Namn | Magnus Olsson
Kurs | Examensarbete
Utbildning | Javautvecklare
Handledare | Jakob Kallin
Datum | 15/01/2023

Kollicon

Konstruktion av en

mobile-first applikation

Abstract

Problem

Kits AB is a continuously growing business with over 30 employees specializing in various IT-related areas. The company hosts an annual conference where their employees participate in activities aimed at improving their teamwork skills and expanding their technical knowledge. The conference schedule and all its activities have historically been announced on the company's website. All information presented on this website has been read from a markdown file, and those in charge have not had the opportunity to edit the current information presented to the website's viewers.

Solution

The product that the company is developing has been named Kollicon. This software is intended to serve as a tool for those at Kits who are responsible for organizing future company conferences. Kollicon is primarily designed to be a mobile service, aiming to be accessible to employees during the conference itself so they can quickly stay updated on the day's events. The application will be a full-stack product and will therefore incorporate a variety of technologies. Additionally, the software is planned to be launched as a cloud service. A more detailed description of this work can be found in the following chapter.

Inledning

Problematik

Kits AB är en kontinuerligt växande verksamhet med över 30 anställda som specialiserat sig inom många olika IT inriktade områden. Företaget anordnar en årlig konferens där deras anställda får delta i aktiviteter ämnade till att förbättra deras gruppsammarbetsförmåga och vidga sina tekniska kunskaper. Konferensens schema och alla dess aktiviteter har historiskt sätt annonserats på företagets hemsida. All information som presenterats på denna hemsida har lästs in ifrån en markdown-fil och ansvariga har inte haft möjligheten att redigera den aktuella informationen som presenterats för hemsidans åskådare.

Lösningen

Produkten som företaget utvecklar han namngivits som Kollicon. Denna programvara ska agera som ett verktyg för de på Kits som är ansvariga för att anordna framtida företagskonferenser. Kollicon är huvudsakligen ämnad till att vara en mobiltjänst, detta är för att applikationen ska vara tillgänglig för de anställda under själva konferensen så att de snabbt kan hålla sig uppdaterade kring dagens händelser. Applikationen kommer att vara en fullstack produkt och kommer därmed inneha en rad olika teknologier. Programvaran ska dessutom lanseras i en molntjänst. En bredare beskrivning utav detta arbete finner du i nästkommande kapitel.

Innehållsförteckning

Forskningsöversikt: Beskriv alla teknologier

Som tidigare nämnt så kommer Kollicon bestå utav både en framsida och baksida innan den slutliga produkten lanseras till dess användare via en molntjänst. För att du som läsare ska kunna begripa hur denna programvara fungerar så krävs det att vi går igenom de olika teknologier som programmet grundar sig på. Vi börjar med en snabb sammanfattning innan vi ger en bredare och mer detaljrik beskrivning utav varje enskild del som möjliggjort konstruktionen utav Kollicon.

- **Figma** är en webbapplikation som används för att skapa ritningar över framtida applikationers design.

- **Amazon Web Services** (AWS) är en molnbaserad tjänst som drivs utav företaget Amazon. Tjänsten erbjuder utvecklare möjligheten att lansera sina webbapplikationer ute i ett moln för att användare runt om i hela världen ska kunna ta del utav den produkten. AWS tillfogar ett brett sortiment av verktyg vilket som gör tjänsten väldigt anpassbar för många olika sorters teknologier som förekommer inom programvaruutveckling.

- **PostgreSQL** är en relationsdatabas, en term som åsyftar databaser som lagrar information med hjälp utav kolumner och tabeller. Relationsdatabaser använder sig av **Standard Query Language** (SQL) för att hämta eller modifiera databasens innehåll.

- **Relational Database Service** (RDS) är en del utav Amazon webservices vars syfte är att skapa och underhålla relationella databaser i molnet. Kollicon kommer använda sig utav PostgreSQL, en utav flera SQL-baserade databaser som RDS är kompatibel med. Tjänsten förenklar underhåll av databasen med hjälp av ett användargränssnitt och en rad andra verktyg som förser användaren med information om databasens innehåll, struktur, status, kostnad, lagringskapacitet mm.

- **Spring boot** är ett Java baserat ramverk som används vid konstruktion utav API-tjänster, Microtjänster och olika sorters applikationer. Spring boot används i detta projekt för att skapa Kollicon`s baksida.

