

Fotografportalens Teamleader: En desktop-applikation/mjukvara för effektiv arbetsflödeshantering och digitalisering med React.js och Electron.js

Projektarbete

Fotografsportalen's Teamleader

En desktop-applikation/mjukvara för effektiv arbetsflödeshantering och digitalisering med React.js och Electron.js

Lucas Hammarstrand Schuber



Mittuniversitetet
MID SWEDEN UNIVERSITY

Fotografsportalen's Teamleader - En desktop-applikation/mjukvara för effektiv uppgiftshantering och digitalisering med React.js och Electron.js

Lucas Hammarstrand Schubert

2024-05-10

MITTUNIVERSITETET

Avdelningen för informationssystem och -teknologi

Författare: Lucas hammarstrand Schubert, luha2200@student.miun.se

Utbildningsprogram: Webbutveckling, 120 hp

Huvudområde: Datateknik

Termin, år: 02, 2024

Sammanfattning

Projektet syftar till att utveckla en desktop-applikation vid namn "Fotografportalens" för fotoföretaget Expresss-Bild, med målet att strukturera, digitalisera och effektivisera arbetsflödet för företagets fotografer. Fotografportalens vision strävar mot att integrera flera inbyggda program och mjukvaror, men det primära fokuset i denna rapport ligger på att designa, testa, utveckla och implementera programmet "Teamleader" - dt första programmet i Fotografportalens.

Genom användning av design thinking och agil metodik har Fotografportalens Teamleader designats i Figma och testats på två användare. Därefter har den utvecklats i Visual Studio Code med React.js och Electron.js för att möjliggöra användning utan internetanslutning och för att kunna ge en snabb och responsiv användarupplevelse.

För offline-funktionalitet integrerades en sqlite-databas med fem tabeller, medan applikationen anropar företagets databas för inloggning och projektstart. Genom detta arbete har Fotografportalens med Teamleader resulterat i en fungerande mjukvara som strukturerar, digitaliserar och effektiviserar fotografernas arbetsflöde på Expresss-Bild.

Nyckelord: Människa-dator-interaktion, Electron.js, React.js, Systemutveckling, Webbutveckling

Abstract

[Sammanfattningen på engelska här](#)

Nyckelord: Human computer interaction, Electron.js, React.js, System development, Web development

Innehållsförteckning

Sammanfattning	iii
Abstract	iv
Förord	vii
1. Introduktion	1
1.1. Bakgrund	1
1.2. Problemmotivering	2
1.3. Etiska aspekter	2
2. Teori	4
3. Metod	7
3.1. Design thinking-metoden	7
3.2. Tillämpning av design thinking-metoden	8
3.3. Agile-metoden	9
3.4. Tillämpning av Agile-metoden	9
4. Konstruktion / Lösningalternativ	11
4.1. Kravspecifikation	11
4.2. Utvecklingsmiljön	12
4.3. Mappstruktur	12
4.4. SQLite-databasen och tabeller	13
4.5. Filinnehåll	14
4.6. Dolda metoder som körs i backend	14
5. Resultat	16
5.1. Fotografportalens och Teamleaders funktionalitet	16
5.2. Teknisk implementation:	16
5.3. Användarfeedback	16
5.4. Användartester	16
5.5. Kvalitetsaspekter	16
5.6. Uppnådda mål	16
6. Slutsatser	17
6.1. Motgångar under projektet	17
6.2. Sammanfattning av huvudresultatet	17
6.3. Betydelse av användartester	17

6.4. Betydelsen av tillämpningen av metoderna	17
6.5. Framgångsfaktorer och lärdomar	17
6.6. Projektets utvecklingspotential	17
6.7. Avslutande reflektioner	18
Källförteckning	19
Bilagor	20
Bilaga A - Controll sheet	20
Bilaga B - Databas och tabeller	21
Bilaga C - X	22

Förord

Tack till Robban och Kajsa för att ni tog er tid och genomför användartesterna. Tack till Express-Bild för den här möjligheten.

1. Introduktion

Den digitala eran har omdefinierat och omformat hur vi arbetar och interagerar med teknik I (Scott MacKenzie, 2024, s.110). För Express-Bild, ett företag specialiserat på fotografi, har utmaningen varit att effektivisera och strukturera fotografernas arbetsflöden samt att övergå till en digital miljö. I detta syfte har Fotografportalens Teamleader kommit till - en desktop-applikation som strukturerar, digitaliserar och effektiviserar fotografernas arbete, med fokus på programmet vi kallar Teamleader.

Denna rapport presenterar processen och resultatet av skapandet av Fotografportalens Teamleader, från koncept till färdig mjukvara. Genom att använda teknologier som Electron.js och React.js har en applikation som erbjuder snabb och responsiv användarupplevelse i både offline- och onlinemiljö tagits fram och testats.

Rapporten inleds med en bakgrund till vad idén om Fotografportalens Teamleader kommer ifrån, följt av en beskrivning av de problem som projektet går ut på att lösa i Fotografportalens Teamleader, samt de etiska aspekter projektet tagit hänsyn till. Därefter finns en teoridel som beskriver viktiga begrepp och termer för att ge läsaren en bättre förståelse för rapportens innehåll. I metoddelen beskrivs de metoder som används och hur de har tillämpats i arbetet, från koncept till färdig produkt. Efter det hittar läsaren konstruktionskapitlet, där en teknisk analys och förklaring av projektets konstruktion presenteras. Här hittar läsaren också översikt över designprocessen och en beskrivning av hur prototyper har används för att iterativt förbättra användarupplevelsen, samt detaljer kring implementeringen av applikationen, inklusive integration av en SQLite-databas och anrop till företagets databas för autentisering och projektstart. I kapitlet efter det, resultatdelen, presenteras resultatet och huruvida målen har uppnåtts. Slutligen, i slutsatser, kommer utmaningar, lärdomar och framtida möjligheter för Fotografportalens Teamleader diskuteras.

