Vecka 1 (v.2)

Denna veckan har jag gjort stora framsteg med Romas[1] testresultat-sida. Fram tills nu har dess innehåll bestått av statiska exempel data medan vi utvecklat gränssnittet och backend lösningen. Men nu när vi har en fungerande backend lösning så tog jag på mig uppgiften att implementera den i gränssnittet. Inledningsvis fick jag skapa funktioner som skulle hantera HTTP requesterna till våran backend. När det var klart kunde jag börja rendera datan som hämtades. Tack vare exempel datan vi hade baserat gränssnittet på fungerade stora delar nästan direkt. Men det visade sig att alla testresultat inte hade riktigt samma struktur och data inkluderad. Det fanns flera anledningar till detta, bland annat att Plejd hade ändrat strukturen sedan något år tillbaka på hur deras databas objekt såg ut. Detta betydde att jag behövde uppdatera vårt gränssnitt så att det hade stöd för det nuvarande strukturen samt den gamla, då denna datan också var viktig att kunna se.

Därefter jobbade jag med att förbättra användarupplevelsen på testresultat-sidan. Exempelvis implementerade jag en laddsymbol för att ge våra användare en visuell återkoppling att deras begäran har kommit fram och bearbetas. Jag lade också till några meddelanden för att förtydliga vad som händer, exempelvis om inga testresultat hittas eller om något i backenden skulle gått snett.

I övrigt jobbade jag också en del med att förbättra sidans UX i sin helhet. Bland annat så uppdaterade jag designen på vår explanation modals[2]. Jag uppdaterade färgen på flikarna för att göra det tydligare när en flik är aktiv och inte, så användaren vet vilken flik som visas just nu. Pop-upens animation var också ostabil och lite laggig, därför uppdaterade jag den också genom att ersätta den nuvarande Svelte Transitions animationen med en standard CSS animation. Detta förbättrade den totala prestandan hos pop-upen och gjorde det mycket finare att titta på.

Avslutningsvis, har jag också uppdaterat sidans navbars design. Detta för att göra navbaren mer konsekvent med resten av sidans design. Dessutom har jag lagt till en funktion för automatiskt stängning efter navigation när navbaren är i fullskärmsläge (behövligt på mindre skärmar, såsom mobiler), vilket förbättrar navigerings flödet och gör det lättare för användaren att fokusera på innehållet på den nya sidan.

Vecka 2 (v.3)

Under denna veckan har jag jobbat mycket med Romas select-komponents[3] funktionalitet och utseende. Komponenten har en sökfunktion som för att lättare hitta ett specifikt alternativ som strulade lite grann. Efter en kortare felsökning kom jag fram till att jämförelsen var "case-sensitive" och detta gjorde att jämförelsen inte såg stor och liten bokstav som samma karaktär, vilket inte var beteendet vi var ute efter. För att få beteendet vi var ute efter uppdaterade jag jämförelsefunktionen till att omvandla alla bokstäver till samma "casing" vilket inte var något större utmaning. Utöver att fixa sökfunktionen uppdaterade jag också utseendet av komponenten genom att minska storleken och uppdatera färgerna för att den skulle passa med resten av sidan mer.

Sedan fixade jag också en bugg med testresultat-sidas resultat-tabell. Tabellens rubrikrad har en skugga som animeras fram när man börjar scrolla nedåt i tabellen (för att skilja rubrikerna från resultaten) men det blev problem när tabellen scrollade tillbaka upp och nådde toppen väldigt snabbt. Det som hände då var att skuggan "blödde" fram på rubrik radens ovansida, vilket jag inte tyckte om. För att lösa detta fick jag redigera skugg animationen samt lägga till en buffertzon för när skuggan skulle börja/sluta visas. Den uppdaterade skugg animationen gjorde att skuggan försvann snabbare än det började visades och i kombination med buffertzonen som berättade när skuggan skulle börja försvinna hinner nu skuggan sluta visas innan tabellen scrollat hela vägen upp till toppen.

