

Hovedopgave

KEA, datamatiker, DAT21A-NY

Kasper Scott Jensen

kasp324f@stud.kea.dk

<https://github.com/kasperscottjensen/final>

Indholdsfortegnelse

| | |
|---|----|
| 1. Indledning..... | 1 |
| 2. Problemformulering..... | 2 |
| 2.1 Indledning..... | 2 |
| 2.2 Problemformulering..... | 2 |
| 2.3 Metode..... | 2 |
| 2.4 Forventet resultat..... | 2 |
| 3. Kundeintroduktion..... | 3 |
| 4. Overblik..... | 4 |
| 4.1 Vision..... | 4 |
| 4.2 Omfang og krav..... | 4 |
| 4.3 Scope..... | 5 |
| 4.4 Skalérbarhed..... | 5 |
| 5. Metode..... | 6 |
| 5.1 Fremgangsmåde..... | 6 |
| 5.2 Teknologier..... | 6 |
| 5.3 Prototyping..... | 7 |
| 5.4 Product Backlog..... | 8 |
| 5.5 Udvikling..... | 8 |
| 5.5.1 Sprint 1: 13. - 19. november..... | 9 |
| 5.5.2 Sprint 2: 20. - 26. november..... | 9 |
| 5.5.3 Sprint 3: 27. november – 3. december..... | 10 |
| 5.5.4 Sprint 4: 4. - 10. december..... | 10 |
| 5.5.5 Sprint 5: 11. - 17. december..... | 11 |
| 5.5.6 Sprint 6: 18. - 24. december..... | 11 |
| 6. Produkt..... | 12 |
| 6.1 Plugins..... | 12 |
| 6.2 Gennemgang af vigtige features..... | 13 |
| 6.2.1 Brugerhåndtering og -autentificering..... | 13 |
| 6.2.2 Databaser..... | 16 |
| 6.2.3 Kalender og ønskeliste..... | 17 |
| 6.2.4 Navigation (sitemap)..... | 18 |
| 6.2.5 Personlig profil og indstillinger..... | 19 |

| | |
|--|----|
| 6.2.6 Sikkerhed og personlige oplysninger..... | 20 |
| 7. Diskussion..... | 21 |
| 7.1 Udfordringer..... | 21 |
| 7.1.1 Tidshåndtering..... | 21 |
| 7.1.2 Fokus..... | 21 |
| 7.2 Mangler..... | 22 |
| 7.2.1 Test..... | 22 |
| 7.2.2 Udvid til flere teams..... | 22 |
| 7.2.3 Administratorpanel..... | 22 |
| 7.2.4 Algoritme..... | 22 |
| 7.2.5 Supportsystem..... | 23 |
| 8. Konklusion..... | 24 |

1. Indledning

Denne rapport dækker udvikling af en vagtplanlægningsapp i forbindelse med den afsluttende prøve på datamatikuddannelsen hos Københavns Erhvervsakademi, vinteren 2023/2024. Rapporten gennemgår den trinvis udvikling af produktet, samt de løbende overvejelser der er gjort. Produktet har til formål at vise forståelse af den viden, der er blevet tilegnet i løbet af uddannelsen, og vil derfor benytte metoder og løsninger, der er gennemgået i undervisningen.

2. Problemformulering

2.1 Indledning

- Identifikation af behovet for en intuitiv og avanceret vagtplanlægningsapp, der kan tilpasses forskellige branchers krav og optimere planlægningsprocessen for at forbedre arbejds effektiviteten.
- Identifikation af vigtigheden af en skalerbar og fleksibel vagtplanlægningsapp, der kan tilpasses individuelle virksomheders behov og imødekomme den stigende efterspørgsel efter digitale planlægningsværktøjer på tværs af forskellige industrier.

2.2 Problemformulering

- Hvordan kan design og udvikling af en vagtplanlægningsapp optimeres for at sikre brugervenlighed og samtidig imødekomme virksomheders behov for tilpasning og skalerbarhed på tværs af forskellige brancher?
- Hvilke teknologiske løsninger og platforme bør overvejes for at sikre en pålidelig ydeevne og brugertilfredshed på tværs af varierende virksomhedskrav?

2.3 Metode

- Anvendelse af agile udviklingsmetoder og prototyping til iterativt at designe og implementere forskellige funktioner og brugergrænseflader til produktet.
- Valg af passende udviklingsrammer og teknologier baseret på en vurdering af appens skalerbarhedsbehov på tværs af forskellige brancher.

2.4 Forventet resultat

- Bidrag til forbedring af virksomheders drift gennem en pålidelig og tilpasselig vagtplanlægningsapp med fokus på brugervenlighed.
- Implementering af en robust teknisk arkitektur, der sikrer stabil ydeevne og tilfredsstillende brugeroplevelse på tværs af varierende behov og aktivitetsniveauer i forskellige industrier.

3. Kundeintroduktion

Opgaven er løst uden en kunde i sigte, da produktet i sin natur er rettet mod et bredt publikum. For dog at have en uvildig tredjepart til review, har der været løbende kontakt med samme firma som jeg var i praktik hos. Dette er Aim Robotics ApS – et firma, der udvikler end-effektorer til kollaborative robotter med hovedsæde i Søborg. Firmaets feedback har fungeret som rettesnor for udviklingen af brugergrænseflade og overordnet funktionalitet, men har ikke haft direkte indflydelse på udviklingen af produktet. Gennem praktikforløbet har jeg haft intim kontakt med hvilke funktioner og hvilken struktur, der forventes af en bruger, der ikke har decideret teknisk kunnen. Disse informationer har dermed lagt grund til selve navigationen i og udseendet af applikationen. Firmaets oplysninger er beskrevet her:

Navn: Aim Robotics ApS

Adresse: Maskinvej 5, 2860 Søborg, Danmark

Tlf.: +45 31 51 55 11

Email: contact@aim-robotics.com

CVR: 40494197

Web: <https://www.aim-robotics.com/>

4. Overblik

4.1 Vision

Visionen er at skabe grundlaget for en vagtplansapplikation, der strømliner og forsimples måden hvorpå firmaer kan holde overblik over deres medarbejderes timer. Dette skal gøres ved udvikling af en platform, der både er nydelig at se på, og som leverer den nødvendige funktionalitet for, at leder såvel som ansat har let adgang til statistikker, ønsker, team og videre. Applikationen skal holde fokus på brugervenlighed, let tilgængelige features og hastighed, således at brugeroplevelsen bliver præget af effektivitet i stedet for frustration.

