

# **Självständigt arbete**

Din Digitala Bovärd

Markus Vickman

**Examinator:** Lars Lundin, [Lars.Lundin@miun.se](mailto:Lars.Lundin@miun.se)

**Handledare:** Mikael Hasselmalm, [Mikael.Hasselmalm@miun.se](mailto:Mikael.Hasselmalm@miun.se)

**Författare:** Markus Vickman, [mavi2302@student.miun.se](mailto: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 uppräkningslista 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

|   |           |
|---|-----------|
| <b>Sammanfattning.....</b>                      | <b>3</b>  |
| <b>Abstract.....</b>                            | <b>4</b>  |
| <b>Förord.....</b>                              | <b>5</b>  |
| <b>Terminologi.....</b>                         | <b>8</b>  |
| <b>1 Introduktion.....</b>                      | <b>1</b>  |
| 1.1 Bakgrund och problemmotivering.....         | 2         |
| 1.2 Övergripande syfte.....                     | 2         |
| 1.3 Avgränsningar.....                          | 3         |
| 1.4 Detaljerad problemformulering.....          | 3         |
| 1.5 Översikt.....                               | 4         |
| 1.6 Författarens bidrag.....                    | 4         |
| <b>2 Teori.....</b>                             | <b>5</b>  |
| 2.1 Definition av termer och förkortningar..... | 5         |
| <b>3 Metod.....</b>                             | <b>8</b>  |
| 3.1 Utvecklingsmiljö.....                       | 8         |
| 3.2 Backend.....                                | 8         |
| 3.3 Databas.....                                | 8         |
| 3.4 Användargränssnitt.....                     | 8         |
| 3.5 Metodlista.....                             | 9         |
| <b>4 Konstruktion.....</b>                      | <b>10</b> |
| 4.1 Grafisk design i Figma.....                 | 10        |
| 4.2 Er-diagram i DrawIO.....                    | 10        |
| 4.3 Backend i Drupal.....                       | 13        |
| 4.4 Frontend i Vue.....                         | 14        |
| 4.5 Anpassning efter DOS-lagen.....             | 14        |
| 4.6 Publicera applikationens.....               | 14        |
| <b>5 Resultat.....</b>                          | <b>15</b> |
| <b>6 Slutsatser / Analys / Diskussion.....</b>  | <b>16</b> |
| 6.1 Analys och resultatdiskussion.....          | 17        |
| 6.2 Projektmetod diskussion.....                | 17        |
| 6.3 Etisk och social diskussion.....            | 17        |
| <b>Källförteckning.....</b>                     | <b>18</b> |

|   |           |
|---|-----------|
| <b>Bilaga A: Första design.....</b>             | <b>21</b> |
| <b>Bilaga B: Slutgiltiga designskisser.....</b> | <b>22</b> |
| <b>Bilaga C: ER-diagram.....</b>                | <b>23</b> |

# 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 arbetsätt 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 både 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 bovärdar.

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 mer värde.

## **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

#### **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 när 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 deklarerar 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]

## VUE

Vue.js är ett JavaScript-ramverk för att skapa användargränssnitt till webbapplikationer. För att skapa en applikation med Vue används HTML, CSS och JavaScript alternativt TypeScript. Vue kan utvecklas som exempelvis en Single-Page Applikation där webbsidor inte behöver laddas om för att visa nytt innehåll eller med Server-Side Rendering där applikationen renderas på servern istället för på klienten.[14]

## 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 ske 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.

För att skapa denna backend ska drupal-moduler installeras genom pakethanteraren Composer. Färdiga moduler ska användas där de fyller ett behov som exempelvis autentisering. En custom-modul ska också skapas för applikationen. En custom-modul är en modul som utvecklaren skapat själv. Den kan bestå av lite olika filer. Detta projekt kommer behöva kod för hantering av automatiska e-postutskick schemalagda med Cron. Specifik kod kommer även behövas för anpassade e-postmeddelanden samt för skapande av pdf-fakturor. För hantering av data mellan frontend och backend behövs även kod skapas för alla REST-ändpunkter. Dessa ändpunkter ska kopplas till applikationens olika datatyper för åtkomst och manipulering av data. REST-ändpunkter ska även skapas för manuella utskick av fakturor.



