**PRODUKTRAPPORT**

Sidernes Verden – Webshop til salg af eventyrlige bøger  
<http://128.199.39.171/>



Michael Aggerholm | Svendeprøve | 9. april - 14. maj 2024

Datatekniker m. speciale i programmering

Indhold

[Indledning 2](#_Toc165309530)

[Læsevejledning 2](#_Toc165309531)

[Case 3](#_Toc165309532)

[Problemformulering 3](#_Toc165309533)

[Arkitektur 4](#_Toc165309534)

[Teknisk produktdokumentation 5](#_Toc165309535)

[Frontend 5](#_Toc165309536)

[Laravel Blade 5](#_Toc165309537)

[Bootstrap 5](#_Toc165309538)

[Backend 5](#_Toc165309539)

[Database 6](#_Toc165309540)

[Server 7](#_Toc165309541)

[Versionsstyring 8](#_Toc165309542)

[Kravspecifikation 9](#_Toc165309543)

[Frontend 9](#_Toc165309544)

[Database 10](#_Toc165309545)

[Backend 11](#_Toc165309546)

[Testspecifikation 13](#_Toc165309547)

[Frontend 13](#_Toc165309548)

[Database 14](#_Toc165309549)

[Backend 15](#_Toc165309550)

[Sikkerhed 18](#_Toc165309551)

[Test 19](#_Toc165309552)

[Laravel Applikationstest Workflow 19](#_Toc165309553)

[Tredjeparts biblioteker sikkerhedstest 19](#_Toc165309554)

[Larastan workflow 19](#_Toc165309555)

[Installationsvejledning 20](#_Toc165309556)

[Brugervejledning 21](#_Toc165309557)

[Forbrugermanual 21](#_Toc165309558)

[Administrator manual 27](#_Toc165309559)

[Konklusion 38](#_Toc165309560)

[Bilag 39](#_Toc165309561)

# Indledning

Sidernes Verden er udarbejdet som svendeprøveprojekt under uddannelsen som datateknikker med speciale i programmering på Tech College i Aalborg.

I rapporten beskrives udviklingsprocessen fra projektstart til endelig aflevering samt hvilke teknologier og metoder, der er valgt.  
Der vil være dokumenteret hvilke udfordringer, som er opstået undervejs, hvordan de er løst, samt hvilke alternativer, der eventuelt ville kunne være valgt.

Formålet med rapporten er at give læseren klart indblik i den udviklingsproces og de valg, som er foretaget under udviklingen af Sidernes Verden.

# Læsevejledning

Denne del er projektets produktrapport. Den bør læses efter procesrapporten da den indeholder beskrivelsen af det endelige produkt.

Produktrapporten indeholder den tekniske produktdokumentation, beskrivelsen af funktioner, specifikationer og tekniske detaljer samt installationsvejledning og brugervejledning.

Under læsningen kan det være en fordel at have adgang til projektet. Det findes frit tilgængeligt via GitHub url:  
<https://github.com/MichaelAggerholm/Apprentice_Test>

Det er også muligt at tilgå webshoppen online via url: <http://128.199.39.171/>   
Guide til brug af webshoppen findes på [side 21](#_Brugervejledning).

# Case

Firmaet Sidernes Verden ønsker en webshop til salg af bøger om eventyrlige fortællinger.

Webshoppen skal give en hurtig og intuitiv oplevelse for kunderne. Det er vigtigt, at kunderne nemt kan finde de ønskede bøger og at grænsefladen er brugervenlig.

Ejeren og medarbejderne hos Sidernes Verden skal have mulighed for at få et fuldt overblik over alle aspekter af webshoppen, herunder bøger, forfattere, udgivere, salg og brugere. Desuden skal medarbejderne have adgang til en log, hvor de kan se, hvilke ændringer, der er foretaget på shoppen, og af hvem.

Det er også vigtigt for virksomheden at have mulighed for at importere nye bøger fra Excel-filer, både før lanceringen af webshoppen og på sigt, hvis der skulle komme aftaler med nye leverandører.

# Problemformulering

Er det muligt at udvikle en webshop, hvor forbrugere kan registrere sig og foretage køb af bøger samtidig med, at medarbejdere kan logge ind, administrere og have overblik over webshoppen?

# Arkitektur

Produktet er struktureret med tre hovedkomponenter, som hver især opfylder specifikke funktioner inden for systemets struktur. En frontend, hvor brugerne kan interagere med systemet, en backend for administration, og en database til opbevaring af data.

**Frontend:**

Frontend-komponenten udgør det brugerorienterede interface, som muliggør interaktion med systemet gennem et web-interface. Denne del er designet til at give brugeren en intuitiv og responsiv brugeroplevelse på diverse enheder og skærmstørrelser.