4.2 Omfang og krav

Nedenstående er en liste over krav, der som udgangspunkt kan danne rammerne for en vagtplansapplikation. Kravene er ikke endegyldige, og vil blive taget til eftertragtning som projektet skrider fremad. Dette grundet opgavens natur og det faktum, at der kun er en enkelt udvikler. Disse krav ses dog som værende ganske overkommelige og fundamentale i et system af denne slags.

| Funktionelle krav | Ikke-funktionelle krav |
|--|---|
| Brugerhåndtering: <ul style="list-style-type: none">- Registrering og autentificering- Personlige profiler- Personlige indstillinger | Design: <ul style="list-style-type: none">- Responsivt- Brugervenlig grænseflade |
| Vagtplan: <ul style="list-style-type: none">- Kalender- Brugerdefinerede views | Ydeevne: <ul style="list-style-type: none">- Skalérbarhed- Minimal ventetid |
| Ønskeliste: <ul style="list-style-type: none">- Kalender- Brugerdefinerede views- Eventhåndtering (ny, rediger, slet)- Flydende og intuitiv interaktion | Platform: <ul style="list-style-type: none">- Skal fungere på kryds af enheder |
| Kollaboration: <ul style="list-style-type: none">- Kommunikation mellem ansatte- Live chat | |
| | |

| | |
|---|--|
| Statistik - Forhenværende vagter - Kommende vagter - Tilstedeværelse | |
| Tilstedeværelse: - Check ind / ud - Overblik | |
| Landing: - Visuelt overblik over funktionalitet - Gennemgang af produkt | |

4.3 Scope

Som nævnt i introduktionen er rammerne om projektet dannet omkring undervisningen i løbet af de forløbne år, og mulige implementeringer som en algoritme eller AI-løsning til generering af vagtplan ud fra brugernes ønsker er altså dermed blevet tilsidesat. I stedet vil der blive lagt vægt på et fundament af layouts og komponenter, der kan bruges til videre udvikling af programmet, skulle muligheden eller nødvendigheden opstå. Scopet vil altså være afgrænset til at møde de angivne krav, men med det forbehold, at applikationens grundsten (læs: kalendersystem, brugerhåndtering etc) vejer tungere end højtekniske implementeringer og afslutningsvise tilføjelser (læs: algoritmer, supportchat etc).

4.4 Skalérbarhed

Produktet til aflevering ved endt projekt bliver funktionsdygtigt til et enkelt firma, men bliver modulært opbygget således, at implementering af skalérbarhed let kan indføres. Dette fordi at der ville tages meget tid fra anden funktionalitet, udelukkende for at udvikle for en situation, der ikke kommer til at indtræffe i dette projekt. Det ses dermed som værende unødvendigt på nuværende tidspunkt. Al brugerhåndtering vil dermed fremgå som det ville i en given virksomhed, men funktionalitet på tværs af disse vil være udeladt fuldstændigt.

5. Metode

5.1 Fremgangsmåde

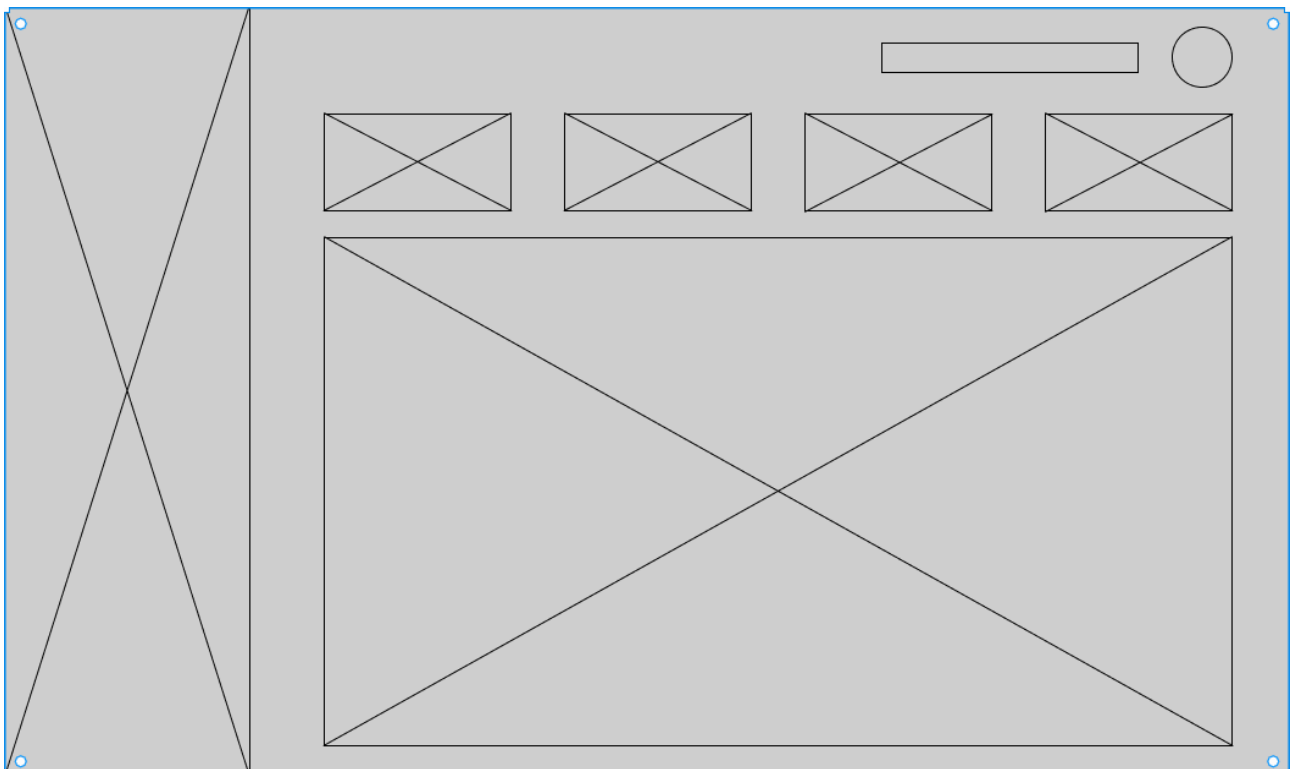
Til udviklingen blev der benyttet en letvægts SCRUM-opsætning, der tog udgangspunkt i de mest essentielle aspekter. Frameworket blev brugt hovedsageligt til at vedligeholde en forsvarlig kvalitet af arbejde og overblik. Det skal dog bemærkes, at en del af frameworket er udeladt, da det som enkeltperson ikke gav mening at overholde alle punkter. Idéen er, at udarbejde en mere eller mindre fuldstændig backlog, der kan nedbrydes i sprint, således at udviklingen ved hvert endt sprint har en hvis funktionalitet og sammenhold. Ved endt sprint er det ligeledes anledning til at lave et review med den uvildige tredjepart.

