slutsatsen att det i praktiken är fullt möjligt att automatisera liknande system med liknande komplexitet.

Vilka programmeringsspråk är mest lämpliga vid utformandet av planeringsverktyget?

> För att skapa ett verktyg som var dynamiskt valdes, som tidigare diskuterats, AngularJS. Det konstaterades tidigt att ett programspråk vars kod kompilerades på användarens enhet var att föredra över

server-kompilerade språk då det annars skulle krävas att användaren skickade förfrågningar varje gång hen ville ändra sin sökning. Detta skulle orsaka en omladdning av sidan och även skapa en onödig komplexitet gällande att spara sökparametrar.

Då ingen av examensarbetets medlemmar hade arbetat med AngularJS tidigare krävdes mycket inläsning för att komma över den relativt höga inlärningströskeln. När den var passerad visade AngularJS sig vara väldigt effektivt och har därmed bidragit till nya kunskaper inom dynamisk och modern webbutveckling för examensarbetets medlemmar.

Hur ska man utforma gränssnittet så att arbetet blir effektivare och snabbare, samtidigt som det ersätter det befintliga systemet utan att försvåra arbetet?

Den största fördelen med slutprodukten jämfört med arbetet som idag utförs av TSAB är att allting är samlat på samma plats. Att navigera på en hemsida som är skräddarsydd för planering och översikt för hur arbetet fortskrider är mer praktiskt än att arbeta med flera Excel-filer som ligger lokalt på datorn. Genom att arbeta i molnet kommer man också från problemet med att en användare av misstag kan radera en viktig fil.

För att inte försvåra arbetet för användaren togs beslutet av examensarbetarna att, där det fanns möjlighet, försöka efterlikna Excel. En del i det arbetet mynnade ut i "klicka och dra"-funktionen för datum då besiktning genomförts och manuell planering av testvecka.

Vilken databashanterare tillgodoser bäst systemets behov?

Efter analys av uppdragsbeskrivningen visade sig att datahanteringen kunde begränsas till två intressanta områden, växlar och personaltimmar. Då TSAB inte hade några som helst kvalitetskrav lades inte mycket tid åt att hitta den bäst lämpade databashanteraren. MySQL bedömdes enligt examensarbetarna vara en fullt fungerande kandidat för hantering av data. Vid de första implementeringarna, framförallt av listan med växlar, testades responstiden för att säkerställa MySQLs trovärdighet. Avsnitt 5.1.1 – Växellistan

redovisar dessa resultat. Effektiviteten hos en framtida server kan anses uppnå de krav som ställts på databashanteringen för systemet. Dessutom är MySQL en av de mest använda och beprövade inom databashantering vilket också motiverar varför den valdes.

Hur ska systemet utformas så att operatörerna på ett enkelt sätt kan redovisa genomförd besiktning?

Frågan diskuterades med Mattias på TSAB huruvida det skulle skapas en ny sida för att markera en växel som testad. Det gick dock att argumentera för att redigeringen bör ske på *Sök*-sidan då där redan fanns funktionalitet för att välja ut en väldigt specifik samling växlar. Resultatet blev som nämnts tidigare i rapporten; att användaren kan välja att redigera ett filtrerat resultat, och sen enkelt klicka och dra för att automatiskt fylla fler celler i tabellen.

6.2 Utvärdering av avgränsningar

Nedan diskuteras de avgränsningar som beskrevs i delkapitel 1.4 – Avgränsningar och hur deras eventuella implementering i systemet kan motiveras.

Prioritera funktionalitet före säkerhet.

Att införa funktionen inloggning kan vara en av de viktigaste implementeringar för att upprätthålla säkerhet. Dock genomfördes aldrig detta, då fokus lades på optimering av funktionaliteter som berörde användarvänligheten eller den bakomliggande datahanteringen. Inloggning är viktigt för hindra obehöriga att få tillgång till systemet, men också för att ge olika användaren olika typer av rättigheter. Detta för att hindra operatörerna att komma åt sidor de inte ska kunna använda.

Systemet kan alltså inte ses som helt funktionsenligt utifrån en säkerhetssynpunkt. Om servern för systemet istället ligger skyddad i ett lokal nätverk utsätts inte systemet för samma risk, och kan som det ser ut nu brukas säkert. Dock är detta inte den tänkta lösningen och inloggningsfunktionen är en av de viktigaste delarna i en framtida vidareutveckling.