### 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 tekniker 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. Testning av webbplatsen
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(Drupal's egna) |                 |
|---------------------|-----------------|
| PK                  | <u>UniqueID</u> |
|                     | 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 bovärderna 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         | <u>UniqueID</u> |
| 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            | <u>UniqueID</u>   |
| FK            | RealEstateID<br>Title<br>Rent<br>EmailAddress<br>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      | <u>UniqueID</u>   |
| FK      | AccommodationId<br>Title<br>InvoiceHTML<br>InvoiceNumber<br>InvoiceStatus<br>TenantName<br>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          | <u>UniqueID</u>                                  |
| FK          | RealEstateID<br>Title<br>InfoText<br>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          | <u>UniqueID</u>   |
| FK          | AccomodationID<br>Type<br>Titel<br>Message<br>Status<br>TenantName<br>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**

Websystem hade redan en Drupal-webbplats igång för utveckling så därför användes den under utvecklingen. Men om en ny installation

behöver göras så installeras Drupal med Composer genom terminal-kommandot `composer create-project drupal/recommended-project:10.4.5 "install-dir"`. Sedan installeras utvecklingsverktyget Drush till projektet med kommandot `composer require drush/drush`. Efter installation kan projektet läggas på exempelvis en Apache-servers public-html katalog för tillgång från webbläsares adressrad. För att installera klart Drupal besöks sedan adressen localhost:port/ där port ska bytas ut till aktuell port för servern alternativt om projektet inte publiceras lokalt ska hela adressraden bytas ut. Väl inne på Drupal-webbplatsen kan utvecklaren följa installationsguiden.

Inställningar för CORS gjordes i filen services.php som finns i katalogen default. I figur 7 visas hur CORS aktiverades samt beviljade metoder.

```
# Configure Cross-Site HTTP requests (CORS).
# Read https://developer.mozilla.org/en-US/docs/Web/HTTP/Access_control_CORS
# for more information about the topic in general.
# Note: By default the configuration is disabled.
cors.config:
  enabled: true
  # Specify allowed headers, like 'x-allowed-header'.
  allowedHeaders: []
  # Specify allowed request methods, specify ['*'] to allow all possible ones.
  allowedMethods: []
  # Configure requests allowed from specific origins. Do not include trailing
  # slashes with URLs.
  allowedOrigins: ['*']
  # Configure requests allowed from origins, matching against regex patterns.
  allowedOriginsPatterns: []
  # Sets the Access-Control-Expose-Headers header.
  exposedHeaders: false
  # Sets the Access-Control-Max-Age header.
  maxAge: false
  # Sets the Access-Control-Allow-Credentials header.
  supportsCredentials: false

queue.config:
  # The maximum number of seconds to wait if a queue is temporarily suspended.
  # This is not applicable when a queue is suspended but does not specify
  # how long to wait before attempting to resume.
  suspendMaximumWait: 30
```

Figur 7: Er-tabell felrapport

#### 4.3.1 Installera moduler

Moduler valdes ut utifrån de funktioner som webbplatsen behövde samt för om de hade kompatibilitet till Drupal version 10.4.5 som användes i projektet. Dessa moduler valdes från Drupals officiella modul-sida[15]. Drupal-moduler kan ofta användas både från PHP-koden i egna moduler samt från det grafiska gränssnittet. Nedan listas de kommandon för moduler som installerades.

**Consumers:** Krävs för att "Simple Oauth" ska fungera och kunna ge tillgång till innehåll. Installationskommando `composer require 'drupal/consumers:^1.19'`

**Simple Oauth:** Modulen används för verifiering av användare för bland annat när en separat frontend används. Den har stöd för OAuth2 med "Bearer token". Installationskommando `composer require 'drupal/simple_oauth:^5.2'`

**RestUI:** Är en modul som ger ett grafiskt interface för konfigurering av modulen "RESTful Web Services" som också installeras när installationskommandot körs. Här kan inställningar göras för vilka REST-resurser som ska vara aktiverade samt med vilka metoder och vilken säkerhet de ska anslutas till. En REST-resurs är en ändpunkt dit externa användare kan skicka förfrågningar till och kan följa med en modul eller så kan utvecklare skriva egna resurser i kod. Installationskommando `composer require 'drupal/restui:^1.22'`

**Symfony mailer:** Modulen är hela e-postsystemet som används i ramverket Symfony. När modulen installeras ersätter den Drupals befintliga e-postfunktion. Den har utökat stöd för att skicka filer och för att formatera e-post med HTML-kod. Installationskommando `composer require 'drupal/symfony_mailer:^1.5'`

**Maillog:** Är en modul som används för att journalföra e-postutskick istället för att skicka iväg dem. Maillog används främst under utveckling och testning av tjänster som skickar e-post. Installationskommando `composer require 'drupal/maillog:^1.0'`

