Requirements and Analysis Document for scheduling application

Markus Grahn, Oliver Andersson, Christian Lind, Victor Cousin, Moa Berglund

28/9 2020

version 5

1. Introduktion

På arbetsplatser är schemaläggning en central arbetsuppgift för att verksamheten ska fungera. Detta är ofta en mycket tidskrävande uppgift, trots att den är repetitiv. Därav är detta projektets syfte att skapa en applikation som automatiskt genererar ett schema utifrån vad verksamheten kräver, och därav minskar det administrativa arbetet på ett företag. Applikationen tar fram schemat utefter certifikat för olika arbetsuppgifter och tillgänglighet hos de anställda. En fördel med generatorn är att den tar hänsyn till egenskaper hos de anställda för att maximera kapaciteten på arbetsstyrkan. Programmet är tillgängligt för olika branscher då innehavaren själv kan bestämma vilka krav som verksamheten har. Den förväntade användaren arbetade tidigare med schemaläggningen, och har ansvar för schemat.

1.1 Ordlista

2. Krav

2.1 User stories

ID är utsatta efter prioritet (01= högst)
"Som x, vill jag kunna y, för att z."

“ID 01: Planering as programstruktur
Som programmerare vill jag tänka igenom strukturen av projektet innan jag börjar koda för att undvika onödiga komplikationer längre in i utvecklingen.”
Acceptans:

- *Det finns genomtänkt domänmodell*
- *Det finns genomtänkt designmodell*
- *Det finns UMLdiagram för klass/paket/modul-nivå*

“ID:02 Anställa/Avskeda personal
Som användare
vill jag kunna lägga till och ta bort personal
för att hålla programmet uppdaterat om de som anställda just nu.”
Acceptans:

<u>funktionell</u>:

- *Det finns en sida där det går att lägga till personal via personuppgifter (textfält) i formulär*
  namn, tel, mail, personnr, typ av anställning, certifikat
- *Personalen som läggs till syns i någon form av lista.*
- *Det går att ta bort personal från listan*

“ID:03 Hantera anställdas certifikat
Som användare
vill jag kunna lägga till och ta bort certifikat hos de anställda
för att hålla programmet uppdaterat om kunskapsbanken just nu.”
Acceptans:

<u>funktionell</u>:

- *Det går att tilldela certifikat till anställd*
- *Det går att skapa nya certifikat*
- *Det går att se vilka certifikat en anställd har*
- *Det går att se vilka anställda som har ett visst certifikat*
- *Det går att ta bort certifikat från anställd genom att trycka på någonknapp någonstans? I listan?*
- *Det går att ta bort certifikat från “möjliga certifikat”*

“ID:04 Kolla bemanning på avdelningar
Som användare
vill jag kunna veta att alla avdelningar är bemannade utan att jag manuellt behöver göra det,
för att frigöra tid till annat för mig.”
Acceptans:

<u>funktionell</u>:

- *Programmet genererar visuellt schema automatiskt*
- *Schemat visar hur många som krävs en viss tid*

<u>ickefunktionell:</u>

- *Användaren behöver inte lägga tid på schemaskapandet annat än när programmet efterfrågar*

“ID:05 Krav på avdelningar
Som användare
vill jag kunna se till att alla avdelningar är bemannade med rätt certifikat

för att verksamheten ska fungera på bästa sätt.”
Acceptans:

<u>funktionell</u>:

- *Certifikat för berörd avdelning bestäms av ägaren*
- *Endast behörig personal med rätt certifikat får arbeta på en avdelning som kräver det certifikatet*
- *För en avdelning går det att lägga till/ta bort certifikat när det kommit nya krav*

“ID: 06 Schemahantering
Som programmerare vill vi kunna hantera schemat på så sätt att vi kan hämta dagar och tider för att lägga in pass.