Avslutningsvis spenderade jag också en del tid på att uppdatera versionerna på våra NPM dependencies då vissa av dem blivit flaggade att ha säkerhetsbrister. Processen jag gick igenom för varje dependency var att kolla om nästa version hade "patchat" problemet eller om jag skulle behöva kolla ännu längre fram. Sedan behövde jag också kolla om dependencyns funktionalitet hade ändrats på något sätt så att det inte längre skulle fungera med våran implementation och i så fall uppdatera våran kod också.

Vecka 3 (v.4)

Även denna veckan spenderade jag mestadels min tid på att förbättra användarupplevelsen av Roma. Av en ren slump upptäckte vi att vår logotyp, som är gjord i SVG med tillhörande CSS, inte såg ut som den skulle i webbläsare baserade på WebKit (exempelvis Safari). Efter en stunds letande kom jag fram till att problemet låg i vår användning utav CSS grid för att centrerar våran logotyp. Det visade sig att WebKit inte hanterade CSS grid inom SVGer på samma sätt som andra webbläsarmotorer (exempelvis Gecko eller Blink). Lösningen jag kom på använde sig istället utav Flexbox, även om implementationen inte blev precis likadan så kunde den ändå uppnå resultatet som vi var ute efter (även i WebKit!).

Därefter fixade jag med sidans filtersektion[4] då dess animation för att öppnas/stängas var väldigt laggig och inte såg så rolig ut. Svelte Transitions implementationen för att animera sektionen upp respektive ner igen behövde alltså bytas ut, jag bestämde mig för att skapa en ny med CSS istället. Resultatet av den nya implementationen blev mycket bättre än den gamla, långt över förväntan. Animationen var mycket jämnare och mjukare vilket gjorde mycket till hur man upplevde sidan när man använde den. Dessutom tror jag att prestandan blev något bättre med den nya implementationen, men det är inget jag är helt säker på då jag inte utförde några tester innan eller efter ändringen. Men det kändes som att sidan presterade bättre!

Vecka 4 (v.5)

I slutet av förra veckan fick vi en förfrågan om vi Platos[5] API var tillgänglig att använda i det fallet man ville automatisera skapandet av frågor. Detta var inget vi tänkt på att någon skulle vilja göra, men som tur var hade vi utvecklat det på ett sätt som gjorde det möjligt! Det visade sig dock att våran dokumentation för APIn inte var uppdaterad och saknade viss information medans annan saknades helt. Så för att förbereda APIn för användning utan webbgränssnittet fick jag spendera en del av veckan till att uppdatera all API dokumentation. Detta gjorde jag genom en kombination utav att kolla på källkoden och att faktiskt testa att använda APIn själv.

Sedan spenderade jag också en del tid på Romas SPC-sida[6] i förberedelse för när den ska börja implementeras med riktig data. Exempelvis upptäckte vi att filtrena på sidan inte riktigt stämmer, vissa saknades och andra var inte konfigurerade riktigt rätt. När du valde vilken enhetstester du ville se så kunde du välja flera samtidigt (en funktion vi vill ha i Roma, men inte på just denna sidan) och att välja en specifik resultatgrupp gick inte, eftersom filtret saknades helt och hållet. Efter jag löst detta gick jag vidare till nästa uppgift på SPC-sidan, att implementera intervall annoteringar på sidans grafer. Annoteringarnas uppgift är att visa användaren var den övre och undre gränsen går för acceptabla värden i grafen. Då graferna är gjorde med ChartJS använde jag mig utav ett av deras plugin för att lösa annoteringarna visuellt, då den faktiska datan inte är redo än kunde jag inte implementera att den faktiskt sätter ut de riktiga värdena och fick nöja mig med temporära fixerade så länge.

Vecka 5 (v.6)

Denna veckan började jag med att skriva dokumentation för Roma. Roma använder sig av ett annat internt projekt (en API) kallat Kraftwerk för att hämta mycket av datan som visas i webbgränssnittet, och detta görs via protokollet Nats för att hålla en högre säkerhetsnivå. Min uppgift var att dokumentera vilka Nats ämnen (topics) som Roma använder sig av och varför de är nödvändiga. Nats ämnen delar in all data i mindre delar och det är endast konton med tillstånd till dessa specifika ämnen som kan hämta datan de håller. Därför behövde jag kolla upp vilka det är vi använder och vilka av dessa vi faktiskt behöver i förberedelse för när vi lanserar produkten så använder vi ett konto som inte har tillgång till fler ämnen än vad det behöver.