- **React** är ett JavaScript baserat bibliotek som används vid konstruktion av mobil- och skrivbordsapplikationer. Biblioteket kommer användas för att skapa Kollicon`s framsida under detta projektarbete.

- **AWS Cognito** är en autentiseringstjänst tillhörande Amazon webservices. Utvecklare kan använda denna tjänst för att identifiera användare och bevilja eller neka tillgång till applikationens resurser.

- **Material UI** är ett bibliotek bestående utav react komponenter som, i kombination med react, kan importeras och användas av utvecklare vid konstruktion av applikationer.

Genomförande

Utvecklingen av denna applikation har skett under en period av 12 veckor. För att ge läsaren en insyn hur projektet skapades från början till slut så kommer jag i denna del utav rapporten gå igenom de olika moment som arbetsgruppen tog sig igenom för att skapa den produkt som idag är Kollicon. Jag kommer att dela detta kapitel i två delar. Första delen ger läsaren en steg för steg beskrivning av applikationens utveckling. Det finns också en teknisk beskrivning av de teknologier som använts i projektet, men det är en väldigt allmän beskrivning som inte går in på någon djupare teknisk nivå. Den andra delen kretsar kring applikationens funktionalitet. Det finns här en lista över applikationens centrala funktionalitet och hur de fungerar på en mer teknisk nivå. Som läsare av denna rapport bör du ta hänsyn till att kvalitén av tekniska beskrivningen kan variera beroende på hur mycket **jag** har varit involverad. Detta är trots allt ett projekt som realiserats av flera personer.

Del 1 av genomförandet - Hur utvecklades Kollicon?

Vem var involverad i produktionen?

Detta arbete har involverat flera personer som besuttit olika roller. **Scrum-master** har haft som uppgift att vägleda utvecklarna i agil arbetsmetodik under arbetets gång. **Produktägaren** har tagit fram den idé som projektet kretsat kring. Denna person har inte direkt varit involverad i utvecklingen av projektet men har med jämna mellanrum övervakat utvecklingen och get nya insikter och förslag under arbetes gång. Slutligen så har vi haft fyra **utvecklare** som utfört själva grundarbetet i projektet, dvs, skapat kodbasen för applikationen. Det har även funnits andra anställda på företaget som givit oss ritningar för applikationens design och en del vägledning i kodbasens struktur, men dessa personer har ej haft en permanent roll i projektet.

Hur arbetade vi?

Vår SCRUM-master anordnade dagliga möten klockan 09:00 där projektets utveckling diskuterades. Vi använde oss utav github-kanban board för att få en översikt kring projektets tasks. Vi använde oss utav 2 veckor långa sprints för att sätta mål och redovisa dessa för projektets produktägare.

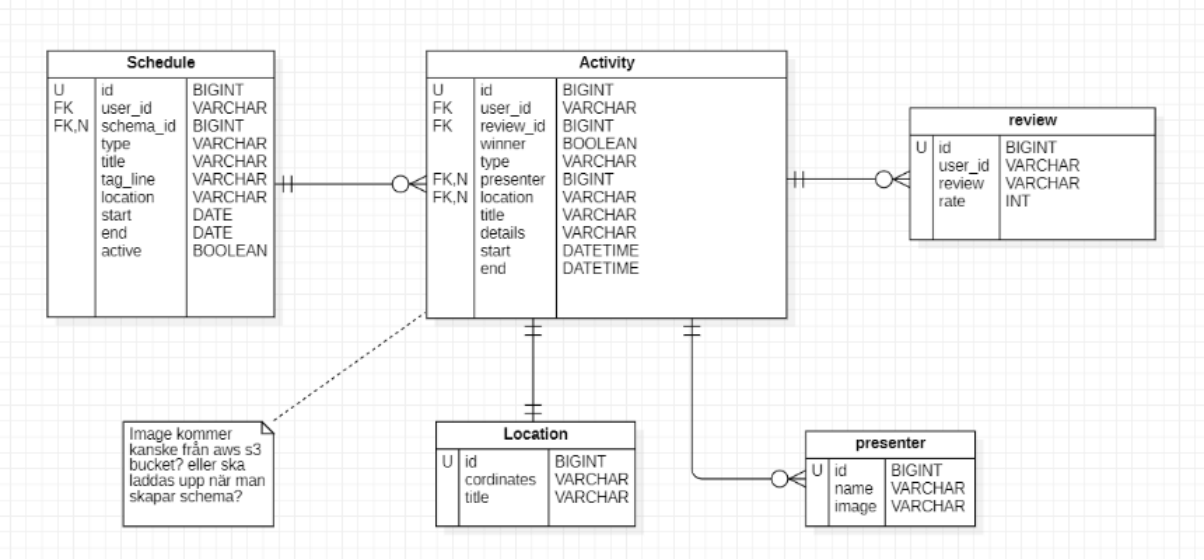
Figma sketch

Produktägaren som tagit fram idén om Kollicon hade gjort vissa förberedelser kring applikationens design. En anställd på företaget, som inte ingick i den arbetsgrupp som

