Nyhetsflöde med Raspberry Pi

Ett roligare sätt att ta del av ett företags affärsrelaterade händelser

Examensarbete inom Högskoleingenjörsprogrammet i Datateknik

ADAM BOOK
DAVID HARALD

Nyhetsflöde med Raspberry Pi

Ett roligare sätt att ta del av ett företags affärsrelaterade händelser

Adam Book, David Harald

Institutionen för data- och informationsteknik
Chalmers tekniska högskola 412 96 Göteborg
Tel: 031-772 1000
Fax: 031-772 3663

FÖRORD

Detta examensarbete gjordes under våren och försommaren 2016. Ett arbete som handlar inte bara om teknisk utveckling utan även personlig utveckling. Efter ett projektarbete som började i samarbete på Gatubolaget och deras idé om smarta postlådor fick vi kontakt med konsultföretaget Sigma. Sigma har mycket kontakt med Chalmers och har många anställda med tidigare kopplingar till Chalmers. Vi fick kontakt med dem då de förvaltar digitala tjänster till Gatubolaget.

Vi har fått gott intryck av Sigma och Rashwan Lazkani som har varit ett bra stöd och vår handledare på företaget. Med påföljden att vi fått goda kontakter med Sigma och med stor sannolikhet inte har sett det sista av dem.

SAMMANFATTNING

Sigma är ett stort konsultföretag i Göteborgsområdet. De har efterfrågat en helhetslösning för att snabbt kunna dela nyheter internt om bland annat nya kunder och nya uppdrag. Denna helhetslösning har utvecklats. En databas, webbtjänst, webbapplikation, applikation till Android och notifikationer med hjälp av en Raspberry Pi med ljusslinga har utvecklats. Databasen lagrar alla meddelanden och användare. Webbtjänsten hanterar förfrågningar mot databasen. Meddelanden skrivs i applikationen till Android. Dessa kan visas i webbapplikationen och i applikationen till Android. När ett meddelande publiceras lyser en ljusslinga i några sekunder. Denna ljusslinga styrs av Raspberry Pi.

ABSTRACT

Sigma is a large company in the Gothenburg region. They have asked for a service to quickly be able to share their most recent news about new customers and assignments among their staff. This service has been developed. A database, web service, web application, Android application and notifications with the help of a Raspberry Pi and a LED strip have been developed. The database stores the messages and the users. The web service handles requests towards the database. People can write messages to the newsfeed with the Android application. Messages published to the newsfeed can be read either with the Android application or the web application. When a new message is published on the newsfeed, a LED-animation is shown at the office. This LED-animation is controlled with the help of a Raspberry Pi.

ORDLISTA

API	-	Application Programming Interface.
Android	-	Ett operativsystem.
DTO	-	Data-Transfer-Object
ER-diagram	-	Entity-Relationship-diagram.
iOS	-	Apple's operativsystem för deras mobiltelefoner
IoT	-	Internet of Things
JSON	-	JavaScript-Object-Notation.
LED	-	Light-Emitting-Diode.
LINQ	-	Languge-Integrated Query.
MD5	-	Message-Digest-algorithm-number-5
MVC	-	Model-View-Controller.
NFPU	-	Tidigt namn för projektet, står för News-Feed-Publisher-Unit.
REST	-	Representational State Transfer.
SCRUM	-	Ett sätt att planera upp ett projekt. Scrum är inget akronym.
SDK	-	Software Development Kit.
SQL	-	Structured-Query-Language.
Trello	-	Trello är en webbapplikation som används som hjälpmedel till SCRUM.
XML	-	Extensible-Markup-Language.

1. INLEDNING

1.1 Bakgrund

Sigma är ett ledande konsultföretag med fokus på teknik som har över 6000 anställda. Sigma IT Consulting Sweden är en avdelning som fokuserar på IT-relaterade tjänster för företag. Några kunder är t ex Saab, Gatubolaget och Liseberg. Som konsultföretag är det vanligt att anställda är ute på annan arbetsplats för att bistå företagskunderna.

1.2 Syfte

På ett konsultbolag är dagarna fyllda med aktiviteter som t ex intervjuer, kundmöten, orderförfrågningar, etc. Oftast gäller devisen att ju fler aktiviteter man har, desto bättre går det för företaget.

För de som har mer stationära aktiviteter knutna till kontoret vill Sigma erbjuda en möjlighet att ta del av vad som händer ”ute på fältet”. Detta ska uppnås med en intern newsfeed (nyhetsflöde) som informerar om när någon på Sigma t ex har bokat en ny kund, slutfört ett projekt eller liknande. Till denna newsfeed kopplas någon form av visuell notifikation för att informera kollegor på kontoret att något nytt har hänt.

1.3 Mål

Målet är att utveckla en helhetslösning för det interna nyhetsflödet. Det ska gå att skicka meddelanden till nyhetsflödet med hjälp av en applikation för Android, se nyhetsflödet i en webbapplikation, och få någon form av visuell notifikation (en ljusslinga som lyser ett mönster) när det kommer ett nytt meddelande till nyhetsflödet.

Utvecklingen kan delas in i följande områden:

- Databas för att lagra informationen
- Webbtjänst för att sköta kommunikationen mellan webbapplikation och databas
- Webbapplikation för att visuellt visa nyhetsflödet
- Applikation till Android för att skicka meddelanden
- Visuell notifikation

1.4 Avgränsningar

En Raspberry Pi kommer att användas för hantering av notifikationerna. Andra alternativ undersöks inte.

Datasäkerhet hade låg prioritet. Enkel textformatering är implementerad för att lösenord inte ska stå i klartext.

Ramverket ASP.NET och designmönstret Model-View-Controller (MVC) med Entity Framework kommer att användas för webbapplikationen då detta har efterfrågats av Sigma.

2. METOD

2.1 Överblick

Arbetsföljden lyder:

1. Planera strukturen (inklusive mockups)
2. Databas
3. Webbtjänst
4. Webbapplikation
5. Applikation till smartphones
6. Applikation för notifikationer med Raspberry Pi

Inledningsvis planeras programstrukturen genom att gå igenom vilka attribut databasen kommer att lagra. Detta ger en överblick över funktioner och anrop som behöver skapas för de olika applikationerna (webbtjänst för att kommunicera med databasen, webbapplikationen för webbvyn, mobilapplikation för att skicka meddelanden, applikation på Raspberry Pi för notifikationer).

När databasen är färdigställd är nästa steg att bygga en webbtjänst, då denna kommunicerar direkt med databasen. Tanken är att webbtjänsten ska sköta kommunikationen mellan databasen och webbapplikationen.

Nästa steg är att bygga webbapplikationen. När både databasen och webbtjänsten är skapade finns de underliggande lager som webbapplikationen behöver för att kunna fungera optimalt.

När databas och webbtjänst är färdiga finns de underliggande byggstenar som behövs för att utveckla en applikation till Android. Med applikationen ska en användare kunna ange sitt namn och ett meddelande som ska skickas till nyhetsflödet.

Den slutgiltiga uppgiften är att koppla en Raspberry Pi med en ljusslinga som ska detektera när meddelanden har skickats till nyhetsflödet och tända ljusslingan under en viss tid.

2.2 Skalbarhet

Prioritet lades vid att bygga en applikation till Android för att skicka meddelanden. Det skulle vid ett senare tillfälle kunna utvecklas en applikation för iOS. Den ska då fungera på ett liknande sätt som applikationen för Android.

För webbapplikationen har det förts diskussioner om olika påbyggnader. Bland de extra funktioner som har diskuterats finns bland annat inloggning (för ökad säkerhet, eftersom nyhetsflödet är en intern tjänst), filtrering (för att se alla meddelanden en viss användare har skickat), topplista för bl a bokade kunder, osv.