Utöver uppdatering av dokumentationen så har jag gjort en del små fixar på Roma. Exempelvis justerade jag storlekarna (avstånd, textstorlek, osv.) av flera av våra komponenter för att göra sidan mer konsekvent och användarvänlig. Sedan gjorde jag också lite förberedelser för Romas Yield-sida[7] i förberedelse för när denna sidan ska implementeras om några veckor.

Jag avslutade veckan med att planera en uppdatering av testresultat-sidan då vi inte riktigt var nöjda med hur den såg ut eller agerade. Processen blev att jag började med att skissa upp flera olika layouter på papper över hur sidan skulle kunna se ut. Därefter hade jag en presentation och diskussion med mina kollegor om vad som var bra och dåligt med de olika layouterna. Efter det började jag på en mockup av sidan i Figma där färger samt storlekar blev definierade tydligare än under skiss stadiet. Sedan hade jag återigen en presentation för mina kollegor där jag fick mer feedback som jag sedan använde för att uppdatera min mockup.

Vecka 6 (v.7)

Hela denna veckan spenderade jag på att implementera den nya testresultat-sidan som jag skapade en ny design för förra veckan. Då jag redan gjort designen i Figma hade jag en ganska bra idé över vilka delar som skulle behöva vara separerade i egna komponenter, hur händelseförloppet skulle se ut och hur den generella layouten var. Jag bestämde mig för att börja implementationer uppifrån och sedan jobba mig ner stegvis.

Det första jag gav mig på blev därför sidans huvud-/titelsektion. Denna delen av sidan har i uppgift att snabbt ge användaren en bild av vad det är för data hen kommer att se, samt hur testen har presterat i helhet. I mer detalj består sektionen av två korta texter, en knapp och två informationskort. De två korta texterna är det första som visas upp och de berättar vilken enhetstyp som är testad och under vilket tids span data som visas är hämtad från. Sedan har vi en knapp med nedladdningsalternativ (JSON och CSV) som består av flera komponenter i sig. Den första är knapp komponenten i sig, denna blev en av de första jag skapade då den skulle behövas på flera ställen. Sedan kunde jag med hjälp av knapp komponenten skapa nedladdning komponenten. Sedan avslutade jag huvud/titel komponenten genom att skapa informations korten. Båda korten blev sin egen komponent som baserades på en kort komponent som jag började med att skapa. Det första kortet visualiserade hur testen hade presterat genom att visa ett procenttal på lyckade test samt antalet lyckade gentemot totalen testade. Det andra kortet visade first och last pass yield.

Efter att den första delen av sidan var klar gick jag vidare till huvuddelen av sidan, test resultat tabellen. Tabellens uppgift är att visa varje individuellt test för sig så användaren lätt ska kunna se vad som gått rätt och fel. I den förra designen var denna delen av sidan också en tabell, men denna gången är designen av tabellen mer användarvänlig då raderna har blivit mer tydligt separerade. Hur du väljer vad som ska inkluderas i nedladdningen har också förtydligats, tidigare markerade du resultaten du ville ha genom att trycka på den (vilket resulterade i att de expanderade) och sedan lämnade dem öppna. Detta var inte bra av flera anledningar, för det första så var det inte så tydligt att öppna resultat betydde att de skulle laddas ner, sedan blev det också osmidigt att fortsätta använda tabellen när flera resultat var öppna samtidigt. Detta löste jag genom att lägga till en kolumn för varje rad med en dedikerad kryssruta där användaren kunde välja om resultatet skulle inkluderas i en eventuell nedladdning. I denna nya designen så expanderar inte heller tabellens rader längre för att visa ytterligare information om testresultaten utan istället så öppnas en modal.

Jag avslutade implementationen med buggfixar och en extra kontroll så alla komponenter var responsiva och fungerade som de skulle.