ansvarade för applikationens utveckling, hade fått i uppgift att skapa en figma sketch. Företaget, Kits AB, som organiserat detta arbete har också en kollektion av färdigutvecklade komponenter som låg till grund för den design som skulle utgöra Kollicon`s framsida.

Databasen

Det första vi som arbetsgrupp gjorde var att ta fram ett klassdiagram i StarUML. Detta diagram skulle ge oss en överblick kring strukturen av applikationens baksida. Till en början så skapade vi ett diagram med fyra klasser. Detta diagram redovisades sedan och blev godkänt. Med tiden så begärde produktägaren att ytterligare funktioner skulle läggas till, arbetsgruppen fick därmed skapa flera klasser i applikationens baksida, och modifiera detta diagram som resultat.



Utvecklingen av Kollicon`s baksida

Efter att vi tagit fram ett klassdiagram över den design som databasen skulle struktureras efter så påbörjades utvecklingen av applikationens baksida. Som nämnt i forskningsöversikten så användes Spring boot för detta ändamål. Både jag och mina kurskamrater var väl bekanta med detta ramverk då tekniken förekom i den utbildning vi genomgått. Du kommer nu få läsa hur Spring boot har använts för att skapa applikationens baksida.

MVC

MVC är en utav många olika arkitekturmönster som utvecklare kan använda i spring boot. Efter att man färdigställt en applikation så övergår arbetsuppgifterna från att skapa till att underhålla. Även om det finns en rad olika arkitekturmönster att förlita sig på så är MVC en utav de absolut vanligaste. När man i framtiden tar in nya medarbetare

för att vidareutveckla eller underhålla Kollicon kan man förvänta sig utav dem att redan ha kännedom om det grundläggande designmönster som applikationen grundats på.

Externa bibliotek med Gradle

Totalt sett så har åtta beroenden (dependencies) angivits i programmets byggfil. Innehållet av dessa beroenden har sedan blivit hämtade och installerade med hjälp utav Gradle. Funktionaliteten som dessa beroenden möjliggjort handlar om dataåtkomst med JPA (Java Persistence API), databasuppkoppling, klass-annotationer, datavalidering, integrering utav e-post, och konstruktion av webbapplikationer.

Relational Database Service

Under konstruktionen av Kollicon`s baksida så har alla utvecklare använt en lokal PostgreSQL, detta har möjliggjort för utvecklarna att simulera hur applikationen kommer fungera innan den lanserats i molnmiljön. Kopplingen mellan en databas i lokal miljö och kopplingen till en databas uppe i AWS molntjänst är relativt lik, om inte identisk. Som utvecklare anger man anslutningsinformation i form av en URL sträng.

```
spring.datasource.url=jdbc:postgresql://localhost:5432/kollicon
```

Vid lokal anslutning.

```
spring.datasource.url=jdbc:postgresql://<RDS_ENDPOINT>:<PORT>/<DATABASE_NAME>
```

Vid koppling mot en databas i molnmiljö.

Utvecklingen av Kollicon`s framsida

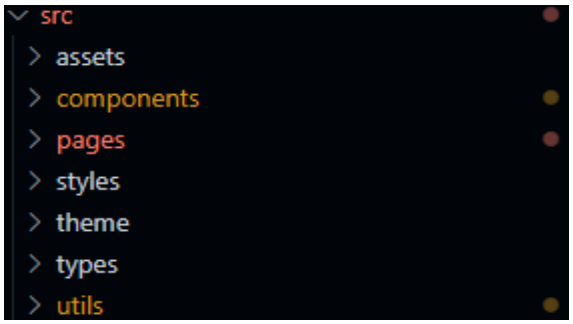
Det huvudsakliga arbetet har kretsar kring utvecklingen av Kollicon`s framsida. Precis som med applikationens baksida så är arbetsgruppens medlemmar väl bekanta med verktygen som ska användas. Du kommer nu få läsa närmare kring hur applikationens framsida skapats med hjälp av React.