Detta arbete har levererat en fungerande mjukvara vid namn Fotografportalens Teamleader, innehållande det första programmet Teamleader, som inte bara uppfyller de tekniska kraven för applikationen, men även en applikation med minimalistisk och stilig design med god användarvänlighet som förhoppningsvis i framtiden ska kunna underlätta företagets fotografers verksamhet och arbetsflöde.

1.1. Bakgrund

Express-Bild har under lång tid varit ett av de ledande företaget inom skol-, idrott- och stundetfotografering, men med den snabba utvecklingen av digitala verktyg och teknologier har företaget inte riktigt hängit med i utveck-

lingen. Fotografernas arbetsflöden har traditionellt varit beroende av manuella och analoga processer för att dokumentera och strukturera arbetet, vilket har resulterat i ineffektivitet och brist på struktur.

Express-bild har länge haft behovet av att skapa en digital lösning som inte bara strukturerar och effektiviserar fotografernas arbete, utan också möjliggör en smidig övergång till en digital miljö. Fotografportalen föddes ur denna vision - en desktop-applikation som ska innehålla flera program, varav ett är Teamleader, som har i syfte att centralisera och digitalisera fotografernas arbetsflöden under fotograferingarna.

1.2. Problemmotivering

Nedan presenteras problemmotiveringen baserat på bland annat kravspecifikationen. För att Fotografportalen och Teamleader ska fungera som önskat, behöver Teamleader innehålla/miljögöra följande:

- Möjliggöra offline-användning av applikationen
- Snabb och responsiv användarupplevelse, vilket innebär användarcentrerad design med fotograferna (användarna) i fokus.
- Gå att köra på windows-datorer då det primärt är vad företagets fotografer använder.
- Ha en integrerad Sqlite-databas för att kunna lagra data offline, med anslutning till Express-Bilds databas när internuppkoppling finns för att skicka in data/projekt samt kunna hämta projekt och användardata.

Genom att adressera dessa problem och utmaningar strävade projektet efter att skapa en lösning som förbättrade effektiviteten och strukturen för Express-Bilds fotografer.

1.3. Etiska aspekter

Eftersom att applikationen behandlar en kunds personuppgifter träder lagen om dataskyddsförordningen in. En metod i applikationen ser till att rensa alla känsliga uppgifter från den lokala integrerade databasen om kunden om den registrerades för mer än 6 månader sedan. Detta för att undvika att personuppgifter lagras efter ändamålen för behandlingen (integritetsskyddsmyndigheten, 2024).

Applikationen för också statistik över fotografens prestationen, vilket kan ge upphov till tävling mellan anställda eller mot sig själv vilket kan innebära ohälsosamma följder.

Fotografsportalen's Teamleader - En desktop-applikation/mjukvara för effektiv uppgiftshantering och digitalisering med React.js och Electron.js

Lucas Hammarstrand Schuber

2024-05-10

Statistiken kan g upphov till tävling och att man arbetar för mycket för att få bra statistik (ännu mer om gamification är implementerat)

Om gamification, uppmuntrar till att jobba mer?

2. Teori

Denna teori-del innehåller vissa termer och begrepp som är viktiga för läsaren att förstå för att vidare förstå rapportens innehåll.

Fotografportalen: Fotografportalen är den mjukvara som har skapats för detta projekt. Syftet med Fotografportalen är att alla framtida program ska finnas inom samma mjukvara, där varje program har ett specifikt syfte korrelerat till fotografernas/användarnas arbete. Genom att integrera olika program och mjukvaror inom applikationen strävar Fotografportalen efter att bli en centraliserad plattform för fotograferna innan, under, och efter fotograferingarna med program som olika möjliggör filuppladdning till företagets databas, tidsrapportering, schemaläggning och jobb-bokningar, kommunikation mellan fotografer samt strukturering, digitalisering och effektivisering av fotografernas arbetsflöden.

Teamleader: Teamleader är det första programmet som ska skapas inom Fotografportalen, och som detta projekt handlar om. Teamleader har i syfte att strukturera, digitalisera och effektivisera fotografernas arbetsflöden, vilket kommer att öka produktiviteten och förbättra användarupplevelsen. Detta inkluderar funktioner som digitalisering av "control sheet", strukturering av de olika jobben och en överblick över fotograferas prestation och tidigare jobb. Genom att använda Teamleader kan fotograferna optimera sin arbetsprocess och jobba på ett mer strukturerat och effektivt sätt.

Electron.js: Electron.js är ett ramverk för att utveckla cross-platform desktop-applikationer med webbt teknologier som HTML, CSS och JavaScript. Genom att använda Electron.js kan utvecklare skapa kraftfulla och responsiva desktop-applikationer som fungerar på olika operativsystem, inklusive Windows, macOS och Linux. Dess flexibilitet och stöd för moderna webbt teknologier gör det till ett idealiskt verktyg för att bygga applikationer som Fotografportalen och Teamleader (Brainhub, 2024).

I Electron är huvudvärlden (main world) huvudprocessen i applikationen och är den första processen som skapas när electron-applikation startas. Huvudprocessen har åtkomst till Node.js API:er och kan användas för att utföra systemnivååtgärder som att skapa och hantera fönster, kommunicera med filsystemet, skapa nya trådar och utföra nätverksanrop. När man arbetar med Electron är det viktigt att förstå gränssnittet mellan huvudprocessen och renderer-processerna, och hur man kommunicerar mellan dessa processer för att utföra olika uppgifter och uppdatera gränssnittet baserat på användarinteraktioner och systemhändelser. I en electron-miljö kör renderer-processerna i en webbläsareliknande miljö och har begränsad åtkomst till systemresurser och användare IPC (Inter-Process Communication) för att kommunicera med huvudprocessen (Electron, 2024).

