Requirements and Analysis Document for scheduling application

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

23/10 2020

version 7

1. Introduktion

På arbetsplatser är schemaläggning en central arbetsuppgift för att verksamheten ska fungera. Detta är ofta en mycket tidskrävande uppgift, samtidigt som att den kan vara repetitiv. Därav är detta projektets syfte att skapa en applikation som schemalägger personal utifrån vad verksamheten kräver, och därav minskar det administrativa arbetet på ett företag. Användaren fyller själv i företagets uppgifter såsom anställda, avdelningar etc. Applikationen sätter sedan schemat utefter certifikat för olika arbetspass som användaren bestämt och tillgänglighet hos de anställda. Programmet är tillgängligt för olika branscher då innehavaren själv kan bestämma vilka krav som verksamheten har. Den förväntade användaren arbetar med schemaläggningen, och har ansvar för schemat. Applikationen begränsar sig till enbart denna användare, och erbjuder ingen funktionalitet eller gränssnitt för personalen som blir schemalagd.

1.1 Ordlista

Certifikat - arbetsrelaterad licens som en anställd kan bli tilldelad

2. Krav

2.1 User stories

ID är utsatta efter prioritet (01= högst). ID markerat med lila är implementerade, grönt är inte implementerade
"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:

funktionell:

- *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:

funktionell:

- *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 någon knapp 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 arbetsdagen bemannad utan att jag manuellt behöver göra det, för att frigöra tid till annat för mig."
Acceptans:

funktionell:

- *Programmet genererar visuellt schema*
- *Schemat visar hur bemanningen är en viss tid*
- *användaren kan lägga in vilka pass en arbetsdag kräver för att vara bemannad korrekt*

ickefunktionell:

- *Användaren behöver inte lägga tid på att skapa ett schema 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:

funktionell:

- *Certifikat för berört arbetspass bestäms av ägaren*
- *Endast behörig personal med rätt certifikat får arbeta på ett pass som kräver vissa certifikat*
- *För ett arbetspass 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:

Funktionell:

- *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:

Funktionell:

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

“ID:09 Välja hur ofta anställda jobbar
Som anställd
vill jag inte kunna ha olika pass tätare än x timmar

för att ha en hälsosam livsstil.”
Acceptans:

funktionell:

- *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”.*

“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 Ledighet
Som schemaläggare vill jag inte att någon ska kunna schemaläggas som har fått godkänd ledighet."
Acceptans:

funktionell:

- Det går att göra en person "upptagen" vid bestämd tid/datum så att det ej går att schemalägga vid ledighet

“ID:12 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:13 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:

<u>funktionell:</u>

- *Användare kan byta pass mellan personal*
- *Programmet låter inte samma person göra två arbetspass samtidigt*
- *Programmet låter inte obehörig personal arbeta på certifikat-krävda pass*
- *Programmet låter inte byta pass med någon som är ifylld "inte tillgänglig" den dagen*
- *Det går inte att byta ett pass till ett annat som ligger mindre än x timmar mellan den berördas redan tilldelade pass*

"ID:15 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:

<u>Funktionell:</u>

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

"ID:16 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:

<u>Icke-funktionell</u>:

- *Koden täcks övergripande av javadoc*

"ID:17 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:

<u>Funktionell</u>:

- *Exception finns för när någon metod returnerar null*

"ID:18 Junit
Som programmerare vill jag kunna testa stor del av programkoden för att undvika buggar."
Acceptans:

<u>Icke-funktionell</u>:

- *Junit är implementerad*

“ID:19 Separerade testklasser
Som programmerare vill jag ha en test-klass för varje klass i modellen för att få en bättre överblick av testerna.”
Acceptans:

<u>Icke-funktionell</u>:

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

“ID:20 Maven
Som programmerare vill jag implementera maven i projektet för att möjliggöra build automation.”
Acceptans:

<u>Icke-funktionell</u>:

- *Maven är implementerad*

2.2 Definition of Done

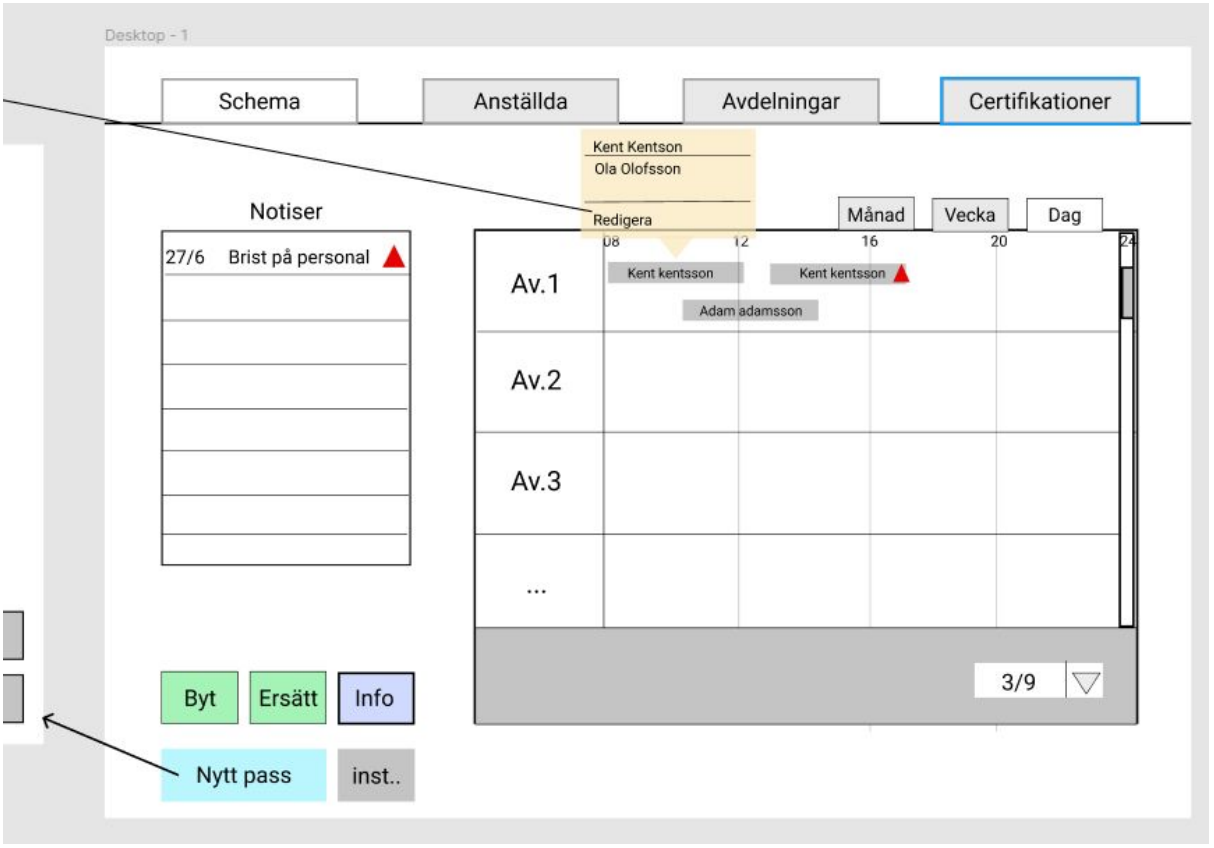
funktionell:

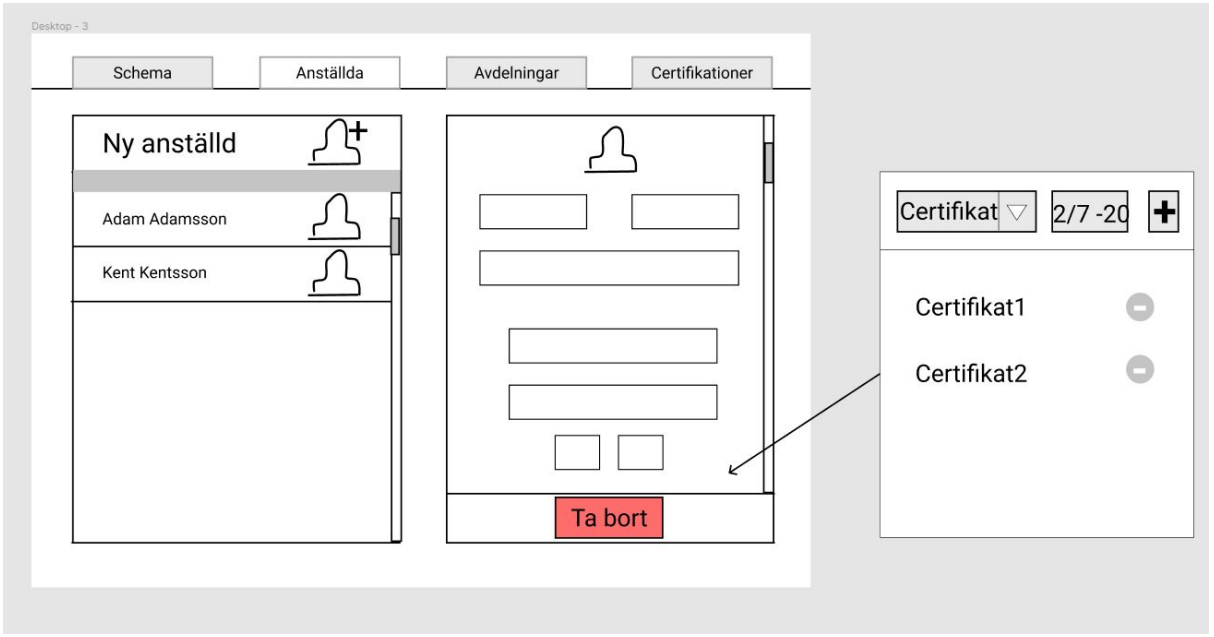
- Arbetsgivare kan mata in företagsspecifika uppgifter såsom anställda, certifikat, avdelningar, x(se ovan) och pass
- visuellt schema genereras
- det är möjligt att uppdatera företagsuppgifter
- det är möjligt att byta pass
- det är möjligt att ta bort och lägga till pass
- schemat visar vilka som jobbar,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

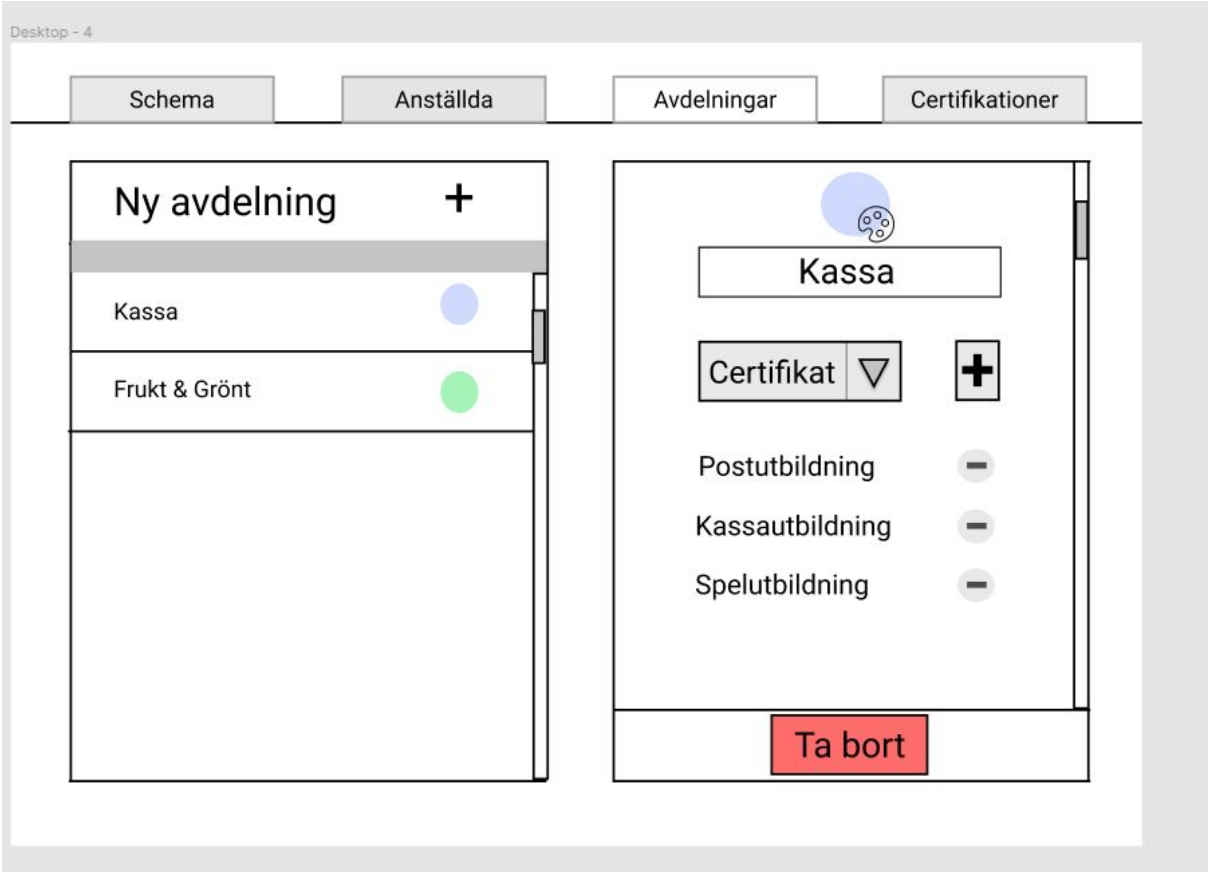
icke-funktionell:

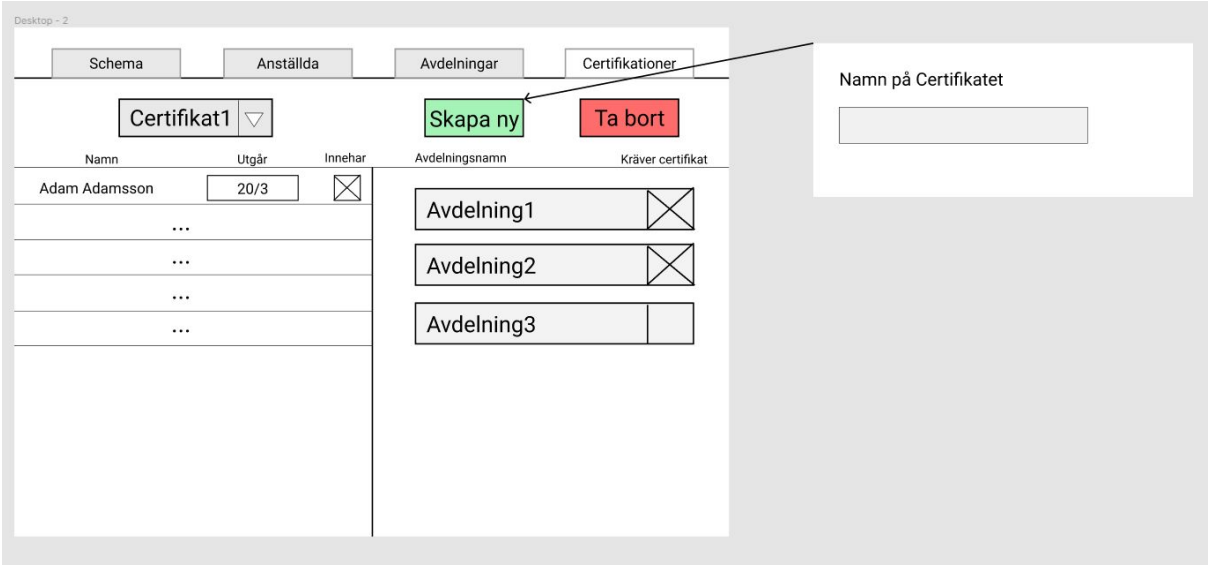
- programmet är utförligt testat med tydlig struktur av tester
- koden är kommenterad med javadoc
- alla klasser innehåller header comments innehållande author, responsibility, used by, uses
- Användaren behöver inte lägga tid på att skapa ett schema annat än när programmet efterfrågar

