

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

MITTUNIVERSITETET
Avdelningen för informationssystem och -teknologi

Författare: Lucas hammarstrand Schuber, 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 "Fotografportalen" 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" - det första programmet i Fotografportalen.

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 Fotografportalen 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. Disposition	1
1.3. Mål	2
1.4. Kravspecifikation	3
1.5. Problemmotivering	3
1.6. Etiska aspekter	4
2. Teori	5
3. Metod	8
3.1. Design thinking-metoden	8
3.2. Tillämpning av design thinking-metoden	9
3.2.1 Planering och tidsschema:	9
3.2.2 Observationer och användarintervjuer:	9
3.2.3 Sammanställda och analys av insamlad data:	9
3.2.4 Flödesschema och skiss:	9
3.2.5 Prototyp - lo-fidelity:	9
3.2.6 Användartester:	10
3.2.7 Prototyp - hi-fidelity:	10
3.3. Agile-metoden	10
3.4. Tillämpning av Agile-metoden	11
4. Konstruktion / Lösningalternativ	13
4.1. Utvecklingsmiljön	13
4.2. Filinnehåll och mappstruktur	13
4.5.1. Rotkatalogen	14
4.5.2. Renderer-mappen	15
4.5.3. Komponenter	16
4.5.4. Pages	19
4.3. Sport-och skolfotografering	20

4.4. Fyra olika språk	20
4.5. SQLite-databasen och tabeller	21
4.6. Dolda metoder som körs i 'bakgrunden'	22
5. Resultat	24
5.1. Fotografportalens och Teamleaders funktionalitet	24
5.2. Användarfeedback	24
5.3. Användartester	24
5.4. Kvalitetsaspekter	24
5.5. Mål	24
5.6. Huvudresultat av projektet	25
6. Slutsatser	26
6.1. Framgångsfaktorer och lärdomar	26
6.1.1. Framgångar	26
6.1.1. Lärdomar	26
6.2. Projektets utvecklingspotential	26
6.3. Betydelse av användartester	27
6.4. Betydelsen av tillämpningen av metoderna	27
6.5. Motgångar under projektet	27
6.6. Sammanfattning av huvudresultatet	28
6.7. Avslutande reflektioner	28
Källförteckning	29
Bilagor	31
Bilaga A - Controll sheet	31
Bilaga B - Databastabeller	32
Bilaga C - Mappstruktur	34
Bilaga D - Pages/undersidor	35
Bilaga E - Se.json-filen	42

Förord

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

1. Introduktion

Den teknologiska eran har förändrat hur människor vi arbetar och interagerar med teknik (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 Fotografportalen 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 Fotografportalen, 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.

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 hängtt med i utvecklingen. Fotografernas arbetsflöden har traditionellt varit beroende av manuella och analoga processer för att dokumentera och strukturerar 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 och total 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. Disposition

Rapporten inleds med en bakgrund till vad idén om Fotografportalen 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 itera-

tivt 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 Fotografportalen och Teamleader diskuteras.

1.3. Mål

Fotografportalen och Teamleader har som övergripande mål att erbjuda en användarcentrerad plattform för hantering av projekt och fotograferingsrelaterade uppgifter för Express-Bilds fotografer. Nedan beskrivs de specifika målen:

- **Hantering av projekt:** Användaren ska kunna starta, redigera och radera projekt för sport- eller skolfotografering som hämtas från företagets databas.
- **Hantering av lag/klasser:** Teamleader ska möjliggöra skapande, redigering och radering av lag eller klasser associerade med fotograferingsprojekten.
- **Offline-användning:** Programmet ska kunna köras felfritt utan internetuppkoppling för att möta fotografernas behov av att arbeta ute på fältet. Internetuppkoppling krävs endast för att skicka in data till företagets databas.
- **Kalenderförsäljning:** Teamleader ska integrera en funktion för försäljning av sportkalendrar under sportfotograferingar, där lagets ledare kan fylla i uppgifter och bestämma om laget vill köpa kalendrar för försäljning. Dessa sidor som berör kalenderförsäljning kommer att ha stöd för fem språk - svenska, norska, danska, finska och tyska.
- **Digitalisering av control sheet:** Den analoga blanketten "Control Sheet" (se bilaga A) ska digitaliseras och integreras i applikationen för att underlätta registreringen av arbetsuppgifter under fotograferingsdagen.
- **Användarhantering:** Teamleader ska möjliggöra registrering och inloggning av användare för att säkerställa att endast auktoriserade personer kan använda plattformen.
- **Statistik och utvärdering:** Fotograferna ska kunna få översiktlig statistik över sina tidigare och aktuella projekt genom visuella diagram för att underlätta utvärderingen av deras arbete.

1.4. Kravspecifikation

För att läsaren ska ha lättare att förstå resten av rapporten presenteras nedan en enklare kravspecifikation över vad Fotografportalen och Teamleader ska innehålla. Kravspecifikationen presenterar specifika funktionella och tekniska krav för att uppfylla målen. Nedan följer en sammanfattning av kraven:

- Möjlighet till offline-användning
- Snabb och responsiv användarupplevelse med fokus på användarcentrerad design
- Kompatibilitet med Windows-operativsystemet
- Stöd för fem olika språk på hemsidan för kalenderförsäljning: svenska, danska, norska, tyska och finska
- Implementering av en sqlite-databas för offline-datalagring med anslutning till Express-Bilds databas för dataöverföring och projektåtkomst vid internetanslutning

1.5. Problemmotivering

Problemmotiveringen identifierar de utmaningar och problemområden som projektet syftar till att lösa genom utvecklingen av Fotografportalen och Teamleader. Nedan följer en sammanfattning av de viktigaste problemområdena:

- Behovet av offline-användning för fotografer som arbetar ute på fältet utan internetuppkoppling
- Kravet på snabb och responsiv användarupplevelse för att effektivisera fotografernas arbetsflöde
- Kompatibilitet med Windows-operativsystemet för att passa fotografernas primära arbetsmiljö
- Kravet på flerspråkig support för att nå en bredare användarbas och möjliggöra internationell användning av plattformen
- Behovet av en robust databashanteringslösning för att säkerställa tillförlitlig datalagring och åtkomst både online och offline.

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

1.6. Etiska aspekter

Eftersom att applikation 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 i sin tur skulle kunna innebära att de anställda tar fler pass än vad hälsan tillåter.

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 arbete. Genom att integrera olika program inom mjukvaran Fotografportalen strävar Fotografportalen efter att bli en centraliserad plattform för fotograferna innan, under, och efter fotograferingarna med program som möjliggör exempelvis 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" (se bilaga A), 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 moderna och responsiva desktop-applikationer som fungerar på flera operativsystem såsom Windows, macOS och Linux (Brainhub, 2024). Electrons flexibilitet och stöd för moderna webbt teknologier gör det till ett utmärkt verktyg att bygga Fotografportalen och Teamleader i.

I Electron är huvudvärlden (main world) huvudprocessen i applikationen och är den första processen som skapas när en 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).