React och Typescript

React är vanligtvis ett JavaScript baserat bibliotek som används för att utveckla framsidor till webbapplikationer. JavaScript som programmeringsspråk är väldigt dynamiskt i och med att variablar inte är bundna till någon specifik datatyp. Produktägaren för Kollicon har angett att applikationens framsida bör vara mer statiskt och har därmed ersatt JavaScript med Typescript. Den största skillnaden här är att variablar är bundna till en viss datatyp. Detta kan leda till potentiella störningar om en viss variabel tar emot ett värde som den inte är ämnad för. Utvecklare tvingas ta hänsyn till detta då de arbetar med dataöverföring.

Struktur

När man har ett projekt som består utav många olika komponenter och flera tusenrader kod så är det väldigt viktigt att tänka på struktur. Källkoden ska vara läsbar samtidigt som det ska bli enkelt för utvecklare att navigera igenom alla de komponenter som utgör applikationens



Som du kan se på bilden så utgörs Kollicon`s framsida utav sju mappar. Exakt vad innehållet i varje folder bidrar med kommer listas här nedanför:

- **Assets** består grafiskt material så som bilder eller GIF-filer.
- **Components** innehåller komponenter som utgör applikationen. Det kan handla om footer, header, user-dashboards etcetera.
- **Pages.** Trots att Kollicon är en single-page applikations så finns det fortfarande olika sidor. Det finns en sida som hanterar inloggning, en sida som användare navigeras till om vid systemfel och slutligen själva huvudsidan, där alla komponenter samverkar med varandra.
- **Styles** innehåller färdigdesignade knappar, menyer och annan kod som förekommer vid flera tillfällen i applikationens olika komponenter.
- **Theme** besitter olika teman. Det kan till exempel handla om en knapp som ändrar färg (tema) om användaren har klickat på den eller ej.
- **Utils** innehåller essentiell funktionalitet som applikationen behöver för att fungera. Det handlar om hooks och redux som deklareras i denna map och sedan tillkallas vid behov utav applikationens komponenter.

Material UI

För att underlätta utvecklingen av applikationens framsida så har vi använt oss utav Material UI. Biblioteket består utav färdigstylade komponenter som utvecklare enkelt kan importera och använda i sina egna projekt. Komponenterna som biblioteket erbjuder består utav HTML taggar, som i sin tur besitter en rad olika CSS regler för att uppnå utseende och beteende. Ett problem som uppstod var att några av dessa komponenter inte passade in i det applikationens tema. Vi som utvecklare fick därmed

”överstyra” de CSS regler som komponenterna ursprungligen designats med. HTML taggarna som utgjorde komponenterna bar på klassnamn, vi som utvecklare använde då dessa klassnamn för att överstyra HTML taggarnas CSS regler.

Exempel på detta kan du se här nedanför:

```
import Dialog from '@mui/material/Dialog';
import DialogTitle from '@mui/material/DialogTitle';
import DialogContent from '@mui/material/DialogContent';
import DialogActions from '@mui/material/DialogActions';
import Checkbox from '@mui/material/Checkbox';
```

Importerade komponenter ifrån Material UI

```
const BootstrapDialog = styled(Dialog)(() => ({
  '& .css-1t1j96h-MuiPaper-root-MuiDialog-paper': {
    backgroundColor: `${Colors.primaryBackground}`,
  },
  '& .css-m64m4g-MuiButtonBase-root-MuiCheckbox-root.Mui-checked': {
    color: `${Colors.primaryAddButton}`,
  },
  '& .css-1m9pwf3': {
    width: '50%',
    position: 'absolute',
    left: '25%',
  },
  '& .css-i4bv87-MuiSvgIcon-root': {
    width: '25px',
    height: '25px',
  },
  '& .css-m64m4g-MuiButtonBase-root-MuiCheckbox-root': {
    padding: '3px',
  },
}));
```

Exempel på hur klassnamnen på HTML taggarna användes för att överstyra CSS regler.

Del 2 – Central funktionalitet

Kollicon`s funktionalitet

Som med alla applikationer finns det vissa funktioner som är mer avgörande än andra. Denna del kommer att ge en teknisk beskrivning av de mest betydande funktioner som Kollicon besitter.

