Självständigt arbete

Din Digitala Bovärd

Markus Vickman

MITTUNIVERSITETET
Institutionen för data- och elektroteknik (DET)

Examinator: Lars Lundin, *Lars.Lundin@miun.se*
Handledare: Mikael Hasselmalm, *Mikael.Hasselmalm@miun.se*
Författare: Markus Vickman, mavi2302@student.miun.se
Utbildningsprogram: TWEUG, Webbutveckling, 120 hp
Huvudområde: Datateknik
Termin, år: VT, 2025

Sammanfattning

Sammanfattningen fungerar som en beskrivning av rapportens innehåll. Den ska underlätta en snabb genomgång av dokumentet och därför utgöra ett koncentrat av rapporten i sin helhet, det vill säga rymma allt från syfte och metod till resultat och slutsats. Exempel: ”Målet med denna undersökning har varit att besvara frågan... . Undersökningen har genomförts med hjälp av.... Undersökningen har visat att...”. Nämn inget stoff som inte behandlas i rapporten. Sammanfattning skrivs i ett stycke. 200-250 ord är en rekommendation. Hänvisningar till rapportens text, källor eller bilagor är inte tillåtet, utan sammanfattningen ska ”stå på egna ben”. Undvik såväl formler och matematiska symboler som kursiv och fet stil. Sammanfattningen kan avslutas med en uppräkning av nyckelord, som kan underlätta sökande efter rapporten i biblioteksdatabaser. Exempel:

Nyckelord: Människa-dator-interaktion, XML, Linux , Java.

Abstract

Abstract, det vill säga motsvarande sammanfattning på engelska, krävs i examensrapporter. Abstract skrivs i ett stycke.

Keywords: Exempel: Human-computer interaction, XML, Linux, Java.

Förord

Förord är inte obligatoriskt men kan tillämpas om du som skribent vill inkludera några personliga ord, till exempel tack till personer som hjälpt dig. Denna text ska alltid skrivas på en egen sida.

Innehållsförteckning

Terminologi

Förkortningar

MoSCoW	Must have, Should have, Could have and Won't have (this time).
MVP	Minimum Viable Product.
SCB	Statistiska centralbyrån även kallat Statistikmyndigheten SCB.
CMS	Content Managment System
DOS-lagen	Lagen om digital offentlig service
WCAG	Web Content Accessibility Guidelines
API	Application programming interface
REST	Representational State Transfer
SQL	Structured Query Language

1 Introduktion

Projektet är del av ett examensarbete inom datateknik hos Mittuniversitetet och syftar till att utveckla en webbapplikation för bovärdar av hyres- och bostadsrätter. Projektet utfördes hos företaget Websystem. I applikationen ska bovärdar kunna lista alla sina objekt ex. hus eller lägenhetshus med alla innehållandes lägenheter. Bovärden ska kunna lägga till namn och e-post till boende som är kopplade till lägenheterna. För varje tillsatt bostad ska hyran faktureras ut vid önskat datum via mejl till hyresgästerna. Projektkrav listas nedan enligt MOSCOW-metoden. MOSCOW-metoden är ett sätt att tydliggöra för både projektledare och beställare vilka krav en produkt har utifrån ett tydligt prioriteringssystem[1].

Must have

- **Lista fastigheter:** I applikationen ska bovärdar kunna lista alla sina objekt ex. hus eller lägenhetshus med alla innehållandes lägenheter.

- **Fakturering:** Bovärden ska kunna lägga till e-post till boende som är kopplade till lägenheterna. För varje tillsatt bostad ska hyran faktureras ut vid önskat datum via mejl till hyresgästerna.

Should have

- **Bovärdar kan lägga upp information till boende:** Gemensam information som angår flera boende ex. Ett helt hyreshus kan publiceras offentligt för boende att tillgå.

Could have

- **Felanmälningar:** Felanmälningar behöver kunna hantera för att underlätta prioritering och effektivisera för bovärden.

- **Inloggning för boende:** För att boende ska kunna se status på sina felanmälningar kan konton skapas även för boende.

- **Meddelandesystem mellan bovärd och boende:** Ett meddelande system mellan bovärd och boende kan vara ett tidseffektivt sätt för bovärdar att svara på frågor.