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 Fotografportalen 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 Fotografportalen 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 Fotografportalen 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 (se bilaga B) 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.

NPM-paket: NPM (Node Package Manager) är en pakethanterare för JavaScript som används för att distribuera och hantera moduler och bibliotek. Det tillåter utvecklare att enkelt installera och använda återanvändbara kodpaket i sina projekt. Genom att använda NPM kan utvecklare snabbt och effektivt integrera extern funktionalitet i sina applikationer, vilket sparar tid och arbete (ericmutta, 2023). Några exempel på NPM-paket som används i Fotografportalen är sqlite3 (sqlite-databasinbindning), electron-log (loggningsmodul för Electron-applikationer som möjliggör enkel loggning av händelser och meddelanden i en Electron-applikation och gör det möjligt att spåra och diagnostisera problem) och axios (erbjuder en enkel och kraftfull HTTP-klient för Node.js- och webbläsarbaserade applikationer och används för att kommunicera med API: er och hämta eller skicka data från eller till en server).

Företagets databas: Utöver den integrerade databasen i SQLite kommunicerar applikationen även med företagets databas dit användardata hämtas ifrån och projektdata (alla fotograferingsjobb) 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 produk-

ten 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.

3.2.1 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.

3.2.2 Observationer och användarintervjuer:

Genom att prata med fotografer över telefon om deras behov och önsningar 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.

3.2.3 Sammanställda och analys av insamlad data:

En problemformulering och kravspecifikation detaljerades av den insamlade datan. En form av designbrief formulerades också.

3.2.4 Flödesschema och skiss:

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

3.2.5 Prototyp - lo-fidelity:

En low-fidilitetsprototyp skapades i Figma för att testa och validera designkoncept och användarflöden.

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

3.2.7 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 demonstrerar slutprodukten och förbereder för nästa fas, 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 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-funktionellt 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 i projektet. Detta var också fördelaktigt då man vill se kodens historik vid eventuella buggar som inte kunde lösas.

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

2024-05-10

4. Konstruktion / Lösningalternativ

Denna del av rapport beskriver hur projektet är konstruerat. Den presenterar bland annat applikationens innehåll i form av mappar och filer, vissa viktiga metoder och den integrerade sqlite-databasen.

4.1. Utvecklingsmiljön

Eftersom att mina tidigare erfarenheter av Electron.js är obefintliga, inleddes 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 i Visual Studio Code med hjälp av ett befintligt GitHub *repo* som omstrukturerades för att passa detta projekt.

Electron- och Reactapplikationen konfigurerades tillsammans med Vite.js och NPM för att underlätta utveckling och felsökning. En sqlite-databas för datalagring implementerades även. De nödvändiga NPM-paketerna installerades

4.2. Filinnehåll och 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 (se bilaga C) som följde bästa praxis för att separera olika delar av applikationen och underlätta återanvändning och skalbarhet.

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 Teamleader.

I detta kapitel ges en mer ingående beskrivning av vad varje fil i applikationen innehåller och har för syfte.

4.5.1. Rotkatalogen

I rotmappen finns mapparna *src* och *resources*. Men det finns även flera filer som är viktiga för applikationens funktionalitet och byggprocess. Nedan beskrivs de viktigaste mappar och filer i projektets rotkatalog:

```
— Rot
  — — resources/
  — — src/
    — — — main/
    — — — preload/
    — — — renderer/
  — — package.json
  — — src
  — — electron.builder.yml
  — — readme.md
  — — electron.vite.config.js
```

Resources-mappen: Denna mappen innehåller applikationsikonen och databasfilen för SQLite.

Src-mappen: Detta är mappen där källkoden för projektet finns. Den innehåller alla React-komponenter, undersidor, stylesheets och bilder. Allt som behövs för att bygga applikationen.

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 Fotografportalen i tillhörande mappar och fortfarande behålla god struktur.

package.json: Denna fil används av npm (Node Package Manager) för att konfigurera och hantera projektet och innehåller metadata om projektet samt information om projektets beroenden, kommandon för att bygga och köra projektet, och annan konfiguration.

electron-builder.yml: Detta är en konfigurationsfil för Electron Builder, vilket används för att bygga och distribuera Fotografportalens. Den innehåller konfiguration för olika bygginställningar och målplattformar för din applikation. Den kommer också att användas för att hålla reda på applikationens uppdateringar efter att den har installerats på fotografernas datorer.

readme.md: Detta är en Markdown-fil som innehåller dokumentation för projektet. Den beskriver bland annat hur man konfigurerar och använder mjukvaran.

electron.vite.config.js: Detta är en konfigurationsfil för Vite specifikt anpassad för att fungera med Electron och React, och den definierar inställningar för huvudprocessen, preload-processen och renderer-processen för din Electron-applikation.

4.5.2. Renderer-mappen

Src-mappen består som tidigare nämnt av tre mappar som representerar olika delar av applikationen - *main*, *preload*, *renderer*. *Renderer* är rotmappen för renderer-processen i electron-applikationen och är här användargränssnittet lever. I mappen *renderer* finns i sin tur ytterligare en *src*-mapp, vilket är hjärtat i frontend-ramverket React, innehållande applikationens källkod som undersidor, komponenter, grafiska material, javascript-filer och css-styling. Applikationens undersidor, komponenter och css-filer. Nedan beskrivs varje fil i den *src*-mappen inuti *renderer*-mappen, samt dess syfte:

```
— main
— preload
— renderer
  — — index.html
  — — src
    — — — assets/
      — — — — css/
        — — — — — teamleader/
      — — — — images/
      — — — — js/
    — — — components/
      — — — — teamleader/
    — — — pages/
      — — — — teamleader/
    — — — App.jsx
    — — — main.jsx
```

index.html: En HTML-fil som fungerar som ingångspunkten för Electron-applikationen. Genom att inkludera main.jsx-filen i bodyn i denna fil, och i main.jsx importera komponenten "App" som pekar på App.jsx där alla

komponenter och undersidor ligger, kan hela applikationen med inkluderande javascript- och css-filer laddas in i applikationen genom denna index-html-fil.

src: Det här är mappen som innehåller all källkod för react-applikationen. Den inkluderar assets-mappen, components-mappen och pages-mappen.

assets: Denna mapp innehåller bilder, css-filer och javascript-filer som används i react-applikationen.

css: I denna mapp för CSS-stilmallar som används i din applikation. De css-filer som hör till programmet Teamleader i programvaran Fotografportalen, finns i mappen Teamleader inuti css-mappen.

images: Denna mapp innehåller alla bilder och ikoner som används genom hela applikationen.

js: Denna mapp innehåller alla externa javascript-filer som används genom hela applikationen. Det är primärt tre metoder som finns där i nuläget. *gdprProtectionMethod* som byter ut persondata i tabellen projects mot bokstav "x". *fetchProjectsByLang* som hämtar projekten från företagets centrala databas efter vilket språk användaren har ställt in i sin profil, samt metoden *fetchProjects* som hämtar alla projekt från företagets centrala databas. Fler metoder som används vid återkommande tillfällen kan placeras i denna mapp för att undvika återupprepning av kod och för god kodstruktur.

components: En mapp som innehåller återanvändbara React-komponenter som används för att bygga användargränssnittet. Komponenterna kan vara allt ifrån bootstraps pop-up-moduler till navigationsmenyer. De komponenter som hör till programmet Teamleader i programvaran Fotografportalen, finns i mappen Teamleader inuti components-mappen.

pages: En mapp som innehåller React-komponenter som representerar olika sidor i din applikation. De undersidor/pages som hör till programmet Teamleader i programvaran Fotografportalen, finns i mappen Teamleader inuti pages-mappen.