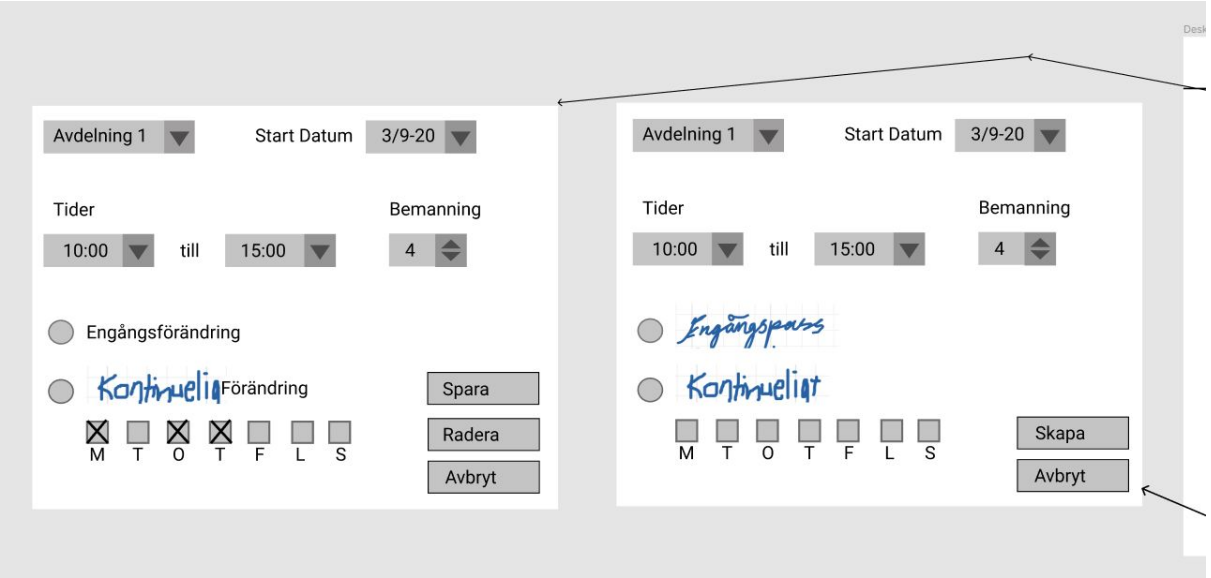
2.3 User Interface



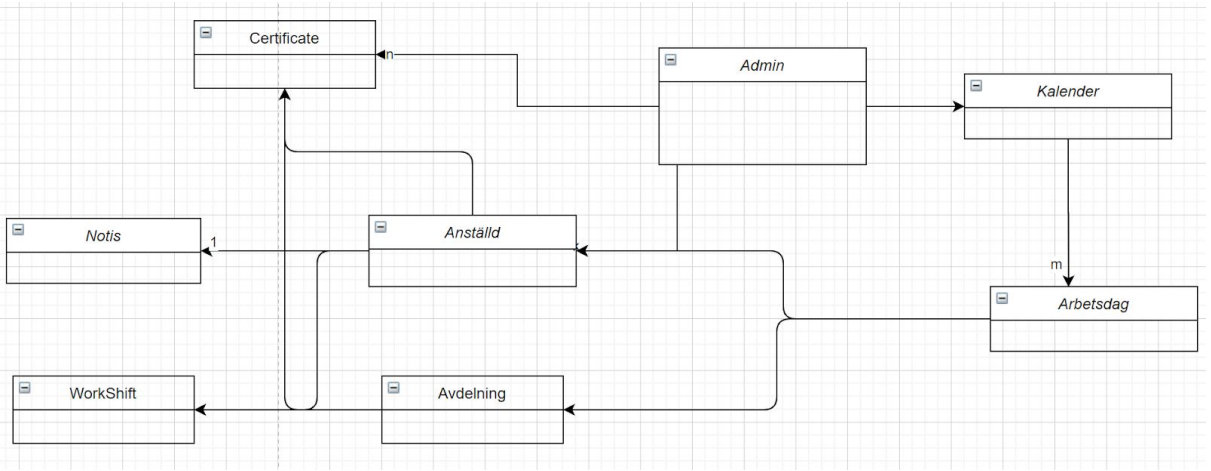



Desktop - 4
Schema
Anställda
Avdelningar
Certifikationer
Ny avdelning
+
Kassa
Frukt & Grönt
Kassa
Certifikat
Postutbildning
Kassautbildning
Spelutbildning
Ta bort


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. **Domänmodell**



3.1. Klassernas ansvarsområden

- Admin: Huvudklassen som ansvarar för kommunikation mellan klasserna. Den delegerar olika uppgifter till de olika klasserna för att inte ha kommunikation/beroenden mellan många olika klasser.
- BreakHandler: Hanterar längden av raster
- Certificate: Representerar certifikat som arbetspass kan kräva av personalen för att kunna jobba på just det arbetspasset.
- CertificateHandler: Hanterar certifikat, vilket innebär att den både förmedlar och tar bort certifikat från anställda (via Admin).
- Department: Representerar en avdelning med alla dess planerade arbetspass. Avdelningen ser till att alla som jobbar samtidigt på avdelningen blir tilldelade raster utifrån hur avdelningen minst måste vara bemannad.
- Employee: Representerar en anställd med dess personuppgifter och tilldelade certifikat. Har även koll på när den inte kan jobba, alltså är beviljad ledighet eller när den redan arbetar.

- EmployeeCertificate: Representerar certifikat som en anställd kan bli tilldelad. EmloyeeCertificate kan göras tidsbestämda.
- EmployeeSorter: Sorterar ut alla anställda på arbetspass utifrån vilka certifikat de har så att anställda inte blir schemalagda på en arbetspass som kräver andra certifikat. Notifierar admin om schemaläggningen resulterar i att pass inte går att fylla med lämplig personal. *Just nu tar programmet inte hänsyn till vilken typ av anställning personal har (ex 50% osv), utan den schemalägger så den anställde med minst ockuperad tid blir tilldelad i första hand om den är lämplig (är inte upptagen och har rätt certifikat)*.