**Serial:** På grund av att Drupal använder innehållstyper istället för rena SQL-tabeller i databasen så har den inte stöd för automatisk ökning av värdet på tabellens id. Serial ger innehållstyper möjligheten att lägga till ett fält med ett unikt värde som ökar för varje ny tabell som skapas. Installationskommando `composer require 'drupal/serial:^2.1'`

**Mpdf:** Modulen gör det möjligt att konvertera HTML-kod till PDF-filer. Den har även stöd för styling, lösenordsskyddade filer samt vattenstämpel. Installationskommando `composer require 'drupal/pdf_using_mpdf:3.x-dev@dev'`

**Admin Toolbar:** Används för att ge ett utökat administrationsgränssnitt för Drupal-webbplatser. Detta gränssnitt visar sig som nedfallande menyer med extra funktionalitet. Installationskommando `composer require 'drupal/admin_toolbar:^3.5'`

Efter att alla tilläggen installerats med Composer navigerade jag på Drupal-webbplatsen till menyvalet utöka. Där aktiverades alla de ovan installerade moduler.

#### 4.3.2 Skapa egen modul

För att ge applikationen dess önskade funktionalitet skapades en egen modul. Modulen innehåller alla REST-ändpunkter för att hantera applikationens data. I Drupal kallas dessa för RestResource. Den innehåller även två nya typer för e-post som används vid e-postutskick från RestResource och för automatisk e-postfakturerings. Dessa e-post typer är typer av EmailBuilder. Nedan redogörs för alla filers struktur och hur funktionaliteten skapades för den egenskapade modulen.

##### Skapa modul

...

##### RestResource

...

##### EmailBuilder

...

##### CRON

...

#### 4.3.3 Inställningar

### 4.4 Frontend i Vue



## **Tillgänglighetsredogörelse**

...

## **Integritetspolicy**

...

## **4.5 Testning av webbplatsen**

Alla tester som utfördes på webbplatsen är på Vue-applikationen. Således gjordes alla ändringar och anpassningar från dessa tester i Vue-applikationen och inte i Drupal.

### **4.5.1 HTML-validering**

Webbplatsens html-kod validerades genom tjänsten "Markup Validation Service". Tjänsten tillhandahålls av "W3C" och är till för att säkerställa att html-kod följer html-standarden[16].

Varje sida validerades och gav några fel som duplicerade id, typfel på textarea samt att html-språket inte var definierat till svenska. Dessa fel åtgärdades genom att överflödiga id togs bort, typningen av textarea togs också bort samt att språk för html lades till.

### **4.5.2 Tester för tillgänglighet**

Alla applikationens vyer samt modals testades för tillgänglighet med det automatiska testverktyget "Ace it". "Ace it" är utvecklat av företaget "Useit" och är ett testverktyg specifikt för tillgänglighet. Enligt "Useit"[17] gör Aceit en fullständig tillgänglighetsgranskning enligt kraven i WCAG samt den europeiska standarden EN 301 549.

Testverktyget går att använda på specifika webbsidor eller för en hel webbapplikation. Men eftersom att min webbapplikation kräver inloggning för att visa alla element så testades den genererade html-koden direkt istället. För att göra det kopierade jag html-koden för varje sida och för varje vue-komponent som renderades på skärmen. Nedan visas resultat och åtgärder från testerna för respektive sida.

**Startsidan** fick inga fel enligt WCAG 2.1 och fick ett betyg på 118%. I

bilaga D. "Ace it" går att se hur resultatet för tillgänglighetstestet presenterades.

**Om tjänsten** lika så fick inga fel och betyget 118%.

**Integritetspolicy** fick följande fel "Förvarna användaren om länken öppnar ett dokument". Texten "Ladda ner integritetspolicy" lades till som aria-label attribut och länktext. Testet fortsatte ändå att anmärka fel för länken och gav betyget 98%.

**Tillgänglighetsredogörelse** fick samma fel, åtgärder och resultat som integritetspolicy.

**Inloggningsidan** gav fel på att nav-element saknades innan användaren loggades in samt att platshållare för inmatningsfält inte är till hjälp för skärmläsare. Nav-elementet för huvudmenyn ändrades så att den alltid syns samt fick attributet aria-label="Huvudmeny". Ett nav-element skapades även för länkarna i sidfoten. Inmatningsfält för registrering och inloggning gavs också attribut för aria-labels. Betyget gick från 91% till 118%.

**Inloggningssidan för hyresgäster** gav fel samma fel för platshållare på inmatningsfält och därför lades det till attribut för aria-labels. Betygen gick från 93% till 118%.