Won't have

- **Underhållsplanering:** Fastigheter har behov av planerade underhåll. En funktion för underhållsplanering ger applikationen mervärde.

Avstämningar kommer regelbundet ske för att säkerställa att projektplanen följs. Projektet följer även MVP. MVP som står för ”Minimum viable product” och syftar till att producera minsta produkt som skapar värde[2]. Denna process möjliggör ett flexibelt arbetssätt och säkerställer att projektet kan anpassas efter eventuella förändringar eller nya krav.

1.1 Bakgrund och problemmotivering

Enligt statistik från SCB så bor cirka 50% av Sveriges hushåll i flerbostadshus[3]. Av dessa bor en större del i hyresrätter. Gemensamt för hyresgästföreningar och bostadsrätter är att en avgift betalas in av de som bor i lägenheterna. Båda boendeformerna behöver hantera underhåll och felanmälningar även om så i olika utsträckning. Det finns ett stort antal potentiella kunder som ett digitalt bovärdssystem skulle kunna underlätta för. Därför skapas det här projektet som ett enkelt digitalt bovärdssystem för mindre bovärdar.

1.2 Övergripande syfte

Syftet är att skapa ett digitalt system för att underlätta för mindre bovärdar. Systemet ska spara tid för bovärdar samt ge större överblick av dennes fastigheter. Det ska även automatisera fakturering och hantera felanmälningar. Med intuitiv och responsiv design kan applikationen bli ett helhetssystem för bovärdar.

1.3 Avgränsningar

Projektet har en tidsbegränsning på 10 veckor. Dessa veckor inkluderar båda det praktiska projektet samt rapportskrivning med projektplanering. Sista veckan i projektet är även avsatt till projektpresentationer. Det kommer krävas att tid avläggs till att sätta mig in i systemutveckling med Drupal. Drupal är ett PHP-baserat ramverk och CMS[4]. För att projektet ska leda till en användbar produkt ska agil systemutveckling användas på ett sådant sätt att en minsta värdes produkt publiceras. Denna produkt ska i mån av tid få extra funktionalitet enligt en MOSCOW-prioritering.

1.4 Detaljerad problemformulering

Ett digitalt system ska skapas för att underlätta för mindre bovärdar. Systemet ska spara tid för bovärdar samt ge större överblick av dennes fastigheter. Det ska även automatisera fakturering för bovärdar. En intuitiv, responsiv design som följer WCAG 2.1 på AA nivå är ett krav. Följande problem behöver lösas:

DOS-lagen
Applikationen ska vara tillgänglig för bovärdar och boende oavsett behov. Därför ska den följa DOS-lagen. DOS-lagen står för ”lagen om digital offentlig service” och är en lag som styr hur innehåll ska struktureras i applikationer från offentliga aktörer[5].

Kontohantering
Bovärdar ska kunna skapa egna konton. Dessa konton ska ge åtkomst till applikationens funktioner. Det ska även gå att återställa lösenord via mejl.

Lista fastigheter och fakturera boende
Det kan vara ett betungande arbete för hyresvärdar att manuellt sammanställa och skicka ut fakturor. Därför ska applikationen kunna skicka ut e-post-fakturor till alla boende.

Extra funktioner i prioriteringsordning

1. Bovärdar kan lägga upp information till boende.

2. Felanmälningar kan hanteras för att underlätta prioritering och effektivisera för bovärden.

3. Ett meddelandesystem mellan bovärd och boende kan vara ett tidseffektivt sätt för bovärdar att svara på frågor.

4. För att boende ska kunna se status på sina felanmälningar kan konton skapas även för boende.

5. En funktion för underhållsplanering ger applikationen mervärde.

1.5 Översikt

I kapitel 2 förklaras teori om olika tekniker och arbetssätt för projektet. Kapitel 3 innefattar projektets tänkta metoder och planerad arbetsgång. Arbetets utförande beskrivs detaljerat i kapitel 4. I Kapitel 5 redogörs för projektets resultat och måluppföljning. Rapporten avslutas i kapitel 6 med en diskussion av projektets genomförande och resultat samt dess etiska aspekter.

1.6 Författarens bidrag

Projektet har till stor del varit självständigt. Val av metoder och tekniker för att lösa projektet gjordes i samråd med Joakim som var min handledare samt frontend och Drupal-utvecklare hos företaget Websystem. Fortlöpande har jag även givits viss rådgivning och genomgång av Drupal.