Endast implementera en webbversion av systemet.

Syftet med systemet var att ge både TSAB och operatörerna ett bättre, specialanpassat verktyg som gör deras jobb enklare. Som tidigare nämnt lades dock mycket fokus på att få in de funktionaliteter som ansågs viktiga, och mindre fokus lades på säkerhet och anpassning till andra plattformar. Möjligheten för en operatör att kunna gå in från sin smartphone eller surfplatta ute på fältet är fullt möjligt idag, men webbgränssnittet är dock inte anpassat för mobila enheter. Om systemet tas emot väl och används av operatörerna kan en mobil applikation komma på tal. Dessutom kan då mer fokus läggas på att optimera gränssnittet att fungera på flera olika skärmstorlekar och plattformar.

7 Framtida utvecklingsmöjligheter

TSAB har uttryckt stort intresse, inte bara för vad produkten vill åstadkomma, men också för att den vidareutvecklas och förbättras, både kvalitativt och kvantitativt. Dessa summeras i nedanstående delkapitel 7.1 – Förbättringar och 7.2 – Tillägg. Förbättringar beskriver funktioner som finns i slutprodukten men som behöver förändras innan lansering. Tillägg beskriver funktioner som ännu ej är implementerade.

7.1 Förbättringar

Förbättring av felhantering i allmänhet.

Som tidigare nämnts i rapporten har fokus för slutprodukten legat på att implementera nödvändig funktionalitet och att göra de funktionerna användarvänliga. Detta har bidragit till en mindre noggrann testning av systemets funktioner vilket bör åtgärdas ifall systemet ska lanseras och användas.

Hantera mätvagnens utslag och justera planeringen.

Ett efterfrågat tillägg från TSAB är att få algoritmen att ta hänsyn till mätvagnens utslag (suspects). Dessa utslag ska verifieras likt en växelbesiktning av operatörerna. Då verifieringarna alltid ska prioriteras först kan dessa resultera i förflyttningar av växelkontroller för att göra plats för verifieringar. På samma sätt kan det leda till att det finns tid över till andra arbetsuppgifter om antalet suspects blev färre än väntat.

Förbättra systemets säkerhet.

Som nämnts tidigare bör systemets säkerhet förbättras avsevärt för att anses helt funktionsdugligt. Det skulle innebära stöd för inloggning och rättighetshantering, men också stöd för prepared statements vid kontakt med databasen.

7.2 Tillägg

Läs in förändringar hos växlarna direkt från Trafikverket.

Försök gjordes att få tillgång till Trafikverkets växeldatabas. De hade dock inget direkt stöd för hämtning av dessa, som exempelvis ett API som kan användas för att hämta data från en utomstående databas. Som tidigare nämnts eftersträvades att få ett system och en databas som speglar verkligheten på ett så korrekt sätt som möjligt, och att införa så att växlarna direkt uppdateras från Trafikverket är en stor del för att uppfylla detta krav.

Stöd för hantering av samtliga typer av besiktningar.

TSAB vill också ha stöd för hantering av den besiktning som också måste göras på AHSP (avvikande huvudtågspår) och kryss. Något som görs av operatörerna men som inte har lika hög prioritet för de lägre besiktningsklasserna. TSAB nämner att en liknande planering behövs för dessa spårdelar.

Spara historik för en växel.

Önskemål om att spara en växels historik finns också. Det behövs någon form av verktyg som kan redovisa vilka problem som uppstått med en viss växel, när och av vem den besiktigats, och annan intressant information. Detta kan exempelvis användas för att se om en växel kanske bör bytas ut helt om den vid varje kontroll haft samma fel.

8 Referenser

[1] Krug, Steve. (2014). Don’t make me think, Revisited. New Riders.

[2] Padron-McCarthy och Risch. (2014). Databasteknik. Studentlitteratur AB.

[3] Riksdagen.se. 2015. Förordning 2010:185 med instruktion för Trafikverket. [online] http://www.riksdagen.se/sv/Dokument-Lagar/Lagar/Svenskforfattningssamling/Forordning-2010185-med-inst_sfs-2010-185/?bet=2010:185 (Hämtad 2015-05-11).