Vecka 7 (v.8)

Denna veckan blev mitt huvudfokus på att räkna ut data som vi ska använda på Romas yield-sida. Datan som ska visas på sidan är uträkningar av flera olika tester (beroende av vilka filter användaren valt) och hur deras resultat ser ut i en helhet. Då det är väldigt mycket data som behöver räknas på så ska det göras direkt i databasen, med hjälp av en MongoDB aggregations pipeline, och eftersom vi behöver hämta data från en av våra databaser så ska beräkningen göras i projektet Kraftwerk. En av anledningarna till varför vi inte hämtar från databasen direkt i Roma är för separering av funktionalitet och en annan är för att vi inte vill öppna upp hela databasen till användaren, utan bara de queries som är nödvändiga för dem.

Att skapa själva endpointen och topicen innebar inga större bekymmer utan var ganska lätt. Jag började med att konfigurera vilka adresser de skulle ha innan jag gick vidare till att definiera vilka parametrar som skulle godkännas. Detta gjorde jag genom att kolla över den inkommande datan och kolla så att alla fält som skickades in, var de jag godkänt skulle få komma in. Några av fälten var obligatoriska (ex. Enhetstyp, Start- och Slutdatum) så dessa var jag också tvungen att kolla så de faktiskt var med också. Sedan kollar jag över alla fälten för att se till så det är av rätt typ och inte kommer skapa problem senare. Om det är något som inte skulle stämma under dessa stegen så skulle anropet avslutas tidigt och användaren skulle få veta vad det är som gått snett.

När den första delen av min uppgift var klar bestämde jag mig för att passa på och skriva dokumentationen för endpointen och topicen. Dokumentationen inkluderar adressen för anropet samt in parametrar (namn, typer, osv.) och vad för respons som kan förväntas.

Den sista delen, själva aggregationen var faktiskt ganska krånglig att få till. Jag började med att konfigurera filtersektionen av pipelinen, denna delen behövde vara ganska dynamisk eftersom anropets filter inte alltid har samma värde. Det fanns också en del edge-cases som jag behövde ta hänsyn till, exempelvis så saknar äldre test resultat fältet TestType som indikerar vad för typ av test det är. Som väl är genomfördes endast test av en specifik typ under denna tiden vilket innebär att vi kan anta att värdet är denna typen när det saknas från ett dokument. Mycket mer än filtrerings delen av pipelinen hann jag inte med denna veckan då vi hade ett catch up-möte om Roma och sedan var veckan slut.

Vecka 8 (v.9)

Veckan började med att jag fortsatte med uträkningarna i Kraftwerk. Det var ganska komplicerat att få till det då det fanns flera krav utöver siffrorna jag skulle räkna ut. Pipelinen behövde klara av att göra uträkningarna på stora mängder dokument samtidigt som den behövde vara väldigt snabb eftersom det kommer sitta en användare i andra änden och vänta på resultatet. Efter många om och men hade jag lyckats skapa en pipeline som räknade ut 95% av all data som jag var ute efter och den var relativt snabb. Men dessa sista 5% var väldigt viktiga att få med och på grund av hur pipelinen såg ut fanns det tyvärr ingen uppdatering jag kunde göra utan att fundamentalt ändra pipelinen. Så för att lösa de sista 5% behövde jag också lösa de första 95% på ett nytt sätt. Efter en stunds jobb lyckades jag att räkna ut all data! Men då visade det sig istället att det inte klarade av lika stora dataset längre, alltså mängden dokument den kunde hantera samtidigt hade sjunkit drastiskt. Min nya lösning visade sig inte vara lika lätt viktig som den tidigare och krävde därför extrema mängder minne för att kunna köra, i vissa fall mer än vad MongoDB kunde erbjuda… så det var bara för mig att gå tillbaka till ritbordet och försöka komma på en ny lösning.

Jag började diskutera med en kollega hur vi skulle kunna gå tillväga för att lösa alla uträkningar utan att överbelasta databasen. Till slut kom vi fram till att begränsa uträkningarna lite grann genom att dela upp det. Tidigare räknade jag ut yield för en specifik produkts alla resultatgrupper och de individuella resultaten i varje grupp, detta är förstås väldigt mycket data så därför ändrade vi till att bara räkna på resultatgrupperna. Om man vill ha datan för de individuella testerna i resultatgrupperna måste man nu specificera en resultatgrupp att göra uträkningarna på.