2 Teori

2.1 Definition av termer och förkortningar

Moscow-metoden

MOSCOW-metoden är ett sätt att tydliggöra för både projektledare och beställare vilka krav en produkt har utifrån ett tydligt prioriteringssystem. MOSCOW är en förkortning för och sorteras efter ”Must have”, ”Should have”, ”Could have” och ”Wont have”. [1]

Minimum viable product

MVP som står för ”Minimum viable product”. Produkten kan i vissa fall vara en prototyp eller en produkt utan substans. Men oftast syftar det till att producera en minsta produkt som skapar värde. Detta snabbar på utvecklingen och gör det tydligare för hur produkten kan utvecklas i framtiden.[2]

Drupal

Drupal är ett open-source ramverk som bygger på PHP-ramverket Symfony. Drupal kan även användas som ett CMS likt WordPress. Det går också att använda Drupal som en komplett backend-lösning. Det finns stöd för att använda färdiga moduler samt programmera egna. Drupal kopplas till en MySQL/MariaDB-databas vid installation.[4]

DOS-lagen

DOS-lagen står för ”lagen om digital offentlig service”. Det är en lag som säger att innehåll på webbplatser och i applikationer från offentliga aktörer och från offentligt finansierade privata aktörer ska vara tillgänglighetsanpassat. Lagen innefattar att dessa ska ha en tillgänglighetsredogörelse för applikationen samt att de ska följa EU-standarden (EN 301 549 V3.2.1). Men eftersom att det redan finns en vedertagen standard inom webbutveckling med samma innehåll vid namn ”WCAG 2.1 (Web Content Accessibility Guidelines)” så går når den också upp till lagkraven.[5]

Symfony

Symfony är ett open-source php-ramverk för webbapplikationer. I Symfony kan färdiga moduler användas. Andra ramverk som Laravel, Drupal och Prestashop bygger på Symfony.[6]

Cron

Cron är en tidsbaserad schemaläggare som kan användas för Unix-liknande system som olika Linux varianter eller macOS. Cronjob är de uppgifter som automatiseras vid olika tidsintervaller med hjälp av Cron.[7]

RESTful Web Services API

RESTful Web Services API är en modul som ger Drupal ett utökat gränssnitt för hantera RESTful-APIer. Den gör så genom att använda klassen RestResource. RestResource är den funktionallitet för RESTful API som är integrerat i Drupal Core. [8]

Entity API

Entity API är Drupals medföljande funktioner för att hantera objekt av olika datatyper. En vanlig datatyp för att hantera innehåll är noder. Noder fungerar som en SQL-tabell även om de i realiteten består av en tabell per datarad. Specifik data kan hämtas från databasen med ”entity query”.[9]

Hooks

Hooks i Drupal är ett sätt att interagera med underliggande funktionalitet eller funktionalitet i moduler. Ett exempel är när en funktion implementerar Cron. Då deklareras Cron direkt i funktionsnamnet genom att avsluta med ”_cron” ex. `function custom_invoice_cron() {}`.[10]

Drush

Drush är troligen det vanligaste verktyget för Drupal-utveckling. Drush funktionalitet innefattar bland annat generering av kod och export av Drupal installationen.[11]

Composer

Composer är en pakethanterare för PHP som även kan hantera paketens beroenden av andra paket. Composer fungerar väldigt likt npm, yarn och bundler som är pakethanterare för andra programmeringsspråk.[12]

Tailwind CSS

Tailwind CSS är ett CSS-ramverk där CSS skrivs direkt som klasser i HTML-koden. Dessa klasser är ofta en direkt referens till en CSS-egenskap. Det gör att utvecklaren får full frihet i hur stylingen ska se ut utan att använda färdig styling.[13]

3 Metod

Projektet ska följa en agil arbetsmetodik. Med principer som MVP som står för ”Minimum valuable product” och MOSCOW-metoden. För att projektet ska uppnå MVP skapades i projektplanen produktkrav som prioriterades enligt MOSCOW-metoden.