Funktionell:

- Hämta tider
- Lägga in pass

“ID:07 GUI ver 1
Som programmerare vill jag skapa ett körbart “GUI-skal” med javaFX för att ha någon form av gränssnitt att utgå och utveckla från”.
Acceptans:

<u>Funktionell</u>:

- *Det finns något körbart av GUI*

“ID:08 Notis vid inkomplett schema
Som användare
vill jag bli underrättad ifall det skulle bli brist på personal någon dag
för att manuellt fylla luckan.”
Acceptans:

<u>Funktionell:</u>

- *“Notis” visas vid brist av personal någon dag*
- *Berörd dag visar en varningstriangel i schemat*

“ID:09 Välja hur ofta man jobbar
Som anställd
vill jag inte kunna ha olika pass tätare än x timmar
för att ha en hälsosam livsstil.”
Acceptans:

<u>funktionell:</u>

- *Ägaren av programmet kan bestämma x antal timmar som de anställda garanteras vara lediga innan nästa arbetspass*
- *Programmet gör anställda vars lediga tid är mindre än x “otillgängliga”.*

- *Det går inte att byta ett pass till ett annat som ligger mindre än x timmar mellan den berördas redan tilldelade pass utan att acceptera förhållandet*
- *Schemat visar vilka tider man inte är tillgänglig för att kunna planera annat*

“ID:10 Garantera alla rast
Som användare
vill jag veta att synkronisering av raster fungerar på så vis att bemanning fortfarande är intakt
för att ha ett bra arbetsflöde.”
Acceptans:

funktionell:

- Användaren kan fylla i hur mycket rast som garanteras efter y timmar
- Användaren kan fylla i hur många som inte får ha rast samtidigt per avdelning
- Schemat visar vilka raster man har

“ID:11 Hantera arbetspass
Som användare
vill jag kunna lägga till och ta bort arbetspass utefter behov
för att det kan hända oväntade företeelser som gör att vi behöver mer/mindre bemanning.”
Acceptans:

Funktionell:

- *Det går att ta bort arbetspass*
- *Användaren kan skapa nya arbetspass (valfri tid och avdelning)*
- *Det går att ha olika scheman olika veckodagar och tider*
- *Ändra bemanning på redan upplagda pass*

“ID:12 Inga felaktiga pass
Som användare
vill jag kunna byta personal på arbetspass i programmet
för att de anställdas scheman ska stämma även när förändringar görs.”
Acceptans:

funktionell:

- *//Användare kan byta pass mellan personal*
- *Programmet låter inte samma person göra två arbetspass samtidigt*
- *Programmet låter inte obehörig personal arbeta på en avdelning*
- *Programmet låter inte byta pass med någon som är ifylld “inte tillgänglig” den dagen*

“ID:13 Visuell separation av anställda i schemat
Som användare

vill jag kunna se vilka som arbetar och deras tider under dagen
för att samtidigt kunna fundera ut eventuella byten när efterfrågan finns.”
Acceptans:

Funktionell:

- Schemat är visuellt synligt
- Schemat visar vilka som arbetar under dagen och deras tider och raster
- Tider och raster är lätt urskiljbara

“ID:14 Visuell separation mellan avdelningar
Som användare
vill jag kunna urskilja i schemat vilka som arbetar på vilken avdelning
för att enkelt kunna uppsöka berörd personal under arbetstid.”
Acceptans:

Funktionell:

- *Schemat är visuellt synligt och särskiljer avdelningarna*
- *Avdelningarna särskiljs mha olika färger*

“ID:15 Kommentera koden
Som deltagare i ett gemensamt programmeringsprojekt vill jag att koden ska kommenteras väl för att enkelt kunna förstå vad de övriga har kodat.”
Acceptans:

Icke-funktionell:

- *Koden täcks övergripande av javadoc*

“ID:16 Exceptions
Som programmerare vill jag att de ska finnas tydliga felmeddelanden när något inte går som det ska för att förstå vad som är fel.”
Acceptans:

Funktionell:

- *Exception finns för när någon metod returnerar null*

“ID:17 Junit
Som programmerare vill jag kunna testa stor del av programkoden för att undvika buggar.”
Acceptans:

Icke-funktionell:

- *Junit är implementerad*

“ID:18 Separerade testklasser

Som programmerare vill jag ha en test-klass för varje klass i modellen för att få en bättre överblick av testerna.”
Acceptans:

Icke-funktionell:

- *Testmetoder är placerade i testklasser för respektive klass som testas*

“ID:19 Maven
Som programmerare vill jag implementera maven i projektet för att möjliggöra build automation.”
Acceptans:

Icke-funktionell:

- *Maven är implementerad*

----------NYA USER STORIES----------

dessa ska prioriteras och utvecklas:

2.2 Definition of Done

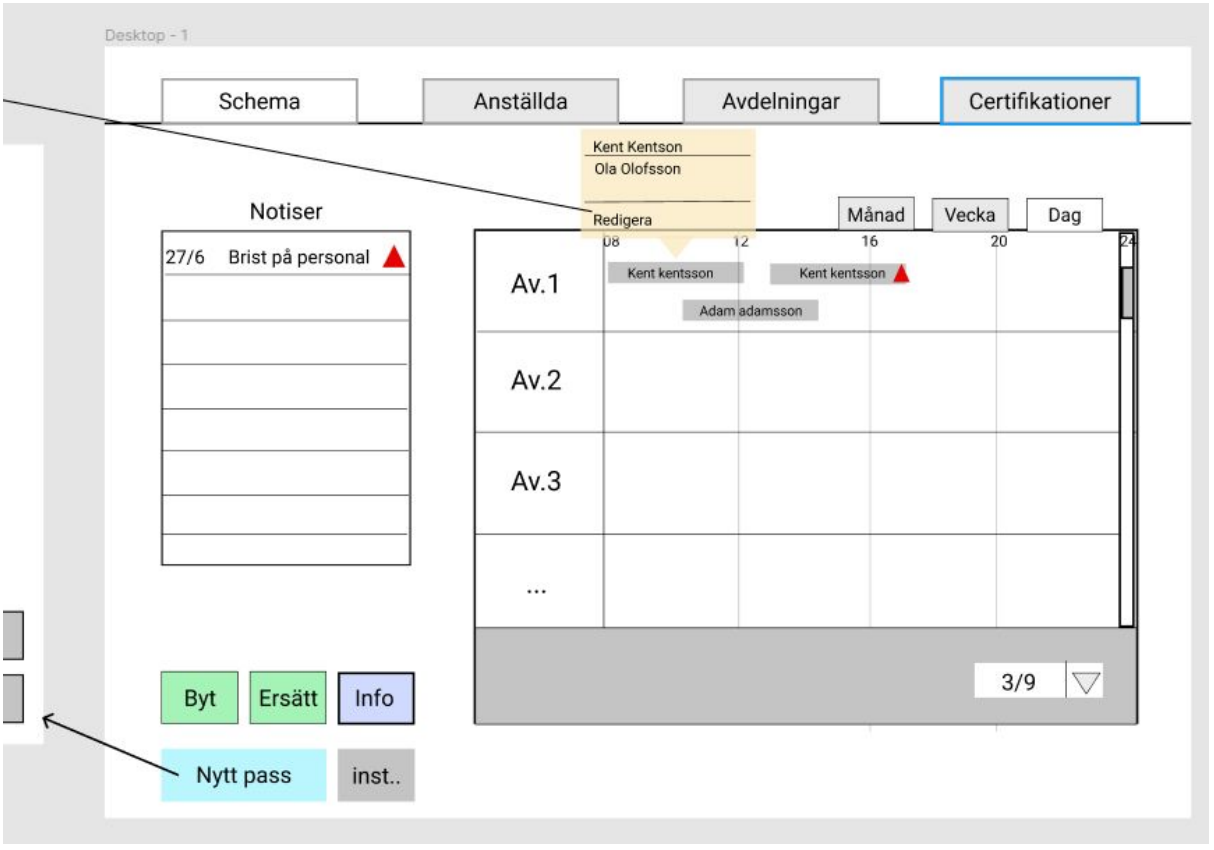
funktionell:

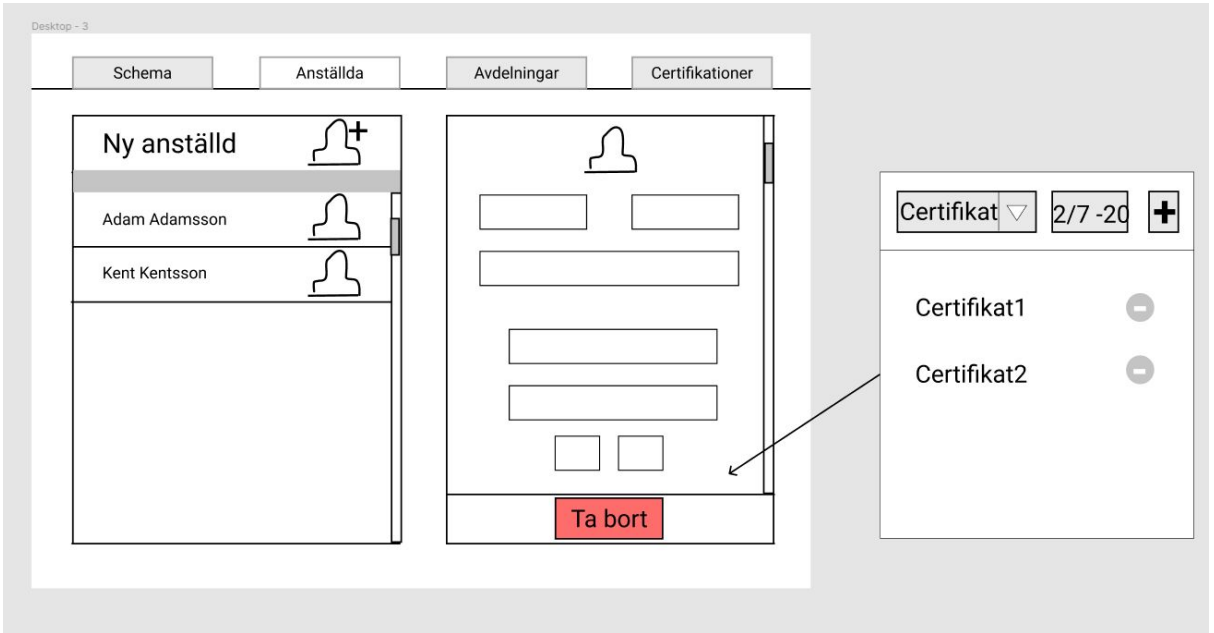
- Arbetsgivare kan mata in företagsspecifika uppgifter såsom anställda, certifikat, avdelningar, x(se ovan) och pass
- visuellt schema genereras automatiskt
- det är möjligt att uppdatera företagsuppgifter
- det är möjligt att byta pass
- det är möjligt att ta bort och lägga till pass
- schemat visar vilka som jobbar, vilken tid och avdelning