**Hyresgästsidan** gav fel på att tomma element renderas på skärmen. Det tomma elementet var ett p-element som ska innehålla eventuella felmeddelanden. För formuläret adderades v-if="errorMessage" till p-elementet för felmeddelande för att det endast skulle skriva om det faktiskt fanns ett felmeddelande. V-if används för att villkorsmässigt rendera en block om uttrycket är sant[18]. Betyget gick från 95% till 118%.

**Fastighetssidan** gav samma fel som uppkommit tidigare som tomma element och saknad attribut för aria-label på inmatningsfält. Dessa åtgärdades med v-if för att kontrollera om felmeddelande fanns samt lades aria label till. Betyget gick från 93% till 116%.

**Fakturasidan** gav även den fel för tomma element samt saknade aria-label attribut och därför utfördes samma åtgärd som för föregående sida. Betyget gick från 93% till 116%.

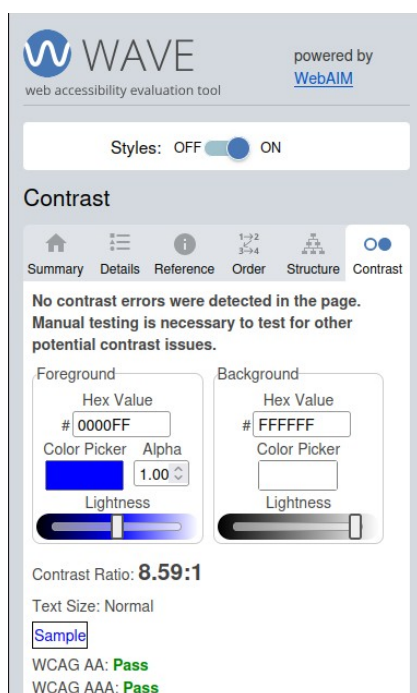
**Felhanteringssidan** gav även den fel för tomma element och saknade attribut för aria-labels. Förutom det gavs fel på att en tabellkolumn saknade attributet `scope="col"`. Attributet används för att specificera att tabellcellen är ett kolumnhuvud. Dessa fel åtgärdades. Betyget gick från 88% till 118%.

**Informationssidan** gav endast fel på ett tomt element och detta fel åtgärdades med en v-if. Betyget gick från 95% till 118%.

#### 4.5.3 Testning med Lighthouse

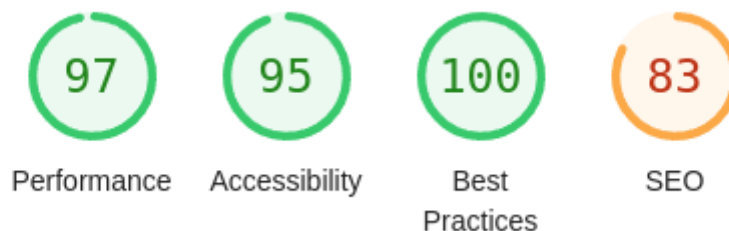
Testverktyget Lighthouse användes för att testa webbapplikationen. Lighthouse är ett inbyggt testverktyg i webbläsaren Google Chrome och är till för att testa webbplatser för prestanda, tillgänglighet, bästa metoder och sökmotoroptimering[19].

Alla webbplatsens sidor testades och resultatet blev ganska lika för alla sidor. För tillgänglighet gav testet fel för kontrasten mellan de blåa knapparna och texten. Eftersom att jag var nöjd med kontrasten kontrollerades även webbplatsens kontraster med webbläsartillägget Wave. Wave är utvecklat av Webaim och är till för att testa webbplatser för tillgänglighet enligt kraven WCAGs olika standarder[20]. I figur 8 kontrasttest visas ett exempel på hur ett kontrasttest ser ut i Wave.

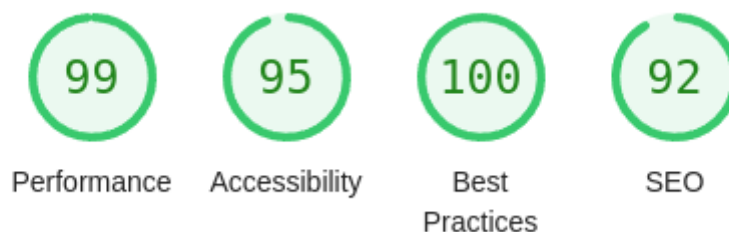


Figur 8. Kontrasttest

Resultatet från Lighthouse var bra med godkända värden på allt utom sökmotoroptimering. Fel för sökmotoroptimering var att en robots.txt-fil saknades. Robots.txt ska innehålla instruktioner för hur webbplatser ska indexeras av bland annat sökmotorer[21]. Webbplatsen saknade även en meta-beskrivning för webbplatsens innehåll. För att lösa dessa problem skapade jag en textfil i Vue-projektets public mapp med namnet robots.txt. I den skrev jag texten `User-agent: * Disallow: /`. Denna text är till för att blockera sökmotorindexering och det passade bra eftersom att jag inte behövde att webbplatsen indexerades. I figur 9. "Lighthouse" och figur 10. "Lighthouse åtgärdat" går det att se hur testresultatet för sökmotor optimering ökade från 83 till godkänt 92.