2.3 Arbetsmetodik

Arbetsmetodiken som har använts är Scrum, ett arbetssätt som är vanligt inom agil utveckling [2]. Det är ett verktyg för att dela upp och effektivisera arbetsuppgifter och sätta stort fokus på det värdeskapande arbetet. Se teknisk bakgrund för mer information.

För versionshantering användes Git, den i dagsläget senaste versionen, 2.8 [12]. Git är ett revisionshanteringsprogram som prioriterar snabbhet, dataintegritet och har stöd för tidsmässigt osammanhängande arbetsgångar. Det är också helt gratis, lanserat med GNU General Public License version 2.
Själva utvecklingen av kod skedde huvudsakligen i Visual Studio 2015 för de webbaserade tjänsterna. Utvecklingen av Android-applikationen har skett i Android Studio.Utveckling av lyssnarprogrammet för notifikationer med Raspberry Pi har skrivits i Python 2.7.9.

2.4 Material

Huvudmaterialet som används är följande:

- Raspberry Pi 3
- BlinkyTape (LED-slinga utformad för användning med datorer)
- Skärm för att visa nyhetsflöde
- Visual Studio 2015 för utveckling av webbtjänst och webbapplikation
- Android Studio för utveckling av applikation till Android
- Postman för testning av webbtjänst
- Google Chrome webbläsare
- Git
- Mockups

Till detta används diverse kablar och standardutrustning för datorer.

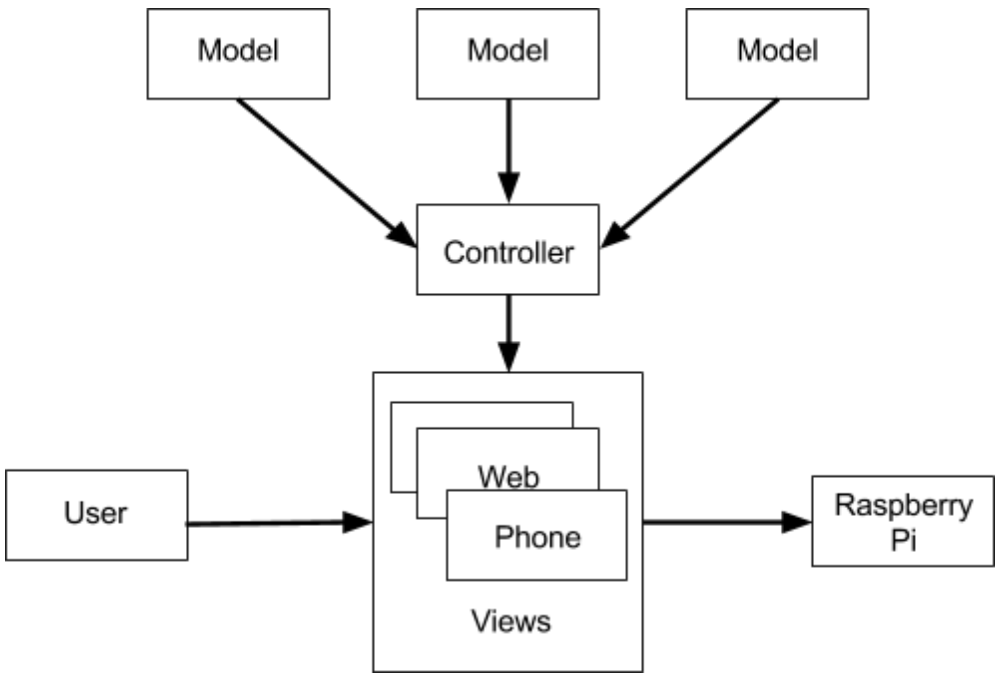
3. TEKNISK BAKGRUND

3.1 MVC-Modellen

MVC är ett designmönster som används brett idag för utveckling av applikationer [5]. MVC står för Model-View-Controller och bygger på att separera viktiga delar av utvecklingen. Detta designmönster är applicerbart i vilken typ av strukturerat projekt som helst, inte nödvändigtvis bara för digital utveckling. MVC-modellen används i stor utsträckning vid professionell mjukvaruutveckling. Tanken med MVC är att uppdelning kan undvika onödigt komplexa lösningar, konflikter mellan de olika instanserna och mutation av data. Samtidigt skapas en tydlig struktur och ökar läsbarheten i koden.

- **(M) Modell:** Strukturerar upp den data som fysiskt ska lagras. Det är inte nödvändigtvis i en databas utan kan vara vilken representation som helst.
- **(V) View:** Är en presentering av data för en användare. Ett typiskt exempel i utveckling av webbaserade projekt är vyn i en webbläsare.
- **(C) Controller:** Hanterar logiska händelser baserat på omständigheter som initierats av en användare eller statisk logik. En majoritet av funktionaliteten finns här.

MVC kan uppfattas som en flyktig term utan kontext, men kommer här att appliceras på utvecklingen av programvara för webbapplikationen (se figur 3.1). Det kommer att innebära en separerad struktur för att främst öka läsbarheten.



Figur 3.1 - En vy över MVC-strukturen som appliceras

3.2 Scrum

Scrum är en arbetsmetodik som är vanligt förekommande i projekt med programmering. Det är ett arbetssätt som räknas som *agil utveckling* [16].

Agil utveckling är ett arbetssätt som togs fram av programutvecklare som hade tröttnat på den överväldigande byråkrati och dokumentation som vid den tiden präglade mjukvaruutveckling. Istället läggs fokus på att underlätta mjukvaruutvecklingen i sig. Det tydliga målet blir att leverera den värdeskapande koden, att ta bort alla hinder som kan dyka upp längs vägen och att lätt kunna anpassa sig till nya omständigheter.

I Scrum kallas projektledaren Product Owner. Det är den person som har det övergripande ansvaret för projektet. Product Owner som tar fram en prioriterad önskelista över vad som ska uppnås i projektet. Denna önskelista kallas Product Backlog. Varje enskild punkt i projektets Product Backlog kallas för en User Story, och är en beskrivning av den funktion som önskas uppfyllas.

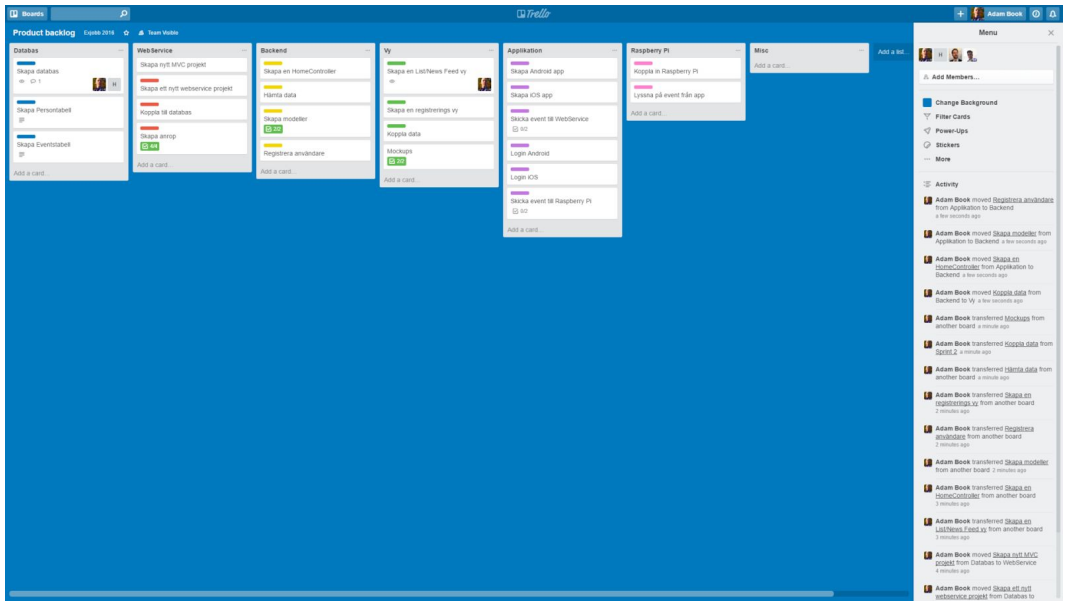
I Scrum sker arbetet i något som kallas Sprints. En Sprint är en begränsad tidsperiod (ofta mellan 2-4 veckor) då fokus ligger på att få utvalda delar av projektets Product Backlog färdiga.