App.jsx: Huvudkomponenten för din applikation. Den renderas vanligtvis i index.html-filen och innehåller rutningslogik och övre nivåkomponenter.

main.jsx: detta kan ses som huvudinmatningspunkten för React-applikationen. Den renderar rotkomponenten, App.jsx, och monterar den i DOM.

4.5.3. Komponenter

Fotografportalen består av flertalet återanvändbara react-komponenter som används igenom hela applikationen. De komponenter som tillhör Teamleader placerades i mappen "teamleader" inuti components-mappen.

Nedan beskrivs samtliga komponenter som tillhör Fotografportalen och Teamleader.

AnomalyReport: Denna komponent ansvarar för att visa anomalirapporten i Teamleader. Den tillhandahåller ett användargränssnitt för att visa, uppdatera fälten *merged_teams* och *anomalies* i tabellen projects.

CalendarConfirmModal: Denna komponent representerar en modal som används för att bekräfta kalenderköp i Teamleader. Modalen visar lagledarens inskrivna uppgifter och två knappar för användaren att integrerar med.

ConfirmControlSheetModal: Denna komponent visar det digitaliserade control sheet innan den skickas in till företagets databas. Innan den skickas in godkänds den.

ControlSheetModal: Denna komponent visar det digitaliserade control sheet efter att den har skickats om användaren i efterhand vill få en överblick över jobbets projekt- och lag/klassdata.

DeleteProjectModal: Denna komponent representerar en modal som används för att bekräfta raderingen av projekt inom Teamleader. Den kräver att användaren skriver in fältet projectname (alltså projektets namn) i ett input fält innan raderingen kan fullföljas.

DeleteTeamModal: Denna komponent representerar en modal som används för att bekräfta raderingen av lag/klasser inom Teamleader. Den kräver att användaren skriver in fältet teamname (alltså lagets/klassens namn) i ett input fält innan raderingen kan fullföljas.

EditTeamModal: I denna modal kan användaren redigera lag/klass-data i Teamleader efter ett lag/klass har skapats. Den tillåter användare att ändra data relaterat till lag/klass samt lagledar-uppgifter. På sportfotograferingsprojekt där kalenderföräsaljning sker, kan användaren också ändra status på om laget vill köpa kalendrar eller ej även i efterhand.

MiniMenu_teamleader: MiniMenu_teamleader-komponenten representerar den högra menyn i Teamleader när användaren befinner sig inne på ett jobb (på portal_teamleader.jsx-sidan) Denna komponent har fyra stycken runda knappar som ger snabbåtkomst till radera projektet, lägg till nytt lag/klass, skicka in projektet och öppna/stäng avvikelse rapporten (*anomaly report*).

NewProjectsModal: Denna komponent öppnar en modal där användaren måste bekräfta ett nytt projekt i Teamleader.

SendProjectModal: Komponenten SendProjectModal representerar en modal och används för att skicka in ett färdigt fotograferingsprojekt till företagets centrala huvuddatabas i Teamleader. För att kunna skicka in projektet kräver den att användaren loggar in och anropar företagets login-

API, med internetanslutning, för att därefter kunna skicka in datan till företagets databas, också med internetanslutning.

Sidemenu_teamleader: Sidemenu_teamleader-komponenten är den vänstra huvudmenyn i Teamleader. Den innehåller länkar/knappar för att komma åt de olika undersidor i Teamleader, samt Fotografportalens ikon högst upp, vilket dirigerar användaren till Fotografportalens startsida.

SubjectChart: Denna komponent visar ett stapeldiagram över antal fotograferad subjekt per projekt, med antalet på y-axeln och projektets datum på x-axeln, och importeras i Teamleader. Den visualiserar data på ett tilltalande sätt och använder javascript-biblioteket 'recharts' som installerades via NPM.

TeamsChart: Denna komponent visar ett stapeldiagram över antal fotograferad lag/klasser per projekt, med antalet på y-axeln och projektets datum på x-axeln, och importeras i Teamleader. Den visualiserar data på ett tilltalande sätt och använder javascript-biblioteket 'recharts' som installerades via NPM.

ConnectUserModal: Den här komponenten representerar en modal för att ansluta användarkonton i Teamleader. Den användas för att hämta en användarprofil från centrala databasen och för att sedan lagra den lokalt i sqlite-databasen. Efter det kan användaren sedan logga in på profilen genom switchUserModal för utföra arbeten.

FeedbackMessage: FeedbackMessage-komponenten används för att visa feedback-meddelanden till användaren genom hela Fotografportalens/mjukvaran. Den visar främst bekräftelsemeddelanden vid användarinteraktioner och systemhändelser som när ett projekt har skickats in, ett nytt lag/klass har skapats, data har uppdaterats osv.

Sidemenu_small: Sidemenu_small-komponenten är en sidomenyn i Fotografportalens och kan användas för att navigera runt mellan Fotografportalens undersidor.

Sidemenu: Sidemenu-komponenten är den vänstra huvudmenyn i Fotografportalens och innehåller länkar/knappar för att komma åt de olika programmet i Fotografportalens, vilket dirigerar användaren in till respektive program.

SwitchUserModal: Den här komponenten är en modal och används för att växla användarprofil inom hela Fotografportalens/mjukvaran. Den kan användas för att byta användarkonto utan att logga ut från applikationen och kan tillåta flera användare att dela en enhet eller session. För att undvika att fotograferna/användarna använder varandras profiler krävs autentisering mot user-tabellen i sqlite-databasen.