3.1 Utvecklingsmiljö

Applikationens utvecklingen kommer att ska i operativsystemet Ubuntu. Programmeringen för backend kommer utföras i IDE’n ”PHP Storm” eftersom den har bättre stöd för php och drupal. Till frontend programmeringen kommer ”Visual Studio Code” att användas som IDE.

3.2 Backend

Drupal ska användas som backend och kopplas som ett RESTful-API till lösningens frontend. Programmeringsspråket är PHP samt drupals egen syntax och grafiska interface. Drupal valdes eftersom att Webbsystem är specialiserade på Drupal. Drupal har även många fördelar ska användas i projektet som exempelvis konto-hantering samt dess databas-koppling för lagring och struktur av data.

3.3 Databas

Drupal använder MySQL/MariaDb som databas. Därför kommer MariaDB att användas för projektet. Er-tabeller med dess relationer ska skapas för databasen.

3.4 Användargränssnitt

Grafiska designskisser ska skapas för applikationen i designverktyget Figma. Användargränssnittet ska skapas med frontend-ramverket Vue och programmeringsspråket som ska användas är TypeScript. För styling av webb-applikationen kommer CSS-ramverket Tailwind att användas. Vue och Tailwind ska användas för att de är moderna relevanta tekniker inom webbutveckling. Det är även tek-

niker som används på företaget Websystem. Eftersom att applikationen vänder sig till en bred grupp av samhället ska applikationen följa DOS-lagen:

- Innehålla en tillgänglighetsredogörelse
- Följ designriktlinjer för WCAG 2.1 på nivå AA

3.5 Metodlista

1. Grafisk design i Figma
2. Er-diagram i DrawIO
3. Backend i Drupal
4. Frontend i Vue
5. Anpassning efter DOS-lagen
6. Publicera applikationen

4 Konstruktion

4.1 Grafisk design i Figma

Utifrån målgrupp och applikationens tänkta funktioner listade i kapitel 1.4 detaljerad problembeskrivning skapades en grafiska designskisser i Figma. För designskisserna skapades moodboards med färgpalett, typsnitt och textstorlekar. I bilaga A ”Första design” visas hur fastigheter och boenden kunde listas samt dess moodboard. Designskisserna presenterades för företagets frontendutvecklare samt två externa bedömare. Utvärderingen visade att designen uppfattades som omodern. I samråd med frontendutvecklaren beslutades därför att en ny, enklare och mer modern design skulle utvecklas. De nya designskisserna som fick bättre feedback går att se i bilaga B ”Slutgiltiga designskisser”. Dessa användes som grund till applikationens fortsatta arbete.

4.2 Er-diagram i DrawIO

I systemutveckling med Drupal används en datatyp istället för en databastabell. De fungerar på liknande sätt för den som utvecklar applikationen. Det är dock viktigt att förstå att för varje rad i en tabell i ER-diagrammet skapas en egen tabell i databasen där alla rader är relaterade till en gemensam rad. Varje datatyp innehåller också många standardrader med data utöver de som skapas manuellt.

För att förstå data och dess relationer behövs ändå ER-diagram för Drupal-utveckling. Ett ER-diagram skapades över databasens datastruktur i draw.io som är en applikation för ritning av grafer och diagram. Tabeller skapades för applikationen tänka funktioner enligt MoSCoW-prioriteringen. Nedan beskrivs alla tabeller men de går även att se i sin helhet i bilaga C. ER-diagram.

User(Must have)

Denna tabell skapades för att innehålla uppgifter till applikationens bovärd. Även fast det inte syns i ER-diagrammet så innehåller alla andra tabeller ett fält för vilken användare som har skapat tabellen. I figur 1 visas hur ER-tabellen för användare är uppbyggd.

User(Drupals egna)	
PK	UniqueID
	UserName
	EmailAdress
	Password
	CreatedDate

Figur 1: Er-tabell användare

RealEstate(Must have)

En tabell skapades även för fastigheter. Varje bovärd kan lägga till flera fastigheter. Förutom fastighetsinformation skapades rader relaterade till fakturering. Dessa rader avgör om bovärden vill ha automatisk fakturering för fastigheter och vilka datum det ska ske. Rader skapades även för betalningsmetod och betalningsnummer vilket visas i figur 2.