Efter varje Sprint utförs en Sprint Review och en Sprint Retrospective. På Sprint Review presenteras det som har åstadkommits.
På Sprint Retrospective reflekterar utvecklarna över hur arbetet blivit utfört och diskuterar förbättringsåtgärder för att effektivisera framtida Sprints.

När nästa Sprint startar väljer utvecklarna återigen nya områden att fokusera på från projektets Product Backlog och påbörjar arbetet.

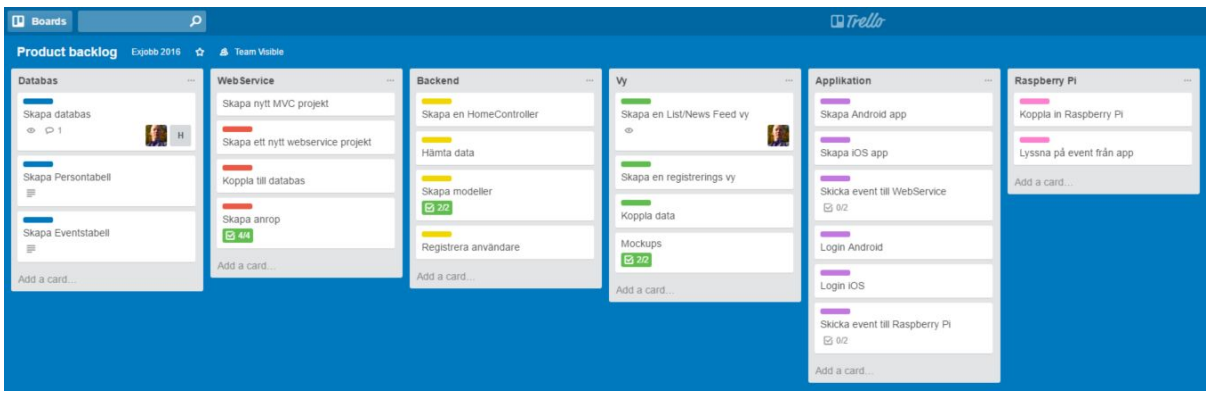
3.3 Trello

För att organisera ett projekt med Scrum är det vanligt att använda verktyget Trello [10]. Trello är en webbapplikation med diverse verktyg för att underlätta hanteringen och översikten av ett projekt (se figur 3.2). Under hela utvecklingen har detta använts i samband med Scrum.

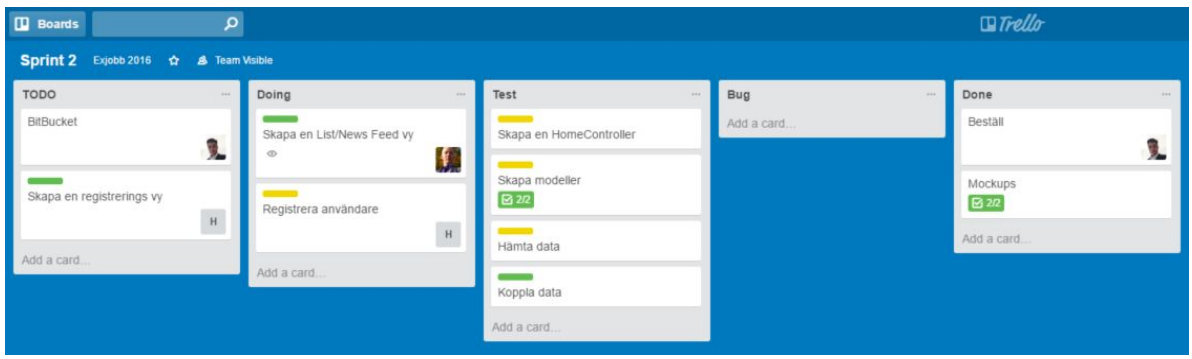


Figur 3.2 Överblick på Trello

Trello är ett utmärkt verktyg för att åskådliggöra Scrum. Genom att skriva små kort i Trello kan dessa användas som User Stories med drag-and-drop funktionalitet. Detta innebär att det blir smidigt och enkelt att skapa en Product Backlog (se figur 3.3) och sedan allt eftersom flytta User Stories till en Sprint (se figur 3.4).



Figur 3.3 Product Backlog i Trello-vy



Figur 3.4 Sprint 2 i Trello-vy

3.4 CSS

Cascading Style Sheets (CSS) är ett språk som ofta används för att utforma designen av en webbapplikation [3]. Det används bland annat till att separera en webbapplikations innehåll och hur detta innehåll presenteras. Genom att använda CSS kan samma stilmall appliceras på flera olika sidor. Visningen kan också anpassas för olika enheter (telefoner, surfplattor, datorer).

3.5 Bootstrap

Bootstrap är en designmall som ofta används till webbdesign för att få en stilren och konsekvent design på en webbapplikation [9]. Det är open source och använder sig av HTML och CSS för att skapa enhetlig design.

3.6 Databas & Code First

En databas är en lagringsplats för data [4]. För att kunna lagra data effektivt är det lämpligast att endast lagra relevant data en gång. Genom att strukturera upp datan i en serie av tabeller med dess data länkad kan man optimera informationslagringen.

Ett av verktygen som används är Microsofts egna utvecklarprogram Visual Studio. I detta program finns ett verktyg som kallas Code First[13]. Denna metod används för att automatiskt generera databasen. Tanken är att koden skrivs först, med de variabler och modeller som önskas. Därefter genereras databasen beroende på den data som behöver hanteras. Databasen är skriven i SQL (Structured Query Language), vilket är det vanligaste språket till databaser.

3.7 Webbtjänst

En webbtjänst är en applikation som har direkt kontakt med databasen [15]. Webbtjänsten hämtar och skriver data i databasen. API står för “Application Programming Interface” och det är i grund och botten bara en samling regler för nämnda applikation. De två största fördelarna med att använda en webbtjänst är lätthanterlighet vid olika plattformar (i detta fallet webbläsare, Android, och iOS) och datasäkerhet.

Genom att ha en webbtjänst är det lätt att kommunicera med databasen oavsett om det är Android, iOS eller en webbläsare som gör en förfrågning. Förfrågningar hanteras på liknande sätt för olika plattformar mot en webbtjänst. Skulle en webbtjänst inte användas skulle specifik kod behöva skrivas för att kunna hantera förfrågningar från varje enskild plattform.

Genom att lägga till en webbtjänst som en mellanhand mellan databasen och applikationerna ökar datasäkerheten. Detta betyder att en användare inte har direkt kontakt med databasen, vilket i sin tur leder till att databasen inte är lika utsatt för attacker och manipulation som den annars skulle vara. Eftersom användare bara kommunicerar med en webbtjänst är det utvecklarna som bestämmer vilka förfrågningar som kan göras mot databasen.

Webbtjänsten bygger på en inbyggd SQL-abstraktion i Visual Studio, som kallas Linq (Languge-Integrated Query), som bygger på många, lättförståeliga mönster för strukturering och hantering av data [6]. Med hjälp av detta hämtas data i form av JSON-objekt eller XML från databasen och kan hanteras av i princip vilken typ av applikation som helst.

JSON står för "JavaScript Object Notation" och är ett sätt att presentera data, egentligen är det inget annat än en syntax i grund och botten.

3.8 Postman

Postman är ett program som används för att skicka och ta emot HTTP-requests [11]. Det är ett program som gör det enkelt att till exempel testa att en webbtjänst fungerar som den ska och klarar att hantera HTTP-requests på det sätt som önskas.