4.5.4. Pages

Fotografportalen består av flertalet *pages* eller undersidor. De undersidor som tillhör Teamleader placerades i mappen "teamleader" inuti pages-mappen. Nedan beskrivs kort samtliga pages som tillhör Fotografportalen och Teamleader.

addleaderinfo_teamleader: Denna sida används för att lägga till information om ett lag i sportfotografering i Teamleader. Sidan består främst av ett formulär där lagledaren skriver in sina personuppgifter i. Denna sida är på det språk användaren har inställt på för att göra det lättare för lagledaren som ska fylla i formuläret (Bilaga D1).

calendarsale_teamleader: Detta är sidan där lagledaren kan läsa om kalendern och trycka i om de är intresserade av att sälja dessa eller ej. Denna sida är också på det språk användaren har inställt på (Bilaga D2).

currwork_teamleader: På denna sida kan användaren se alla startade och pågående projekt. Härifrån kan användaren navigera in på projekten för att fortsätta ett arbete (Bilaga D3).

home_teamleader: Detta är startsidan i Teamleader, som visar en översikt över information och statistiskt kopplat till användaren tidigare/inskickade projekt. (Bilaga D4).

newproject_teamleader: På denna sida kan användaren skapa ett nytt projekt i Teamelader. Användaren kan välja projekt i dropdown-lista, och sedan välja antingen om det är en sport-eller skolfotografering (Bilaga D5).

newteam_teamleader: Här kan användaren skapa ett nytt lag eller en ny klass, beroende på om det är skol-eller sportfotografering. Användaren skriver in antal fotograferade subject, och kryssar i om den tagit bild på grupp, porträttbilder och om det fanns en elev/lagmedlem med skyddat id (Bilaga D6).

portal_teamleader: Portalen är den sida fotografen befinner sig på när den har startat ett projekt och arbetar med ett jobb. Här har fotografen en översikt över de lag/klasser som den har fotograferat, samt tillgång till 'minimenu_teamleader'-komponenten (Bilaga D7).

prevwork_teamleader: Visar de tidigare och inskickade projekten. Här kan användaren trycka på en knapp för att se den digitaliserade versionen av control sheet (Bilaga D8).

yescalendarsale_teamleader: Om lagledaren vill köpa kalendrar, navigeras den hit. Denna sida består av ett formulär för kalenderförsäljning där lagledaren ska skriva in personuppgifter som personnummer och adress, samt måste godkänna försäljningsvillkoren. Denna sida är på det språk användaren har inställt på (Bilaga D9).

account: Används för att visa användarkontot, samt möjliggör anslutning och byte av fotograf. (Bilaga D10).

index: Huvudsidan för applikationen Fotografportalens (Bilaga D11).

login_window: Inloggningssidan för användare (Bilaga D12).

register_window: Registreringssidan för nya användare (Bilaga D12).

settings: Inställningssidan där användare kan ändra sin användardata som namn, stad och språk (Bilaga D13).

4.3. Sport-och skolfotografering

Igenom hela applikationen skiljer applikationen på om det är en skol-eller sportfotografering genom fältet 'type' i projects-tabellen. Denna lagras i sessionstorage variabeln 'project_type' när ett projekt skapas eller startas och används för att styra användargränssnittet och dess innehåll. Beroende på om projektet är en skol-eller sportfotografering, ser alltså gränssnittet olika ut och behandlar data på olika sätt. Förutom att många ord som 'student' och 'team member' eller 'team' och 'class' skiljer sig åt beroende på 'type', innefattar sportfotografering också kalenderförsäljning vilket innebär att undersidorna är fler och datan som behandlas i en sportfotografering är betydligt mer. Sidorna 'addleaderinfo_teamleader', 'clandarsale_teamleader' och 'yescalendarsale_teamleader' är endast synliga om det är en sportfotografering då dessa sidor samlar in data om lagledaren och kalenderförsäljning som är korrelerat med just sportfotografering.

Denna lösning ansågs vara det enklaste lösningen, istället för att skapa två olika versioner av samma undersida - en först sport- och en för skolfotografering.

4.4. Fyra olika språk

Applikationens undersidor som berör kalenderförsäljningen finns på fem olika språk, vilket var ett krav i applikationen. I inställningar, undersidan 'settings', kan användaren byta språk om den så önskar. En förkortning för språket (SV, NO, FI, DE, DK) lagras i sessionstorage is variabeln 'lang' och kontrolleras innan kalenderförsäljningssidorna addleaderinfo_teamleader', 'clandarsale_teamleader' och 'yescalendarsale_teamleader' laddas. Den förkortning som finns lagrad i 'lang' laddas då in på dessa sidor från en json-fil i 'language'-mappen i assets. Det finns en json-fil för varje språk - svenska, danska, norska, finska och tyska som laddas in med en switch-sats. Se bilaga E för en bild hur *se.json* (den svenska json-filen) ser ut.

4.5. *SQLite-databasen och tabeller*

I mappen *resources* finns databasfilen för SQLite. SQLite-databasen och dess idag 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 under bilaga B. Nedan följer en beskrivning av varje tabell och dess struktur:

Users-tabellen: 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 användaraутентisering och användarinformation för en förbättrad användarupplevelsen.

Projects-tabellen: 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 anomalier. Projektens status och metadata sparas här för att möjliggöra effektiv hantering och överblick över alla pågående och avslutade projekt.

Teams-tabellen: 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-tabellen: 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-tabellen: 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 lokalt 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.

4.6. Dolda metoder som körs i 'bakgrunden'

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 användaren 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 internetuppkoppling 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

get 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.

Detta arbete har levererat en fungerande mjukvara vid namn Fotografportalen, 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.

5.2. *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.

5.3. *Användartester*

Frågor, antal testpersoner

5.4. *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.

5.5. *Mål*

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.

Fotografportalen och Teamleader hade som övergripande mål att erbjuda en användarcentrerad plattform för hantering av projekt och fotograferingsrelaterade uppgifter för Express-Bilds fotografer. Nedan beskrivs de specifika målen och hur det har uppnåtts:

- **Hantering av projekt:** Användaren ska kunna starta, redigera och radera projekt för sport- eller skolfotografering som hämtas från företagets databas.
- **Hantering av lag/klasser:** Teamleader ska möjliggöra skapande, redigering och radering av lag eller klasser associerade med fotograferingsprojektet.
- **Offline-användning:** Programmet ska kunna köras felfritt utan internetuppkoppling för att möta fotografernas behov av att arbeta ute på fältet. Internetuppkoppling krävs endast för att skicka in data till företagets databas.
- **Kalenderförsäljning:** Teamleader ska integrera en funktion för försäljning av sportkalendrar under sportfotograferingar, där lagets ledare kan fylla i uppgifter och bestämma om laget vill köpa kalendrar för försäljning.
- **Digitalisering av control sheet:** Den analoga blanketten "Control Sheet" ska digitaliseras och integreras i applikationen för att underlätta registreringen av arbetsuppgifter under fotograferingsdagen.
- **Användarhantering:** Teamleader ska möjliggöra registrering och inloggning av användare för att säkerställa att endast auktoriserade personer kan använda plattformen.
- **Statistik och utvärdering:** Fotograferna ska kunna få översiktlig statistik över sina tidigare och aktuella projekt genom visuella diagram för att underlätta utvärderingen av deras arbete.

5.6. Huvudresultat av projektet

Samtliga mål har uppnåtts och mjukvaran kan erbjuda en användarcentrerad design med god kvalitet. För att installera mjukvaran eller för köra lokalt på datorn, se GitHub-länken nedan. Läs readme.md filen för vägledning.

GitHub: <https://github.com/LucasHSchubert/fotografportalen.git>

6. Slutsatser

6.1. Framgångsfaktorer och lärdomar

6.1.1. Framgångar

Färdigställande av projektet: En av de största framgångar har varit att lyckas färdigställa projektet och kunna uppvisa en fungerande applikation till Express-bild. Genom hårt arbete har samtliga mål och krav uppnåtts och samtliga planerade funktioner har implementerats som planerat. Detta har resulterat i en produkt av hög kvalitet vid namn Fotografportalen, med det första innehållande programmet Teamleader.