RealEstate	
PK	UniqueID
FK	UserID
	Title
	InvoiceDueDate
	InvoiceSendDate
	PaymentMethod
	PaymentNumber
	AutoInvoice
	StreetAddress

Figur 2: Er-tabell fastighet

Accommodation(Must have)

Varje fastighet kan innehålla flera bostäder och därför skapades även en tabell för bostad. Figur 3 visar att bostadstabellen innehåller rader för titel(bostadsnamn), hyra sam namn och e-postadress till boendegästen. Bostäder är relaterade till fastigheter.

Accommodation	
PK	UniqueID
FK	RealEstateID
	Title
	Rent
	EmailAdress
	TenantName

Figur 3: Er-tabell bostad

Invoice(Must have)

För att applikationen skulle bli mer användbar behövdes att fakturor kunde sparas och därför skapades även en tabell för fakturor. Denna tabell strukturerades så att den ska kunna hämta data från andra tabeller när den skapas. Figur 4 visar innehållet i tabellen där även relationen till bostad visas.

Invoice	
PK	UniqueID
FK	AccommodationId
	Title
	InvoiceHTML
	InvoiceNumber
	InvoiceStatus
	TenantName
	EmailAddress

Figur 4: Er-tabell användare

Information(Should have)

Eftersom att bovärdarna bör kunna skicka information till boende i deras fastigheter skapades en ER-tabell för information. I figur 5 visas den relativt enkla tabellen för information till boende.

Informationstabellen är relaterad till fastighetstabellen.

Information	
PK	UniqueID
FK	RealEstateID
	Title
	InfoText
	CreatedDate

Figur 5: Er-tabell information

ErrorReport(Could have)

I mån av tid kunde applikationen få ett system för felhantering och därför skapades även en ER-tabell för felrapport. Tabellen innehåller rader för meddelandetyp, meddelande, status och titel. Den innehåller även namn och e-postadress om vem som skrev felrapporten. Relationen till bostad går att se i figur 6.

ErrorReport	
PK	UniqueID
FK	AccomodationID
	Type
	Titel
	Message
	Status
	TenantName
	EmailAdress

Figur 6: Er-tabell felrapport

ServicePlan(Won't have)

Funktionalitet för att planera underhåll kommer inte implementeras i detta projekt utan finns med i projektet som en vision. Underhållsplanering tas med som ett förbättringsförslag. Ett ER-diagram skapades ändå för underhållsplan och går att se i bilaga C. ER-diagram.

4.3 Backend i Drupal

4.4 Frontend i Vue

4.5 Anpassning efter DOS-lagen

4.6 Publicera applikationens

5 Resultat

Resultatkapitlet ingår när du har genomfört en systematisk undersökning, t ex en utvärdering av ett datorprogram som du har utvecklat, vilket krävs inom examensarbeten på C- och D-nivå. I resultatkapitlet redovisas objektiva resultat av en empirisk undersökning., t ex en sådan utvärdering som nämns ovan. Tänk på att eventuella kommentarer i detta kapitel endast får vara av förtydligande art. Dina egna synpunkter och subjektiva (personliga) kommentarer hör hemma i kapitlet Slutsatser/Analys/Diskussion.

Sträva efter att redovisa resultaten, till exempel enkät-, test-, mät-, beräknings- och/eller simuleringsresultat, så överskådligt och lättbegripligt som möjligt. Resultaten presenteras med fördel i diagram- eller tabellform. Redovisning av intervjuer kan bestå av sammanfattningar, eventuellt kompletterade med några konkreta exempel.

Omfattande resultat, till exempel fullständiga sammanställningar av enkätresultat, stora tabeller och långa matematiska härledningar, placeras med fördel i bilagor.

6 Slutsatser / Analys / Diskussion

Efter de objektiva resultaten följer kapitlet Slutsatser/Analys/Diskussion (välj en rubrik), där du presenter dina egna slutsatser, din subjektiva uppfattning, samt kritiskt analyserar resultatens tillförlitlighet och generaliserbarhet.

Om denna del är omfattande kan den indelas i flera kapitel eller under-kapitel, t ex ett analys- eller diskussionskapitel med förklaringar till och kritisk granskning av resultaten, ett slutsatskapitel där de viktigaste resultaten och slutsatserna presenteras, samt ett avsnitt med förslag på fortsatt arbete inom området.

