# System design document for scheduling application

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

02/10 2020

version 1

## 1 Introduktion

Att skapa ett väl fungerande schema är en mycket viktig del för att få till en stabil arbetsmiljö. Men scheman och personalen som ska följa schemat är dynamiska och kan förändras snabbt vilket kan rubba stora delar av schemat då alla krav kan vara svåra att fylla upp. Det är därför NF (new five) har utvecklat en schemaläggare som snabbt och effektivt ska kunna placera ut anställda på ett schema efter deras behörigheter och erbjuda ett lätt sätt för att snabbt kunna strukturera om ett schema.

### 1.1 Ordlista

## 2 Systemarkitektur

Det första som händer när applikationen startar är en prompt om handlingar, “Ny fil”, “Ladda fil”, “Spara & Avsluta”. Väljer man “ny fil” körs programmet utan argument och man får en blank kanfas att arbeta på. Väljer man istället “Ladda fil” så läser programmet in all gammal data från särskilda textfiler om alla anställda, avdelningar och tillhörande information såsom certifikat. Detta för att man enkelt ska kunna spara all information när man stänger programmet samt att man ska kunna använda olika versioner av programmet fast ändå ha kvar samma data.

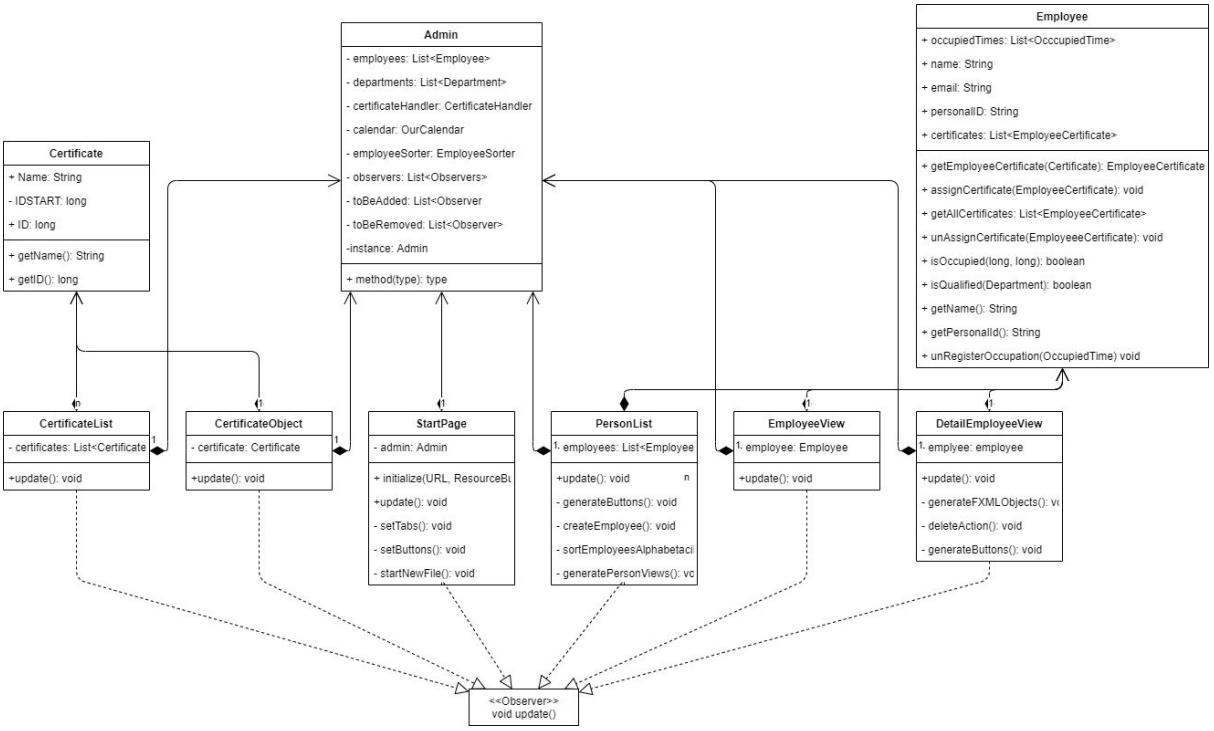
När man väl startat kommer man till en vy där man kan redigera information om anställda, certifikat, avdelningar och även ändra arbetspass. Det är möjligt att skapa nya avdelningar och därefter skapa arbetspass och olika krav i form av certifikat för den specifika avdelningen. Sedan kan man delegera arbetare som har rätt utbildning i form av certifikat på dessa arbetspass. Detta går att göra både manuellt av schemaläggaren men också automatiskt med hjälp av en sorterare som finns i programmet. Denna sorterare har en algoritm som ser till att alla pass fylls med behörig personal.