Figur 9. Lighthouse



Figur 10. Lighthouse åtgärdat

## 4.6 Publicera applikationens

### 4.6.1 Publicera Drupal

Som webshotell för Drupal valdes Inleed eftersom att det har stöd för många funktioner. Tjänsten erbjuder egna mariaDb-databaser, mailkonton, Cron-funktionalitet[22]. Dessutom går det att specificera PHP-version samt komma åt sitt webshotell via SSH om behov finns.

Inne på inleed valdes databas i kontrollpanelen. Sedan valdes "skapa databas". Namn för databasen valdes till ddb. Användarnamnet för databasen blev då samma samt så skapades ett slumpmässigt lösenord. Dessa uppgifter sparades då de skulle komma att behövas vid Drupal installationen.

Ett e-postkonto skapades också för applikationen. I inleeds kontrollpanel valdes e-post samt skapa konto. Där valdes användarnamn och lösenord samt domän för e-postkontot. Valen gav e-postadressen "din.digitala.bovard@markuswebb.se".

Drupal installerades om igen på samma sätt som vid nyinstallation. Först installerades Drupal med Composer och sedan installerades alla publika tillägg även de med Composer på samma sätt som i kapitel 4.3.1. I projektets modul-katalog skapades en ny katalog vid namn custom där det egenskapade tillägget kopierades in. Inställningar för CORS gjordes lika dant som i kapitel 4.3.

Nu kopierades hela projektkatalogen över till katalogen public-html hos webbhotellet. Installationsprocessen för Drupal startades genom att index.php-filen besöks genom webbläsaren. Webbadressen blev då domännamn/web. I installationsguiden valdes rekommenderad installation samt att uppgifter för databas samt e-post fylldes i.

Inställningar för applikationen gjordes nu på samma sätt som i kapitel 4.3.3.

För att aktivera CRON på webbhotellet navigerade jag in på "Cron jobs" under "Advanced Features" i Inleeds kontrollpanel. I bilaga E. CRON visas hur jag skapade ett nytt CRON job i Inleed. Väl i inställningarna valde jag att varje morgon klockan fem skulle kommandot "curl -L -s" följas av Drupal-webbplatsens webbadress för att köra CRON. Webbadressen för detta återfinns i Drupal-webbplatsens meny under konfiguration/system/schemalagda aktiviteter.

#### 4.6.2 Publicera Vue

Vue-applikationen publicerades till Netlify. Netlify är en webbhost för statiska webbsidor och webbapplikationer. För mer tillförlitlig

routing vid sidladdning skapades en fil med namnet `_redirects` innehållandes koden `/* / 200`. Denna fil hjälper Netlify att veta hur applikationens routing ska hanteras. Därefter kördes kommandot "npm run build" i terminalen för att transpilera Vue-applikationen. Applikationens dist-katalog kopierades nu in i deploy-gränssnittet hos Netlify. Efter publicering kunde applikationens namn väljas.

## 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, anknyttande 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 lekmanperspektiv, 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.