Vecka 9 (v.10)

I slutet på förra veckan lade jag upp min endpoint/topic för att räkna ut yield som en merge request, detta för att få avslutande feedback innan den kan officiellt börja användas. Så under flera dagar denna veckan så har jag fått feedback på små saker som jag kunde ändra för att förbättra min kod samt om det var några fel så fick jag också fixa till detta. Det visade sig exempelvis att hur jag räknade ut en siffra inte var så kunden hade tänkt sig. Så detta fick jag ändra, även om det jag gjort inte var fel så var det inte det som de hade tänkt sig. Vi bestämde oss också för att räkna ut ytterligare några siffror under denna veckan så detta lade jag till också. Lite senare i veckan var merge-requesten klar och accepterad av våra seniora utvecklare och jag kunde publicera koden.

Därefter fortsatte jag implementera yield genom att skapa en endpoint i Romas backend som kopplar vidare till Kraftwerk topicen jag precis fått publicerad. För att skapa denna endpointen behövde jag bland annat konfigurera adressen, lägga till validering av inkommande parametrar och skriva dokumentation.

Jag avslutade veckan med att implementera en funktion som sparade användarens valda filter lokalt så att nästa gång användaren skulle använda sidan så skulle den komma ihåg vad för filter hen tidigare valt.

Strax innan det var dags att ta helg och bege sig hemåt fick vi beskedet att Roma skulle byta namn till Spreed. Spreed är ett annat verktyg som används på företaget men detta har mindre funktionalitet än vad Roma och QRM (verktyget Roma skulle ersätta) har, men det är redan publicerat och har väldigt många användare. Tanken bakom namnbytet är att fånga upp många av användarna som redan använder Spreed när vi lanserar Roma. Funktionaliteten som redan finns i Spreed är därför något som vi kommer behöva implementera i Roma på sikt om det verkligen ska kunna byta ut Spreed också.

Vecka 10 (v.11)

Denna veckan fokuserade jag helt på att få klart yield funktionaliteten i Roma/Spreed. Det som var kvar att implementeras i detta steget var frontenden, alltså att hämta datan från vår backend samt att visa upp den i diverse grafer.

Jag började med att skapa graf komponenterna som skulle behövas. Sidan skulle bestå av fyra grafer totalt varav två skulle vara stapeldiagram och två linjediagram. Sedan skapade jag layouten på sidan utifrån de nya graf komponenterna samt andra tidigare komponenter vi redan hade skapat.

När sidan visuellt såg ut som den skulle började jag jobba med logiken bakom. Det första jag gjorde var att koppla ihop våra filter till sidan så att om användaren ändrar något av sidans filter så hämtar den riktigt data från vår backend. Sedan fixade jag så att den inkommande datan formaterades så den kunde användas i graferna. Nu var yield-sidan alltså funktionell.

Jag avslutade veckan med att försöka förbättra yield-sidan genom att sätta vissa begränsningar och fånga eventuella buggar som jag missat när jag skapade sidan tidigare under veckan.

Vecka 11 (v.12)

Mot slutet på denna veckan skulle vi ha ett möte där vi skulle visa upp Spreed för resten av webb- och data-teamet. Därför började vi veckan med att fixa iordning alla små saker som saknades för att utseendet skulle se så klart ut som möjligt. Vi jobbade också på att fixa alla kända buggar vi hade kvar och bestämde oss för att inte implementera någon ny större funktionalitet denna veckan. Detta för att det inte skulle plötsligt sluta fungera precis innan mötet (vilket vi redan hade erfarenhet av från ett tidigare möte).

Sedan var det dags för själva mötet där personer från både web- och data-teamen deltog. Målet med mötet var att visa upp vad vi hade gjort och ge de andra i mötet en bättre bild av vad det är vi jobbat med, samt att de får en chans att komma med förslag på förbättringar och dylikt. Mötet var väldigt lyckat, det vi valde att visa upp fungerade jättebra och de andra var väldigt imponerade.