Användarfeedback: Vi fick positiv feedback från våra användare under utvecklingsprocessen. Deras input och kommentarer har varit ovärderliga för att förbättra användarupplevelsen och identifiera områden för vidareutveckling.

6.1.1. Lärdomar

Electron: Eftersom att mina tidigare erfarenheter av electron.js var obefintliga vid projektets start, tillägnades den första veckan med att sätta mig in i electron.js och hur det kan kombineras med ramverket react, och en av de största lärdomarna är nog just hur man med webbutveckling kan skapa en desktop-applikationen med electron och react för flera operativ-system som fungerar både med och utan internetanslutning. Detta är något som har varit väldigt roligt och jag hoppas på att fortsätta med det länge framöver.

Agil metodik: Att jobba iterativt i en agil metodik var också nytt. Med sprintar och iterativa mötet där varje sprint diskuteras och testas har varit nytt och lärorikt för mig. Att kunna vara flexibilitet och snabbt ändra riktning i och med lärdomar efter mötet och involveringen av användare, har under metodiken varit viktig då ändringar kunnat ske under processen.

6.2. Projektets utvecklingspotential

Både Fotografportalen och Teamleader har mycket utvecklingspotential och internt diskuteras det hur man kan vidareutveckla mjukvaran. Några av de punkter som de talar mycket om är följande:

- **Gamification** - Genom att implementera gamification och ge användaren antingen monetära eller digitala belöningar vid uppnådda mål, som vid ett visst antal sålda kalendrar eller vid ett viss antal avslutade projekt, kan förbättra användarupplevelsen och göra både jobbet och användningen av mjukvaran mer rolig och intressant. Det kan också göra att fotograferna vill ta fler projekt för att avancera och nå högre poäng

för att få belöningar. Vi kan se en stor utvecklingspotential inom gamification, vilket kan leda till många förbättringar.

- **Multiprogram-mjukvara:** En framtidsvision företaget har är att göra en centraliserad mjukvara som innehåller alla nödvändiga program för fotografens arbete, istället för flera olika program. Några program som vi hoppas på att kunna implementera i mjukvaran Fotografportalens utöver Teamleader på kort sikt är Filetransfer, ett program för fotograferna att skicka över bilder på till företagets databas. Tanken är att detta program redan ska påbörjas i mitten/slutet av juni och förhoppningsvis ska vara klart innan skolfotosäsongen startar i slutet av augusti. På längre sikt hoppas vi på att även skapa program för schemahantering, tidsrapportering och en plattform för interna diskussioner mellan fotograferna.
- **Automatsikaapplikations-uppdateringar med electron-updater:** Electron updater är ett npm-paket som tillhandahåller automatisk uppdateringsfunktionalitet för electron-applikationer vilket innebär att applikationen gör automatiska uppdateringar av applikationen när nya uppdateringar har gjorts. Detta går att konfigurera genom ett GitHub-repository och gör det enklare att leverera nya funktioner och rätta till buggar för användarna i efterhand - efter det har installerats på användarens dator. Detta är något som vi hoppas kunna implementeras i applikationen i början av juni månad.

6.3. Betydelse av användartester

6.4. Betydelsen av tillämpningen av metoderna

6.5. Motgångar under projektet

Under projektet stötte jag på flera motgångar som krävde att jag sökte ny kunskap och flera timmar av felsökning. Några av dessa motgångar beskrivs nedan:

- **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 utmanande 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.

6.6. *Sammanfattning av huvudresultatet*

Mjukvaran Fotografportalen innehållande programmet Teamleader färdigställdes och uppnådde samtliga mål som initialt sattes vid projektets start. Jag tillsammans med min avdelning på Express-bild är alla nöjda med resultatet och ser en stor förbättring i både fotografernas arbete, men också hur det kommer underlätta arbetet för de anställda och de avdelningar som är involverade i fotograferingarna på kontoret.

För att installera mjukvaran och köra lokalt på datorn, se GitHub-länken nedan. Läs readme.md filen för vägledning.

GitHub: <https://github.com/LucasHSchubert/fotografportalen.git>

6.7. *Avslutande reflektioner*

Projektet nådde de uppsatta målen och uppfyllde kravspecifikationen. Resultatet blev en fullt fungerande desktop-applikationen som går att köra i både en online- som offline-miljö. Applikationen har installerats lokalt på min MacBook-dator och fungerar utmärkt. Detta resultat är jag både väldigt nöjd och stolt över och resan hit har varit tuff men väldigt lärorik. Jag har lärt mig ett helt nytt område inom webbutveckling som har breddat min kompetens från att kunna utveckla till webben och Progressive-Webb-applications till att kunna skapa cross-platsform desktop-applikationer med elektron, något jag tror kommer vara meriterande när jag söker jobb i framtiden. Jag har också fått en plats på IT-avdelning på Express-bild och ska fortsätta att utveckla Fotografportalen till att bli en ännu mer omfattande och komplex mjukvara med flera innehållande program. Ett uppdrag som jag ser mycket fram emot att fortsätta med i framtiden.

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>
12. ericmutta. (2023, January 12). npm CLI commands. [Online]. npm Documentation. Hämtad 8 maj 2024, från <https://docs.npmjs.com/cli/v10/commands/npm>

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

2024-05-10

Bilagor

Bilaga A - Controll sheet

EXPRESS-BILD

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

EXPRESS-BILD

LAG SOM EJ TOG PORTRÄTT

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

SAMMANSLAGNA GRUPPER

	GRUPP/LAG/KLASS	IHOPLAGEN 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 - Databastabeller

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 STRING 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>
-------	-------	-------	---	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