3.9 Back-End

Back-End är ett bildligt ord som definierar en applikations funktionalitet. Back-End kan också ses som en synonym till Controller. En sorts intern Controller för applikationen. Här kan bland annat tillfälliga beräkningsbara attribut till diverse data distribueras och presenteras. Back-End är C(Controller) i MVC modellen.

3.10 Webbapplikation

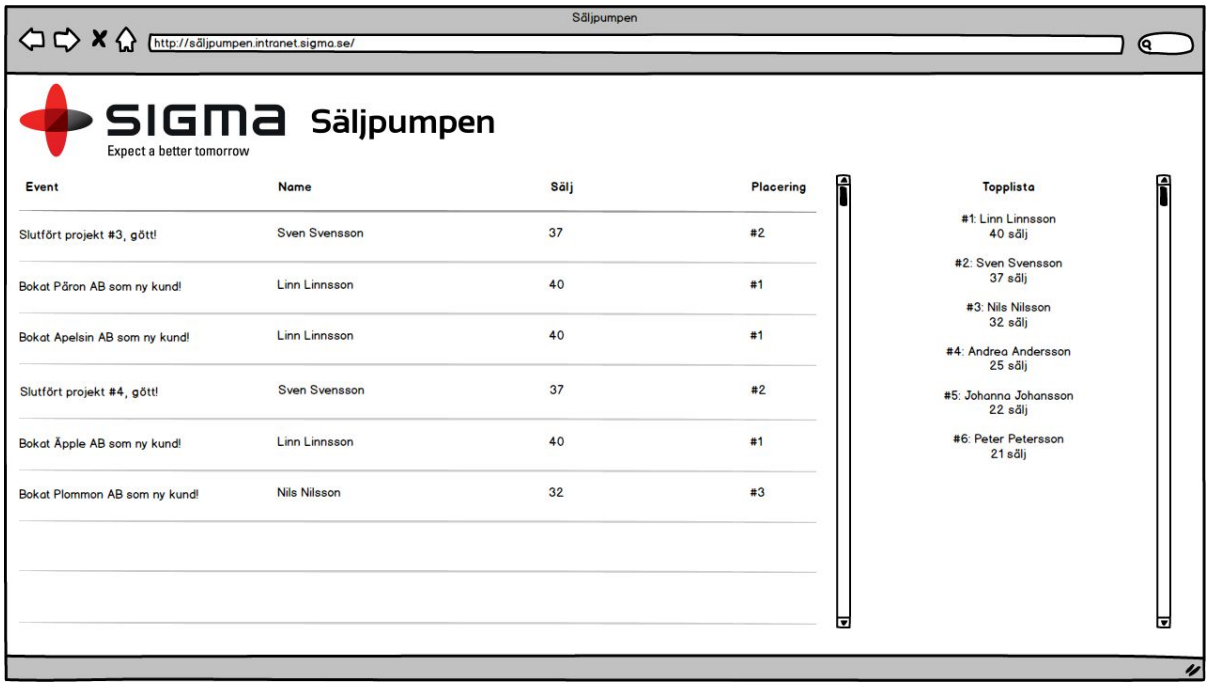
Webbapplikationen är en ASP.NET-applikation, precis som webbtjänsten, med skillnaden att denna även har vyer för presentation av data [14]. Det är webbapplikationen som kommer att synas för användaren. Webapplikationen bör inte påbörjas förrän webbtjänstens funktionalitet för datatrafik är fullt utvecklad. Ändringar kan göras på tjänsten efter att applikationen påbörjats, men det kan skapa många konflikter i koden. Webbapplikationen är V(View) i MVC-modellen. Webbapplikationen innehåller dock också logik. Därav är webbapplikationen också C(Controller) i MVC.

3.11 Mockups

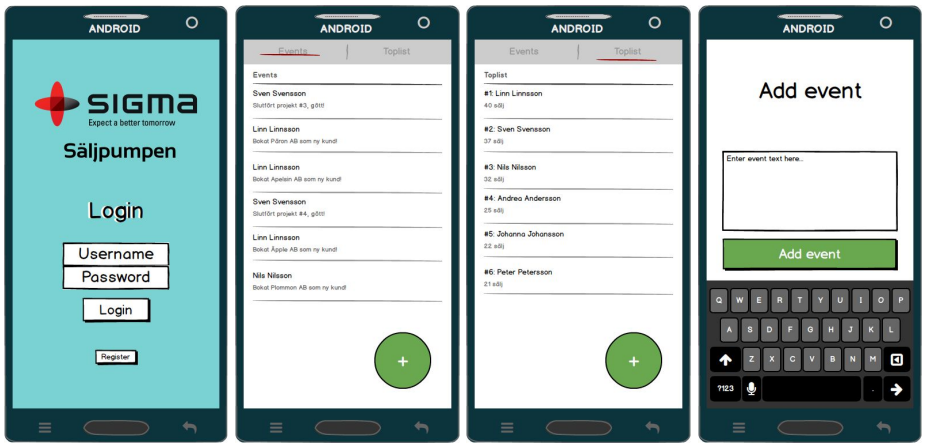
En mockup är en mycket enkel prototyp som används för att inleda designprocessen (både vad gäller gränssnitt och funktionalitet) och kommunicera med kunden om önskat utseende och funktionalitet[1]. Tanken är att börja få respons från kunden så snart som möjligt.

Eftersom relativt lite resurser läggs på varje mockup har kunden lättare att komma med kritik och förslag. Skulle stora resurser användas för en första version av mjukvaran till kunden är det mycket besvärligare för kunden att kritisera produkten och be om ändringar. Det är dessutom dyrare och mer omständligt för utvecklaren.

Eftersom det är lätt och billigt att göra ändringar i en mockup används de även internt bland utvecklarna för att diskutera olika designförslag och framtida funktionstillägg.



Figur 3.5 Mockup för webbapplikationen.



Figur 3.6 Mockups för telefonappen.

3.12 Blinkytape (LED-slinga)

Blinkytape är en LED-slinga som består av ett en meter långt flexibelt band på totalt 60 LED-lampor [7]. Den är avsedd för användning med datorer och kan programmeras för att visa de ljusmönster som önskas. Man kan programmera färgen, ljusstyrkan och animationsmönstret för varje enskild LED-lampa på slingan och därmed uppnå exakt ljussekvens som önskas.

Figur 3.7 BlinkyTape med tända LED-lampor

3.13 Raspberry Pi

Figur 3.8 Raspberry Pi 3 (utan chassi)

Raspberry Pi [8] är en liten enkortsdator (8,6 cm lång, 5,6 cm bred och 2,1 cm hög) som har blivit mycket populär på senaste tid för Internet-of-Things-projekt (IoT). IoT handlar om

teknikprylar som är uppkopplade till Internet, till exempel en kaffebryggare. På grund av dess låga pris på ca 450 kr har IoT-projekt blivit allt mer alldagliga och något som privatpersoner kan ta del av.

Figur 3.9 Den Raspberry Pi som använts (med taket på chassit avplockat)

3.14 Linq

Linq används för kommunikation och koppling av modellobjekt till databasen. Linq är en serie mönster som gör det möjligt att hantera JSON, XML eller liknande, helt utan några kunskaper om hur den underliggande koden kommer att exekveras. Linq är ett bra verktyg för enkel hantering av data.