Sammanfattningsvis är huvudvärlden (main world) den primära processen i din Electron-applikation och ansvarar för att hantera huvuddelen av

systemnivåfunktionaliteten. Det är viktigt att förstå gränssnittet mellan huvudprocessen och renderer-processerna för att bygga robusta och tillförlitliga desktopapplikationer med Electron.

React.js: React.js är ett populärt JavaScript-bibliotek för att bygga användargränssnitt och komponentbaserade webbapplikationer. Med React.js kan utvecklare skapa modulära och återanvändbara komponenter som gör det enkelt att hantera användargränssnitt och tillstånd i applikationer (React, s.f.). Genom att använda React.js i kombination med Electron.js kan Fotografportalens och Teamleader dra nytta av fördelarna med en modern och skalbar arkitektur för att leverera en användarvänlig och responsiv desktop-applikation.

React.js komponent: En React.js-komponent är en självständig och återanvändbar del av användargränssnittet som är byggd med hjälp av React.js-biblioteket och representerar en specifik del av det visuella gränssnittet och som kan interagera med användaren eller visa dynamiskt innehåll (ITHS, 2023). En React-komponent kan vara allt från en enkel knapp eller ett formulär till mer komplexa användargränssnittselement som en meny eller en inloggningssida. I Fotografportalens och Teamleader består gränssnittet av flertalet komponenter, både för att enkelt kunna återanvändas men också för att behålla en mer strukturerad och läsbar projektkod.

Axios: Axios är en Javascript-baserad HTTP-klient för webbläsaren och Node.js, som används för att göra HTTP-förfrågningar från en webbläsarapplikation eller från en Node.js-baserad server. Det används vanligtvis för att kommunicera med webb-API:er och hämta data från externa resurser (Axios, årtal saknas). Användningen av Express-bilds interna API:er med Axios möjliggjorde kommunikation med bakomliggande system och databaser för att hämta och hantera data relaterad till Fotografportalens funktioner. Genom att använda Axios kunde teamet smidigt integrera applikationen med befintliga system och säkerställa att den fungerade sömlöst för användarna.

SQLite.js: SQLite är ett inbäddat relationsdatabashanterare som möjliggör lagring och hantering av data lokalt på användarens enhet. SQLite är också serverlös, vilket innebär att applikationen kan läsa och skriva data direkt utan klient-serverarkitektur samt heller inte kräver någon installation eller konfiguration, vilket gör det fristående och mindre beroende av operativsystemet (Ravoof, 2023). Genom att integrera SQLite i Fotografportalens kan applikationen hantera användarinformation, fotoprojekt och annan relaterad data både online och offline. Detta möjliggör en god användarupplevelse även när användaren inte är ansluten till internet. I Teamleader finns det fem olika tabeller som har designats för att skapa en så snabb och smidig upplevelse som möjligt.

Vite.js: Vite.js är en byggverktyg som används för att utveckla moderna webbapplikationer med snabb utveckling och byggtider. Utvecklare väljer

Vite för dess enkelhet, hastighet och det moderna verktygsekosystemet som det stöder. Det förenklar byggprocessen, minskar konfigurationsbördan och optimerar utvecklings- och produktionsbyggen för prestanda (LoadFocus, s.f.). Genom att använda Vite.js i utvecklingsprocessen för Fotografportalen kan utvecklare dra nytta av dess snabbhet och effektivitet för att skapa och testa applikationen med React.js och Electron.js. Vite.js kan också användas för att logga och felsöka i en Electron-miljö på samma sätt som det används i en vanlig webbmiljö. Användningen av Vite.js i projektet har bidragit till att förbättra utvecklingsflödet för Fotografportalen.

Företagets databas: Utöver den integrerade databasen i SQLite kommunicerar applikationen även med företagets databas där användardata hämtas ifrån och projektdata (all fotojobb) skickas till, givet att datorn är uppkopplad mot internet. Denna databas är byggd i PHP och innehåller flera endpoints.

3. Metod

Metoden är uppdelad i två olika sektioner - *designprocessen* och *utvecklingsprocessen*. Designprocessen innefattar förarbetet - det arbete från planering till färdig prototyp. Utvecklingsprocessen innefattar arbetet från efter prototypen är klar, - programmeringsfasen.

Den metod som används för designprocessen är en *light* version av *design thinking*-metoden, och den metod som används för utvecklingsprocessen är *agile-metoden*.

3.1. Design thinking-metoden

I början av projektet användes design thinking för att förstå användarnas behov och definiera problemet eller utmaningen som projektet ska lösa. Genom att använda metoder som empatiövningar, användarintervjuer och problemdefinieringsövningar gav det en djupare förståelse för användarna och deras behov.

Design thinking-metoden består av följande fem steg (Wolniak 2017):

1. **Empatisera:** Empatistadiet handlar om att sätta sig in i användarnas situation och förstå deras behov, önskemål och utmaningar. Det innebär att samla in information genom observationer, intervjuer och användarundersökningar för att få en djupare förståelse för användarnas perspektiv. Målet är att identifiera och definiera problemet utifrån användarnas synvinkel, vilket kommer att ligga till grund för utformningen av lösningar.
2. **Definiera:** I detta steg används den insamlade informationen för att definiera och klargöra problemet eller utmaningen som ska lösas. Det handlar om att sammanställa och analysera data för att identifiera användarbehov och formulera en tydlig och konkret problembeskrivning. En viktig del av detta steg är att formulera en designbrief som tydligt beskriver målen och kraven för den framtida lösningen.
3. **Idégenerera:** I detta steg genereras en mängd olika idéer och koncept för att lösa det definierade problemet. Genom brainstorming, skissande och andra kreativa metoder utforskar teamet olika möjligheter och utforskar nya infallsvinklar. Inget idé är för absurd eller omöjlig att överväga under idégenereringsfasen, eftersom det är viktigt att främja kreativitet och innovation.
4. **Prototypa:** Prototypstadiet innebär att skapa en eller flera prototyper baserat på de bästa idéerna som genererats tidigare. Prototyperna är förenklade versioner av den slutliga produkten eller lösningen, men de är tillräckligt detaljerade för att testas och utvärderas av användarna.