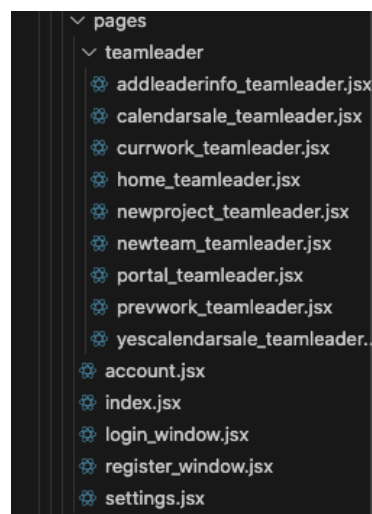
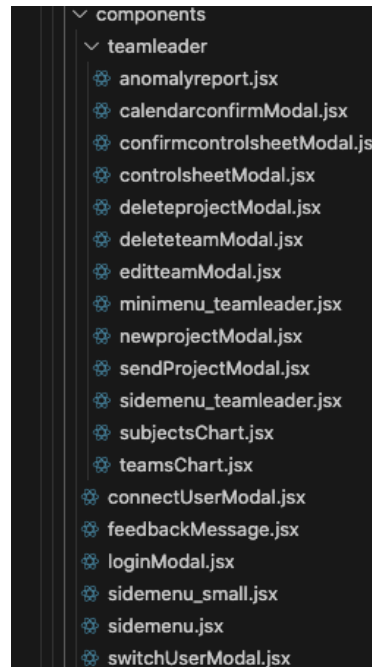
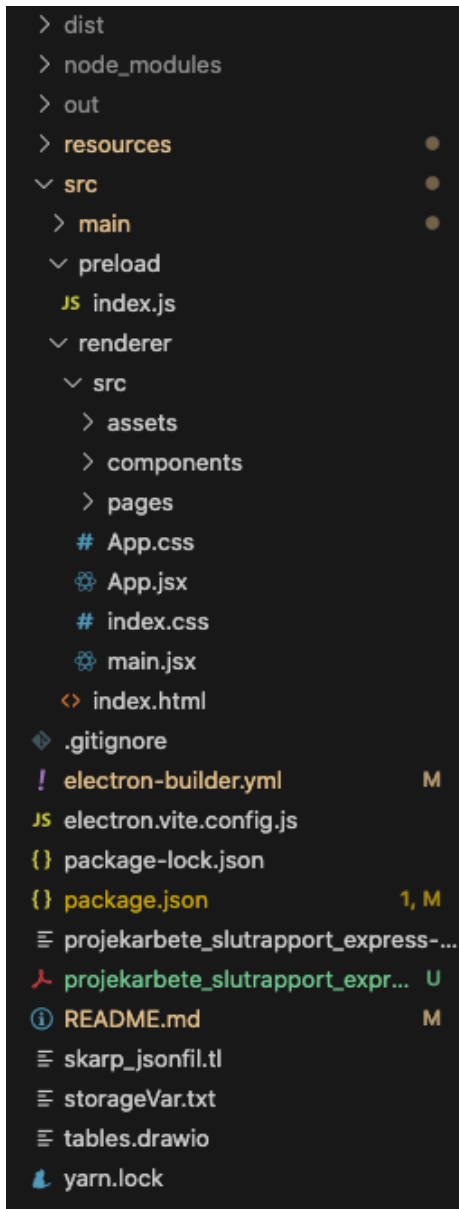
Teams history-tabellen

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 - Mappstruktur



Bilaga D - Pages/undersidor

1. Addleaderinfo_teamleader



Home

Previous work

Current work

New project

1. Formuläruppgifter2. Kalenderinformation3. Köp av kalender

+ Nytt lag






Lag information:

Team leader info:

Avbryt


Nästa

2. Calendarsale_teamleader



Home

Previous work

Current work

New project

1. Formuläruppgifter2. Kalenderinformation3. Köp av kalender

Kalenderinformation



Sportkalendern

Att sälja sportkalendrar är ett sätt för ert lag att tjäna pengar till lagkassan. Genom att sälja tre kalendrar per spelare tjänar ni 300 kr per spelare. Om ni inte lyckas sälja alla kalendrar, returneras dessa konstnadsfritt tillbaka till oss, och ni förlorar inga pengar.

Vinst: 300 kr/spelare till lagkassan!
Kostnad: 200 kr/st (varav 100 kr/per såld kalender tillfaller er lagkassa)

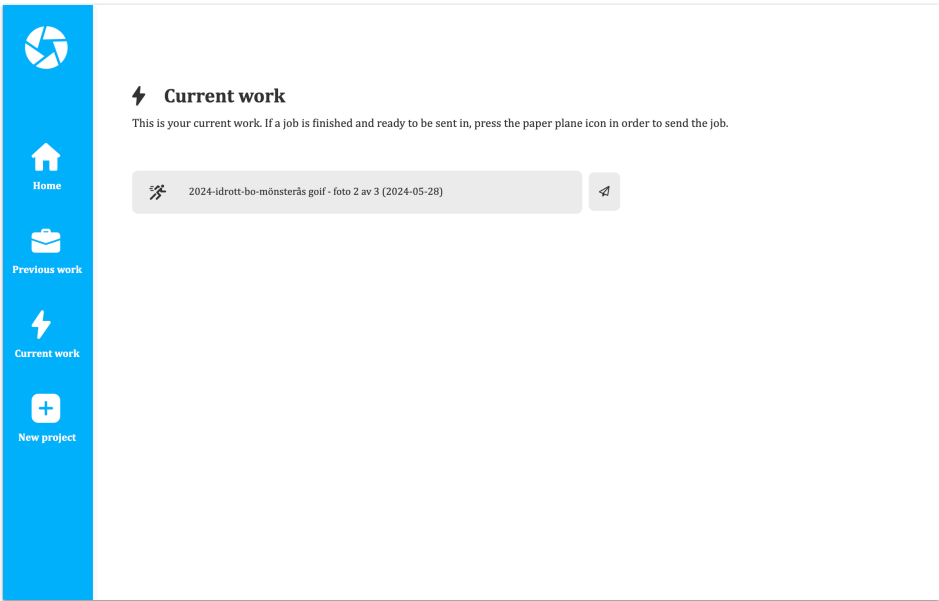
Är ni intresserade av att helt riskfritt sälja kalendrar som ett sätt för ert lag att tjäna in pengar till lagkassan?

Tillbaka

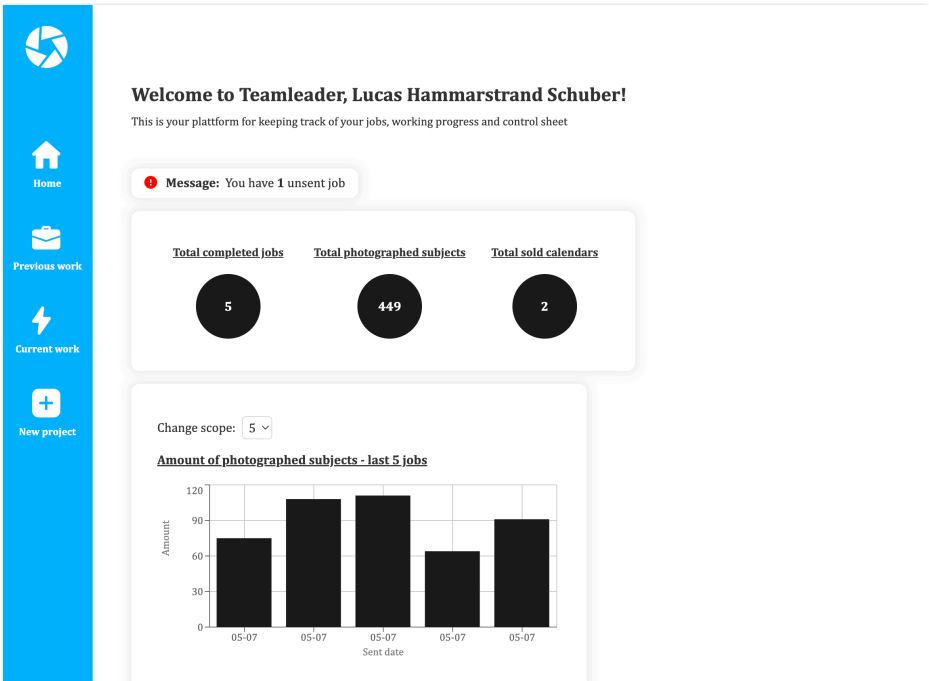
Ja, tack!