Linq kan t ex skrivas i C# (se figur 3.10). Linq skrivs väldigt likt vanlig SQL-kod och fungerar på samma sätt. I detta fallet begärs information från databasens tabell Events och sparar denna data till en digital representation där varje enskild sammanhängande data sparas till ett EventDTO-objekt. DTO används som inte en helt exakt representation av innehållet i databasen utan en representation av den data som skickas, vilket inte alltid är exakt det som finns i databasen.

```
1.    private NFPU5Context db = new NFPU5Context();
2.
3.            // GET: api/Events
4.            public IQueryable<EventDTO> GetEvents()
5.            {
6.                DateTime pastThirtyDays = DateTime.Today.AddDays(-30);
7.
8.                var Event = from eventObj in db.Events
9.                            where eventObj.dateOfPublish >= pastThirtyDays
10.                           orderby eventObj.dateOfPublish descending
11.                           select new EventDTO()
12.                           {
13.                               id = eventObj.id,
14.                               text = eventObj.text,
15.                               category = eventObj.category,
16.                               dateOfPublish = eventObj.dateOfPublish,
17.                               personID = eventObj.personID
18.                           };
19.               return Event;
20.           }
```

Figur 3.10 - Exempel med Linq

4. GENOMFÖRANDE

4.1 Allmänt

Eftersom databasen är grunden till designen utvecklades denna först. Därefter utvecklades en webbtjänst för att hantera förfrågningar till databasen. När dessa två grundstenar var på plats fanns de underliggande delar som webbapplikationen behöver för att kunna visa vyer med data. Valet att utveckla webapplikationen som nästa steg gjordes då även Android-enheter kan använda sig av webbläsare och kan då läsa nyhetsflödet via webben. Därefter utvecklades en applikation till Android.

Det är lättare att utveckla de olika instanserna var för sig och i ordning (databas, webbtjänst, webbapplikation, applikation till Android och notifikationer med Raspberry Pi). De flesta instanser bygger på att en annan instans existerar och opererar. En applikation kan visserligen kodas och hantera inkommande information, men om det inte finns någon tjänst som tillhandahåller information så kommer ingen data presenteras. Ändringar kommer oundvikligen att behöva göras vid ett flertal tillfällen. Dessa kan vara mycket svåra att implementera och kan ta lång tid.

All utveckling av kod har skett i Visual Studio 2015 om det inte framgår att något annat verktyg har använts. Testning av API sker med Postman eller webbapplikationen.

4.2 Databasen

Databasen är mycket enkel. Den består av två tabeller som innehåller användare och event som läggs upp. Själva skapandet av databasen skrivs med hjälp av Visual Studios egna implementering av Code First. För att demonstrera strukturen används ett ER-diagram (se figur 4.1).

Figur 4.1 - ER-diagram över databasstrukturen

Tabellen innehållande alla användare finns i entiteten People. Alla event finns i entiteten Event. Pilen beskriver att en användare kan ha många events medan ett event måste vara kopplat till exakt en användare. Databasen skulle kunna göras mer komplex men Sigma uttryckte specifikt ett önskemål om att hantera all logik rörande databasen i webbtjänsten. Detta kan kosta prestanda men ökar läsbarheten.

Tanken är att skalbarheten ska vara stor. Detta innebar att databasen kunde innehålla fler attribut och även fler tabeller som i slutändan kanske inte kommer att användas.

En annan ändring som gjordes var att skapa statiska datavärden för att kunna dela dessa bland alla plattformar och undvika hård statisk kodning. Dessa statiska datavärden interagerar inte med användaren på något sätt.

Tidigare attribut vars användningsområde upphörde togs egentligen inte bort ur databasen utan ignorerades. De antog antingen null-värden eller autogenererade värden.

Det är givetvis svårt att veta i förväg vad som ska tilläggas under en vidareutveckling. Om det är känt vilka funktioner som skall utvecklas i framtiden är det önskvärt att implementera stöd för det så tidigt som möjligt.

Databasen hanterar även känslig information och gör så genom en algoritm som krypterar datan på en låg nivå. Det gjordes tidigare en avgränsning mot skyddande av känslig data. En låg krypteringsnivå implementerades ändå för att undvika att datan står i klartext. Algoritmen är en standard MD5-algoritm som konverterar den indata den får till en lång sträng av oigenkännlig representation av indatan. Tanken är att det inte går att återskapa den ursprungliga datan utan endast dess krypterade version. Denna version går heller inte att återskapa genom att köra algoritmen baklänges. Databasens hantering av lösenord blir därmed säkrare. Om någon skulle komma över informationen i databasen så får angriparen då inte tillgång till lösenorden i klartext.

4.3 Webbtjänst

För läsning och skrivning av data till och från applikationen används en webbtjänst. Webbtjänsten är utvecklad i ASP.NET 4.5. Webbtjänsten är ett så kallat RESTful API, vilket innebär att det lätt kan göras förfrågningar baserat på CRUD (create, read, update, delete). Förfrågningar mot webbtjänsten görs via HTTP (POST, GET, PUT, och DELETE) och JSON-objekt skickas mellan webbtjänsten och förfrågaren. Sigma utvecklar mycket i samma miljö. För att de enklare ska kunna använda alternativt vidareutveckla nyhetsflödet kommer utvecklingen att hålla sig till de normer satta av Sigma.

Webbtjänsten kommer att hållas som en separat applikation. Tanken är att alla övriga delar, webbapplikation och mobil-applikation, ska kommunicera med webbtjänsten och genomföra önskade förfrågningar för att behandla och presentera data. Fördelen med detta är att det endast behövs en webbtjänst istället för en till varje plattform.

API:n skrivs i huvudsak i programspråket C#. Själva kommunikationen mellan de olika applikationerna sker genom att skicka och ta emot JSON-paket. Fördelen med JSON är att data kan behandlas asynkront mycket enkelt. Asynkront menas att begäran behandlas i bakgrunden istället för att hela applikationen står stilla och väntar på svar från en server som kanske inte är igång.

Efter skapandet av webbtjänsten har minimala ändringar gjorts i efterhand. De ändringar som har tillkommit är utbyggnaden av topplista, hantering av kategorier och sortering av events.

Webbtjänsten är designad med MVC-mönstret och Code First. Detta innebär att data är strukturerad i modeller och hanteras i controllers. I modellen beskrivs vad för data som önskas sparas (se figur 4.2). Därefter skapas databasen med de tabeller som krävs baserat på hur modellerna ser ut. Sedan kan man hantera datan i webbapplikationen.

```
1.    public class Event
2.        {
3.            public string id { get; set; }
4.            [Required]
5.            public string text { get; set; }
6.            public string category { get; set; }
7.            public DateTime dateOfPublish { get; set; }
8.
9.            // Foreign Key
10.           public string personID { get; set; }
11.       }
```

Figur 4.2 Modellen för Events

Kodexemplet i figur 4.2 resulterar i en tabell i databasen nämnd “Events” med följande utseende:

id	text	category	dateOfPublish	personID

När det senare görs en förfrågning mot webbtjänsten med HTTP kan det se ut som nedan. I detta fall efterfrågas webbtjänsten att skicka alla events.

Förfrågningen görs mot webbtjänsten:

```
GET http://localhost:56422/api/Events
```

JSON-objekt kan då se ut som följande (se figur 4.3):

```
[
  {
    "id": "902743870927",
    "text": "Erik på Företag AB.",
    "category": "Ringde upp : ",
    "dateOfPublish": "2016-05-31T13:39:48.113",
    "personID": "Michen"
  },
  {
    "id": "806725553461",
    "text": "Lisa på Företag AB.",
    "category": "Gjorde en affär med : ",
    "dateOfPublish": "2016-05-31T13:33:15.597",
    "personID": "Michen"
  },...
]
```

Figur 4.3 Webbtjänstens svar på förfrågan "GET Events"

För enkelhetens skull syns enbart två event-objekt i detta exempel, men i verkligheten hade en fullständig lista på alla events skickats som svar.

På liknande sätt kan man göra en förfrågning mot databasen för att lista alla registrerade användare av nyhetsflödet.

Förfrågningen görs mot webbtjänsten:

```
GET http://localhost:56422/api/People
```

JSON-svaret blir då (se figur 4.4):

Figur 4.4 Webbtjänstens svar på förfrågan "GET People"

Även i detta fall visas enbart de två första svarsobjekten i exemplet för enkelhetens skull.