**Backend:**

Backend-komponenten fungerer som systemets administrative kerne, hvor det er muligt for administratorer at håndtere og overvåge systemets forskellige funktioner.  
Denne del af systemet tilbyder administrering af alle webshoppens entiteter.

**Database:**

Databasen er ansvarlig for lagring af al data og giver en struktureret og effektiv håndtering af webshoppens data, som er essentielle for systemets drift.

**Modulær struktur:**

Arkitekturens modulære design betyder, at de tre hovedkomponenter er adskilte og kan opdateres, testes og vedligeholdes uafhængigt af hinanden. Det giver en høj grad af fleksibilitet og skalerbarhed for webshoppen, hvilket gør det muligt at tilpasse, forbedre eller udskifte hver del uden at påvirke resten af systemets elementer.

# Teknisk produktdokumentation

Beskrivelse af den tekniske opbygning og miljøet for de tre udviklede komponenter.

Frontend

Webshoppens frontend er bygget i Laravel Blade med Bootstrap, hvilket danner et solidt grundlag for en web frontend. Kombinationen giver en effektiv udviklingsproces som prioriterer genbrug af kode, høj modularitet og et responsivt design.

### Laravel Blade

Blade er et templatingværktøj som integreres sømløst med Laravel-frameworket. Det giver mulighed for at opdele frontend kode i genanvendelige komponenter, samt tilbyder brugen af PHP-logik direkte i HTML skabeloner, som fx løkker, betingelser, variabler.

### Bootstrap

Bootstrap er et CSS-framework som tilbyder en stor samling af foruddefinerede komponenter og layouts. Bootstrap komponenter understøtter responsiv visning på tværs af forskellige enhedsstørrelser.

## Backend

Valget af Laravel som backend framework er baseret på en række tekniske overvejelser, blandt andet er hele kombinationen af Laravel, Blade og SQLite en meget stærk og elegant sammensætning som giver et hurtigt og let arbejdsmiljø med et stort antal foruddefinerede funktionaliteter.

Laravel benytter MVC arkitektur (Model-View-Controller), som giver en struktureret tilgang til at organisere og separere logikken i backenden. Det bidrager til at skabe en mere skalerbar og testbar kodebase.

Eloquent ORM kommer inkluderet i Laravel og tilbyder en let måde at interagere med databasen på, ved hjælp af PHP-objekter.

Migrations er endnu et stærkt element i Laravel, som giver mulighed for at versionsstyre og administrere databaseændringer. Der kan blandt andet rulles tilbage til forrige database-migreringer hvis der skulle opstå fejl eller mangler i nyere migrationer.

Laravel kommer med indbygget sikkerhed som CSRF-beskyttelse, inputvalidering og XSS (Cross-site scripting) beskyttelse, som reducerer risikoen for sikkerhedsbrud.

Database

Webshoppens backend er konfigureret til at benytte en SQLite database, SQLite er et letvægtsdatabasesystem hvilket er ideelt til brug for mindre til mellemstore webapplikationer som denne.

I Laravel genereres tabeller ved hjælp af migrations, hvilket giver en struktureret tilgang til håndtering af databaseændringer, databasen defineres i koden og kan opdateres til databasen ved hjælp af Artisan-kommandoer.  
Artisan kommandoer er Laravel’s indbyggede kommando sprog som letter arbejdet for eksempelvis at slette en hel database og køre den frisk ind igen, eller opdatere en eksisterende database.

SQLite er en standard understøttelse i Laravel hvilket gør det let at interagere med tabeller og data som PHP-objekter gennem Laravel's Eloquent ORM.

For at sikre kvaliteten af koden kan SQLite også bruges til at køre tests i et isoleret miljø.

SQLite og Laravel som en kombination giver en strømlinet og effektiv tilgang til databaseadministration via Laravel’s backend ved brug af code-first princippet.

[Findes også i bilag side 39.](#Database_diagram)

****

# Server

Projektet er opsat og udviklet på en lokalt Ubuntu server, herefter pakket som en container og flyttet til en online web server i form af en Ubuntu 22.04, i dette afsnit beskrives begge miljøer samt opsætning.

**Udviklingsmiljø**

Webshoppens udviklingsmiljø er sat op ved hjælp af Laravel Sail på en lokal Ubuntu-server, som kører via WSL2 (Windows Subsystem for Linux 2). Dette miljø udnytter Docker til at skabe en ensartet udviklingsplatform, hvor alle nødvendige komponenter er containeriserede.

Laravel Sail er optimeret til Laravel-applikationer og automatiserer opsætningen af Docker-miljøet. Under opsætningen kan konfigurationer tilpasses, herunder valget af web-server og database.

Sail bruger PHP til at køre applikationen, PHP er et serverside scripting sprog, her har man under opsætningen selv mulighed for at vælge hvilken version af PHP der skal tages i brug, ellers bruger opsætningen som standard den seneste stabile (LTS) version.