Att återknyta till undersökningens syftes- och målformulering hör till det viktigaste i detta kapitel.

Ge gärna utrymme åt svaren på följande frågor: Vad är projektets nyhetsvärde och viktigaste bidrag till forskningen eller teknikutvecklingen? Har projektets mål uppnåtts? Har uppdraget utförts? Vad är svaret på den inledande problemformuleringen? Har resultatet blivit det väntade? Är slutsatserna generella, eller gäller de bara under vissa förutsättningar? Vilken betydelse har metod- och modellvalet för resultaten? Har nya frågor väckts på grund av resultatet?

Den sista frågan inbjuder till möjligheten att ge förslag till andra, anknytande undersökningar, d.v.s. förslag dels till åtgärder och rekommendationer, dels till fortsatt forskning eller utveckling för den som vill bygga vidare på ditt arbete.

I tekniska rapporter på uppdrag av företag presenterar du här den rekommenderade lösningen på ett problem. Du kan då göra en konsekvensanalys av lösningen ur tekniskt såväl som lekmannaperspektiv, till exempel i fråga om ekonomi, miljö och förändrade arbetsrutiner. Kapitlet innehåller då rekommenderade åtgärder samt förslag på vidare utveckling eller forskning, och utgör således beslutsunderlag för uppdragsgivaren.

6.1 Analys och resultatdiskussion

Gör en djupanalys och diskutera din applikation/resultat/mätresultat. Här kan man vara mer subjektiv i sin beskrivning.

6.2 Projektmetod diskussion

Analysera och diskutera din valda metod, infallsvinkel och/eller valda startvärden.

6.3 Etisk och social diskussion

I detta underkapitel ska du diskutera etiska frågeställningar, hur ditt projekt påverkar människor och dess omgivning socialt. Använd ett mänskligt perspektiv, hur kommer människor att påverkas av ditt arbete, var andra involverade i ditt arbete, enskilda människors personliga integritet? Visa att du har tänkt på hur ditt projekt kan användas av människor och i övrigt av samhället och på vilket sätt detta sker.

Källförteckning

[1] Projektledning, ”MoSCoW metoden” https://projektledning.se/moscow-metoden/ Uppdaterad 2021-08-10. Hämtad 2025-04-25.

[2] Into the commerce, ”What is A minimum viable product(MVP)? – A Complete Guide” https://intothecommerce.com/business/what-is-a-minimum-viable-product-mvp/ Uppdaterad 2025-03-31. Hämtad 2025-04-25.

[3] SCB, ”Boende i Sverige” https://www.scb.se/hitta-statistik/sverige-i-siffror/manniskorna-i-sverige/boende-i-sverige/ Uppdaterad 2023-07-05. Hämtad 2025-03-24.

[4] Drupal, ”Overview of Drupal” https://www.drupal.org/docs/getting-started/understanding-drupal/overview-of-drupal Uppdaterad 2025-02-03. Hämtad 2025-04-25.

[5] DIGG, ”Om lagen om tillgänglighet till digital offentlig service” https://www.digg.se/analys-och-uppfoljning/lagen-om-tillganglighet-till-digital-offentlig-service-dos-lagen/om-lagen Uppdaterad 2024-01-04. Hämtad 2025-04-25.

[6] Symfony, ”What is Symfony” https://symfony.com/what-is-symfony Hämtad 2025-04-25.

[7] W3Schools, ”Bash crontab Command – Schedule Tasks” https://www.w3schools.com/bash/bash_cron.php Hämtad 2025-04-28.

[8] Drupal, ”RESTful Web Services API overview” https://www.drupal.org/docs/drupal-apis/restful-web-services-api/restful-web-services-api-overview Uppdaterad 2024-03-29. Hämtad 2025-04-25.

[9] Drupal, ”Working with the Entity API” https://www.drupal.org/docs/drupal-apis/entity-api/working-with-the-entity-api Uppdaterad 2024-03-25. Hämtad 2025-04-25.