Efteråt kan man med hjälp av ett schema se de olika arbetspassen, där man även får information om det skulle saknas arbetare på något pass. För att man inte ska missa detta finns en funktion som skickar både ett mejl och en notis i applikationen utifall ett arbetspass

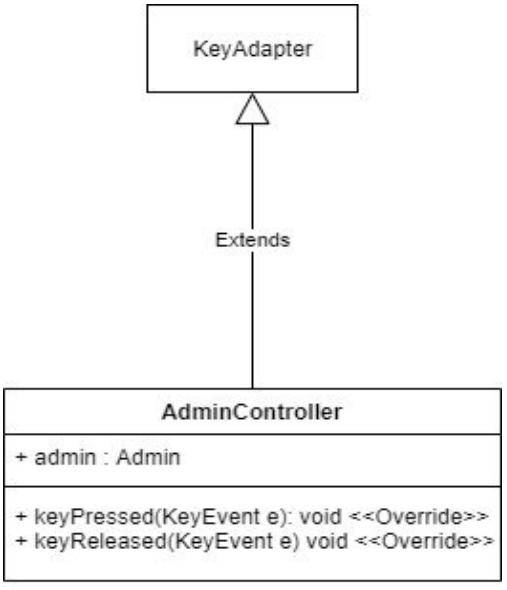
inte skulle vara bemannat så att användaren kan ordna detta, antingen manuellt eller via sorteraren.

# 3 Systemdesign

View:
Använder sig av observer pattern för att behålla mvc. Admin har också en getInstance()-funktion för att kunna skapa nya viewklasser utan att ändra tidigare skapade klasser.