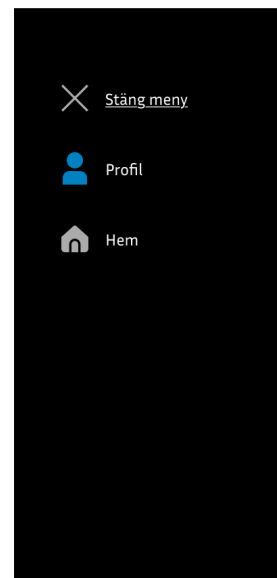
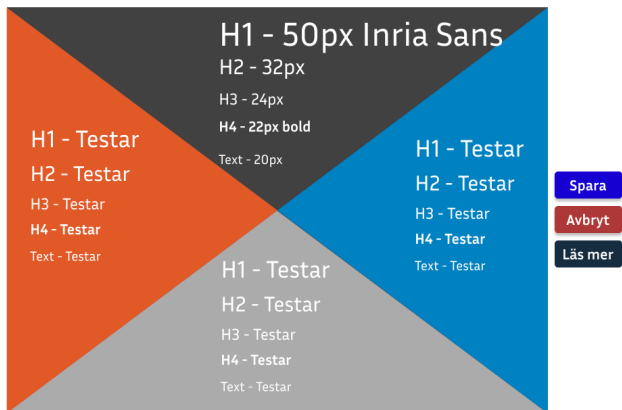
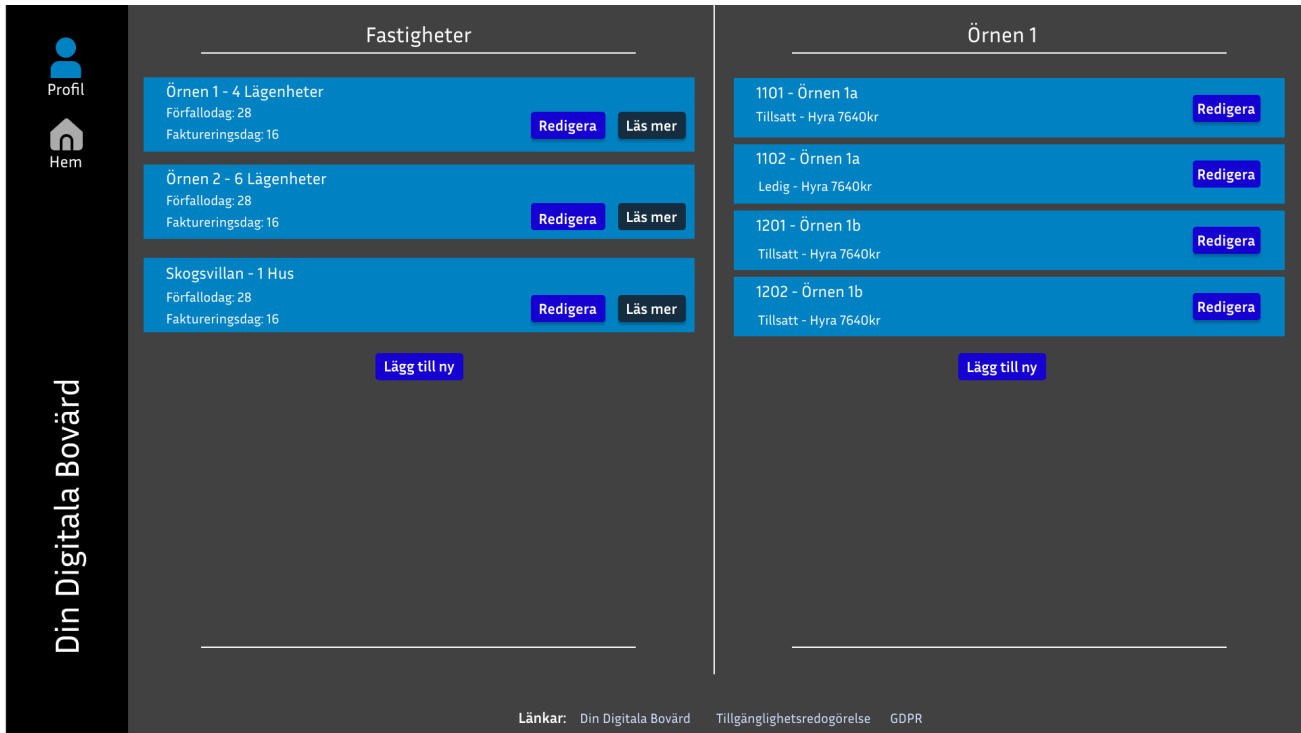
## Källförteckning

- [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](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.
- [14] Vue.js, "Introduction" <https://vuejs.org/guide/introduction.html> Hämtad 2025-05-01.
- [15] Drupal, "Drush"  
[https://www.drupal.org/project/project\\_module](https://www.drupal.org/project/project_module) Hämtad 2025-05-08.
- [16] W3C, "Markup Validation Service" <https://validator.w3.org/> Hämtad 2025-05-08.
- [17] Useit, "Testplattform för digital tillgänglighet" <https://ace.useit.se/ax/about.php> Hämtad 2025-05-08.
- [18] Vue.js, "Conditional Rendering" <https://vuejs.org/guide/essentials/conditional.html> Hämtad 2025-05-08.
- [19] Chrome for developers, "Introduction to Lighthouse" <https://developer.chrome.com/docs/lighthouse/overview> Uppdaterad 2016-09-27. Hämtad 2025-05-08.