Det finns även möjlighet att göra en GET-förfrågan på en specifik ID- eller mailadress:

```
GET http://localhost:56422/api/People?mail=mail@mailserver.com
```

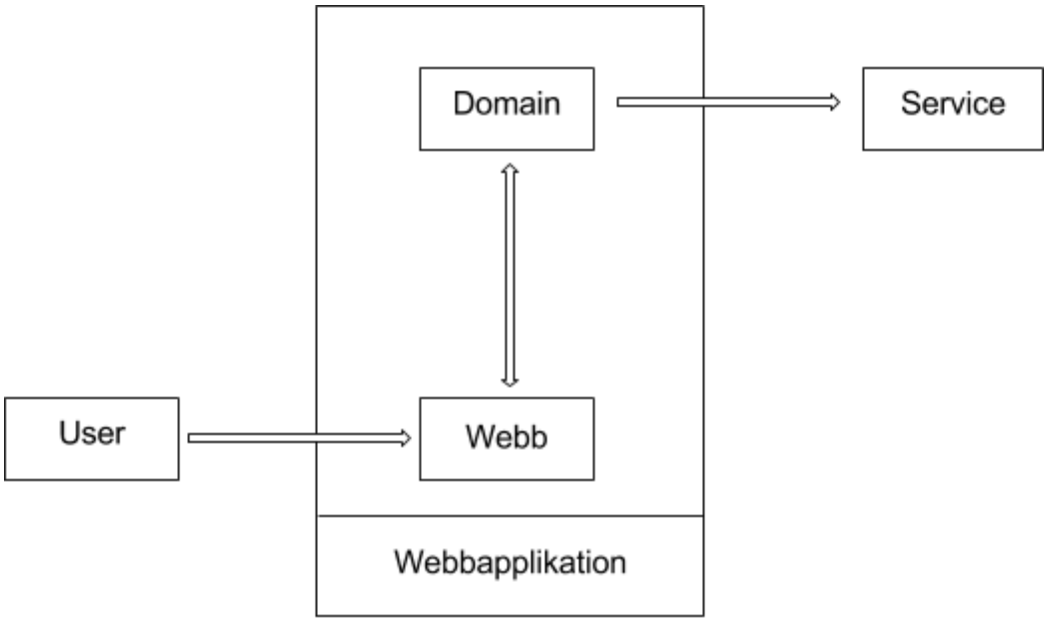
```
GET http://localhost:56422/api/People?id=AdamHa
```

4.4 Webbapplikation

Webbapplikationen utvecklas efter att webbtjänsten har slutförts och opererar korrekt. Applikationens funktioner bygger på att tjänsten inte ändrar på sin utgående trafik. Om ändringar måste göras på trafik så måste ändringar även i applikationen göras på den delen som använder sig utav just den utgående trafiken från webbtjänsten.
Applikationen är den vy som ska synas utåt och hanteras av användare. Därför är det viktigt att just applikationen innehåller en estetiskt tilltalande och enkel design. Kunden har fått ge sin åsikt om designen, och ändringar har gjorts därefter för att tillfredställa kundens önskemål.

4.4.1 Domain

Domain är ett internt hjälpprojekt i webbapplikationen som innehåller alla logiska operationer och metoder. Eftersom en webbaserad applikation kan bli mycket omfattande underlättar en pålitlig och enkel struktur. Med Domain delas logik upp för sig och webbvy för sig. Detta gör det enklare för en utomstående att förstå och eventuellt vidareutvekla. I figur 4.5 innehåller Domain all logik, Webb innehåller vyer och förfrågningar till Domain, Service är webbtjänsten som gör förfrågningar mot databasen.



Figur 4.5 Domain hjälp-projekt

En användare gör en förfrågan till webbapplikationen, webbapplikationen frågar sedan Domain om svaret på denna förfrågan, Domain svarar och webbapplikationen levererar resultatet i sin vy för användaren.

4.5 Android-Applikationen

Android-applikationen utvecklades i Android Studio och det bestämdes att den skulle ha ett minimum SDK av 8. SDK 8 valdes för att stödja ett så brett utbud av Android-enheter som möjligt. En användare loggar in i applikationen och får då tillgång till att läsa alla event, lägga upp nya event, hantera sina event och ta del av den allmänna topplistan, som då ger en representation av vilka som bidragit mest till eventlistan.
Utvecklingen började med att skapa indexsidan, som består av en inloggningsfunktion. Inloggningsfunktionen går sedan vidare till de vyer som är av intresse för användaren. Mycket tid gick åt till att bekanta sig med utvecklingsmiljön. Många av applikationens funktioner bygger, precis som i webbapplikationen, på att ett fungerande API redan finns. Detta medförde att funktionerna inte behövde skapas i en viss ordning.

Utvecklingen skedde i ett flertal steg:

- Indexsida skapades.
- Eventlista skapades.
- Event kunde skapas av en användare.
- Toplista.
- Registrering.
- Redigering av befintlig angiven information.

När presenteringen av data sker, oavsett plattform, hämtas datan från webbtjänsten i form av JSON-objekt. Figur 4.6 visar ett utdrag för hur ett HTTP-Request i Android-applikationen kan anropas.

```
1.    case "GET": {
2.                    URL testUrl = new URL(params[0] + "/api/" + params[1]);
3.                    testConnect = (HttpURLConnection) testUrl.openConnection();
4.                    testConnect.setRequestMethod("GET");
5.                    testConnect.connect();
6.
7.                    InputStream testInput = testConnect.getInputStream();
8.
9.                    testReader = new BufferedReader(new InputStreamReader(testInput));
10.                   buffer = new StringBuffer(); // This holds the data
11.
12.
13.                   String line = "";
14.                   while ((line = testReader.readLine()) != null) {
15.                       buffer.append(line);
16.                   }
17.
18.                   return buffer.toString();
19.               }
20.
```

Figur 4.6 - En java klass i Android-applikationen, en metod för att hämta data.

4.6 Raspberry Pi och notifikationer

När ett meddelande publiceras lyser en LED-slinga på kontoret för att signalera att ett nytt meddelande har skrivits. LED-slingan lyser olika beroende på vilken kategori meddelandet tillhör. Detta sker med hjälp av en Raspberry Pi och en BlinkyTape. Enheten är uppkopplad till Internet via kontorets WiFi och har ett lyssnarprogram som körs i bakgrunden.

Varje gång ett meddelande publiceras skickas ett meddelande från Android-applikationen till Raspberry Pi-datorn på kontoret för att initiera LED-slingan. Meddelandet från Android-applikationen informerar Raspberry Pi:n om vilken kategori meddelandet tillhör varefter en ljussekvens lyser på LED-slingan baserat på vilken kategori som valdes.

4.6.1 Lyssnarprogrammet för notifikationer

Lyssnarprogrammets funktion är att lyssna efter meddelanden och tända ljusslingan då ett nytt meddelande tas emot. Det är skrivet i Python 2.7.9. Python valdes då BlinkyTape har

förbyggda bibliotek för Python som delvis kunde användas för att effektivisera programmeringen.

UDP valdes som kommunikationsprotokoll [17]. Detta beror på att UDP är ett protokoll som är effektivt att implementera. I det osannolika fall att ett UDP-paket inte skulle nå lyssnarprogrammet innebär detta enbart att ljusslingan inte kommer lysa för det meddelandet. Det kommer därefter att fungera som vanligt igen.

Eftersom BlinkyTape är en LED-slinga med 60 lysdioder skickas kommandon till varje lysdiod. För varje animation som visas går programmet igenom varje enskild lysdiod för att meddela hur den ska lysa.

Lyssnarprogrammets första läge är att lyssna efter meddelanden. Om ett meddelande tas emot läses det av för att bekräfta om det motsvarar en av de meddelanden som initierar en animation på ljusslingan.