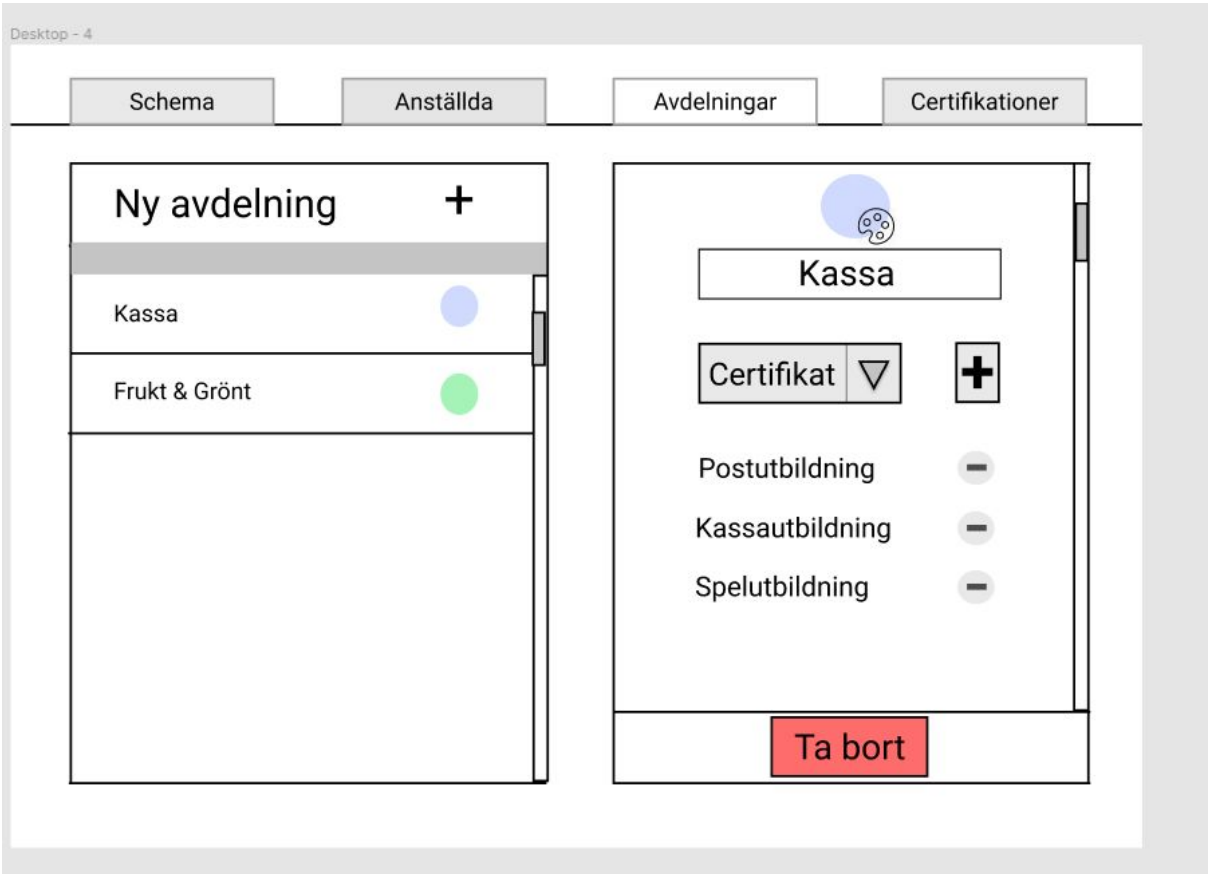
icke-funktionell:

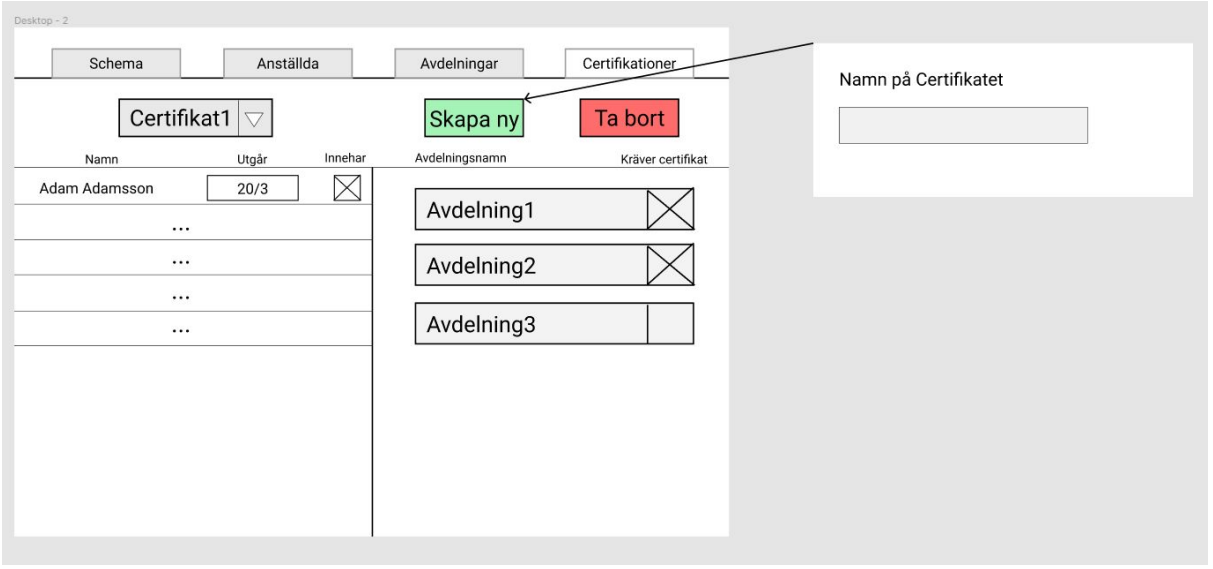
- programmet är utförligt testat med tydlig struktur av tester
- koden är kommenterad med javadoc