- [20] Webaim, "Wave web accessibility evaluation tool"  
<https://wave.webaim.org/> Uppdaterad 2025. Hämtad 2025-05-08.
- [21] SEO-guide, "Robots.txt" <https://www.seo-guide.se/robots-txt>  
Uppdaterad 2014-07-07. Hämtad 2025-05-08.
- [22] Inleed, "Hjälpcenter" <https://login.inleed.net/helpcenter?>  
Hämtad 2025-05-08.
- [23] SEO-guide, "Robots.txt" <https://www.seo-guide.se/robots-txt>  
Uppdaterad 2014-07-07. Hämtad 2025-05-08.

## Bilaga A: Första design



## Bilaga B: Slutgiltiga designskisser

Logga in

### Din digitala bovärd

Tjänsten skapar värde till din förening. Innehåller funktioner som faktura hantering och felanmälningar med mera..

Länkar: [Din Digitala Bovärd](#) [Tillgänglighetsredogörelse](#) [GDPR](#) [Om tjänsten](#)

### Din digitala bovärd

Fastigheter

Profil

Logga ut

#### Fastigheter

Örnen 1 - 4 Lägenheter  
Förfallodag: 28  
Faktureringsdag: 16

Redigera

Läs mer

Örnen 2 - 6 Lägenheter  
Förfallodag: 28  
Faktureringsdag: 16

Redigera

Läs mer

Skogsvillan - 1 Hus  
Förfallodag: 28  
Faktureringsdag: 16

Redigera

Läs mer

Lägg till

#### Örnen 1

1101 - Örnen 1a  
Tillsatt - Hyra 7640kr

Redigera

1102 - Örnen 1a  
Ledig - Hyra 7640kr

Redigera

1201 - Örnen 1b  
Tillsatt - Hyra 7640kr

Redigera

1202 - Örnen 1b  
Tillsatt - Hyra 7640kr

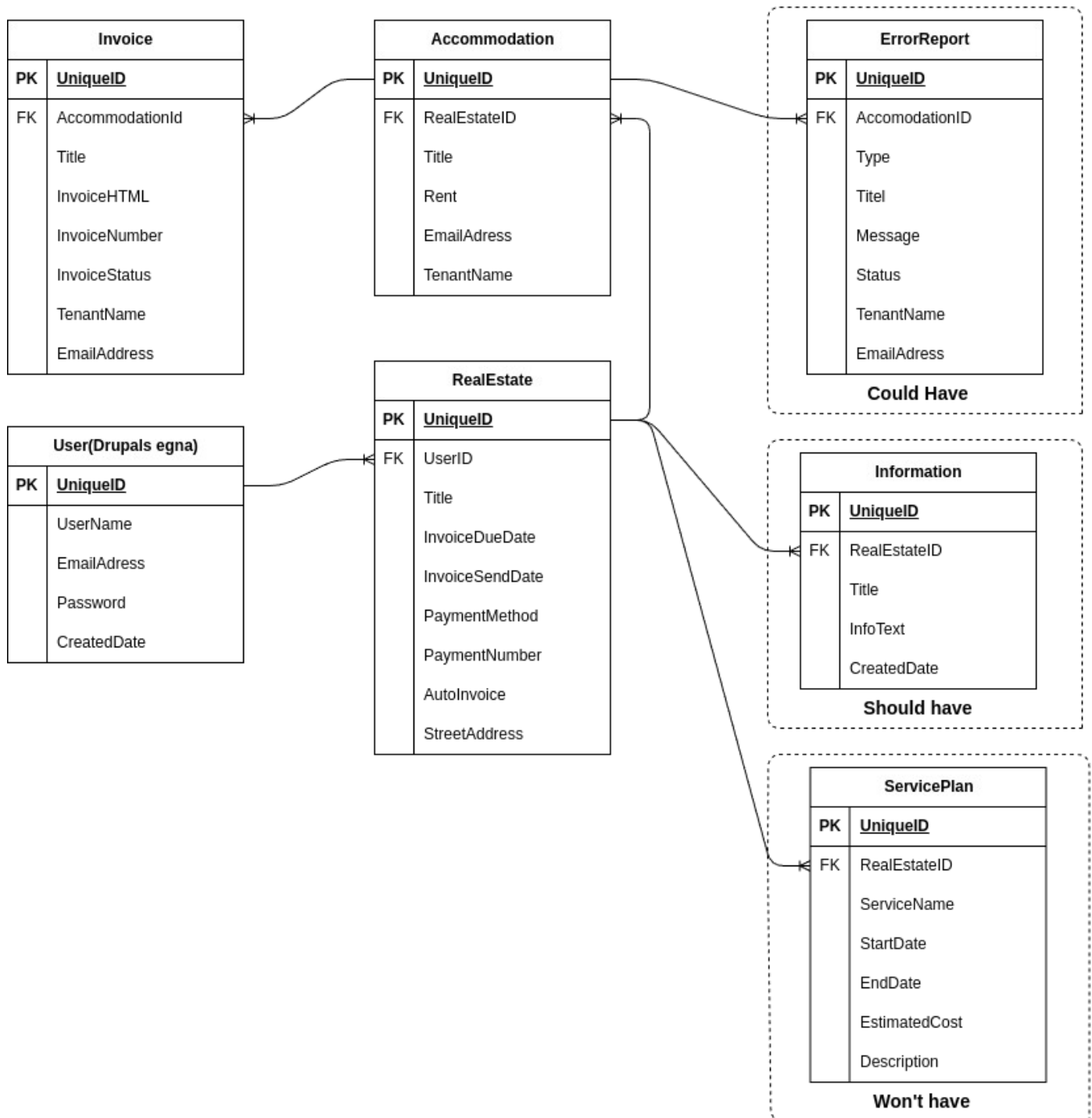
Redigera

Lägg till

Länkar: [Din Digitala Bovärd](#) [Tillgänglighetsredogörelse](#) [GDPR](#) [Om tjänsten](#)

## Bilaga C: ER-diagram

### ER-diagram Din Digitala Bovärd



## Bilaga D: Ace it

### Sammanfattning

#### Analys av:

- Din digitala bovärd  
Direktinmatad\_html-kod (1)
- Analyserar bara inmatade sidor

Analyserade  
sidor

1



#### Antal tester som hittat fel, i snitt per sida

| WCAG 2.1 nivå AA |             |         | Bortom WCAG 2.1 nivå AA |         |
|------------------|-------------|---------|-------------------------|---------|
| Fel              | Troliga fel | Klarade | Fel                     | Klarade |
| 0                | 0           | 26      | 2                       | 10      |

#### Tillgänglighetsindex

118%

- Över 99: Verktöget har inte hittat några fel eller troliga fel kopplade mot WCAG.