Autentisering med AWS Cognito

Kollicon är en produkt som ska användas internt i företaget. Det betyder att enbart anställda hos Kits AB ska få tillgång till applikationens resurser. Som tidigare nämnt i forskningsöversikten så kommer AWS Cognito vara den tjänst som applikationen förlitar sig på när det kommer till autentisering. Denna process sker i följande steg:

1. Först så deklarerar vi ett konfigurationsobjekt.

```
const userManagerConfig = {
  authority: 'https://cognito-idp.eu-west-1.amazonaws.com/eu-west-1_KNn5zQbLW',
  client_id: '2vbat83e7ac8cubv23cbgq6ufe',
  redirect_uri: 'http://localhost:5173/handlelogin',
  response_type: 'code',
  scope: 'openid email profile',
  loadUserInfo: true,
  automaticSilentRenew: true,
  extraQueryParams: { identity_provider: 'AzureServerlessTest' },
  revokeTokensOnSignout: true,
  revokeTokenTypes: ['refresh_token'] as ('refresh_token' | 'access_token')[],
};
```

Detta objekt besitter en del inställningar gentemot AWS Cognito.

2. Vi skapar en instans mot detta konfigurationsobjekt.

```
const userManager = new UserManager(userManagerConfig);
```

```
export const signinCallback = () => userManager.signinCallback();
```

3. Sedan loggar vi in med ett Microsoft 365 konto

```
onClick={() => signinRedirect()}
```

Som du kan se på första bilden så innehåller vårt konfigurationsobjekt en rad attribut som möjliggör för applikationen att använda sig utav AWS Cognito. En utav dessa attribut är autentiseringsprotokoll OpenID, som tillåter användare att logga in på applikationer en identifierare som inte är direkt kopplad till applikationen som dem försöker logga in på, i detta fall Kollicon. Användare kommer därför att logga in på Kollicon via Microsoft 365.

Kollicon kommer därefter att matcha inloggningsuppgifterna som den får via Microsoft 365 med de användare som finns i AWS Cognito identity pool för att godkänna eller neka tillgång till applikationen.

Skapa schema och aktiviteter

Syftet med Kollicon är att skapa ett schema över företagskonferenser som anordnas av Kits AB. Varje schema som skapas med hjälp av applikationen kommer bestå utav x antal aktiviteter. I applikationens framsida så har koden som ansvarar för att skapa ett schema delats upp i 4 komponenter som alla placerats i en folder benämnd som ”CreateSchedule”.

- **DatePickerComponent.tsx**. Som tillåter användaren att välja det datum som schemat ska inneha.

- **ImageSelectComponent.tsx**. Varje schema som lagras i databasen besitter en avatar. Användaren får i denna komponent välja avatarbild för schemat.
- **InputComponent.tsx**. Består utav inputfält som användaren kan addera relevant information för schemat. Detta kan vara saker som beskrivning, plats, titel med mera.
- **ScheduleComponent.tsx**. Registrerar den information som användaren har deklarerar i InputComponent.tsx.

De komponenter som ansvarar för att skapa aktiviteter tillhörande ett schema har placerats i en folder benämnt som RegisterActivity. Innehållet består utav 7 komponenter.

- **DatetimePickerComponent.tsx**. Där tid för aktiviteten deklareras.
- **PresenterComponent.tsx**. Skapar en aktivitet för presentatörer som är anställda på företaget.
- **ExtraPresenterComponent.tsx**. Skapat en aktivitet för presentatörer som inte är anställda på företaget.
- **InputComponent.tsx**. Tillåter användaren att fylla i beskrivning och titel för skapad aktivitet.
- **KitsConTypeComponent.tsx**. Är en komponent där användaren får deklarera vilken typ av aktivitet som skapas. Användaren får här en lista över engelskt benämnda aktiviteter att välja bland. Några exempel på benämningar som ingår i listan är: lunch, drink, presentation, airplane, boat.
- **LocationComponent.tsx**. tillåter användaren att deklarera vilken plats som aktiviteten ska vara.
- **RegisterActivityComponent.tsx**. Är den komponent som registrerar aktiviteten som ett objekt.

Alla dessa komponenter har använt sig utav post-förfrågningar gentemot applikationens baksida för att skapa schema och aktiviteter.