Nginx bliver benyttet som web-server samt reverse proxy, det vil sige at Nginx står for at levere indhold til klienter, men også at håndtere anmodninger fra klienter til applikationen.

Xdebug et værktøj til fejlfinding i PHP-kode som er integreret i servermiljøet for at lette udviklingsprocessen.

Guide til opsætning af lokalmiljø kan findes på [side 20.](#_Installationsvejledning)

**Online web-server**

Webshoppens live servermiljø er opsat på en såkaldt ”Droplet” oprettet på DigitalOcean, som kører Ubuntu 22.04, dropletten er konfigureret til at spejle det lokale udviklingsmiljø, hvilket sikrer en sømløs overgang fra udvikling til drift.

Efterfølgende ændres konfigurationen i Laravel’s miljøvariabel fra Development til Production, samt npm køres som live fremfor Development, dette påvirker flere aspekter af webshoppens adfærd og ydeevne, der vises blandt andet ikke fejlbeskeder længere da debug deaktiveres under produktion, caching aktiveres for at forbedre applikationens respons samt nedsætte belastningen af server ressourcer.

Css og JavaScript filer komprimeres og caches på serveren som ”minifed” versioner, hvilket reducerer leveringstiden for levering til klienten, samt indlæsningstiden for klienten, hvilket giver en bedre og hurtigere brugeroplevelse for forbrugere af siden.

Web-serveren kan tilgås via url: <http://128.199.39.171/>   
Brugervejledning af webshoppen findes på [side 21.](#_Brugervejledning)

# Versionsstyring

Projektet anvender Git som versionsstyringsteknologi, valget er grundet Gits pålidelighed og effektive måde at administrere og opretholde en velorganiseret og pålidelig kodehistorik.

Der anvendes GitHub som platform til visuelt at danne overblik over projektet samt at vedligeholde diverse pipelines, også kaldet GitHub Actions / Workflows som projektet benytter til tests og at finde kendte sikkerhedsbrud i koden samt i inkluderede tredjeparts biblioteker.

[Se bilag: GitHub Repository s. 41](#GitHub_Repository)

# Kravspecifikation

Kravspecifikation opdelt i henholdsvis Frontend, Backend og Database.

## Frontend

Frontend delen af projektet skal sikre overskuelig og brugervenlig præsentation for sidens kunder, samt give medarbejdere adgang til et struktureret administratorpanel via login.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Krav | Prioritet | Type |
| F1 | Funktionalitet til at søge efter bøger | Høj | Funktionelt |
| F2 | Købsfunktionalitet | Høj | Funktionelt |
| F3 | Brugerregistrering og login | Høj | Funktionel |
| F4 | Adminpanel til webshopstyring | Høj | Funktionel |
| F5 | Produktvisning | Høj | Funktionel |
| F6 | Betalingsgateway | Høj | Funktionel |
| F7 | Responsiv design | Høj | Ikke-funktionel |
| F8 | Sikkerhed | Høj | Ikke-funktionel |

## Database

Databasekravspecifikationen skal sikre en struktureret og effektiv lagring af data for webshoppen.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Krav | Prioritet | Type |
| D1 | Brugertabel | Høj | Funktionelt |
| D2 | Udgivertabel | Høj | Funktionelt |
| D3 | Formattabel | Høj | Funktionel |
| D4 | Tilstandstabel | Høj | Funktionel |
| D5 | Genretabel | Høj | Funktionel |
| D6 | Sprogtabel | Høj | Funktionel |
| D7 | Forfattertabel | Høj | Funktionel |
| D8 | bogtabel | Høj | Funktionel |
| D9 | Bog-Genre relations tabel | Høj | Funktionel |
| D10 | Bog-forfatter relations tabel | Høj | Funktionel |
| D11 | Aktivitetslog tabel | Høj | Funktionel |
| D12 | Ordretabel | Høj | Funktionel |

## Backend

Backend-delen af projektet skal sikre en robust og sikker håndtering af data, forretningslogik og administration af webshoppen.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Krav | Prioritet | Type |
| B1 | Fejlhåndtering og logging | Høj | Funktionelt |
| B2 | Autentificering og autorisation | Høj | Funktionelt |
| B3 | API ‘er til frontend | Høj | Funktionel |
| B4 | Administrationspanel | Høj | Funktionel |
| B5 | Ordrehåndtering | Høj | Funktionel |
| B6 | Brugerhåndtering | Høj | Funktionel |
| B7 | Udgiverhåndtering | Høj | Funktionel |
| B8 | Formathåndtering | Høj | Funktionel |
| B9 | Tilstandshåndtering | Høj | Funktionel |
| B10 | Genrehåndtering | Høj | Funktionel |
| B11 | Sproghåndtering | Høj | Funktionel |
| B12 | Forfatterhåndtering | Høj | Funktionel |
| ID | Krav | Prioritet | Type |
| B13 | Boghåndtering | Høj | Funktionelt |
| B14 | Administratorhåndtering | Høj | Funktionelt |
| B15 | Fejlhåndtering | Høj | Funktionel |
| B16 | Hændelseslogging | Høj | Funktionel |
| B17 | Søgefunktionalitet | Høj | Funktionel |
| B18 | Filtrering af bøger | Høj | Funktionel |
| B19 | Sikkerhed | Høj | Funktionel |
| B20 | Gendannelse af slettede objekter | Høj | Funktionel |
| B21 | Mulighed for download af alle hændelser i Excel format | Høj | Funktionel |