[4] Banverket.se. 1997. Handbok rörande Banverkets växeldriv. [online] http://pwidastreamerext.banverket.se/StreamService/StreamDocument.ashx?AppKey=ce01c31b-0d2a-4247-a219-c4465dbded51&DocumentKey=6cc790ad-5713-4db8-9d4a-399da0218e6a (Hämtad 2015-05-11).

[5] AngularJS. AngularJS API Docs. 2015. [online] https://docs.angularjs.org/ (Hämtad 2015-03-19).

[6] AngularJS. API: $http. 2015. [online] https://docs.angularjs.org/api/ng/service/$http (Hämtad 2015-03-19).

[7] Bootstrap.com. CSS. http://getbootstrap.com/css/ (Hämtad 2015-03-10).

[8] PHP.net. Language Reference. 2015. [online] http://php.net/manual/en/langref.php (Hämtad 2015-03-10).

[9] PHP.net. MySQLi - Manual. 2015. [online] http://php.net/manual/en/book.mysqli.php (Hämtad 2015-03-10)

[10] WDG Web Design Group. HTML 4 Reference. http://www.htmlhelp.com/reference/html40/ (Hämtad 2015-04-13).

[11] Mozilla Developer Network. JavaScript reference. 2015. [online] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference (Hämtad 2015-03-19).

[12] Microsoft Support. Developer Reference for SQL Server 2014. [online] https://msdn.microsoft.com/en-us/library/dd206988.aspx (Hämtad 2015-03-12).

[13] IBM.com. DB2 for Linux, UNIX and Windows. 2013. [online] http://www-01.ibm.com/software/data/db2/linux-unix-windows/features.html (Hämtad 2015-03-12).

[14] IBM Knowledge Center. Isolation levels. [online] http://www-01.ibm.com/support/knowledgecenter/SSEPGG_9.7.0/com.ibm.db2.luw.admin.perf.doc/doc/c0004121.html (Hämtad 2015-03-12).