Nej, vi är inte intresserad av att tjäna pengar till lagkassan genom att sälja sportkalendrar!

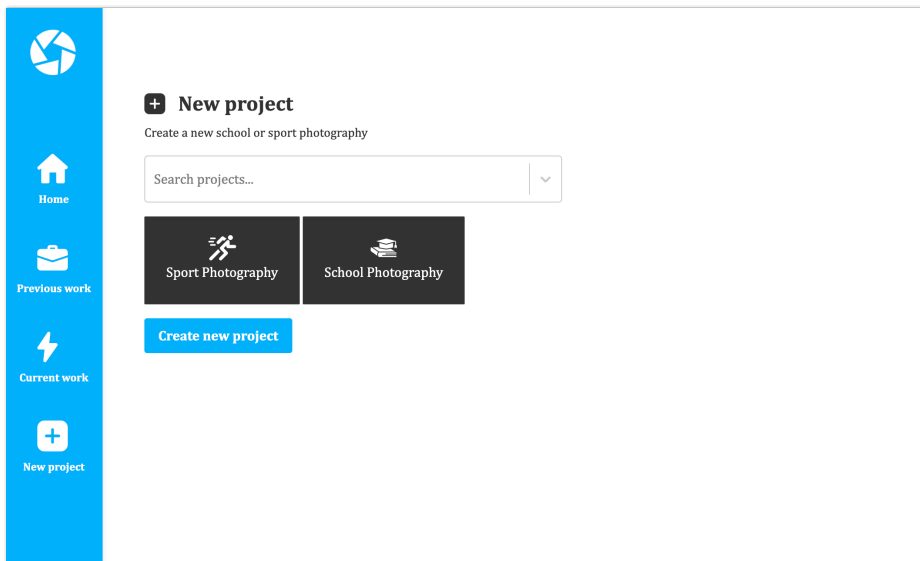
3. Currwork_teamleader



4. Home_teamleader

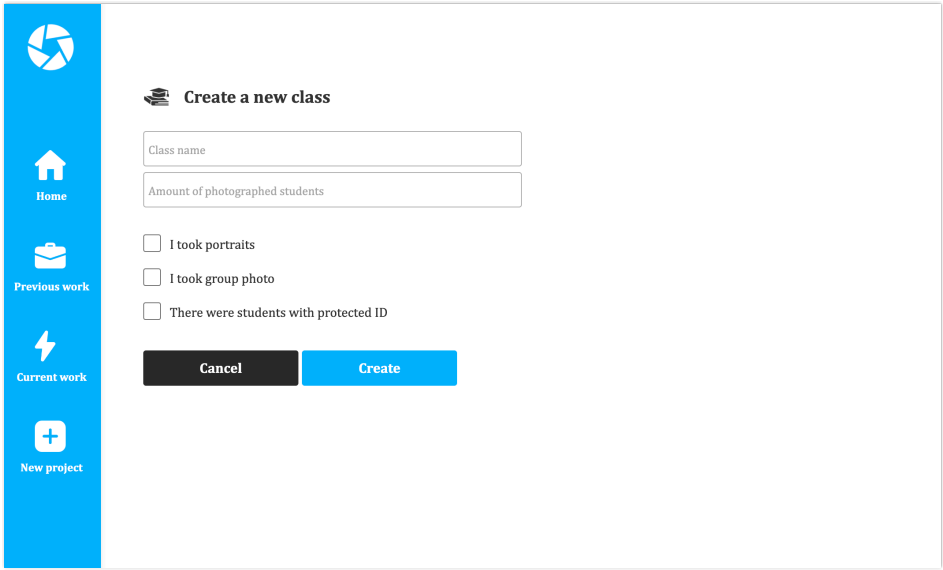
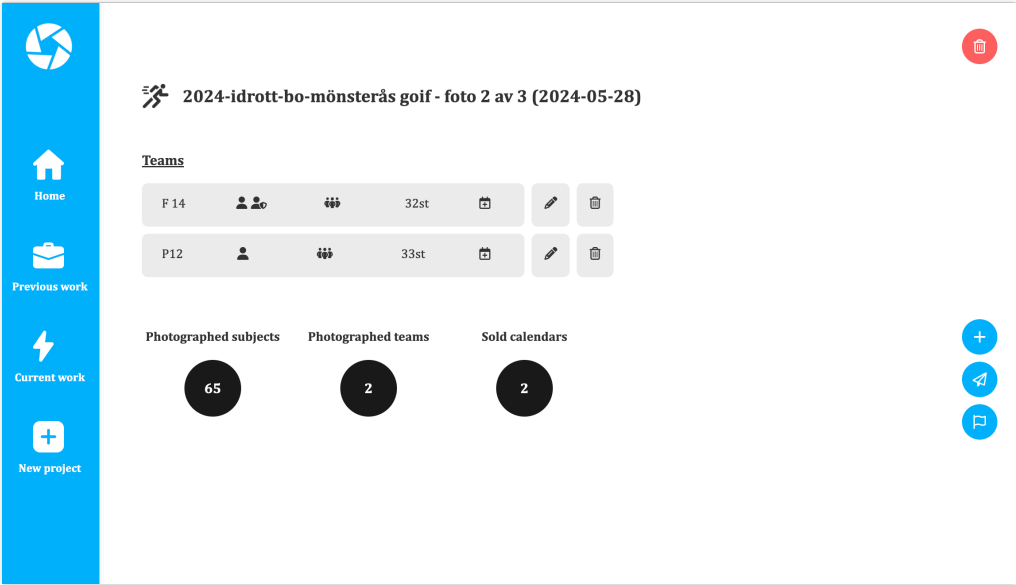