# Testspecifikation

Testspecifikation opdelt i henholdsvis Frontend, Backend og Database.

## Frontend

Oversigt over test krav til frontend.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Acceptkrav | Kritisk | Godkendt | Afvist |
| F1 | Brugere kan søge efter bøger ved hjælp af forskellige søgekriterier som titel, forfatter, genre samt at søgeresultaterne er nøjagtige og relevante for søgningen | Ja | Ja |  |
| F2 | Brugere kan tilføje varer til deres indkøbskurv og gennemføre købet korrekt samt betalingsprocessen fungerer problemfrit og sikkert | Ja | Ja |  |
| F3 | Brugere kan registrere sig med succes og logge ind på deres konto | Ja | Ja |  |
| F4 | Administratorer kan logge ind på Adminpanel og har adgang til nødvendige funktioner til styring af bøger, brugere, ordrer osv. Og at ændringer foretaget i Adminpanel afspejles korrekt i webshoppen | Ja | Ja |  |
| F5 | Produkter vises korrekt på webshoppen med alle relevante oplysninger såsom titel, forfatter, pris, beskrivelse osv. | Ja | Ja |  |
| F6 | Kunder kan gennemføre betalinger uden problemer og betalingen kan findes i Stripe | Ja | Ja |  |
| F7 | Webshoppen reagerer korrekt på forskellige enheder, herunder desktopcomputere, bærbare computere, tablets og smartphones | Ja |  | Ja |
| F8 | Der anvendes HTTPS-protokollen for at kryptere kommunikationen mellem webbrowseren og web-serveren | Ja | Nej | Ja |

## Database

Database testkrav for at verificere at databasen understøtter webshoppen og opfylder de krav som er nødvendige for at kunne levere data.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Acceptkrav | Kritisk | Godkendt | Afvist |
| D1 | brugertabelen indeholder alle nødvendige felter til lagring af brugeroplysninger | Ja | Ja |  |
| D2 | Indeholder alle nødvendige felter til lagring af udgivere | Ja | Ja |  |
| D3 | Indeholder alle nødvendige felter til lagring af bog formater | Ja | Ja |  |
| D4 | Indeholder alle nødvendige felter til lagring af bog tilstande | Ja | Ja |  |
| D5 | Indeholder alle nødvendige felter til lagring af genre | Ja | Ja |  |
| D6 | Indeholder alle nødvendige felter til lagring af sprog | Ja | Ja |  |
| D7 | Indeholder alle nødvendige felter til lagring af forfattere | Ja | Ja |  |
| D8 | Indeholder alle nødvendige felter til lagring af bøger | Ja | Ja |  |
| D9 | Indeholder bog id og genre id for at lave en bog-genre relation | Ja | Ja |  |
| D10 | Indeholder bog id og forfatter id for at lave en bog-forfatter relation | Ja | Ja |  |
| D11 | Indeholder alle nødvendige felter til lagring af administrator aktiviteter | Ja | Ja |  |
| D12 | Indeholder alle nødvendige felter til lagring af ordreoplysninger | Ja | Ja |  |