5.2 Teknologier

Da en applikation af denne typer kræver en hurtig og responsiv brugeroplevelse for at fungere efter hensigten, blev frontend øjeblikket sat til at være Svelte. Svelte indbyggede store-funktionalitet gør det oplagt til udvikling af applikationer, hvor værdier skal både opdateres og vises hyppigt. Subscriptions til disse stores gør det let at arbejde som en SPA, og dynamisk at hente og vise data fra hvilken som helst kilde det måtte være. Sveltes routingsystem gør det ligeledes let at åbne eller lukke navigation baseret på – for eksempel – en brugers autentificering. Disse kombineret vil give et hurtigt og let grundlag til en velfrundet klient. Til backend er det svært at undgå Node.js. Node har en overlegen evne til at inkorporere hvilken som helst tredjepart, det skulle være. Dette gælder både serverteknologi såvel som databaser. Man kan lynhurtigt opsætte routing ved hjælp af Express, og derefter skrive til en af mange databaser, for eksempel SQLite, MongoDB eller lignende. En versatil løsning, der er nem at koble til en Svelte-klient.

5.3 Prototyping

Da klienten bliver skrevet i Svelte, har jeg valgt at benytte Tailwind CSS til templating. Tailwind har et indbygget row/column-system, der gør det let at lave en struktureret og gennemført opbygning af grænsefladen. Disse teknologier gør det også muligt at udvikle modulært, og dermed genbruge mange af de komponenter, der bliver udviklet løbende. Baseret på dette gav det mening at udvikle det generelle layout i samme modulære natur. Grænsefladen er bygget op af en sidebar til navigation og et view. Viewet opdateres når der navigeres, men beholder dog en header, der kan indeholde vigtig information til brugeren. Denne indeholder også et cirkulært avatar, samt et søgefelt.



Disse elementer danner det grove layout efter succesfuldt login. De vigtigste informationer og statistikker kan vises uanset view, og efterlader samtidig masser af plads til gnidningsfri navigation og overblik. Landingsiden er en simpelt opbygget one-page med scroll-navigation.

5.4 Product Backlog

| Layout | Landing | Brugerhåndtering |
|---|--|---|
| <ul style="list-style-type: none">- Stylingsheets- Layoutmoduler- Grafikker- Opsætning RWD | <ul style="list-style-type: none">- Navigation- CTA- Grafikker- Introduktion til features | <ul style="list-style-type: none">- Login- Registrering- Personlige profiler- Personlige indstillinger |
| Grænseflade | Planlægningsfunktioner | Kommunikation |
| <ul style="list-style-type: none">- Navigation- Basemoduler | <ul style="list-style-type: none">- Kalender- Ønskeliste- Brugerdefinerede visninger- Events CRUD- Tilstedeværelsessystem- Check ind / ud | <ul style="list-style-type: none">- Live chat- Team overview- Beskedsystem- Søgefunktion |
| Statistik | Test | Support |
| <ul style="list-style-type: none">- Statistikmoduler- Grafer- Tilstedeværelsesstatistik | <ul style="list-style-type: none">- API- RWD- Data | <ul style="list-style-type: none">- Brugerdokumentation- Supportsystem |

5.5 Udvikling

Herunder ses et overblik over de seks sprints. Det bemærkes at kun fem sprints blev gennemført under projektet. Sprint seks er bestående af de resterende elementer fra backloggen, og blev ikke nået i denne udviklingsperiode. Estimer er givet før hvert sprint baseret på viden fra tidligere udviklet lignende elementer. Prioritet er sat efter hvor mange andre komponenter, der er afhængig af denne. Som eksempel er layoutmoduler i sprint 1 sat til høj, da disse skal benyttes som byggeklodser i resten af applikationen.

Høj = Vil tilbageholde andre elementer hvis forsinket

Normal = Kan tilbageholde ikke-kritiske elementer hvis forsinket

Lav = Har lille til ingen chance for at tilbageholder andre elementer

Sprintene er forsøgt udarbejdet således at der forekommer et mere eller mindre brugbart produkt ved enden. Dette gælder naturligvis ikke fra opstart, da meget af grundarbejdet bliver udviklet her.

Ved endt første sprint var der knap nogen visuel fremgang, men til gengæld et godt fundament af genbrugelige komponenter til videre udvikling. Aim Roboticsrepræsentant og designguidelines blev benyttet efter hvert sprint til et kort review. Hovedsageligt angående brugergrænsefladen, men også for at se om brugsflowet var som en uvildig bruger ville forvente det. Dette hjalp til at udregne hvordan man lettere imødekommer en brugers naturlige vaner og forventninger til placering af elementer, navigation, farveskema og lignende.

5.5.1 Sprint 1: 13. - 19. november

| Task | Estimat | Prioritet |
|------------------|---------|-----------|
| Stylingsheets | 8 | Høj |
| Layoutmoduler | 16 | Høj |
| Grafikker | 8 | Normal |
| Opsætning af RWD | 8 | Lav |

I dette sprint blev de fundamentale komponenter udviklet. Det vil sige de byggesten som kan bruges til hurtigt at sammensætte elementer. Dette ved hjælp af de standard Tailwind stylesheets, og Svelte Material UI pluginet. Det vil sige at elementer som knapper, content-kort, skrifttyper, farver, og lignende alt sammen kunne skabes med disse værktøjer. Her blev også udviklet en grov skitse for layoutet for både dashboard og landing page. Grafikker er hentet fra Tailwind Resources.