Efter mötet satte jag och en person från data-teamet och stress testade Spreed. Detta gjorde vi genom att göra väldigt breda sökningar, många olika filter kombinationer, etc. Vi passade också på att leta efter eventuella buggar som vi annars missat och skriva ner eventuella ändringar vi vill göra.

Vecka 12 (v.13)

Denna veckan låg fokuset på att försöka fixa de fel som vi hittade i slutet på förra veckan. Vi hade skrivit upp en lista med saker som skulle ändras eller inte fungerade som det skulle, på listan fanns det allt från visuella buggar och logik fel till saker som vi kunde förbättra för att göra sidan mer användarvänlig.

Ett exempel på vad jag fixade var grupperingen av datan på vår yield-sida. Det visade sig nämligen att efter att vi hämtat yield datan till vår frontend och formaterat den till att fungera i våra charts att den ibland grupperades fel. Detta behövde jag åtgärda eftersom detta felet resulterade i att viss data inte visas som den skulle.

Vecka 13 (v.14)

Denna veckan gjorde jag en del små fixar på Spreed. Exempelvis saknades det en återställningsknapp på en av graferna på spc-sidan. Utöver det var det också vissa saker som var för stora på mindre skärmar, så detta fixade jag också.

Jag lade också till ett "outcome" filter i Kraftwerk för en av våra endpoints, detta för att ge användaren mer kontroll över vilken data det är som kommer att hämtas.

Vecka 14 (v.15)

Då Spreed börjar närma sig en första release så finns det inte så mycket kvar att implementera vid detta lag. Därför blev denna veckan framförallt en testnings vecka för att försöka hitta eventuella fel vi missat tidigare. Vi lyckades hitta några mindre fel och lyckades åtgärda dessa.

Vecka 15 (v.16)

Denna veckan fortsatte jag att testa Spreed för att hitta fler fel jag kunde fixa. Vi bokade också in ett sista möte inför releasen nästa vecka för att se till att allt var som det skulle och vi inte missat något.

Vecka 16 (v.17)

Denna veckan släpptes den första versionen av Spreed och tanken är att QA-teamet bland andra ska börja gå över till att använda vårt verktyg istället för QRM och det före detta Spreed. Under övergångsperioden hoppas jag vi kommer få mycket feedback om vad de tycker som Spreed samt om det är något som saknas eller inte riktigt stämmer så vi kan fixa till det.

Förklaringar

1. Roma

Ett verktyg för Plejds QA-team som hjälper till att visualisera och analysera data från flera källor för att underlätta planering och prognos.

2. Explanation Modal

Ett pop-up fönster vars uppgift är att förklara vissa sektioner av sidan. Detta skulle exempelvis kunna vara en förkortning eller en längre matematisk formel.

3. Select-komponent

En komponent där användaren kan välja ett eller flera alternativ utifrån en bestämd lista av värden.

4. Filtersektion

Högst upp på varje sida av Roma (exempelvis testresultat-sidan) finns det en filtersektion. Det är här användaren filtrerar vilka resultat det är som ska hämtas. Eftersom det är ganska detaljerad data finns det många punkter man vill kunna filtrera på och detta innebär att filtersektionen har blivit väldigt stor. För att lösa detta har vi implementerat ett "stängt" läge vilket innebär att sektionen förminskas och döljer filterna när du inte behöver dem.

5. Plato

Ett internt utbildningssystem som jag och två andra LIA studenter har utvecklat kontinuerligt sedan i somras. Projektets uppgift är att hantera företagets interna utbildningar genom att ge utbildare möjligheten att boka anställda för kurser samt skicka ut uppföljningsfrågor efter avslutat utbildningstillfälle. Plato består av en egenbyggd REST API samt ett webbgränssnitt som är den huvudsakliga delen utbildarna integrerar med.

6. SPC-sida

Romas SPC-sida är en sida för statistisk processkontroll. Här ska användarna kunna se utförlig statistik av olika tester och tidsspann. Exempelvis standardavvikelse, Anderson-Darling test och mycket mer.

7. Yield

En del av Roma där användarna ska kunna se statistik av testers utfall. Här ska man kunna se hur stor andel av tester som lyckas, misslyckas eller avbryts. Både totalt och över tid.