Exportera markdown-fil

Produktägaren för Kollicon hade vid tidigare skede annonserat konferensens schema på företagets hemsida. All information om schemat angavs i en markdown-fil och denna fil laddades senare upp på företagets hemsida för beskådning. Även efter att Kollicon skapats och möjlighet till annonsering kunde göras via applikationen, så ville produktägaren fortfarande annonsera schemat på företagets hemsida. Scheman som skapades i Kollicon behövde därmed kunna exporteras till en markdown-fil.

Användare i rollen som administratör av Kollicon har en möjlighet att markera scheman för export i en lista av scheman som finns i databasen. När ett eller flera scheman valts så omvandlas valt schema till ett YAML objekt ifrån applikationens baksida. Detta YAML objekt hämtas sedan till applikationen framsida. När objektet befinner sig i framsidan så

omvandlas det till ett BLOB-objekt. Detta objekt blir nerladdat till den lokala enheten i egenskap av en markdown-fil.

Hitta tidigare kitscon

Produktägaren ställde kravet på arbetsgruppen om att Kollicon skulle kunna hämta schemat för tidigare konferenser. Utav de centrala funktioner som finns med i Kollicon så var detta den enklaste. Scheman som finns lagrade i databasen hämtas med hjälp utav en get-förfrågan.

Resultat och analys

Kollicon har nu gått från idé till färdig produkt och det är nu vi som arbetsgrupp vill testa hur applikationen fungerar. Jag och mina skolkamrater som utgjort arbetsgruppen bjöd därmed in personal ifrån företaget för att få en inblick i hur Kollicon upplevs utav dem. Detta är särskilt positivt eftersom produkten kommer användas intern av företaget, och de som nu ska delta i våra användare tester är samma personer som kommer använda produkten när den väl lanserats.

Hur användartesterna utfördes

Användare av Kollicon kommer att delas in i två olika roller: Administratör eller användare. Det viktigt är att dessa användartester reflekterar båda rollerna. samtidigt som applikationens funktioner undersöks. Arbetsgruppen började med att skriva ner scenarion för administratör och användare. Produktägaren för Kollicon fick genomföra de scenarion som var utformade för administratör, därefter så erbjöds tre anställda på företaget att genomgå de scenarion som var utformad för användare. Efter att dessa scenarion utförts så fick samtliga testare svara på en enkät. Produktägaren som genomgick administratörscenarion svarade på en enkät specifikt framtagen för den rollen. Övriga anställda som genomförde användarscenarion fick svara på en enkät specifik framtagen för den rollen.

Scenarion

Scenarion framtagna för rollen som administratör

1. Logga in med ett giltigt Microsoft 365 konto
2. Skapa ett nytt schema med lämpliga attribut
3. Lätt till aktiviteter med olika typer

- Lägg till en aktivitet med typen flygplan klockan 07:00 – 09:30
- Skapa en aktivitet med en specifik plats
- Skapa en aktivitet med presentatörer från företaget

- Skapa en aktivitet med presentatörer utifrån företaget
- Skapa nya aktiviteter som ligger parallellt i tiden med övriga aktiviteter.
4. Hitta funktionen för att exportera schema till en markdown-fil
5. Redigera informationen för en redan existerande aktivitet
6. Skicka ut en notifikation i applikationen

Scenarion framtagna för rollen som användare

1. Logga in med ett giltigt Microsoft 365 konto.
2. Hitta följande aktiviteter.
- Hitta en aktivitet som har titulerats som ”Chefen presenterar” där Patrik är talare
- Hitta en aktivitet som har titulerats som ”Lunch på båt” och ta reda på platsen där denna skall hållas.
- Hitta lunch aktiviteten som hålls på fredag och ta reda på vilken tid det sket.
3. Skapa en egen aktivitet på schemat.
4. Redigera den aktivitet som du nyligen skapat.
5. Kontrollera om du har fått några notiser.
6. Hitta tidigare Kitscon scheman. Ta fram det schemat som skett i Amsterdam.
7. Gå till websidan BeerWithMe via en länk som finns på applikationen.
8. Gå till din profilsida på applikationen.

Enkätundersökning

Som tidigare nämnt så hade vi två olika enkäter: En som var framtagen för de som genomgick scenarion i som administratör och en som var framtagen för de som genomgått scenarion i rollen som användare.

Våra enkäter skapades i Google document där varje frågeställning var direkt inriktad gentemot hur svårt det hade varit att utföra de scenarion som deltagarna hade fått göra under våra scenariotester. Respondenterna fick besvara varje fråga genom att välja en siffra från 1 till 5, där 1 indikerade att ett scenario var **mycket svårt** och 5 indikerar att ett scenario var **mycket lätt**.