5. Newproject_teamleader



6. Newteam_teamleader

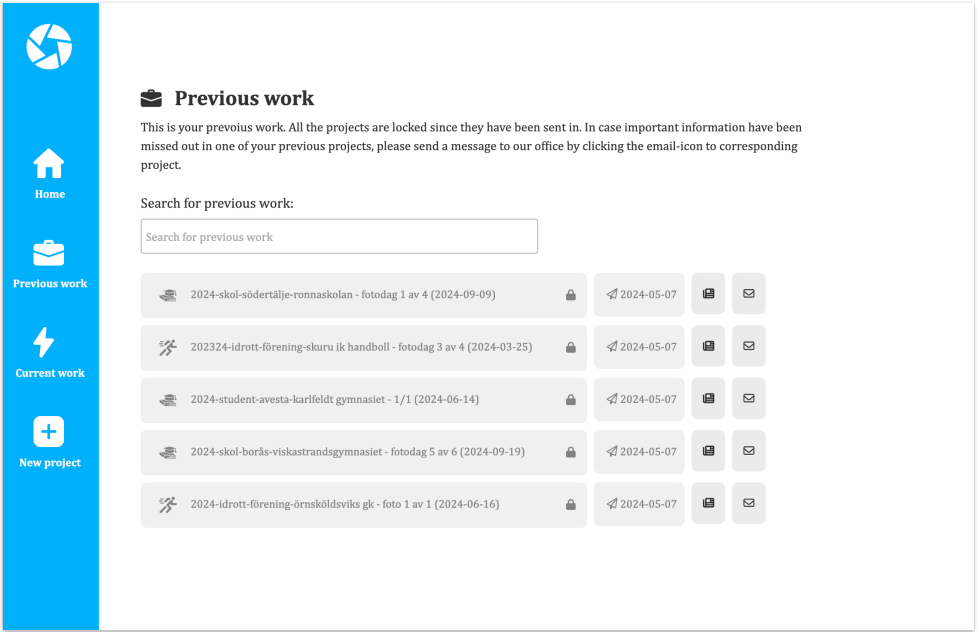
7. Portal_teamlader



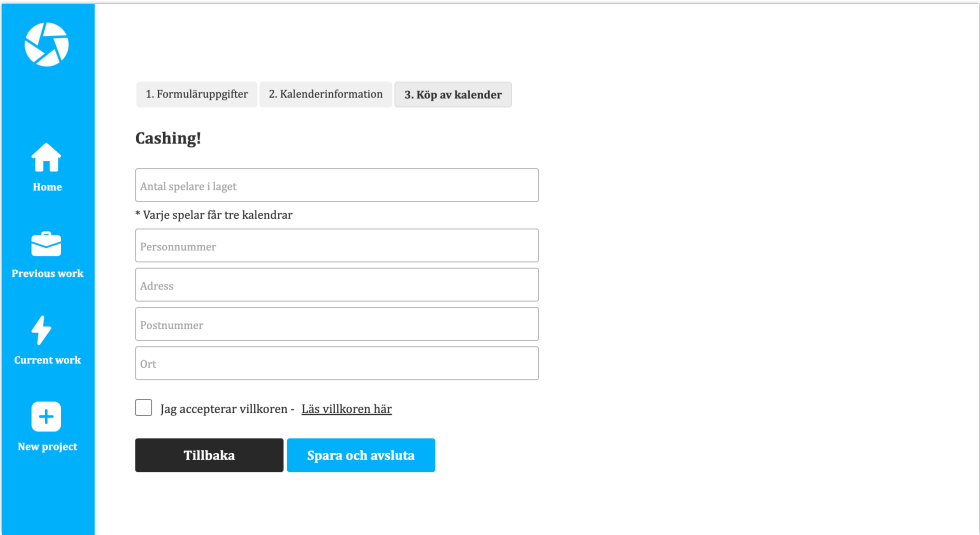
8 .

vwork_teamleader

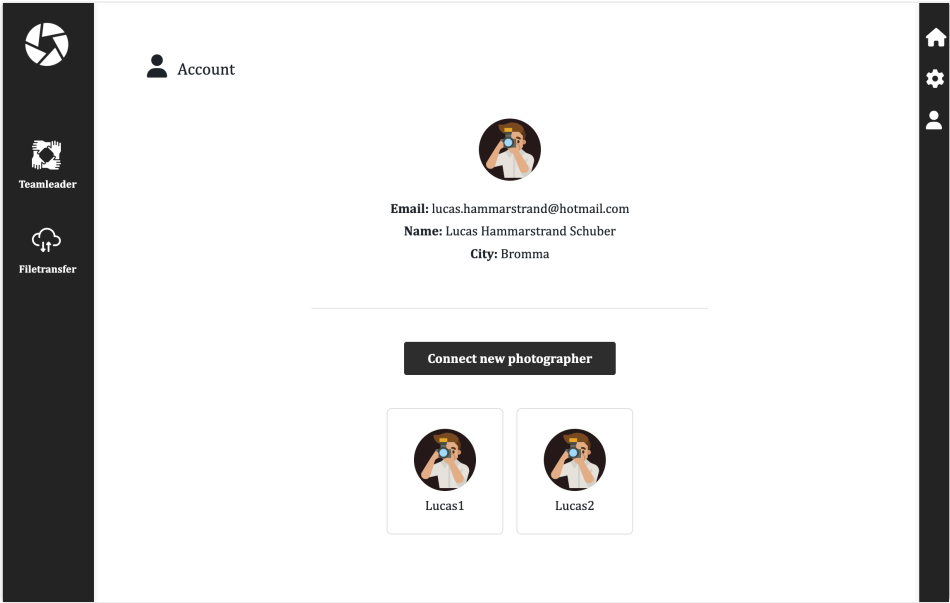
Pre-



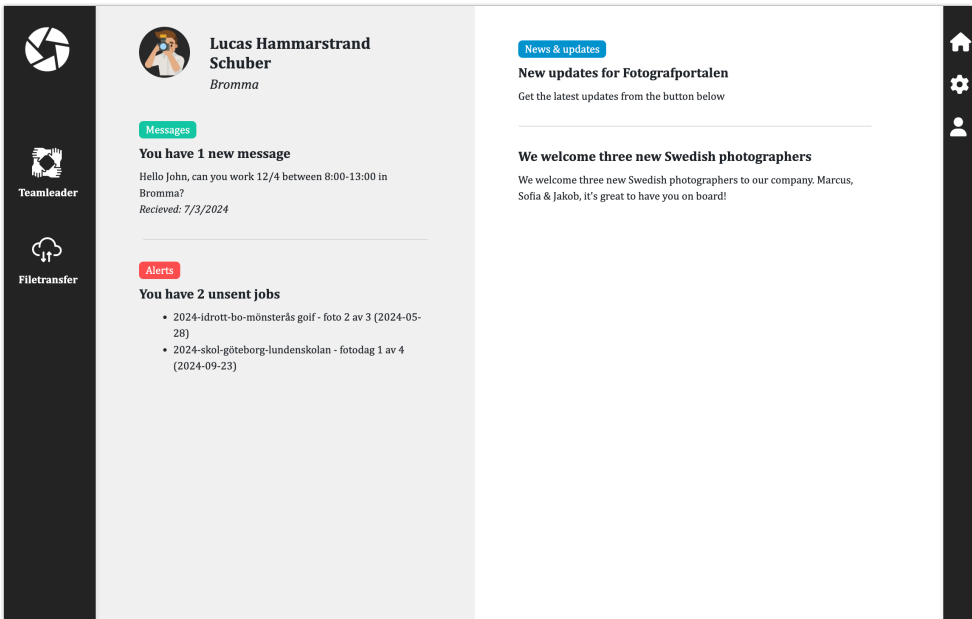
9. yescalendarsale_teamleader



10. Account



11. Index



12. login_window och register_window

Log in

lucas.hammarstrand@ho

Password

Log in

Dont have an account? Register here!

Register

lucas.hammarstrand@ho

Password

Register

Already have an account? Log in here!

13. Settings

Teamleader

Filetransfer

Settings

Here you can change your user setting

Email:

lucas.hammarstrand@hotmail.com

Firstname:

Lucas

Lastname:

Hammarstrand Schubert

City:

Bromma

Language:

Swedish

Update ✓

Home

Settings

User