Figur 4.7 visar en förenklad version av koden för avlyssningen av meddelanden.

```
1.  import socket
2.
3.  UDP_IP = "10.66.195.69"
4.  UDP_PORT = 5005
5.
6.  sock = socket.socket(socket.AF_INET, # Internet
7.                       socket.SOCK_DGRAM) # UDP
8.  sock.bind((UDP_IP, UDP_PORT))
9.
10. print("Waiting for messages...")
11.
12. while True:
13.     data, addr = sock.recvfrom(1024) # Buffer size is 1024 bytes
14.     print ("Received message:"), data
```

Figur 4.7 - utdrag ur koden för att ta emot UDP paket på Raspberry Pi

UDP_IP är i detta fall värddatorns IP. UDP_PORT anger vilken port programmet ska lyssna efter UDP-meddelanden på.

Om till exempel meddelandet “RED” skickas till lyssnarprogrammet kommer det att initiera en ljussekvens på ljusslingan i röd färg. RGB-koder skickas till varje lysdiod (i detta fallet mörkröd till 3 lysdioder och därefter ljusröd till 3 lysdioder tills alla 60 lysdioder har fått en färginstruktion). När varje lysdiod har fått en startfärg skickas en puls varje 8 millisekunder då programmet “flyttar” färgerna en lysdiod framåt i ledet. Lysdiodens färg i slutet av slingan skickas till den första lysdioden.

Figur 4.8 visar ett utdrag ur lyssnarprogrammet för hantering av den röda animationen.

```
1.    if(data=="RED"):    #If the message received is "RED", do the following
2.           print("Message was RED!")
3.           for x in range(0, 800): #Loop the animation sequence 800 times
4.                # Run a tick on each block
5.              for block in redblocks:
6.                    block.tick(redblocks)
7.
8.                 # Once we've ticked all blocks, redraw everything
9.                 # Let's set everything to blank first
10.             for i in range(0,btr.getSize()):
11.                btr.setPixel(i,0,0,0)
12.                # Now loop all blocks and get colours
13.             for block in redblocks:
14.                colour = block.getColour()
15.                for x in block.getOccupiedSpaces():
16.                   btr.setPixel(x,colour[0],colour[1],colour[2])
17.
18.             btr.sendUpdate()
19.             time.sleep(tickTime) #tickTime = 0.002
20.          #Block.clear(blocks)
21.          for block in redblocks:
22.                   bt.clear()      #Turn off the LEDs
```

Figur 4.8 - koden för att tända ljusslingan i rött

Sammantaget ger detta ett intryck av en animation där lysdioderna snabbt “rör” sig igenom slingan. Efter att ljusslingan har lyst i ca 7 sekunder återgår den till att vara släckt tills ett nytt meddelande kommer.

5. SLUTSATS

5.1 Databas

Databasen utvecklades utifrån vad Sigma angav i sin ursprungliga kravspecifikation - en tabell för användare och en tabell för inläggen. Databasen är skapad genom Visual Studios "Code First" metod och det innebär att den är mer eller mindre autogenererad utifrån de modeller som har ställts upp (se figur 4.2 för modell-exempel). På begäran från Sigma önskades att databasens logik ska vara i C# och inte SQL, vilket medför att logiken och hanteringen av data sker i webbtjänsten. All data sparas som textsträngar (nvarchar i databasen).

5.2 Webbtjänst

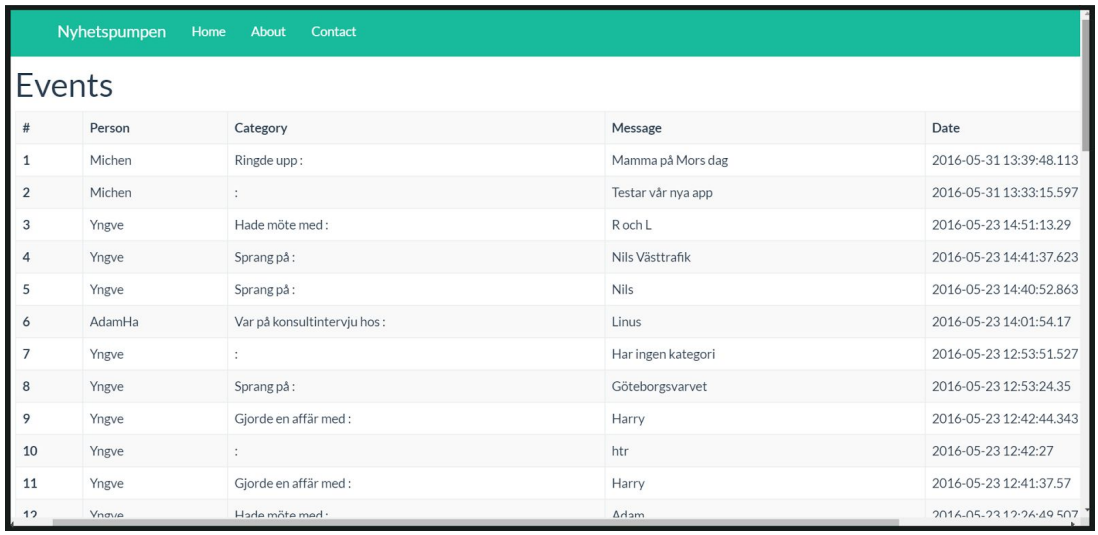
Webbtjänsten innehåller logiken för att läsa och skriva i databasen. All trafik från användarna passerar webbtjänsten. Tjänsten som utvecklades har stöd för CRUD-operationer (Create, Read, Update, Delete) på både användare och inlägg. Tjänsten innehåller även extra regler, regler som databasen inte tar hänsyn till, vilka begränsar möjligheten att ändra vissa attribut. Den största anledningen till detta är att inte helt låsa fast sig i formateringen av datatrafik. En administratör kanske vill lägga in oformaterad data.

I dagsläget finns det inget gränssnitt för en eventuell administratör, däremot kan det läggas in manuellt direkt i koden.

Webbtjänsten tillhör helt Back-End och har ingen tydlig interaktiv funktionalitet med en användare. Den returnerar JSON-objekt till alla som skickar HTTP-förfrågningar vilket kan vara en säkerhetsrisk. MD5-kryptering har använts för att undvika detta. Därmed sparas inte den känsligaste datan i klartext.

5.3 Webbapplikation

Webbapplikationen är den som ändrats mest under utvecklingens gång. Eftersom detta är det som användaren ser och det är den del som innehåller mest kod, är det ytterst viktigt att både kod och design är väl genomtänkt. Det visade sig också vara den del vars funktioner var svårast att implementera.

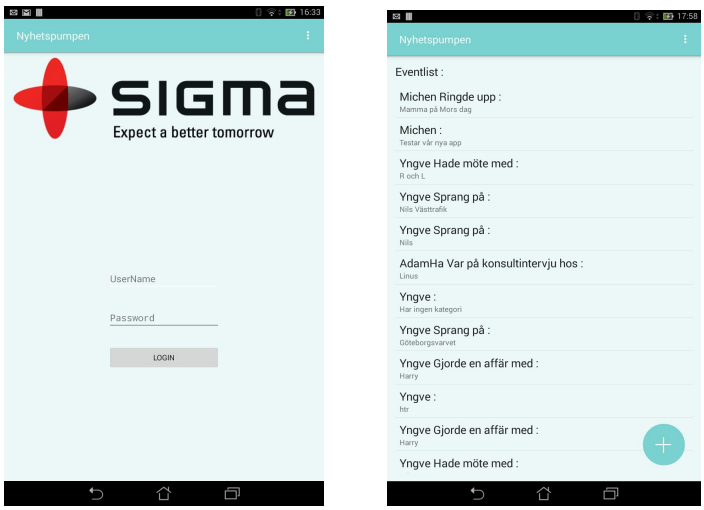


Figur 5.1 - Eventlistan i webbmiljö

Resultatet av webbmiljön är, i dess funktionella aspekt, lyckad. Alla funktioner som efterfrågades är utvecklade och implementerade. Designen av webbmiljön uppfyllde inte riktigt de önskemål som fanns för att få ett tilltalande utseende som skulle locka de anställda på Sigma att ta del av det nya nyhetsflödet. Det saknas ett par vanliga funktioner till webbmiljön, bland annat inloggningsfunktion, möjlighet att ändra givna uppgifter, skapa eller ta bort nytt event och detaljerad informationssida om varje event. Dessa delar är några av de funktioner och tillägg som hade varit önskvärda. Men sammantaget blev det funktionell presentering av nyhetsflöden med stora möjligheter för förbättring och utbyggnad.