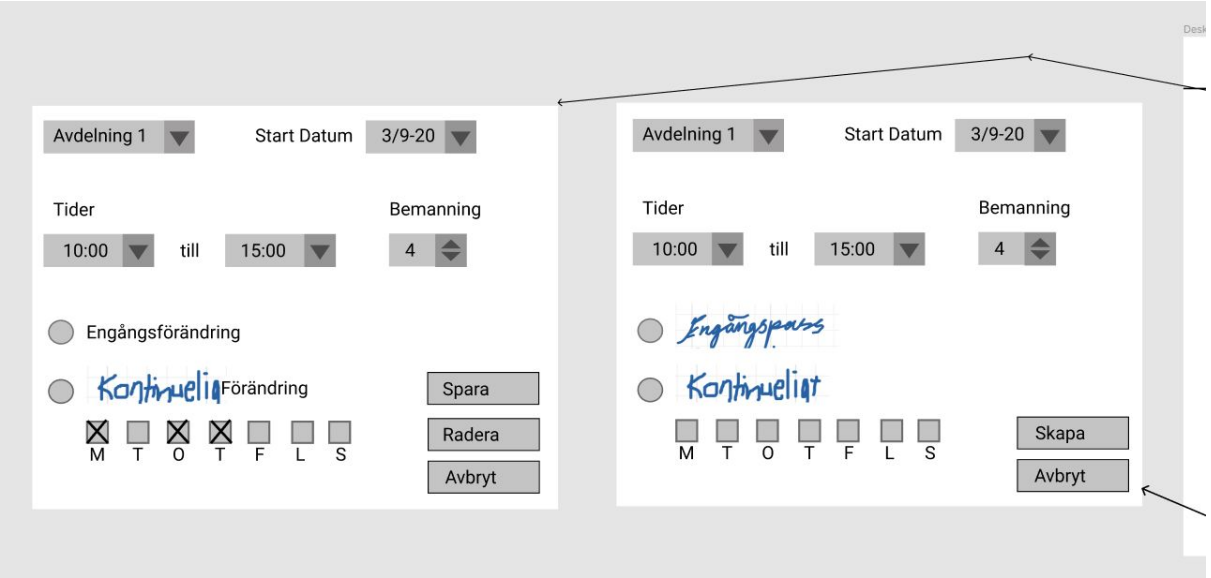
2.3 User Interface






Desktop - 4
Schema
Anställda
Avdelningar
Certifikationer
Ny avdelning
+
Kassa
Frukt & Grönt
Kassa
Certifikat
Postutbildning
Kassautbildning
Spelutbildning
Ta bort