Syftet med prototyperna är att ge en känsla av hur den slutliga produkten kommer att fungera och se ut, samtidigt som det ger möjlighet att identifiera och åtgärda eventuella brister eller förbättringar.

5. **Testa:** I teststadiet utvärderas prototyperna genom att de testas av användarna i en verklig eller simulerad miljö. Målet är att samla in feedback och utvärdera hur väl prototyperna uppfyller användarnas behov och mål. Testningen kan leda till ytterligare iterationer och förbättringar av designen baserat på användarnas respons, vilket bidrar till att skapa en mer effektiv och användarcentrerad lösning.

3.2. Tillämpning av design thinking-metoden

Genom att använda metoder som intervjuer och användartester skapades en djupare förståelse för användarnas perspektiv och behov.

Planering och tidsschema: Initialt i planeringsprocessen för utvecklingen av Fotografportalen och Teamleader skapades en tidsplan och en kravspecifikation. Denna del av arbetet inkluderade identifiering av projektets omfattning, tidsuppskattning för olika uppgifter och mjukvarans innehåll och funktioner.

Observationer och användarintervjuer Genom att prata med fotografer över telefon om deras behov och önskningar om vilka funktioner Teamleader skulle inkludera för att underlätta deras arbete, samt genom att observera deras sätt att arbeta på ute på fält, kunde en förståelse för deras arbetsprocess skapas. Dessa observationer och samtal kunde tas vidare in i nästa fas.

Sammanställda och analys av insamlad data: En problemformulering och kravspecifikation detaljerades och av den insamlade datan. En form av designbrief formulerades även.

Flödesschema och skiss: Flödesscheman och skisser för att visualisera och planera användargränssnittet och navigationsflödet i Fotografportalen skapades på penna och papper. Skisserna inkluderade olika designalternativ och flödesscheman olika användarinteraktioner.

Prototyp - lo-fidelity: En low-fidilitetsprototyp skapades i Figma för att testa och validera designkoncept och användarflöden.

Användartester: Med hjälp av två användare kunde low-fidelityprototypen testas över Zoom. Feedback och insikter från användarna samlades in och togs vidare in i nästa fas. Genomförandet av användartester gjordes för att utvärdera och validera användarupplevelsen i mjukvaran. Användartestet gick ut på att sätta testpersonen i olika testscenarier för att se hur använ-

daren interagerade med prototypen. Samtidigt antecknades testresultat för att identifiera användarbehov och förbättringsmöjligheter.

Prototyp - hi-fidelity: En hi-fidilitetsprototyp baserat på feedback från användartester och användarinteraktioner utvecklades i Figma. Denna mer interaktiva prototyp med en mer detaljerad design och funktionalitet för att demonstrera slutprodukten och förbereda för implementering, testades på samma två testpersoner.

3.3. Agile-metoden

Efter att ha definierat problemet och förstått användarnas behov, började planeringen av utvecklingsarbetet med agile-metoden. Agile-metoden används för att effektivt utveckla och leverera lösningar baserat på den insamlade informationen från designprocessen. Agile möjliggör snabba iterationer och anpassningar baserat på feedback från användare och teamet. (Kumar & Bhatia, 2012).

Agil metodik erbjuder flera fördelar för mjukvaruutvecklingsprocessen som gör att den bör antas vid utveckling av programvara. För det första möjliggör den en förbättring av planeringsfasen genom att involvera användare direkt i utvecklingsprocessen. Detta resulterar i att kraven verkligen återspeglar slutanvändarnas aktuella behov. Agil metodik möjliggör även tidig upptäckt av fel eftersom testning utförs under varje iteration. Detta gör det möjligt att upptäcka och åtgärda fel tidigare innan de ökar i allvarlighetsgrad. De kontinuerliga mötena ger en möjlighet till värdefull information och kontinuerliga förbättringar. Flexibilitet definierar förmågan att snabbt ändra riktning, vilket är en viktig egenskap hos agil metodik. Eftersom hantering av ändrade krav är huvudfunktionen hos agil metodik måste designen vara flexibel och kunna hantera ändringar enkelt (Kumar & Bhatia, 2012).

I slutändan betonar Agil metodik för mjukvaruutveckling utvecklingen av krav med direkt användarinvolvering i utvecklingsprocessen, snabba iterationer och små och frekventa frisläppanden. De förbättringar som görs i mjukvaruutvecklingsprocessen inkluderar mer stabila krav, tidigare upptäckt av fel, kortare ledtider för testning, ökad kommunikation och ökad anpassningsförmåga. Trots sina begränsningar har antagandet av agil metodik en positiv inverkan på både produktivitet och kvalitet, vilket gör både utvecklingsteamet och kunden nöjda med dess genomförande i mjukvaruutvecklingsprocessen (Kumar & Bhatia, 2012).

3.4. Tillämpning av Agile-metoden

Genom intern kommunikation och tvärfunktionella möten kombinerat med sprintar och tester i en iterativ process kunde en mjukvara tas fram som

mötte både användarnas och företags behov och önskemål.

Utveckling av mjukvara: Utvecklingen av Fotografportalen och Teamleader med användning av Electron.js och React.js innefattade kodning, testning och iteration för att skapa en användarvänlig desktop-applikation. Genom att tillämpa Agile-metoden kunde teamet effektivt hantera och prioritera uppgifter, vilket resulterade i en snabb utveckling av mjukvaran.