## Backend

Backend testspecifikation for at sikre at webshoppen fungerer korrekt og opfylder specifikke krav og forventninger.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Acceptkrav | Kritisk | Godkendt | Afvist |
| B1 | backend håndterer alle typer fejl og undtagelser korrekt samt logger hændelsesaktivitet til databasen. | Ja | Ja |  |
| B2 | brugere kan logge ind ved brug af brugernavn/adgangskode og systemet kan skelne mellem normal bruger og administrator brugere | Ja | Ja |  |
| B3 | API 'er til frontend er korrekt implementeret og leverer data som vises på websitet samt håndtere fejl korrekt og returnere passende fejl/succes beskeder | Ja | Ja |  |
| B4 | Administrationspanelet indeholder alle nødvendige funktioner og værktøjer til at administrere bøger, kunder, ordrer osv. Samt se hændelseslog for admin aktivitet | Ja | Ja |  |
| B5 | Applikationen kan modtage, behandle og opdatere ordrer korrekt | Ja | Ja |  |
| B6 | Applikationen giver mulighed for oprettelse, redigering og sletning af brugere med tilhørende informationer | Ja | Ja |  |
| B7 | Applikationen giver mulighed for oprettelse, redigering og sletning af udgivere med tilhørende informationer | Ja | Ja |  |
| B8 | Applikationen giver mulighed for oprettelse, redigering og sletning af formater med tilhørende informationer | Ja | Ja |  |
| B9 | Applikationen giver mulighed for oprettelse, redigering og sletning af tilstande med tilhørende informationer | Ja | Ja |  |
| B10 | Applikationen giver mulighed for oprettelse, redigering og sletning af genre med tilhørende informationer | Ja | Ja |  |
| B11 | Applikationen giver mulighed for oprettelse, redigering og sletning af sprog med tilhørende informationer | Ja | Ja |  |
| B12 | Applikationen giver mulighed for oprettelse, redigering og sletning af forfattere med tilhørende informationer | Ja | Ja |  |
| B13 | Applikationen giver mulighed for oprettelse, redigering og sletning af bøger med tilhørende informationer | Ja | Ja |  |
| B14 | Mulighed for at give brugere administrator adgang samt at fratage admin privileger igen | Ja | Ja |  |
| B15 | Backend håndterer fejl og sender relevante fejlbeskeder videre til frontend hvis muligt | Ja | Ja |  |
| B16 | Alle admin aktiviteter logges til hændelsesloggen og vises korrekt i admin panelet med brugernavn samt hvilken aktivitet der er foretaget | Ja | Ja |  |
| B17 | Korrekt håndtering af brugerens søgeforespørgsler og returnering af relevante resultater baseret på søgekriterier som titel, forfatter, kategori | Ja | Ja |  |
| B18 | Korrekt visning af bøger ved valg af filtrering samt valg af kombineret filtrering ved tjek i flere muligheder. | Ja | Ja |  |
| B19 | Der er begrænsning af adgang til beskyttede ressourcer og funktioner baseret på brugerens rolle samt inputvalidering for at forhindre SQL-injektioner, XSS og lignende former for angreb, alle forms er beskyttet med CSRF beskyttelse | Ja | Ja |  |
| B20 | Efter sletning af ethvert objekt som forfatter, sprog eller udgiver, skal det slettede objekt kunne gendannes fra administratorpanel. | Ja | Ja |  |
| B21 | Der bliver aktivt downloadet en korrekt Excel fil med liste over alle hændelser i systemet. | Ja | Ja |  |

# Sikkerhed

Sikkerhed udgør en fundamental del af enhver udviklingsproces, især når der er involveret brugerlogin med opbevaring af følsomme oplysninger. Det åbner op for en lang række muligheder for hackere at udnytte, og derfor er det af afgørende betydning at implementere effektive sikkerhedsforanstaltninger for at beskytte mod potentielle trusler. Webshoppen tager dette ansvar alvorligt og benytter en række værktøjer og teknologier til at opretholde en høj sikkerhed.

En central del af sikkerheden er adgangskontrol, som administreres gennem brug af forskellige roller og tilladelser. Dette sikrer, at hver bruger kun har adgang til de funktioner og ressourcer, der er relevante for deres rolle i systemet. Samtidig anvendes CSRF-tokens til at forhindre Cross-Site Request Forgery (CSRF), som er et angreb hvor uautoriserede handlinger kan udføres af en bruger, der er blevet narret til at udføre en handling på en anden hjemmeside, eksempelvis en falsk hjemmeside eller en forklædt url.

Passwords bliver sikret med bcrypt algoritmen, som tilføjer en unik 'salt' til hver adgangskode for at øge tilfældigheden og sikkerheden. Dette sikrer, at selv om to brugere har samme adgangskode, vil deres hashes - eller digitale signaturer - være forskellige. Denne metode hjælper med at beskytte passwords mod hacking.

Laravel’s indbyggede Middlewares giver mulighed for at filtrere og håndtere indgående anmodninger, hvor brugeren sendes hen ift. Rolle eller ip-adresse, eksempelvis hvis webshoppens administration ikke skal være tilgængelig via udenlandsk ip-adresse.  
På den måde kan der skræddersyes sikkerhedsregler alt efter specifikke krav og behov.

Et vigtigt sikkerhedsaspekt er beskyttelse mod SQL-injektioner, en almindelig form for angreb hvor hackere indsætter SQL kode igennem inputfelter, dette forhindres gennem en kombination af Laravel’s indbyggede SQL-Injektionsbeskyttelse og inputvalidering samt rense data (også kaldet at data-sanitizing) i Controllers inden data bruges til eksempelvis at oprette eller opdatere data på siden.