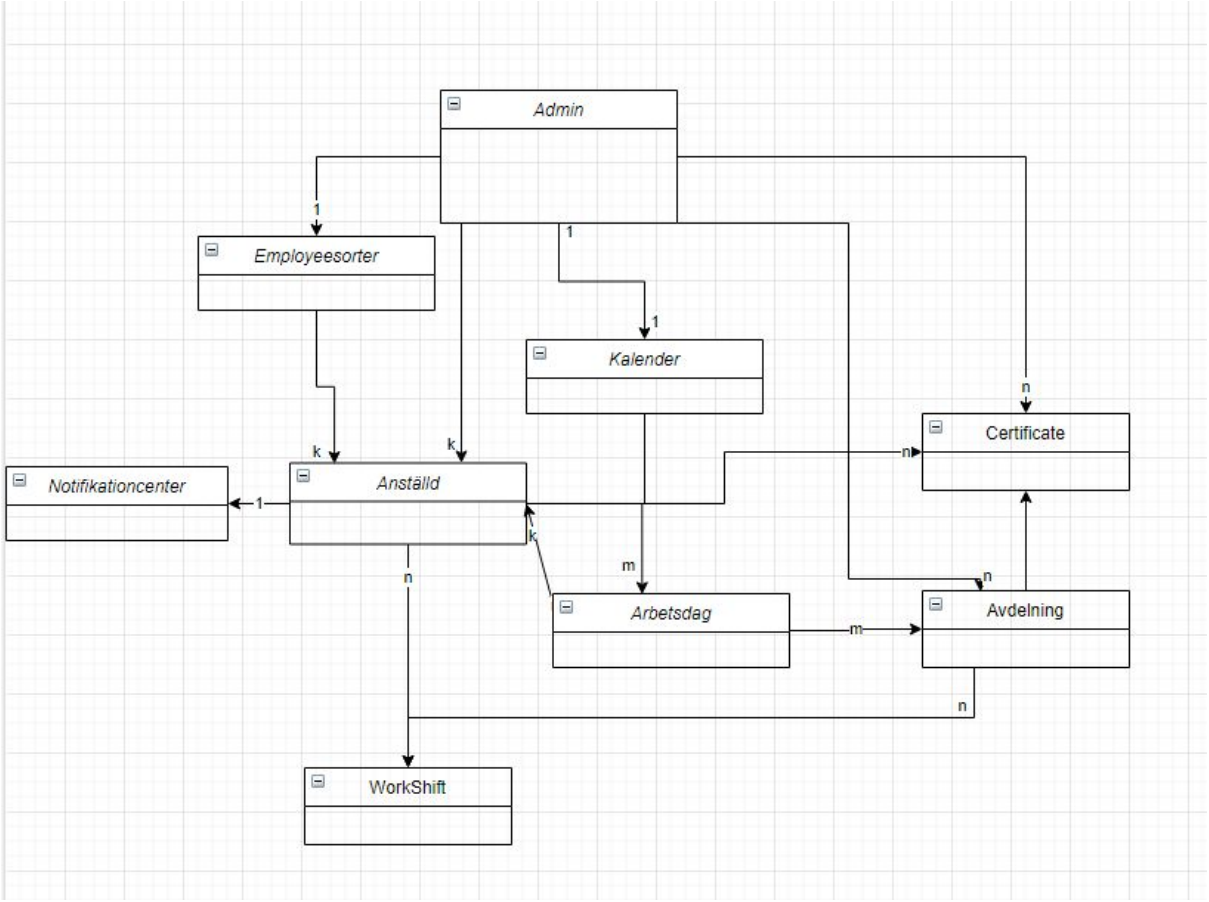

Desktop - 2
Schema
Anställda
Avdelningar
Certifikationer
Certifikat1
Skapa ny
Ta bort
Namn
Utgår
Innehar
Avdelningsnamn
Kräver certifikat
Adam Adamsson
20/3
...
...
...
...
Avdelning1
Avdelning2
Avdelning3
Namn på Certifikatet



3. Domain model

3.1 Class responsibilities

- Admin:Huvudklassen som tar ser till att all kommunikation mellan klasserna fungerar som det ska. Delegerar även olika uppgifter till de olika klasserna.
- Exporter: Används så att man kan exportera sin kalender till valfritt program så att man enklare kan komma åt sitt schema.
- EmployeeSorter: Sorterar alla anställda utifrån vilka certifikat den anställda innehar så att man inte hamnar på en avdelning som behöver andra certifikat. Ser även till så att antalet som jobbar på en avdelning inte understiger minst antal personer samtidigt.
- Employee: Representerar en anställd med personuppgifter.
- Calender: Representerar en kalender där man kan se månad, vecka och dag samt vilka tider olika anställda är upptagna.
- Date: Gör så att man kan se vecka, dag och månad samt att man kan få vilka anställda som jobbar en specifik tid. Kan även kolla på olika avdelningar om vilka som är insatta olika tider för att representeras i Calender.
- Apartment: En avdelning där man har vilka certifikat som krävs och hur mycket personal som behöver jobba där.
- Notificationcenter: Skickar en notifikation när man har blivit tilldelad ett pass.
- OccupiedTime: Visar vilka tider man kan jobba och inte



4. References