Sprintar: De viktigaste funktionerna och målen för projektet identifierades och bröts ner i mindre, hanterbara uppgifter, som kallas sprintar, med mål att implementeras i applikationen. Under utvecklingsfasen användes Agile-principer och tekniker som exempelvis just sprintar och kontinuerlig integration för att utveckla och implementera mjukvaran. Tillsammans med teamet arbetade vi i iterationer för att leverera fungerande delar av produkten för att sedan få kontinuerlig feedback från användare och teammedlemmar. Efter varje sprint genomfördes tester för att validera den utvecklade funktionaliteten och säkerställa att den uppfyllde användarnas behov. Feedbacken användes sedan för att iterera och förbättra produkten i nästa sprint.

Kundinriktad utveckling: Genom att kontinuerligt involvera och engagera användare och samla deras feedback i utvecklingsprocessen, kunde mjukvaran möta användarens behov och förväntningar.

Tvär-funktionell samarbete: Genom att samarbeta och kommunicera med olika team inom organisationen skapades en mer holistisk och väl avrundad produkt som främjar hela företaget. Möten med olika avdelningar kunde göra att mjukvaran möter flera avdelningars behov och förväntningar, och på så sätt även underlättar deras arbetsflöde.

Testing och felhantering: Slutligen användes olika strategier och metoder för testning och felhantering för att säkerställa Teamleaders prestanda och funktionalitet. Genom att implementera enhetstester, integrationstester och användartester kunde buggar och fel upptäckas och åtgärdas tidigt i utvecklingsprocessen och därför tillslut också säkerställa en hög kvalitet på den slutgiltiga produkten.

Versionshantering till GitHub: Versionshantering via GitHub gav teamet möjlighet att effektivt hantera kodändringar och samarbeta om projektet.

4. Konstruktion / Lösningalternativ

Konstruktionsavsnitt ingår ofta i tekniska rapporter, men inte alltid i vetenskapliga rapporter. Här genomför du din analys av problemställningen och formulerar en teknisk kravspecifikation. Här beskriver du de viktigaste principerna i de lösningalternativ som du föreslår, utformar och senare i rapporten kommer att utvärdera. Beskrivningen placeras ibland före, men oftast efter metod- /modellkapitlet.

Tänk på att läsaren sällan är intresserad av alltför detaljerad dokumentation av datorprogramkod, algoritmer, kretsscheman, användarhandledning, med mera. Sådana detaljer placeras med fördel i bilagor, om de över huvud taget inkluderas.

4.1. Kravspecifikation

För att läsaren ska ha lättare att förstå resten av rapporten presenteras nedan en enklare kravspecifikation över vad Teamleader ska innehålla.

- I Teamleader ska användaren kunna starta nya projekt hämtade från företagets databas, antingen sport eller skolfotografering.
- Kunna skapa nya lag/klasser, redigera lag/klasser, radera lag/klasser, samt kunna radera projekt.
- Offline-användning, då arbetet kan ske ute på fält utan internetuppkoppling. Programmet måste därför kunna köras felfritt utan internetuppkoppling, med undantag för när användaren ska skicka in data/jobbet till företagets stora databas då internetuppkoppling krävs för att kommunicera med backend/API't.
- Kalenderförsäljning - Teamleader ska möjliggöra kalenderförsäljning vid sportfotografering vilket innebär att under en sportfotografering ska lagets lagledare integrera med applikationen där den fyller i uppgifter och om den vill köpa sportkalendrar som laget kan tjäna pengar på genom försäljning av dessa eller ej.
- Digitalisera control sheet - en del av Teamleader är att blanketten "control sheet" (bilaga 1) som fylls i under arbetsdagen på en blankett ska digitaliseras och på ett smidigt sätt registreras/integreras i applikationen istället, vilket kommer innebära att den delen av det analoga arbetet kan slopas helt.
- Kunna registrera och logga in användare
- Presentera aktuella jobb och tidigare jobb på ett överskådligt sätt - Fotograferna ska kunna se statistik över deras tidigare jobb i form av vi-

suella diagram för att lätt kunna utvärdera deras arbete. Exempel på vad diagrammen skulle kunna visa är antal genomförda jobb, antal fotograferade subjekt, antal sålda kalendrar, med mera.

4.2. Utvecklingsmiljön

Eftersom att mina tidigare erfarenheter av electron.js är obefintliga, inleds projektet med en vecka av att förstå hur man sätter upp och arbetar med Electron.js. När jag kommit till underfund med hur Electron.js och React.js fungerar tillsammans, skapades utvecklingsmiljön med hjälp av ett befintligt GitHub *repo* som omstrukturerades för detta projekt.

Detta kapitel beskriver hur utvecklingsmiljön är utsatt med electron.js, react.js och vite.js.

4.3. Mappstruktur

Utvecklingsmiljön konstruerades med noggrann tanke på att skapa en välstrukturerad och organiserad mappstruktur för att underlätta utvecklingen och underhållet av projektet. För att uppnå detta skapades en hierarkisk mappstruktur som följde bästa praxis för att separera olika delar av applikationen och underlätta återanvändning och skalbarhet.

I rotmappen finns mapparna *src* och *resources*. I *resources* återfinns applikationens databas och app-ikon. Inuti *src*-mappen strukturerades filerna för att klargöra ansvarsområden och funktionalitet. Här finns de tre mappar som representerar olika delar av applikationen - *main*, *preload*, *renderer*: I *main* finns filen *Index.js* som ansvarar för huvudprocessen (main process) som hanterar systemnivååtgärder med Electron.js. Här placerades huvudskriptet för att starta Electron-applikationen samt alla filer som hanterar fönster, filsystemåtgärder och kommunikation med renderer-processen. I *preload* finns också en *index.js* fil, men som är de *preload*-skript som körs i renderer-processen och möjliggör säker kommunikation mellan huvudprocessen och webbsidorna i renderer-processen. I mappen *renderer* finns *src*-mappen, hjärtat i frontend-ramverket React, innehållande applikationens undersidor, komponenter, grafiska material, javascript-filer och css-styling. Applikationens undersidor, komponenter och css-filer delades upp i olika mappar beroende på om det tillhörde mjukvaran Fotografportalen eller programmet Teamleader. På så sätt kan ytterligare program läggas till i Fotografportalens i tillhörande mappar och fortfarande behålla god struktur.