5.4 Android-Applikation

Applikationen till Android har utvecklats med ett minimum SDK-version av 8. Detta gör att den kan köras på i princip alla Android-enheter som finns idag. Det gör den dock lite svårare att förstå kodmässigt. Detta beror på att många hjälpmetoder som finns idag inte kunde användas.



Figur 5.2 Inloggning och nyhetsflöde för Android

Fullt fungerande funktioner skapades för applikationen, men på grund av tidsbrist kan dessa funktioner vara mindre optimalt kodade. Detta kan påverka den optimala prestandan på hur snabbt och smidigt applikationen kan köras.

Resultatet med Android blev mer eller mindre som väntat - en fungerande, enkel applikation med stöd för i princip alla Android-enheter ute på marknaden idag. Den kan styras överallt där det finns internetuppkoppling. Den är enkel att använda med givna standardfraser. Med ett par enkla knapptryckningar kan hela företaget ta del av nyhetsmeddelandet. Tyvärr utvecklades inte något stöd för iOS eller något annat mobilt operativsystem, på grund av tidsbrist.

5.5 Raspberry Pi

Målet med att använda Raspberry Pi var att kunna visa en tydlig notifikation på kontoret då ett nytt meddelande har skrivits. Detta görs med hjälp av en LED-slinga kopplad till Raspberry Pi-enheten. När ett meddelande publiceras från Android-applikationen framför LED-slingan en ljussekvens.

Dessa mål har mötts. Raspberry Pi initierar LED-slingan så fort ett meddelande har publicerats via Android-applikationen. LED-slingan lyser i olika mönster och färger beroende på vilken kategori meddelandet tillhör.

Raspberry Pi lämpar sig utmärkt för hantering av notifikationer. På grund av dess användarvänlighet är det lätt att komma igång och snabbt att utveckla sina applikationer. Det finns många verktyg och resurser för att bygga projekt.

Efter några veckor kraschade minneskortet till Raspberry Pi-enheten, men tack vare att det är enkelt och smidigt att använda Raspberry Pi var allt återställt och fungerande inom några timmar.

5.6 Hållbarhetsaspekt

Nyhetsflödet med Raspberry Pi är ett verktyg som kan användas internt för att hålla sig uppdaterad, informerad och motiverad kring de miljömässiga ansträngningar man gör på Sigma. Genom nyhetsflödet kan man informera de anställda om nya och pågående initiativ för att främja miljön. Det är ett verktyg som kan användas för att engagera de anställda i miljöarbetet och strävan mot en hållbar utveckling.

5.7 Planering

Planering med hjälp av Scrum, Trello och tvåveckors Sprints har åskådliggjort arbetet och konkretiserat vad som behöver göras och när. Detta har varit en viktig del i att få rätt arbetsuppgifter gjorda och i rätt tid. Tack vare en väl utarbetat planering har det alltid framgått hur väl utvecklingen går och om arbetet utförs i tillräckligt hög takt. Detta har lett till ökat fokus på rätt områden, vilket har visat sig vara oerhört viktigt. Därmed värdesätts planeringsarbetet mycket högt.

Tack vare tvåveckors Sprint-möten har kontakten med kunden underhållits och eventuella nya önskemål kunde tas om hand direkt. På grund av den kontinuerliga kommunikationen har meningsskiljaktigheter kunnat lösas innan eventuella problem har blivit för stora.

5.8 Helhetsintryck

Sigma uttryckte en önskan om fokus på funktionalitet. Detta har lyckats, men har lett till att betydligt mindre tid har kunnat läggas på gränssnittet och utseendet på applikationerna. Sett till helhet har dock den funktionalitet som önskades uppfyllts.

5.9 Fortsatt arbete

Utbyggnaden och potentialen för fortsatt arbete är stor. Sigma ämnar att fortsätta detta arbete. Främsta fokusområden är bland annat en applikation för iOS, att förbättra grafikdesignen och att implementera användarlogin.

6. REFERENSER

[1] Interaction Design Foundation. 2016. *Mock-ups*. [ONLINE] Available at: https://www.interaction-design.org/literature/book/the-glossary-of-human-computer-interaction/mock-ups. [Accessed 1 April 2016].
[2] Ken Schwaber, Jeff Sutherland. 2013. *The Scrum Guide*. [ONLINE] Available at: http://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-us.pdf. [Accessed 30 March 2016].
[3] Mozilla Developer Network. 2015. *CSS Developer Guide*. [ONLINE] Available at: https://developer.mozilla.org/en-US/docs/Web/Guide/CSS. [Accessed 15 April 2016].
[4] Garcia-Molina H., Ullman J.D., Widom J., 2008. *Database Systems: The Complete Book, Second Edition*
[5] Microsoft ASP.NET. 2016. *MVC ASP.NET Documentation*. [ONLINE] Available at: https://docs.asp.net/en/latest/mvc/index.html. [Accessed 1 April 2016].
[6] Microsoft Developer Network. 2016. *Language-Integrated Query (LINQ) (C#)*. [ONLINE] Available at:https://msdn.microsoft.com/en-us/library/mt693024.aspx. [Accessed 12 April 2016].
[7] BlinkinLabs. 2016. *BlinkyTape - BlinkinLabs*. [ONLINE] Available at: http://blinkinlabs.com/blinkytape/. [Accessed 10 April 2016].
[8] Raspberry Pi Foundation. 2016. *What is a Raspberry Pi?*. [ONLINE] Available at:https://www.raspberrypi.org/help/what-is-a-raspberry-pi/. [Accessed 3 May 2016].
[9] Bootstrap. 2016. *Bootstrap - The world's most popular mobile-first and responsive front-end framework*. [ONLINE] Available at: https://getbootstrap.com/. [Accessed 17 April 2016].
[10] Trello. 2016. *Trello*. [ONLINE] Available at: https://trello.com/. [Accessed 3 April 2016].
[11] Postman. 2016. *Postman | Supercharge your API workflow*. [ONLINE] Available at:https://www.getpostman.com/. [Accessed 4 April 2016].
[12] Git. 2016. *Git*. [ONLINE] Available at: https://git-scm.com/. [Accessed 12 April 2016].
[13] Entity Framework Tutorial. 2016. *What is Code-First?*. [ONLINE] Available at:http://www.entityframeworktutorial.net/code-first/what-is-code-first.aspx. [Accessed 1 April 2016].
[14] ASP.NET. 2016. *Get started with ASP.NET Web Sites | The ASP.NET Site*. [ONLINE] Available at:https://www.asp.net/get-started/websites. [Accessed 31 March 2016].
[15] Dot Net Tricks. 2013. *What is Web API and why to use it?*. [ONLINE] Available at: http://www.dotnet-tricks.com/Tutorial/webapi/VG9K040413-What-is-Web-API-and-why-to-use-it-?.html. [Accessed 31 March 2016].
[16] Agile Methodology. 2016. *The Agile Movement*. [ONLINE] Available at: http://agilemethodology.org/. [Accessed 4 April 2016].
[17] Microsoft TechNet. 2016. *User Datagram Protocol (UDP): TCP/IP*. [ONLINE] Available at:https://technet.microsoft.com/sv-se/library/cc785220(v=ws.10).aspx. [Accessed 18 April 2016].