5.5.2 Sprint 2: 20. - 26. november

| Task | Estimat | Prioritet |
|---------------------------|---------|-----------|
| Ekstern navigation | 4 | Høj |
| CTA | 12 | Normal |
| Grafikker | 8 | Lav |
| Introduktion til features | 16 | Lav |

Sprint 2 handlede hovedsageligt om opbygningen af en landing side. Komponenter fra sidste sprint blev benyttet til hurtigt at opsætte en one-page side, der forklarer om applikationen og dens features.

5.5.3 Sprint 3: 27. november – 3. december

| Task | Estimat | Prioritet |
|--------------------------|---------|-----------|
| Login | 8 | Høj |
| Registrering | 4 | Lav |
| Personlige profiler | 8 | Normal |
| Personlige indstillinger | 8 | Normal |
| Intern navigation | 2 | Høj |
| Basemoduler | 16 | Høj |

I dette sprint blev brugerhåndtering udviklet. Til dette benyttes Firebase. Firebase holder brugernes information i Firestore og Firebase Storage og kan ligeledes håndtere autentificering via Firebase Auth. Klienten benytter firebase-client-sdk og serveren benytter firebase-admin-sdk. Med disse er det let og overskueligt at administrere brugeres kommen og gåen, registrering og så videre. Der blev ligeledes udviklet moduler som forme, content-kort (fortsat), sidebar og lignende.

5.5.4 Sprint 4: 4. - 10. december

| Task | Estimat | Prioritet |
|----------------------------|---------|-----------|
| Kalender | 8 | Høj |
| Ønskeliste | 8 | Høj |
| Brugerdefinerede visninger | 4 | Lav |
| Events CRUD | 8 | Normal |
| Tilstedeværelsessystem | 8 | Normal |
| Check ind / ud | 8 | Normal |

I dette sprint blev selve kalenderne til planlægningen implementeret. Til dette blev benyttet et plugin til Svelte, der hedder Event Calendar. Der blev ligeledes udvikling serverside funktionalitet til behandling af events til visning i klienten. Udover kalender og ønskeliste, blev der udviklet et tilstedeværelsessystem, der logger en brugers check ind og ud.

5.5.5 Sprint 5: 11. - 17. december

| Task | Estimat | Prioritet |
|---------------------------|---------|-----------|
| Live Chat | 4 | Normal |
| Team overview | 8 | Normal |
| Beskedsystem | 4 | Lav |
| Søgefunktion | 4 | Normal |
| Statistikmoduler | 8 | Normal |
| Grafer | 8 | Normal |
| Tilstedeværelsesstatistik | 8 | Normal |

Her blev der udviklet resterende mindre kritiske elementer. En live chat blev implementeret via websocket. Beskedsystemet blev fraseret og erstattet med simple maillinks. Statistikmoduler blev udviklet og placeret i cards, og grafer blev udviklet med pluginet D3. Der blev oprettet stores til at holde den nødvendige information til at udregne statistikker og information, og Firestores tilladelser blev justeret således at alle teammedlemmer har adgang til ikke-kritisk information om hinanden (læs: information, de har tilgængelig andetsteds).

5.5.6 Sprint 6: 18. - 24. december (IKKE FULDFØRT)

| Task | Estimat | Prioritet |
|---------------------|---------|-----------|
| API Test | 8 | Normal |
| RWD Test | 8 | Normal |
| Data Test | 8 | Normal |
| Brugerdokumentation | 8 | Lav |
| Supportsystem | 8 | Lav |

6. Produkt

6.1 Plugins

For at lette arbejdsbyrden, blev der benyttet en række plugins. Nedenfor ses et udsnit af package.json for både klient og server. Der er ligeledes angivet links til de mest væsentlige.

6.1.1 Klient

```
"devDependencies": {
  "@beyonk/svelte-notifications": "^4.2.0",
  "@event-calendar/core": "^2.5.0",
  "@sveltejs/vite-plugin-svelte": "^2.0.3",
  "svelte": "^4.2.5",
  "svelte-check": "^3.6.2",
  "svelte-preprocess": "^5.1.1",
  "typescript": "^5.3.3",
  "vite": "^4.2.0"
},
"dependencies": {
  "@event-calendar/day-grid": "^2.5.0",
  "@event-calendar/interaction": "^2.5.0",
  "@event-calendar/list": "^2.5.0",
  "@event-calendar/resource-time-grid": "^2.5.0",
  "@event-calendar/time-grid": "^2.5.0",
  "@popperjs/core": "^2.11.8",
  "@rollup/plugin-typescript": "^11.1.5",
  "@tailwindcss/forms": "^0.5.7",
  "d3": "^7.8.5",
  "dotenv": "^16.3.1",
  "firebase": "^10.4.0",
  "js-cookie": "^3.0.5",
  "svelte-material-ui": "^7.0.0-beta.16",
  "svelte-routing": "^2.6.0",
  "tailwindcss": "^3.3.5",
  "uuid": "^9.0.1"
}
```

6.1.2 Server

```
"dependencies": {
  "cors": "^2.8.5",
  "dotenv": "^16.3.1",
  "express": "^4.18.2",
  "express-validator": "^7.0.1",
  "firebase-admin": "^11.10.1",
  "multer": "^1.4.5-lts.1",
  "sqlite3": "^5.1.6",
  "uuid": "^9.0.1",
  "ws": "^8.16.0"
},
"type": "module"
```

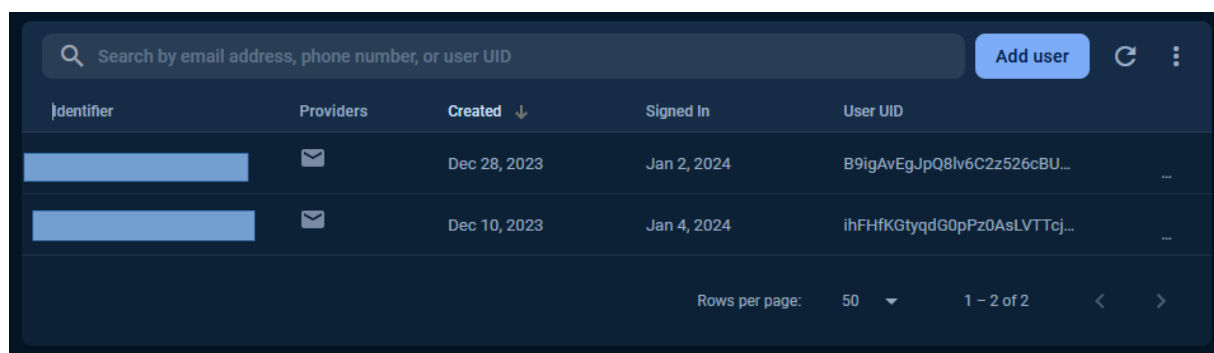
| Plugin | Brug | Link |
|--------------------|----------------|---|
| @event-calendar | Kalendersystem | https://github.com/vkurko/calendar |
| d3 | Grafer | https://www.npmjs.com/package/d3.chart |
| tailwindcss | Styling | https://tailwindcss.com/ |
| firebase | Storage / Auth | https://firebase.google.com/ |
| svelte-material-ui | UI komponenter | https://sveltematerialui.com |
| sqlite3 | Database | https://www.sqlite.org/ |