För att underlätta hanteringen av externa bibliotek och beroenden användes en pakethanterare som npm tillsammans med Vite.js, vilket gjorde det möjligt att installera och uppdatera bibliotek och verktyg som behövdes för

projektet samt att bygga projektet för produktion med hög prestanda och snabb byggtid.

Sammanfattningsvis konstruerades utvecklingsmiljön med en tydlig och organiserad mappstruktur som underlättade utvecklingen och underhållet av projektet. Genom att följa bästa praxis för strukturering och organisering av filer och mappar, samt genom att använda Vite.js för snabb byggtid och utvecklingsserver, kunde vi skapa en stabil och skalbar grund för att bygga Fotografportalens och Teamleader.

4.4. SQLite-databasen och tabeller

I mappen *resources* finns databasfilen för SQLite. SQLite-databasen och dess (nuvarande) fem tabeller som är konfigurerad i huvudprocessen. I Fotografportalens databasstruktur används SQLite för att hantera och lagra användar- och projektrelaterad data. Databasen och dess tabeller går att hitta i bilaga B. Nedan följer en beskrivning av varje tabell och dess struktur:

Users Table: Denna tabell lagrar information om användare av Fotografportalens. Varje rad representerar en enskild användare och innehåller detaljer som e-postadress, för- och efternamn, lösenord, stad, språkval och autentiserings-token. Genom denna tabell hanteras användarautentisering och användarinformation för en förbättrad användarupplevelsen.

Projects Table: I denna tabell sparas data relaterad till de olika projekt som startas i Teamleader. Ett projekt har olika attribut såsom projektnamn, datum, fotograferingstyp (sport- eller skolfoto), fotografens namn och anmärkningar. Projektens status och metadata sparas här för att möjliggöra effektiv hantering och överblick över alla pågående och avslutade projekt.

Teams Table: Team-tabellen håller information om de olika team som är involverade i projekten i Teamleader. Det inkluderar detaljer om varje team, såsom lag/klass-namn, ledarens kontaktinformation och specifika detaljer om teamets aktiviteter och attribut. Varje lag/klass i teams-tabellen har en främmannyckel som refererar till ett projekt i projects-tabellen. På så vis kan alla lag/klasser kopplat till varje projekt hämtas på ett lätt och smidigt sätt.

Teams history Table: Denna tabell lagrar historisk information om team som har varit involverade i projektet. Den uppdateras varje gång ett lag/klass ändras av användaren för att kunna spåra uppdateringar som görs vid eventuella behov.

Projects Table: Denna tabell hämtas från företagets databas, och inkluderar de skol- och sportprojekt som användaren kan skapa. För att kunna skapa projekt utan internetuppkoppling behöver dessa projekt lagras lo-

kalt i databasen och hämtas in därifrån. När användaren har internetuppkoppling hämtas därför alla projekt från företagets databas och skriver över de befintliga raderna för att få de senast uppdaterade. När användaren senare ska skapa ett projekt utan internetuppkoppling, kan den välja mellan alla projekten i en lista som hämtas från tabellen `_projects`. De fält i tabellen är `projectname`, `project_uuid`, datumet projektet äger rum samt vilket språk projektet tillhör.

4.5. Filinnehåll

Beskriv alla filer och dess innehåll

4.6. Dolda metoder som körs i backend

I bakgrunden, bakom användargränssnittet, körs tre metodiska processer som underlättar hanteringen och underhållet av systemet samt bidrar till en god användarupplevelse.

Hämta och lagra projektdata: En av de bakgrundsprocesser som stöder funktionaliteten i systemet är den metod som hämtar projektdata med hjälp av `axios` från företagets centrala databas och lagrar den i tabellen `"_projects"` i den integrerade SQLite-databasen. Denna metod arbetar regelbundet och triggas så fort användaren befinner sig på index-sidan för att säkerställa att den lokala databasen alltid är uppdaterad med de senaste projekten från företagets databas.

GDPR datarensning: En annan viktig metod är den som ansvarar för att rensa GDPR-relaterade data från `projects`-tabellen. Efter 12 månader från det att personuppgifter har lagts till i tabellen, ersätts alla känsliga personuppgifter med ett `"x"`. Denna process säkerställer att systemet följer de lagstadgade kraven och riktlinjerna för dataskydd och integritet. Genom att regelbundet rensa och anonymisera personuppgifter upprätthåller systemet en hög standard för integritet och skyddar användarnas personliga information.

Internetuppkopplings-feedback: En annan metod som körs i bakgrunden är en som kontrollerar om användaren har en aktiv internetuppkoppling eller. Denna metod är särskild viktig i tre olika situationer där applikationen kräver internetuppkoppling för att åtgärden ska fungera - när användaren ska registrera en ny fotografprofil, när användaren ska skicka in ett färdigt fotograferingsprojekt till företagets centrala databas, och när an-

vändaren ska hämta in alla projekt från centrala databasen och lagra dessa i projects-tabellen.