- ImportServicePackage: Hårdkodad data som modellen hanterar som om den kom från en laddad fil om användaren väljer att “ladda in fil” i GUI
- Login: Möjliggör att användaren måste skriva in användarnamn och lösenord för att komma till startsidan av applikationen
- OccupiedTime: Representerar vilka tider en anställd inte är tillgänglig för nya arbetspass. Admin väljer hur lång ledighet som lovas efter ett arbetspass.
- OurCalendar: Representerar en kalender fylld med arbetsdagar(Workday) som är möjlig att hantera i månader, veckor och dagar.
- SendNotification: Skickar ett mail till anställdas mailadress när de blivit tilldelad ett pass och mail till admins mailadress när ett pass inte gick att fylla vid schemaläggning.
- WeekHandler: Har statiska metoder som möjliggör att enkelt addera tid till ett date-objekt utan att varje gång gör om till millisekunder.
- WorkDay: Representerar en arbetsdag på ett visst datum som håller koll på alla arbetspass på den specifika dagen, vilka avdelningar dessa tillhör och om de är fyllda eller inte. Möjliggör byten mellan olika arbetspass och förändring av bemanning om de är kvalificerade.
- WorkShift: Representerar ett arbetspass som kan upprepas om så är valt. Detta har själv koll på om det är bemannat eller ej och kan tilldela anställda till det.

4. Referenser

- JavaFX https://openjfx.io/
- Maven https://maven.apache.org/
- IntelliJ stöd för Dependency Analys
- Javax.activation https://docs.oracle.com/javase/8/docs/api/javax/activation/package-summary.html
- Javax.mail https://javaee.github.io/javamail/