6.2 Gennemgang af vigtige features

I dette afsnit gennemgår jeg et par af de vigtigste features i programmet. Disse skal ikke forstås som guides, men blot give en hurtigt overblik over hvordan applikationen er sat op inde for disse aspekter.

6.2.1 Brugerhåndtering og -autentificering

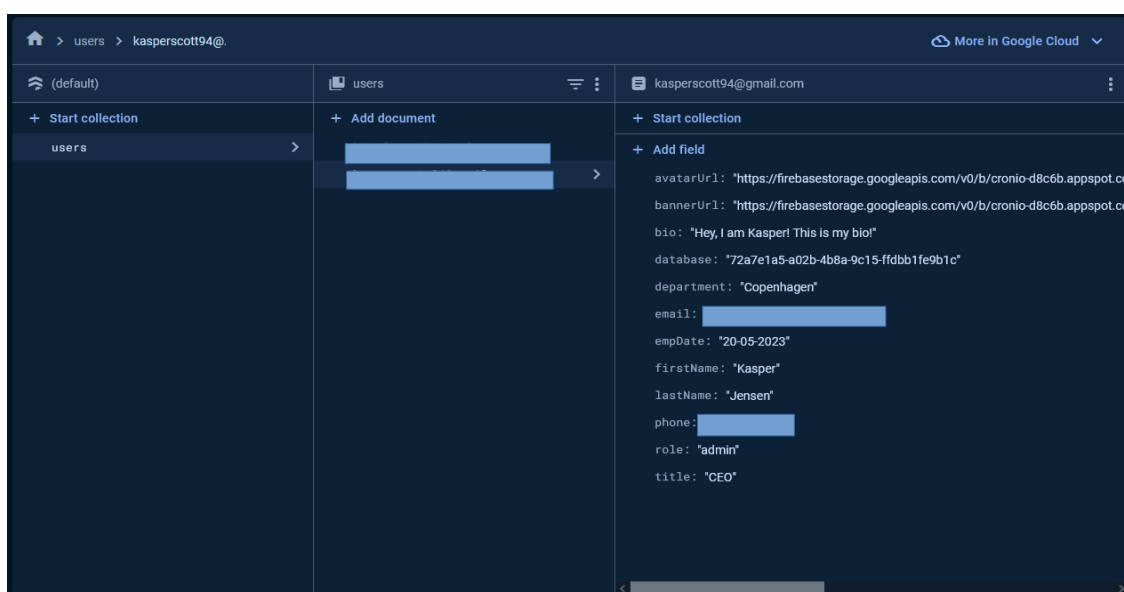
Autentificering foregår via Firebase. Når en bruger bliver oprettet, bliver denne automatisk overvåget af Firebases autentificeringssystem, og bruger får oprettet separate ressource- og datalokationer. Her ses brugerovervågningen:



The screenshot shows the Firebase console's 'Users' page. At the top, there is a search bar with the placeholder text 'Search by email address, phone number, or user UID' and an 'Add user' button. Below the search bar is a table with the following columns: Identifier, Providers, Created, Signed In, and User UID. There are two rows of user data. The first row shows a user created on Dec 28, 2023, signed in on Jan 2, 2024, with a User UID of B9igAvEgJpQ8lv6C2z526cBU... The second row shows a user created on Dec 10, 2023, signed in on Jan 4, 2024, with a User UID of ihFHfKGtyqdG0pPz0AsLVTTCj... At the bottom right, there is a pagination control showing 'Rows per page: 50' and '1 - 2 of 2'.

| Identifier | Providers | Created ↓ | Signed In | User UID |
|------------|--------------|--------------|-------------|------------------------------|
| [Redacted] | [Email Icon] | Dec 28, 2023 | Jan 2, 2024 | B9igAvEgJpQ8lv6C2z526cBU... |
| [Redacted] | [Email Icon] | Dec 10, 2023 | Jan 4, 2024 | ihFHfKGtyqdG0pPz0AsLVTTCj... |

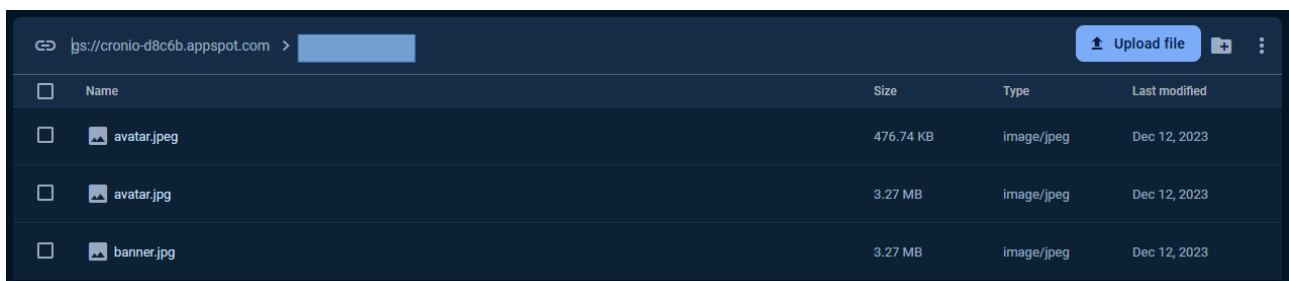
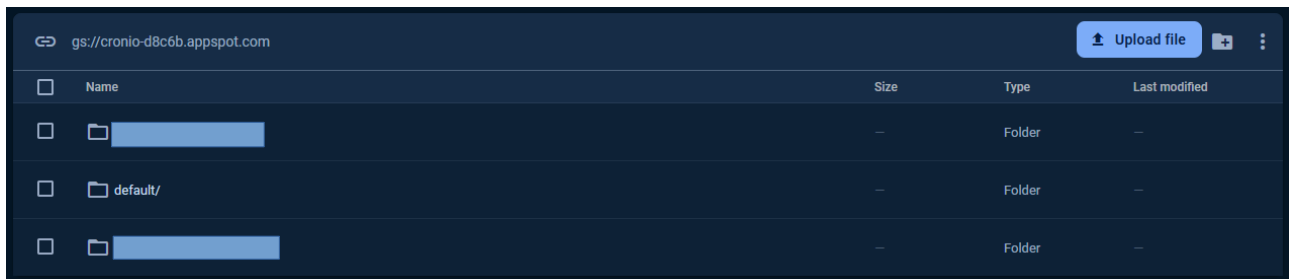
Brugerens personlige indstillinger og oplysninger opbevares som nedenfor. Det bemærkes at brugere ikke har adgang til andres information, kun til sin egen.