De två första processerna finns i index-sidan, vilket innebär att de initieras när användaren startar applikationen, eller besöker index-sidan. Den tredje metoden ligger i sidomenyn vilket är en komponent på varje undersida och ligger i en *useeffect hake*, vilket innebär att den ständigt kontrollerar om en internetanslutning hittas eller försvinner. Om datorn är uppkopplad på internet, tillges logotyp-klassen en klass som startar en snurrande animation på logotypen som en visuell indikation på att internet är tillgängligt. Det signalerar för användaren att systemet är online och att interaktioner och funktioner som kräver interuppkoppling är möjligt. Å andra sidan, om det inte finns någon aktiv internetanslutning, avbryts snurrandet av logotypen. Denna åtgärd informerar användaren om att det inte finns någon internetuppkoppling tillgänglig för tillfället. Trots att mjukvaran körs i en offline-miljö, kan användare fortfarande använda lokal lagrad data och använda den precis som om det vore med en internetanslutning.

Dessa dolda metoder utgör ryggraden i systemet och arbetar tyst i bakgrunden för att säkerställa att dataflödet hanteras effektivt och att systemet förblir i linje med gällande lagar och bestämmelser om dataskydd. Det ger även användaren en smidig och användarvänlig upplevelse genom att anpassa gränssnittet efter användarnas aktuella anslutningsstatus. Genom att automatisera dessa processer minskar systemet bördan på användarna och ger en smidigare och mer säker användarupplevelse

5. Resultat

5.1. *Fotografportalens och Teamleaders funktionalitet*

Beskriv vilka funktioner och egenskaper som har implementerats, inklusive användargränssnitt, interaktionsmöjligheter och funktionalitet för att hantera fotografprojekt.

5.2. *Teknisk implementation:*

En översikt av den tekniska arkitekturen för Fotografportalen och Teamleader, inklusive vilka teknologier och ramverk som har använts, som Electron.js, React.js, och andra relevanta verktyg och bibliotek. Ex, att det blev Electron.js och React.js för att möta kravet om att applikationen ska gå att köras på windowsdatorer i offline-miljö.

5.3. *Användarfeedback*

Diskutera den feedback som fått från användare och teammedlemmar under utvecklingsprocessen. Identifiera eventuella positiva eller negativa reaktioner och hur dessa har påverkat design och utveckling av produkten.

5.4. *Användartester*

Frågor, antal testpersoner

5.5. *Kvalitetsaspekter*

Utvärdera den övergripande kvaliteten på Fotografportalen och Teamleader, inklusive prestanda, stabilitet och användarupplevelse. Diskutera hur olika teststrategier har bidragit till att uppnå hög kvalitet och tillförlitlighet.

5.6. *Uppnådda mål*

Bedöm i vilken utsträckning mål och krav har uppfyllts genom projektet samt eventuella avvikelser eller utmaningar som uppstod under projektets gång och hur de hanterades.

6. Slutsatser

6.1. *Motgångar under projektet*

Hela Electron-miljön och main world: Electron kräver navigation genom att kommunicera mellan processer och hantera gränssnittet mellan applikation och operativsystem. Som webbutvecklare var detta också nytt för mig som är van vid att utveckla för webben enbart och som inte var bekant med detta sedan tidigare. Det tog att tag att förstå gränssnittet mellan huvudprocessen och renderer-processerna för att bygga applikationen. Att kommunicera mellan dessa processer för att utföra olika uppgifter och uppdatera gränssnittet baserat på användarinteraktioner och systemhändelser var helt nytt för mig.

Integrera SQLite-databasen i Electron.js-miljön: Att integrera en SQLite-databas i Electron.js-miljön var därför en utmanade uppgift. Att både kunna koppla en databas till applikationen och sedan kommunicera med denna från frontend var nytt och krävande.

Desktop-applikation: Att utveckla en desktop-applikation för första gången innebär ett annat tankesätt jämfört med att utveckla en webbplats, både system-mässigt men också gränssnitts-mässigt. I en webbläsare kan användaren enkelt navigera fram och tillbaka och ha tillgång till en mängd olika webbaserade verktyg. I en desktop-applikation är användarupplevelsen annorlunda vilket gjorde att jag behövde tänka på saker som fönsterhantering, dialogrutor och tillgång till systemresurser på ett nytt sätt.

6.2. *Sammanfattning av huvudresultatet*

6.3. *Betydelse av användartester*

6.4. *Betydelsen av tillämpningen av metoderna*

6.5. *Framgångsfaktorer och lärdomar*

6.6. *Projektets utvecklingspotential*

Ge förslag på hur projektet kan vidareutvecklas och förbättras i framtiden. Nya funktioner eller förbättringar för Fotografportalens Teamleader, *Gamification för att göra teamleader roligare att använda och kanske fp

arbetare att vilja jobba mer/ta mer pass.

*Implementera fler program så att användaren inte behöver sitta i flera olika program utan i ett och samma program och kunna göra samma saker = en centraliserad mjukvara

*Ladda upp egen profilbild och implementera en chattfunktion mellan användarna

6.7. Avslutande reflektioner


Personliga reflektioner och lärdomar från arbetet samt hur erfarenheterna har bidragit till professionella utveckling och vilka insikter jag fått som kan vara värdefulla för framtida projekt.