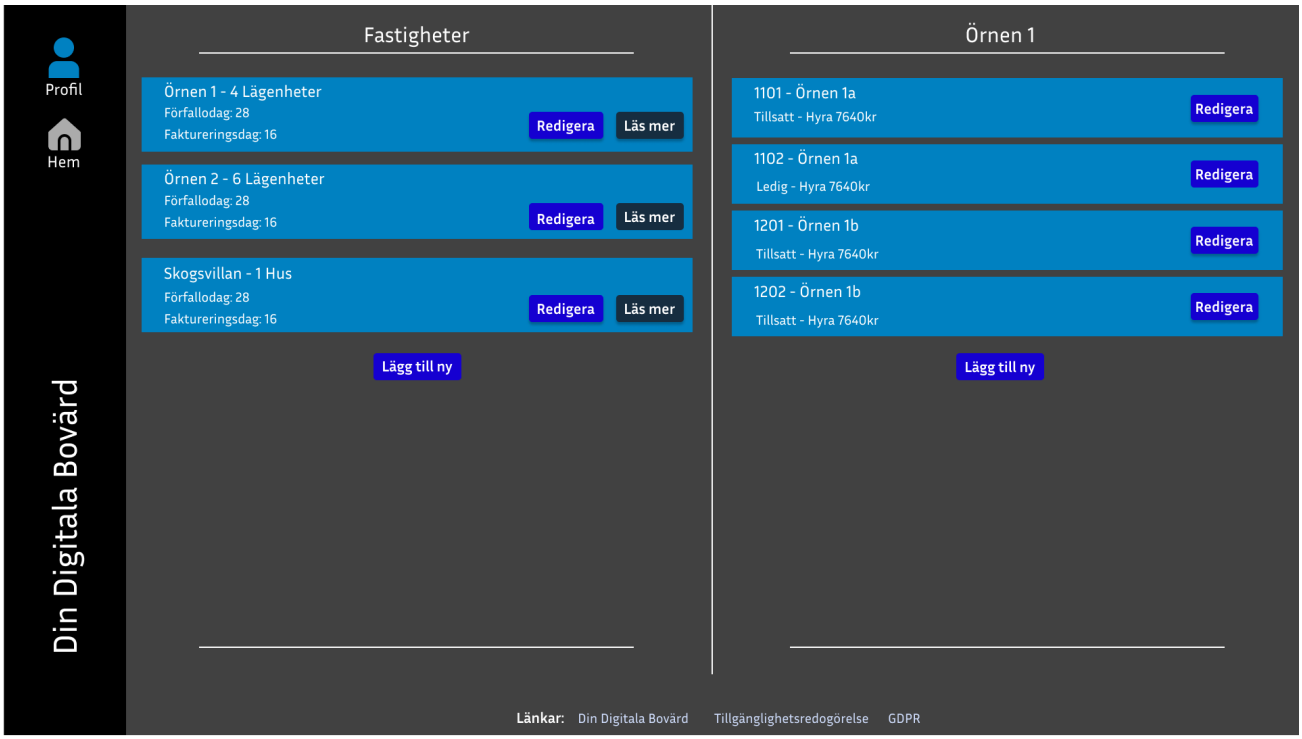
[10] Drupal, ”Understanding hooks” https://www.drupal.org/docs/develop/creating-modules/understanding-hooks Uppdaterad 2025-03-14. Hämtad 2025-04-25.

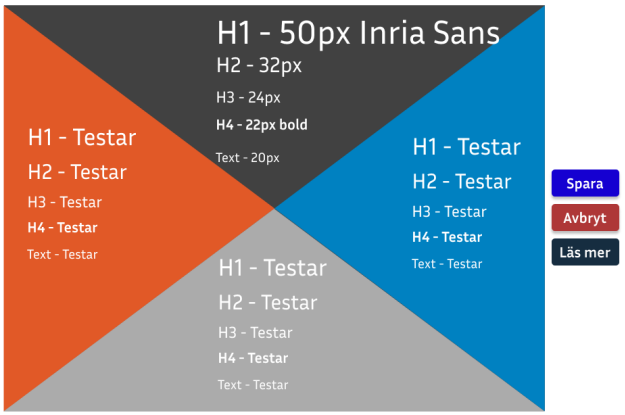
[11] Drupal, ”Drush” https://www.drupal.org/docs/develop/development-tools/drush Uppdaterad 2024-04-03. Hämtad 2025-04-25.