The screenshot shows the Firebase console's 'Users' page with a user profile selected. The breadcrumb navigation at the top reads 'users > kasperscott94@'. The left sidebar shows a tree view with 'users' selected. The main content area displays the profile details for the user 'kasperscott94@gmail.com'. The profile details are organized into three sections: 'Avatar', 'Banner', and 'Bio'. The 'Avatar' section shows a placeholder image. The 'Banner' section shows a placeholder image. The 'Bio' section contains the following fields: 'bio' (text), 'database' (text), 'department' (text), 'email' (text), 'empDate' (text), 'firstName' (text), 'lastName' (text), 'phone' (text), 'role' (text), and 'title' (text). The 'email' field is highlighted in blue.

| Avatar | Banner | Bio |
|------------|------------|------------|
| [Redacted] | [Redacted] | [Redacted] |

Tilsvarende har hver bruger sin egen separate storage bucket:



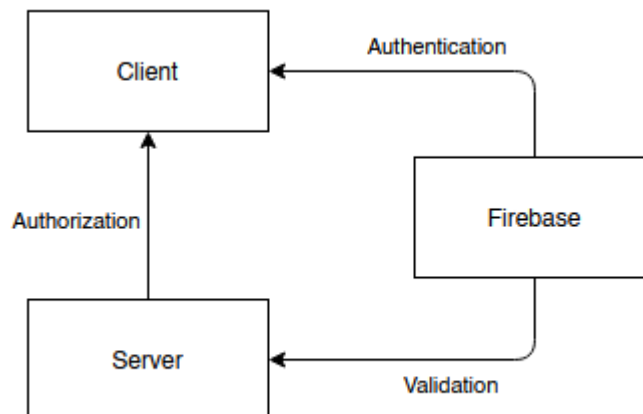
Personlig information er udskraveret. Når en bruger forsøger at logge ind benyttes altså Firebase Client SDK, og en request sendes til Firebase, der varetager autentificeringen.

```
export async function loginAttempt(data) {  
  try {  
    const userCredential = await signInWithEmailAndPassword(auth, data.email, data.password);  
    const user = userCredential.user;  
    if (data.remember) {  
      const sessionToken = await user.getIdToken();  
      Cookies.set('session_token', sessionToken, { secure: true, sameSite: 'strict' });  
    }  
    loginAndGetData(user.email)  
  } catch (error) {  
    setUserAuth(false, null)  
    notifier.danger('Invalid credentials', 3000)  
    console.log(error);  
  }  
}
```

Baseret på responsen fra denne request gemmes en ID token (JWT) som cookie. Denne cookie bliver verificeret gennem serverens Firebase Admin SDK, både ved login og løbende med et interval. Denne serverside-verifikation lægger til grund for brugerens autorisation til navigation.

```
<NotificationDisplay />
<Router>
  <Route path="/" component={Home} />
  <Route path="/login" component={Login} />
  <Route path="*" component={NotFound} />
  {#if isAuthenticated}
    {#each Object.entries(authRoutes) as [path, component]}
      <Route {path} component={component} />
    {/each}
  {:else}
    {#each Object.entries(authRoutes) as [path, component]}
      <Route {path} component={Login} />
    {/each}
  {/if}
</Router>
```

Efter endt validering hentes brugerens information, database og lignende. Disse informationer gemmes i stores og hentes dynamisk til visning når behovet opstår. Informationsflowet kan ses nedenfor.



6.2.2 Databaser

Ved oprettelse af en bruger, sker der en række af ting i serverens storage. Som nævnt benyttes SQLite3 til at opbevare en brugers data. Brugeren bliver altså først implementeret i databasen Status, der udelukkende har til formål at håndtere hvornår en bruger er checket ind eller ej, da denne information ellers ikke ville persistere på tværs af enheder eller sessions. Tabellen ser således ud:

```
CREATE TABLE "status"  
  ("id" INTEGER PRIMARY KEY AUTOINCREMENT,  
   "user" TEXT,  
   "is_checked_in" BOOLEAN)
```

Desuden bliver hver bruger tildelt en separat database, der skal indeholde al information om kalender, ønskeliste og tilstedeværelse. Databasen bliver navngivet et version 4 UUID, der bliver gemt under brugerens information i Firestore. Databasen for hver bruger har følgende tabeller:

```
CREATE TABLE "events"  
  ( "id" INTEGER PRIMARY KEY AUTOINCREMENT,  
    "user" TEXT, "start_date" TEXT,  
    "end_date" TEXT,  
    "start_time" TEXT,  
    "end_time" TEXT,  
    "title" TEXT,  
    "backgroundColor" TEXT)
```

```
CREATE TABLE "attendance"  
  ( "id" INTEGER PRIMARY KEY AUTOINCREMENT,  
    "user" TEXT, "start_date" TEXT,  
    end_date TEXT,  
    "start_time" TEXT,  
    "end_time" TEXT)
```

```
CREATE TABLE "wishlist"  
( "id" INTEGER PRIMARY KEY AUTOINCREMENT,  
  "user" TEXT,  
  "start_date" TEXT,  
  "end_date" TEXT,  
  "start_time" TEXT,  
  "end_time" TEXT,  
  "title" TEXT,  
  "backgroundColor" TEXT)
```