Källförteckning

1. IMY. (2024). Det här gäller enligt GDPR: Grundläggande principer.- Hämtad från <https://www.imy.se/verksamhet/dataskydd/det-har-galler-enligt-gdpr/grundlaggande-principer/>. [Åtkomstdatum: 30 april 2024].
2. Wolniak, R. (2017). The Design Thinking method and its stages. *Systemy wspomagania w inżynierii produkcji*, 6(6), 247-255.
3. Kumar, G., & Bhatia, P. K. (2012). Impact of agile methodology on software development process. *International Journal of Computer Technology and Electronics Engineering (IJCTEE)*, 2(4), 46-50.
4. Axios. (s.f.). Introduction - Axios. Hämtad från <https://axios-http.com/docs/intro> [Åtkomstdatum: 30 april 2024].
5. I. Scott MacKenzie (2024). Human-Computer Interaction: An Empirical Research Perspective [Elektronisk resurs]. Elsevier Ltd.
6. Brainhub. (2024, January 11). What is Electron.js? Hämtad från <https://brainhub.eu/library/what-is-electron-js> [Åtkomstdatum: 1 maj 2024]
7. Electron. (2024). Process model. Hämtad 5 maj, 2024, från <https://www.electronjs.org/docs/latest/tutorial/process-model>
8. React. (s.f.). Hämtad 5 maj, från <https://react.dev>
9. ITHS. (2023, 16 oktober). Vad är React? Hämtad 5 maj, från <https://www.iths.se/vad-ar-react/>
10. Ravoof, S. (2023, 21 september). SQLite vs MySQL: A Comparison Guide. Hämtad 5 maj, från <https://kinsta.com/se/blog/sqlite-vs-mysql/>
11. LoadFocus. (s.f.). Vad är Vite? Hämtad från <https://loadfocus.com/sv-se/glossary/vite>

Bilagor

Bilaga A - Controll sheet



CONTROL SHEET

JOBBNAMN: _____ ORT: _____

FOTOGRAF: _____ DATUM: _____

ANTAL GRUPPBILDER: _____

	GRUPPER SOM NÄRVARAT GRUPP-/LAG-/KLASSNAMN	ANTAL SPELARE/ELEVER	PORTRÄTT (JA/NEJ)	GRUPP (JA/NEJ)
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

SIDA 1 AV 2

VÄND



LAG SOM EJ TOG PORTRÄTT

	GRUPP-/LAG-/KLASSNAMN
1	
2	
3	
4	
5	
6	

SAMMANSLAGNA GRUPPER

	GRUPP/LAG/KLASS	IHOPS LAGEN MED	GRUPP/LAG/KLASS
1			
2			
3			
4			
5			
6			

☐ JAG HAR SKRIVIT IN ALLA LAGLEDARE IN I TEAMLEADER

☐ JAG HAR LAGT IN OCH NAMNSATT ALLA GRUPPBILDER I PHOTOLINK

☐ JAG HAR SKANNAT ALLA FOTOLAPPAR OCH SÄNDER IN DEM INOM IHOP MED ALLA BILDER (PORTRÄTT/GRUPPBILDER) OCH DENNA CONTROL SHEET.

Fotografunderskrift: _____

Avvikelse rapport: _____

SIDA 2 AV 2

Bilaga B - Databas och tabeller

Users-tabellen

type	name	tbl_name	rootpage	sql
table	users	users	2	<pre>CREATE TABLE users (user_id INTEGER PRIMARY KEY AUTOINCREMENT, email TEXT NOT NULL, firstname TEXT NOT NULL, lastname TEXT NOT NULL, password TEXT NOT NULL, city TEXT, lang STRING NOT NULL, token STRING, created TEXT NOT NULL DEFAULT CURRENT_TIMESTAMP)</pre>

Projects-tabellen

table	projects	projects	7	<pre>CREATE TABLE projects (project_id INTEGER PRIMARY KEY AUTOINCREMENT, project_uuid STRING NOT NULL, projectname TEXT NOT NULL, photographername TEXT, project_date TEXT NOT NULL, type SRTING NOT NULL, anomaly TEXT, merged_teams TEXT, unit BOOLEAN, created TEXT NOT NULL DEFAULT CURRENT_TIMESTAMP, alert_sale BOOLEAN, is_deleted BOOLEAN DEFAULT 0, is_sent BOOLEAN DEFAULT 0, sent_date TEXT, user_id INTEGER NOT NULL, FOREIGN KEY (user_id) REFERENCES users(user_id))</pre>
-------	----------	----------	---	---

Teams-tabellen

table	teams	teams	5	<pre>CREATE TABLE teams (team_id INTEGER PRIMARY KEY AUTOINCREMENT, teamname TEXT NOT NULL, amount INT, leader_firstname STRING, leader_lastname STRING, leader_address STRING, leader_postalcode STRING, leader_county STRING, leader_mobile STRING, leader_email STRING, leader_ssn INTEGER, calendar_amount INTEGER, portrait BOOLEAN, crowd BOOLEAN, protected_id BOOLEAN, named_photolink BOOLEAN, sold_calendar BOOLEAN, is_deleted BOOLEAN DEFAULT 0, created TEXT NOT NULL DEFAULT CURRENT_TIMESTAMP, project_id INTEGER NOT NULL, FOREIGN KEY (project_id) REFERENCES projects(project_id))</pre>
-------	-------	-------	---	--

Teams history-tabellen

table	teams_history	teams_history	6	CREATE TABLE teams_history (team_history_id INTEGER PRIMARY KEY AUTOINCREMENT, teamname TEXT, amount INT, leader_firstname STRING, leader_lastname STRING, leader_address STRING, leader_postalcode STRING, leader_county STRING, leader_mobile STRING, leader_email STRING, leader_ssn INTEGER, calendar_amount INTEGER, portrait BOOLEAN, crowd BOOLEAN, protected_id BOOLEAN, named_photolink BOOLEAN, sold_calendar BOOLEAN, is_deleted BOOLEAN DEFAULT 0, created TEXT NOT NULL DEFAULT CURRENT_TIMESTAMP, team_id INTEGER NOT NULL, FOREIGN KEY (team_id) REFERENCES teams(team_id))
-------	---------------	---------------	---	--

Projects-tabellen

table	sqlite_sequence	sqlite_sequence	3	CREATE TABLE sqlite_sequence(name,seq)
table	_projects	_projects	4	CREATE TABLE _projects (project_id_ INTEGER PRIMARY KEY AUTOINCREMENT, project_uuid STRING NOT NULL, projectname STRING NOT NULL, start TEXT NOT NULL, lang STRING NOT NULL)

Bilaga C - X