Ved go-live af webshoppen vil der skulle benyttes HTTPS (Hypertext Transfer Protocol Secure) som er en afgørende faktor for at overholde høj sikkerhedsstandard. HTTPS er en krypteret protokol som sikrer at kommunikationen mellem brugerens browser og web-serveren er krypteret og dermed beskyttet mod potentielle angreb eller aflytninger af data under kommunikation mellem browser og web-serveren.  
Ved at bruge HTTPS sikres følsomme oplysninger, såsom adgangskoder og betalingsoplysninger. HTTPS giver også brugerne tillid til, at webshoppen er autentisk og legitim, da det er forbundet med et SSL-certifikat, der bekræfter webstedets identitet, hvilket også bidrager til en god brugeroplevelse.

Projektet benytter en environment fil (.env), som gavner projektet af flere sikkerhedsmæssige årsager, her isoleres konfigurationsnøgler ved at adskille oplysninger fra kodebasen som blandt andet API nøgler, MYSQL-adgang og Stripe nøglen.  
Det reducerer fejlrisikoen for utilsigtet eksponering af oplysninger, .env filen er eksempelvis heller ikke en del af versionsstyringen, men skal opsætte på ny ved hver installation.

# Test

Webshoppen er opsat med en testproces for at sikre en høj kvalitet, sikkerhed og forhindrer nedbrud. Der er implementeret tre GitHub Action-workflows for at automatisere testprocessen og sikre at kodebasen forbliver stabil og pålidelig efter enhver ændring.

([Se bilag: Workflow Runs, s. 40](#Workflow_runs))

## Laravel Applikationstest Workflow

Dette workflow tester Laravel applikationen ved at bygge den og verificere at den bygges korrekt.   
Først installeres alle tredjeparts biblioteker ved hjælp af composer og NPM, Derefter oprettes der forbindelse til en SQLite test database, efterfølgende migreres der til databasen, alle PHPunit tests køres. Såfremt alt kører igennem med succes, meldes der gennemført workflow tilbage.

## Tredjeparts biblioteker sikkerhedstest

Her fokuseres der på at teste tredjeparts biblioteker for kendte sikkerhedsbrud eller forældede versioner.  
Kørslen starter med at scanne alle tredjeparts biblioteker for kendte sikkerhedsbrud, efterfølgende kontrolleres det om der i projektet benyttes forældede versioner samt sikrer at de versioner som benyttes, er sikkerhedsmæssigt opdateret.

## Larastan workflow

Larastan er en udvidelse af Phpstan, tilpasset Laravel miljøet, det er et PHP analyseværktøj som benyttes til at opdage fejl og potentielle problemer i Laravel koden og PHP-koden.

Udvidelsen er tilpasset til at forstå Laravel specifikke elementer som Eloquent ORM ‘en, migrationer og mange andre indbyggede funktioner.

Dette workflow installerer alle tredjeparts biblioteker, validerer PHP-koden for eventuelle syntaksfejl, efterfølgende køres der en dybdegående analyse af kodebasen for at identificere potentielle problemer eller brug af indbyggede forældede PHP-funktionaliteter, der tjekkers også for kendte sikkerheds fejl eller fejllogikker i koden.  
Sidst meldes der tilbage med en detaljeret rapport over eventuelle fundne problemer, hvilket gør det let at forbedre kvaliteten og forhindre fejl i koden.

Larastan har ikke en officiel website men er open source og kan findes på <https://github.com/larastan/larastan> - Larastan er baseret på PHPStan som kan tilgås via <https://phpstan.org/>

# Installationsvejledning

I guiden gennemgås hvordan lokalmiljøet opsættes på en Windows / Linux maskine, det er også muligt at springe denne del over og gå direkte til live versionen af webshoppen på url: <http://128.199.39.171/>

For at begynde med skal du sikre dig, at dit system opfylder de nødvendige krav:

* **Windows Subsystem for Linux (WSL2)** på Windows maskiner
* **Docker Desktop**: Installeret og kørende
* **Git**: Installeret til versionskontrol
* **Composer**: Til PHP afhængighedshåndtering
* **Node.js og npm**: Til frontend afhængighedshåndtering

Herefter følges guide trin for trin for opsætning af projekt på lokal maskine:

1. Åbn en terminal (Brug WSL2 terminalen på Windows).
2. Naviger til den mappe, hvor du ønsker projektet skal være.
3. Kør git clone kommando for at klone projektet ned i ønsket mappe:

~ git clone <https://github.com/MichaelAggerholm/Apprentice_Test>

1. Skift til projektets rodmappe:

~ cd Apprentice\_Test

1. Installer PHP-afhængigheder:

~ composer install

1. Installer Nods.js afhængigheder med NPM:

~ npm install

1. Opsætning af Laravel, kopier miljø fil med cp kommando:~

~ cp .env.example .env

1. Generer Laravel applikationsnøgle:

~ php artisan key:generate

1. Start docker containers i ”detached mode” med sail kommando:

~ ./vendor/bin/sail up -d

1. Kør migrationer for at opsætte databasestrukturen:

~ .vendor/bin/sail artisan migrate

1. Seed databasen med test data:

~ .vendor/bin/sail artisan db:seed

1. kør frontend miljø for at kunne tilgå webshoppen med følgende npm kommando:

~ npm run build

1. Tilgå nu webshoppen via computerens localhost adresse på følgende url:

<http://localhost>

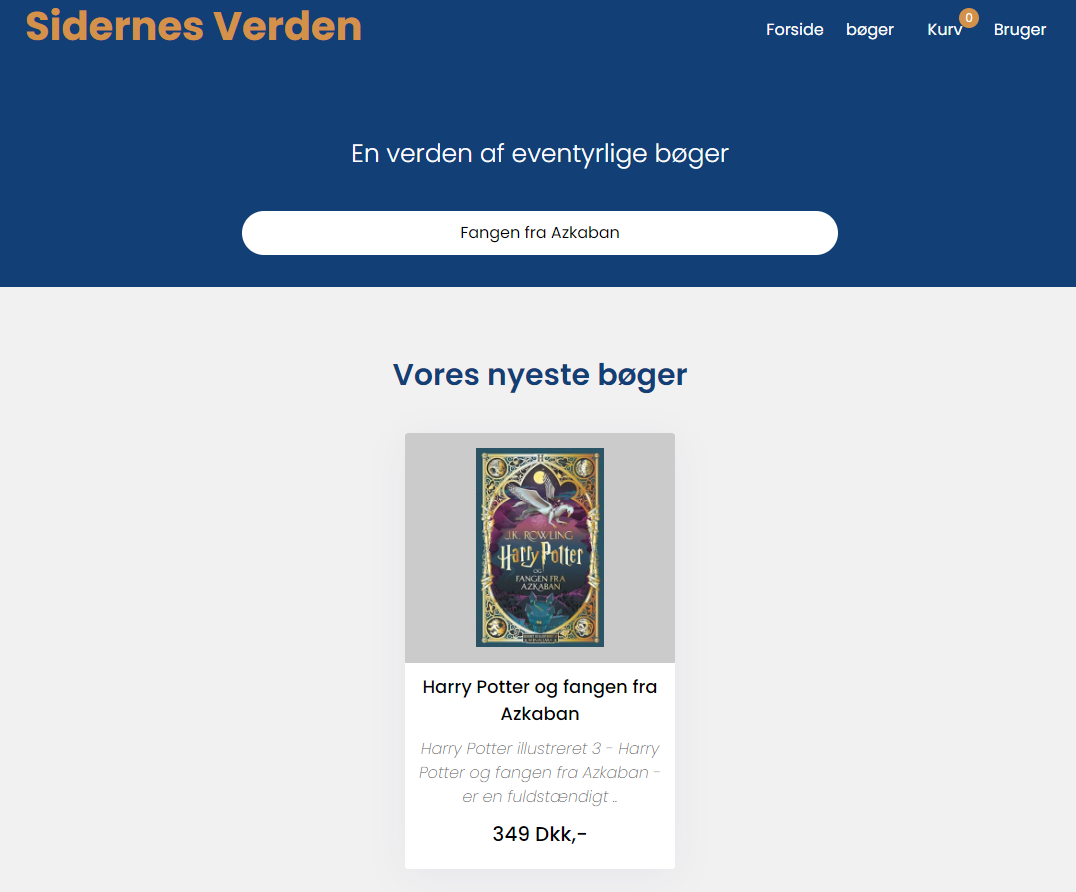
# Brugervejledning

I dette afsnit beskrives hvordan websitet benyttes og hvordan der navigeres i henholdsvis frontend for forbrugeren samt i administrationspanelet for medarbejdere.