6.2.3 Kalender og ønskeliste

Til visningen af kalender og ønskeliste er benyttet et plugin, der hedder Event Calendar (se pluginliste for Github link). Pluginet supporterer en interagerbar kalender, der kan vise og redigere eventobjekter med minimal opsætning. Opsætning kan ses nedenfor:

```
$: eventData.subscribe(value => {  
  data = value.events  
  if (data) {  
    displayEvents(data)  
  }  
})  
  
let plugins = [TimeGrid, List, DayGrid];  
let options = {  
  view: "timeGridWeek",  
  dragScroll: true,  
  allDaySlot: false,  
  firstDay: 1,  
  headerToolbar: {  
    start: 'prev,next today',  
    center: 'title',  
    end: 'dayGridMonth,timeGridWeek,timeGridDay,listWeek'  
  },  
  events: []  
};  
  
function displayEvents(data) {  
  options.events = data.map(event => ({  
    id: event.id.toString(),  
    start: `${event.start_date} ${event.start_time}`,  
    end: `${event.end_date} ${event.end_time}`,  
    title: event.title,  
    backgroundColor: event.backgroundColor  
  }));  
}
```

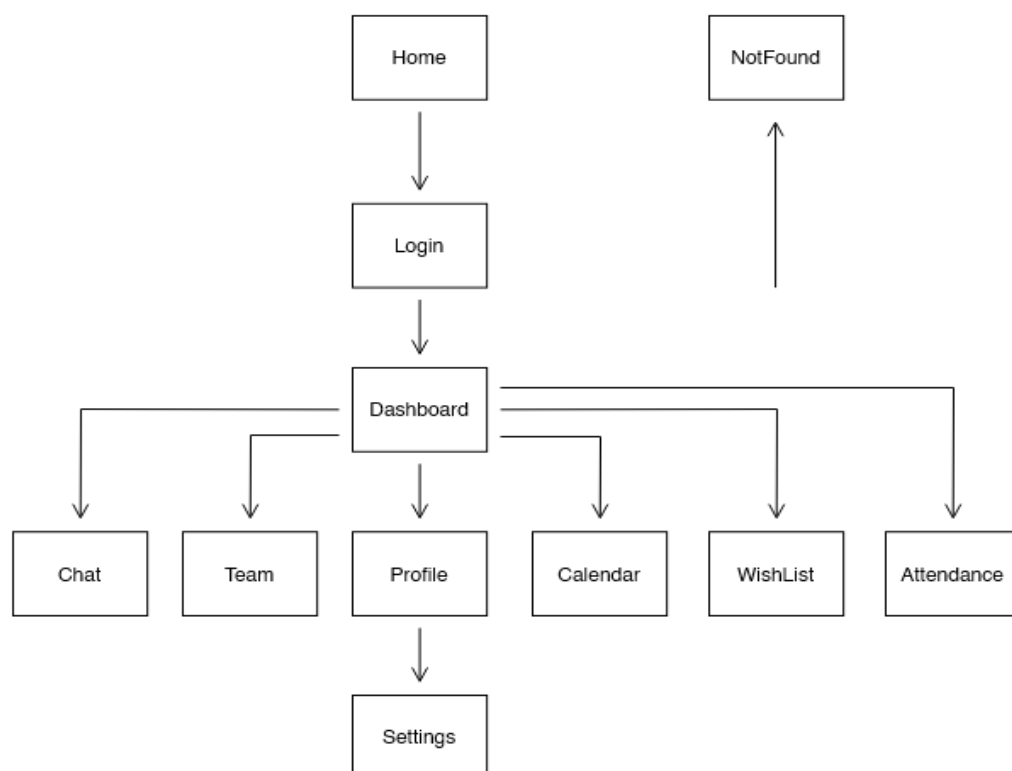
Eventdata er subscribed til en store, der indeholder resultatsættet fra en select-query fra databasen på serveren. Da denne subscription er trigger til displayfunktionen, vil kalenderen blive opdateret dynamisk når værdien af storen ændres. Pluginet indeholder ligeledes funktionalitet til at tilføje, ændre eller esktrahere data on the fly. Dette er naturligvis blevet brugt til ønskelisten, hvor brugeren selv kan tilføje hvilke vagter denne måtte ønske sig.

Wish List
Use the calendar to update your schedule wishes.

< > today 1-7 Jan 2024 month week day list

| | Mon, 01/01 | Tue, 02/01 | Wed, 03/01 | Thu, 04/01 | Fri, 05/01 | Sat, 06/01 | Sun, 07/01 |
|-------|------------|------------|------------|------------|------------|------------|------------|
| 01:00 | | | | | | | |
| 02:00 | | | | | | | |
| 03:00 | | | | | | | |

6.2.4 Navigation (sitemap)



6.2.5 Personlig profil og indstillinger

Hver brugers personlige oplysninger er opbevaret som angivet i afsnittet om brugerhåndtering. Disse bliver ved validering af ID token hentet til en store, og vises under brugerens personlige side. Denne indeholder blandt andet kontaktinformation, avatar, bannerbillede osv. Alt dette kan selvfølgelig ændres efter brugerens eget ønske. Noget af denne information (læs: ikke-kritisk) er tilgængelig de resterende medlemmer under Team-overviewet. Dette er med til at personliggøre applikationen. Billedet er skraveret af hensyn til brugeren.

My Profile

SAVE CHANGES

USER INFORMATION

FIRST NAME

Kasper

LAST NAME

Jensen

TITLE

CEO

DEPARTMENT

Copenhagen

CONTACT INFORMATION

EMAIL

PHONE

ABOUT ME

BIO

Hey, I am Kasper! This is my bio!

PICTURES

AVATAR

Browse...

No file selected.

BANNER

Browse...

No file selected.

20-05-2023
Employed since

MESSAGE

Kasper Jensen

COPENHAGEN

CEO

Hey, I am Kasper! This is my bio!

6.2.6 Sikkerhed og personlige oplysninger

Da delingen af personfølsomme oplysninger er en nødvendig del af arbejdsprocess mellem arbejdstager og -giver, er det ligeledes vigtigt at inkorporere standarder for sikkerhed i applikationen. Derfor er den, selv under development underlagt et sæt regler for behandlingen af disse. For at gøre det lettere, er de indført via Firestore, således at ingen har adgang til følsomme oplysninger, de ikke selv er autentificeret til.