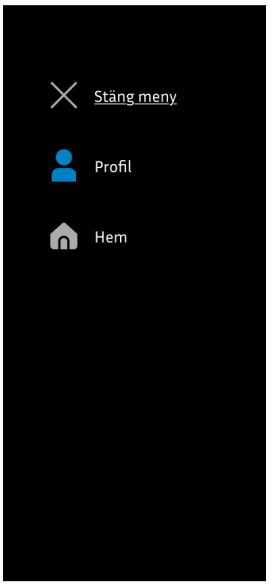
[12] getcomposer, ”Introduction” https://getcomposer.org/doc/00-intro.md Hämtad 2025-04-28.

[13] tailwindcss, ”Rapidly build modern websites without ever leaving your HTML” https://tailwindcss.com/ Hämtad 2025-04-25.

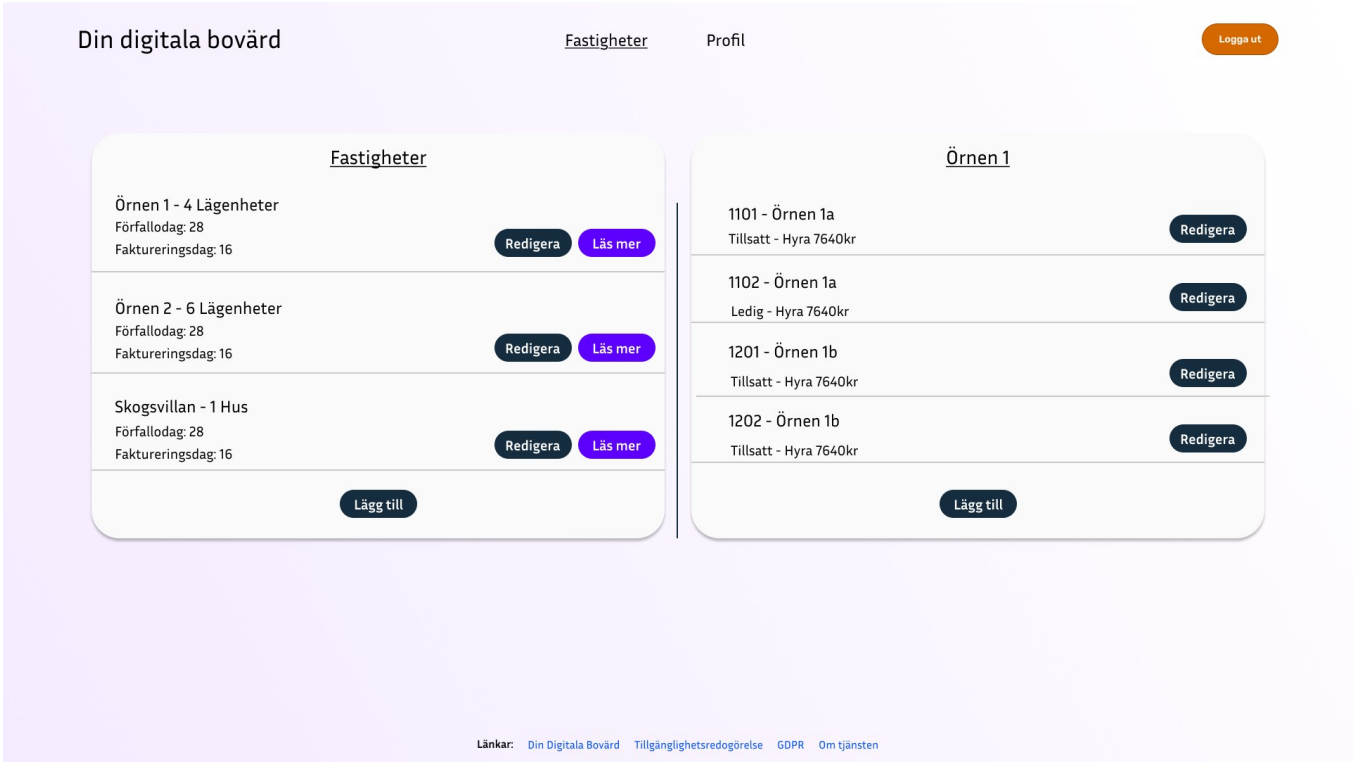
Bilaga A: Första design







Bilaga B: Slutgiltiga designskisser



Bilaga C: ER-diagram