Den enkät som var utformat för rollen som administratör består av följande frågor:

1. Hur lätt var det att logga in med ditt Microsoft 365 konto i applikationen?
2. Hur lätt var det att skapa ett nytt schema?
3. Hur lätt var det att skapa aktiviteter?
4. Hur lätt var det att exportera ett schema till en markdown-fil?
5. Hur lätt var det att hitta en exporterad markdown-fil?
6. Hur lätt var det att förstå hur man kunde redigera aktiviteter?
7. Hur lätt var det att redigera aktiviteter?
8. Hur lätt var det att hitta notisfunktionen?
9. Hur lätt var det att skicka ut notiser?

Den enkät som var utformad för rollen som användare består av följande frågor:

1. Hur lätt var det att hitta aktiviteter som vi efterfrågade?
2. Hur lätt var det att skapa aktiviteter?
3. Hur lätt vad det att redigera egna aktiviteter?
4. Hur lätt var det att hitta notiser?
5. Hur lätt var det att hitta tidigare Kitscon event?
6. Hur lätt var det att hitta din profil?

Resultat för enkätundersökning

Administratör

Enkätresultaten för de som utfört scenarion i rollen som administratör antydde att våra testscenarion var mycket lätta. Fråga 2 på vår enkät, ”Hur lätt var det att skapa ett nytt schema” besvarades med en 4. De övriga frågorna på enkäten besvarades med 5. Detta resultat indikerar att användandet av Kollicon i rollen som administratör är mycket enkelt.

Användare

Enkätresultatet från de som utfört scenarion i rollen som användare antydde att våra testscenarion var mycket lätta att utföra. På fråga 3, ”Hur lätt var det att redigera egna aktiviteter”, och på fråga 5, ”Hur lätt var det att hitta tidigare Kitscon?” svarade 33% av deltagarna med en 4. Övriga frågor besvarades med 5 av alla de tillfrågade.

Utvärdering av testresultaten

Kollicon kommer vara en produkt som enbart används då företaget anordnar konferenser. Vi som utvecklat denna produkt under de 12 veckor som arbetet pågick kommer tyvärr att lämna företaget innan applikationen sätts i bruk och används för första gången. Vi kan däremot se på resultaten av våra scenariotester och de efterföljande enkäterna att applikationen anses vara mycket användarvänlig.

Utvärdering

Företaget som anordnade praktiken för oss och som begärde konstruktionen av Kollicon hade ett klart och tydligt mål: Att skapa programvara som underlättade för dem att schemalägga deras konferenser. Det 12 veckor långa arbete som jag och mina kamrater utfört hade slutat med ett lyckat resultat. Kollicon står idag som en färdig produkt och uppfyller de förväntningar som produktägaren ställt på arbetsgruppen.

Slutreflektion

Jag känner mig väldigt nöjd med min praktik och det arbetet som utfördes här. Ur ett eget perspektiv så lärde jag mig väldigt mycket under denna praktik. Jag exponerades för teknologier och tillvägagångsätt som för mig var helt nya. Den design som applikationens framsida skulle grundas på var angiven i en figma-sketch. En del detaljer i denna design var inte helt färdigritade. Detta tillät arbetsgruppen att vara kreativ och komma med egna förslag. Kits AB planerar att hålla en konferens i mitten av Maj 2024, där Kollicon gör sitt första framträdande. Jag och mina kurskamrater kan därmed lämna företaget med vetenskapen om att vårt arbete haft en positiv påverkan i näringslivet.

Referenser

Amazon Web Services

URL: https://aws.amazon.com/what-is-aws/

W3schools

URL: https://www.w3schools.com/whatis/whatis_aws_rds.asp

Techtarget

URL: https://www.techtarget.com/searchaws/definition/Amazon-Cognito

Microsoft

URL: https://www.microsoft.com/en-us/security/business/security-101/what-is-openid-connect-oidc

Spring boot dokumentation

URL: https://spring.io/projects/spring-boot

Jet brains

URL: https://www.jetbrains.com/help/idea/work-with-gradle-dependency-diagram.html

W3schools

URL: https://www.w3schools.com/typescript/typescript_intro.php

Material UI dokumentation

URL: https://mui.com/material-ui/getting-started/