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /users/{userId} {
      allow read: if request.auth != null;
      allow create, update, delete: if request.auth != null && request.auth.uid == userId;
      allow update: if request.auth != null && request.auth.uid == userId;
      allow read: if areFieldsAllowed(request.resource.data, ['firstName', 'lastName', 'title', 'bio', 'department', 'empDate']);
    }
  }
}

function areFieldAllowed(data, allowedFields) {
  return allowedFields.filter(field => request.resource.data[field] != null && request.resource.data[field] == data[field]).size() == allowedFields.size();
}
```

7. Diskussion

7.1 Udfordringer

7.1.1 Tidshåndtering

Det blev meget hurtigt klart for mig at jeg var presset på tid. Allerede efter første sprint kunne jeg se at selve layoutet og de enkelte komponenter blev udviklet alt for langsomt. Grundstenene for resten af appen var vigtige, men de tog hurtigt så stor en mængde tid, at jeg vidste at jeg kom i tidspres. Jeg skulle dog have indset dette hurtigere, for i stedet for at omlægge strategien, valgte jeg at lægge flere timer i projektet. Om det lykkes efter hensigten er svært at sige, men en ting er helt sikkert: den ekstra tid der blev lagt i det fundamentale er ikke spildt. Til al videre udvikling er det skabt et godt fundament, således at man let kan genbruge alle de udviklede komponenter, og altså udvide applikationen smertefrit. Så kortsigtet var det ikke det rigtige at gøre, men på lang sigt var det klart den bedre løsning.

7.1.2 Fokus

En af glæderne ved at have arbejdet alene er, at jeg har kunnet arbejde dynamisk. Jeg har på et hvilket givent tidspunkt kunne lægge arbejdet fra mig, komme tilbage til det senere, eller hurtigt at prøve en anden retning end planlagt fordi jeg fik en god idé. Jeg oplevede en mangel på strømlining og argumentation for sådanne valg – en sådan diskussion ville man naturligt have haft i gruppeøjemed, men manglede altså for mig. Det har resulteret i en mindre struktureret arbejdsform end jeg har oplevet i tidligere projekter. Det har ligeledes været mindre personlige, men praktiske udfordringer. Mange metoder og frameworks er bygget med tanke på to eller flere mennesker. Det betyder at det som enkeltperson kan være sværere at efterleve et frameworks faste struktur, og således sætte sig selv i en situation for fejl kan opstå. Her tænker jeg på peer reviews, dag til dag kommunikation med et team, og lignende. Hvor stor en effekt det har haft kan jeg ikke sige, men det er uden tvivl en faktor at tage hensyn til fremover.

7.2 Mangler

7.2.1 Test

Da jeg måtte aflyse sidste sprint er der ikke udviklet meget på tests i applikationen. Optimalt ville jeg gerne have lavet startup-tests på serverens endpoints og på databehandling. En ting jeg dog nåede at få implementeret er request-body tests på endpoints. Med Express Validator bliver alle requests typetestet, og hvis disse ikke er som forventet bliver der sendt en HTTP status 200 fejlrespons. Dog ville det være optimalt at teste alle endpoints både med og uden data ved server startup.

7.2.2 Udvid med flere teams

Som nævnt i starten af rapporten, er applikation i denne tilstand kun udviklet til et enkelt team. Som beskrevet, var dette grundet at det ville være dumt at allokere ressourcer til en funktion, der ikke vil blive benyttet i projektets forløb. Dette er helt sikkert et af de første ting, der skal implementeres, hvis projektet skal videreudvikles. Database, Firestore, Firebase Auth og Firebase Storage er alle klargjort, både i regelsæt og format, til at blive udvidet. Strukturerne som de er nu, kan alle indkapsles som et "team", og er dermed klar til at skalere. Der mangler dog den programmatisk implementering.

7.2.3 Administratorpanel

På nuværende tidspunkt har jeg udelukkende fodret applikation med mockup data. I en senere version ser jeg gerne at en "teamleader" har rollen som administrator, og kan tilgå et separat administratorpanel, hvor vedkommende har mulighed for at håndtere brugere, ønsker, statistikker og så videre på et administrativt niveau. Dette bringer mig videre til det næste punkt, som i mine øjne er den største mangel i dette projekt.

7.2.4 Algoritme

Optimalt ville jeg gerne have udviklet en algoritme til at fordele vagter til brugernes kalender baseret på en række af faktorer. Først og fremmest, deres ønskeliste, naturligvis. Men endvidere på baggrund af lokation, færdigheder, stilling, er vedkommende blevet forfordelt tidligere og lignende. Jeg indså dog hurtigt at en sådan operation ikke kun er kompliceret, men også tidskrævende. Og som før nævnt var tid ikke en udtømmelig ressource i dette projekt.

7.2.5 Supportsystem

Endnu et punkt på sidste sprint var at få udarbejdet en form for dokumentation til applikationen. Enten en brugermanual eller implementering af en chatbot med simple supportoplysninger og muligheden for at oprette en ticket.

8. Konklusion

Samlet set har udviklingen af denne vagtplanlægningsapp skabt et skelet, der opfylder grundlæggende behov, men der er plads til forbedringer og udvidelser. Fremtidigt arbejde bør fokusere på at løse de nævnte udfordringer, forbedre funktionaliteter og implementere manglende features som algoritmebaseret vagtfordeling og et supportsystem. Applikationen udgør et solidt fundament, der kan videreudvikles og tilpasses behovene hos forskellige virksomheder og teams. En mere struktureret tilgang til udvikling, testdækning og tættere samarbejde med brugere eller kunder kan bidrage til at skabe en mere omfattende og brugervenlig løsning, der imødekommer specifikke behov. Alt i alt har det været en læringsrig proces, der har givet mig et bedre overblik over nøjagtig hvad det kræver af tid og fokus at udvikle og fremtidssikre en applikation.

Kildeliste

Dokumentation

| | |
|--------------------|---|
| Svelte | https://svelte.dev/ |
| Node.js | https://nodejs.org/en |
| Tailwind CSS | https://tailwindcss.com/ |
| Svelte Material UI | https://sveltematerialui.com/ |
| SQLite3 | https://www.sqlite.org/index.html |
| Firebase | https://firebase.google.com/ |
| Express | https://expressjs.com/ |
| Event Calendar | https://github.com/vkurko/calendar |

Bøger

Craig Larman, 2005, Applying UML And Patterns, 3. udg.

Joel Murach, 2015, MySQL, 2. udg.

Roger Pressman, 2015, Software Engineering, 8. udg.