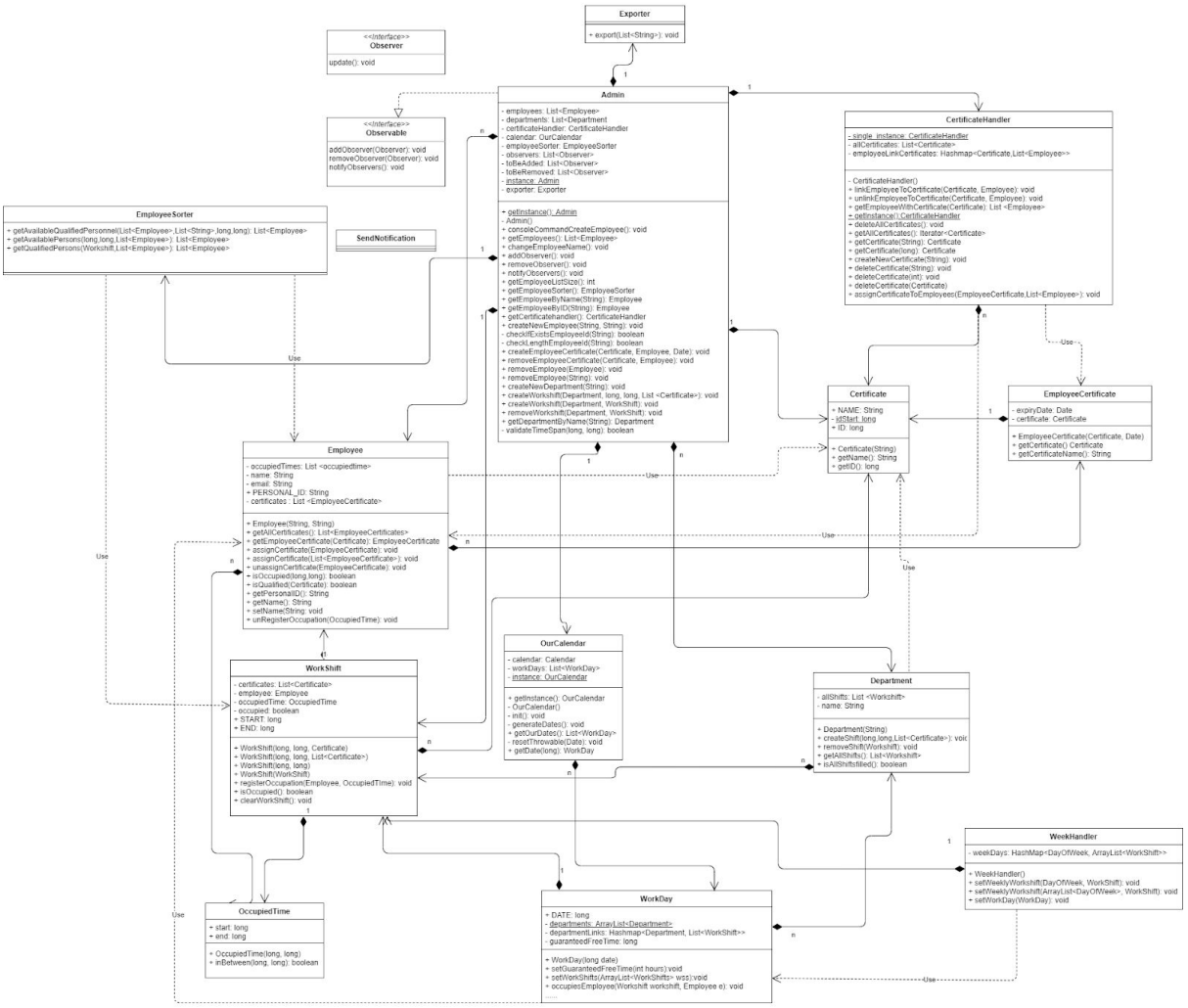



Controller:

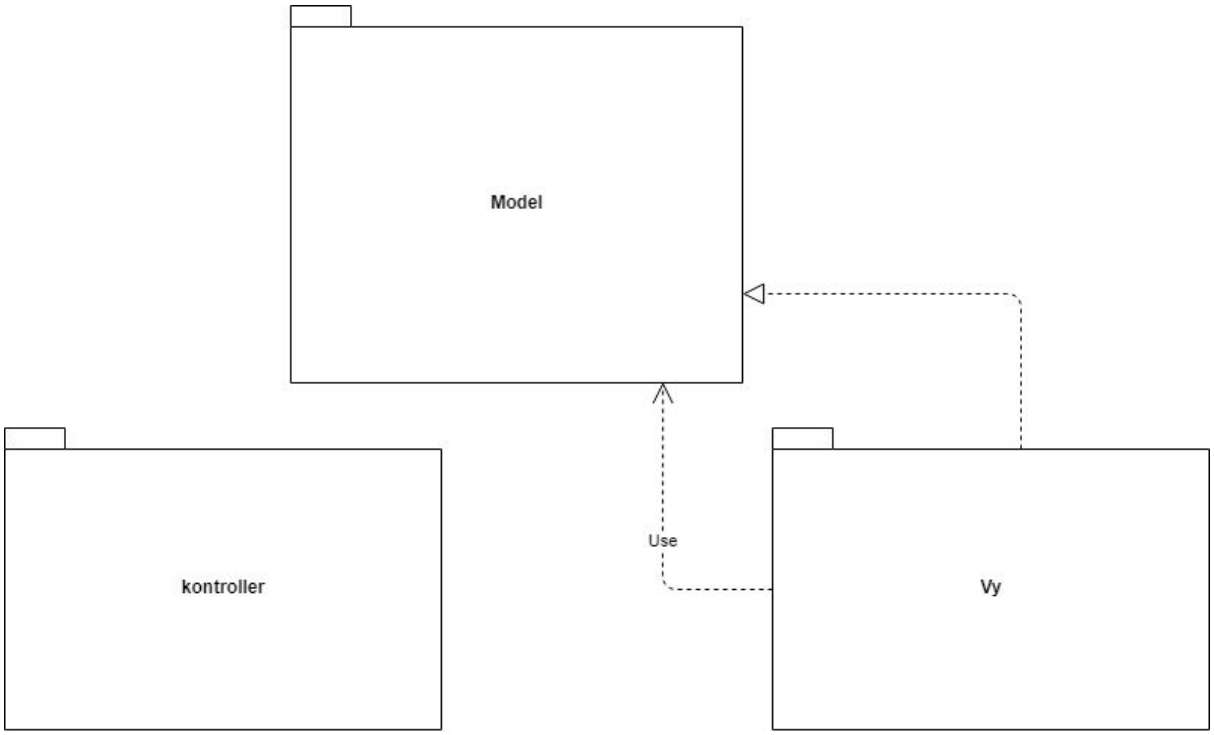



(används ej för närvarande)

Model Package:



Module-level:




Design Patterns:

- Singleton - används då Admin, OurCalender, CertificateHandler enbart går att skapa en gång, för att alltid använda enbart den instansen
- Observer- för att kunna uppdatera vyn utan att vyen ska behöva fråga efter uppdatering från model- den uppdateras när uppdatering finns då modellen notifierar vyn om det

# 4 Datahantering

För att på ett smidigt och enkelt sätt lagra data så valdes att alla listor (ArrayList och List) ska lagras i varsin textfil (.txt) eftersom man i förväg aldrig vet hur stora listorna är och det kan bli väldigt svårt att hitta information om allt ligger i samma fil. När det kommer till alla HashMap (HashMap och Map) så valdes att det ligger en fil bland de andra textfilerna med varje nyckel. Detta ligger i nummerordning vilken gör det lätt att skapa en ny mapp som har samma namn som variabelnamnet på HashMap. Detta gör att det är enkelt att hitta vilken lista eller variabel som korresponderar till nyckeln. Eftersom det finns andra variabler som man redan i förväg vet längden på så valdes här istället att det finns en textfil per Javapaket där de finns ordnade i en logisk struktur.

Eftersom man ska kunna ladda gamla scheman så kommer allt att sparas i en .zip där all data läggs och för att det ska vara enkelt att hitta döps den till dagens datum och tid. t.ex. 11-06-2020-09-29 för den tiden som är nu. Detta gör att man sedan enkelt kan ladda den .zip man vill. När man valt en .zip som man vill använda så unzippar applikationen filen och

lägger den i applikationens mapp så att den enkelt kan nå all data och sedan omzippa den nya datan.

# 5 Kvalitet

För varje klass så skapas en tillhörande testklass under en testmapp med samma namn + test. Målet med dessa testklasser är att testa varje funktion av den tillhörande klassen. Problemet är att man inte kan förutspå när eller om det kommer fallera i ett gränsfall då gränsfall inte är lätt definierade när variablerna är så ickelinjära.
För stunden(2020-10-02) finns det inte “header comments”. Alla metoder är inte testade ännu.
Dependency-analys finns i git-projektet under dependencyAnalysis/“dependency” + versionnummer

## 5.1 Åtkomstkontroll och säkerhet

I framtiden ska något liknande en login-funktion finnas men för närvarande(2020-10-02) saknas det stöd både view och model. Anledningen till att en säkerhetskontroll såsom log in är åtråvärt kan vara att vi hanterar personlig information samt att en obehörig ej ska kunna ändra i schemat, t.ex. en anställd som inte är nöjd med sitt schema.

# 6 Referenser

- JavaFX
- Maven
- IntelliJ stöd för dependency analys