- 90-99: Det finns avsteg men troligen inte så omfattande.
- 75-89: Det finns klara problem.
- Under 75: Det finns stora problem på sidan.

0

Antal individuella fel mot WCAG 2.1 nivå AA, i snitt per sida

0

Antal individuella troliga fel mot WCAG 2.1 nivå AA, i snitt per sida

1

Antal individuella övriga fel i snitt per sida

✓ Vad säger siffrorna?

### Resultattabell


✓ Berätta om resultattabellen

Visa samtliga fel som lista

| #    | Sida                 | Struktur              | Rubriker | Bilder | Länkar | Tabeller | Formulär | Ramar | Html | Texter |
|------|----------------------|-----------------------|----------|--------|--------|----------|----------|-------|------|--------|
| 1    | Din digitala bovärd  | Dator, med JavaScript |          |        |        |          |          |       |      |        |
| 118% | Förekomster på sidan | 5                     | 3        | 5      | 8      | 0        | 3        | 0     | 0    | 3      |
|      | WCAG 2.1 nivå AA:    | -                     | 0 0      | 0      | 0 0 0  | 0 0 0    | 0 0 0    | 0     | -    | 0 0    |
|      | Troligt fel:         | 0 0                   | 0 0      | 0 0    | 0 0    | -        | 0        | -     | 0 0  | -      |
|      | Bortom WCAG:         | 0 0 1                 | 0        | 0      | 0      | 0        | 0        | 0     | -    | 0 0    |



## Bilaga E: CRON



Dashboard > Cron Jobs > Edit Cron Job

markus.webb

### Edit Cron Job

Current Time

5/8/2025, 9:49 AM

Minute

0

?

Hour

5

?

Day of Month

\*

?

Month

\*

?

Day of Week

\*

?

Cron job will run: At 05:00 AM

Command

?

curl -L -s https://www.markuswebb.se/theproject/web/cron/hbeoqxI7nAlpNVqJCJpxCFkzV5

PREVENT E-MAIL

SAVE

Explanation:

- Valid Cron time values are the numbers indicated and \*.
- You can specify exact times using commas to separate them. e.g. 1,2,3 (minutes 1,2 and 3)
- You can specify spans using a dash. e.g. 5-7 (minutes 5 to 7)
- You can specify intervals using a star and a forward slash. e.g. \*/2 (every 2nd minute)
- You can combine them to create a more precise schedule. e.g. 1,5,11-15,30-59/2 (minutes 1, 5, 11 to 15 and every 2nd minute between 30 and 59)