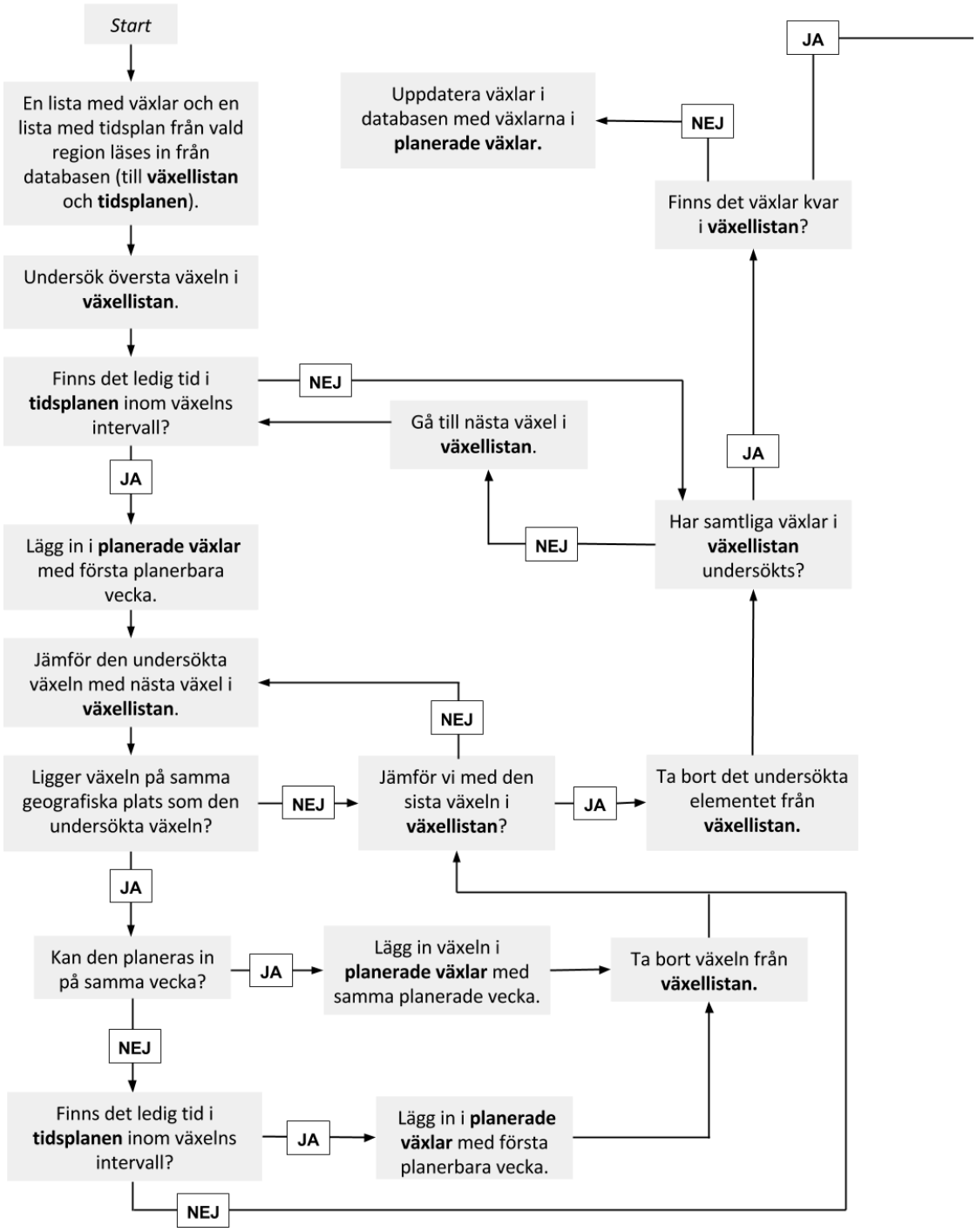
[15] Microsoft Developer Center. 2012. [online] https://msdn.microsoft.com/en-us/library/office/dn142571.aspx (Hämtad 2015-03-12).
[16] dev.mysql.com. MySQL 5.6 Reference Manual. 2015. [online] http://dev.mysql.com/doc/refman/5.6/en/index.html (Hämtad 2015-03-12).
[17] Oracle Help Center. Database Concepts. 2015. [online] http://docs.oracle.com/cd/B19306_01/server.102/b14220/toc.htm (Hämtad 2015-03-12).
[18] infoq.com. Kanban and Scrum - making the most of both. 2009. [online] http://www.infoq.com/minibooks/kanban-scrum-minibook (Hämtad 2015-03-20).
[19] B. B. Agarwal, S. P. Tayal, M. Gupta (2009). Software Engineering and Testing. Infinity Science Press.
[20] Stack Overflow [online] http://stackoverflow.com (Hämtad 2015-06-21).
[21] Gustafsson, Mattias. Verkställande direktör, Tirawa Solutions AB.

9 Terminologi

Term	Definition
Användaren	Den personen som använder systemet. Det förutsätts att användaren har förståelse över växlar, hur de besiktigas och av vem. Användaren vet också vad de olika parametrarna för varje växel står för och har goda kunskaper kring det svenska järnvägsnätet.
Bootstrap	Ett ramverktyg för HTML, CSS och javascripthantering.
Databas	Den samling data som tillhör systemet och hanteras av databashanteraren.
Databashanterare	Hanterar läsning/skrivning till systemets databas.
Excel-filen	Den fil som fås av TSAB. Den innehåller samtlig information om växlarna. Den måste importeras till systemets databas. Filen är egentligen en rådatafil av typen CSV, som är exporterad från en Excel-fil.
Operatör	En person som arbetar ute på fältet och har i uppgift att underhålla tågrälsen. En operatör genomför bland annat besiktningar av växlar och verifieringar av misstänksamma mätpunkter.
Planeringsalgoritmen/ automatiska planeringen	Den algoritm som genomför den automatiska planeringen av växlar.
Systemet	Beskriver webbgränssnittet, databas/databashanteraren och programmeringskoden som helhet.
TSAB	Förkortning för Tirawa Solutions AB.
Webbgränssnittet	Beskriver det gränssnitt som användaren interagerar med.
Prepared statement	Används för att skydda systemet mot försök att modifiera databasen.

Bilaga A1 – Flödesdiagram del 1

Figuren beskriver ett flödesdiagram över första delen av planeringsalgoritmen.
Flödet inleds i *start*-rutan. Algoritmens andra del fortsätter på nästa sida.



Bilaga A2 – Flödesdiagram del 2

Figuren beskriver ett flödesdiagram över andra delen av planeringsalgoritmen.