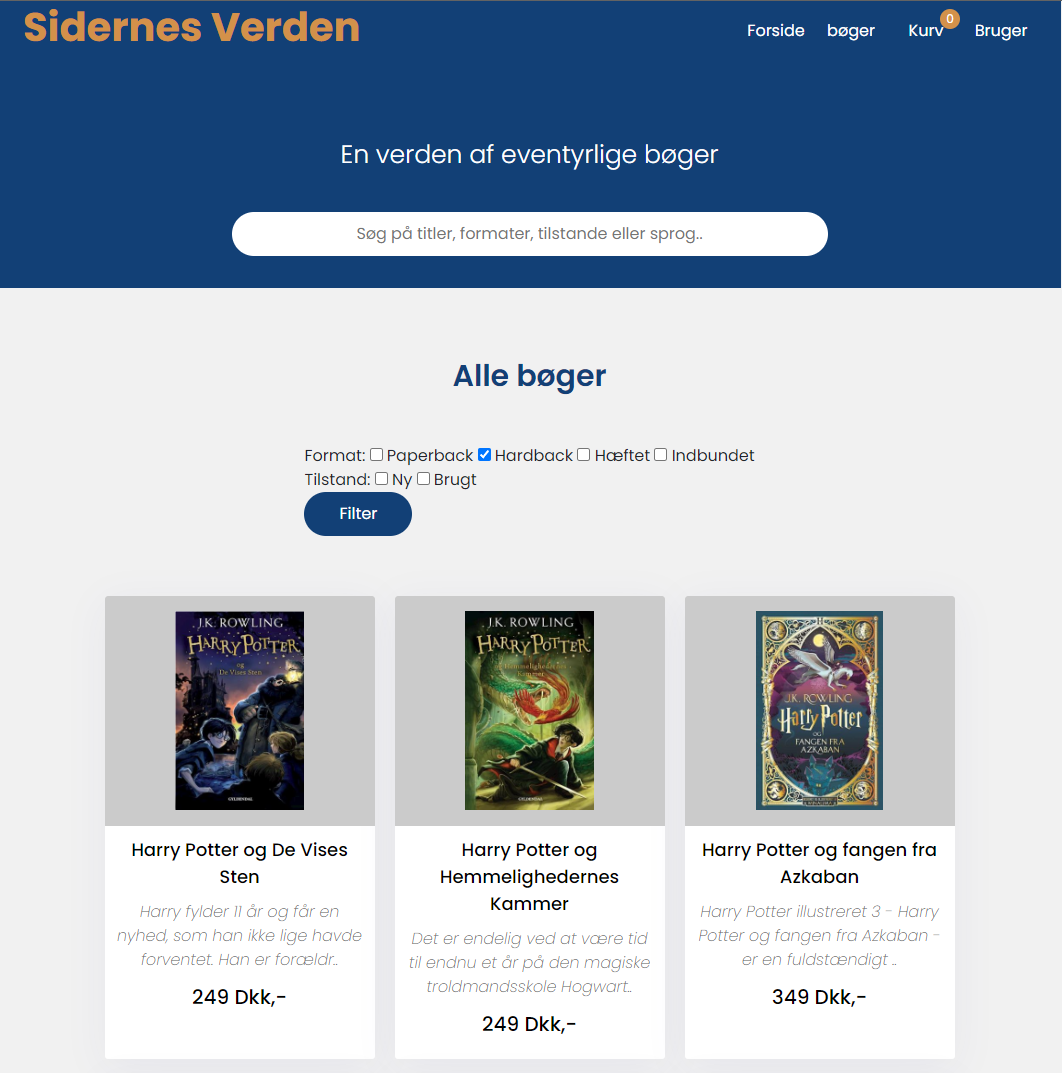
## Forbrugermanual

Forsiden byder på en oversigt af bøger samt en navigationsmenu og en søgefunktionalitet for hurtig at finde den ønskede bog.

Naviger til søgefeltet og skriv eksempelvis ”Fangen fra Azkaban”, afslut ved klik på Enter.



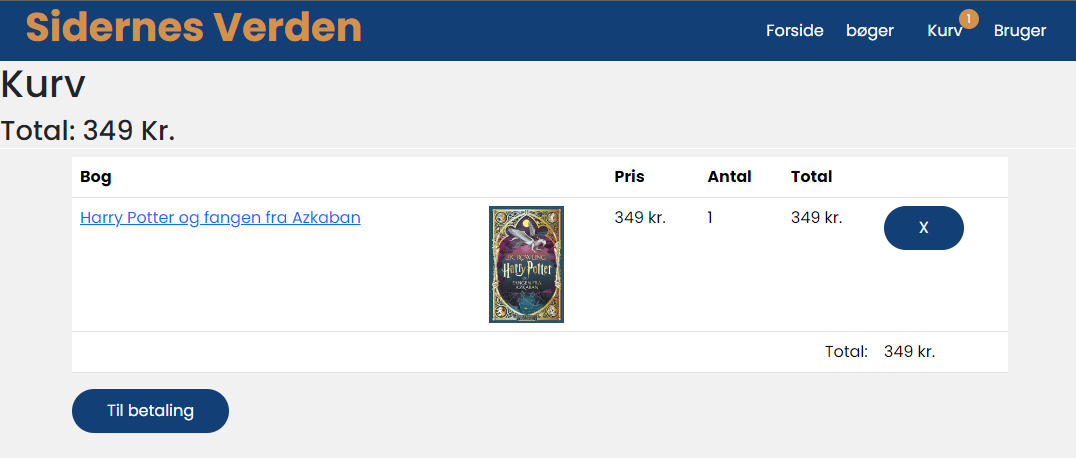
Det er også muligt at navigere til menupunktet ”Bøger”, her kan bøger filtreres på format eller tilstand, samt søges frem fra søgefeltet. Vælg ”Hardback” og derefter klik på den ønskede bog:



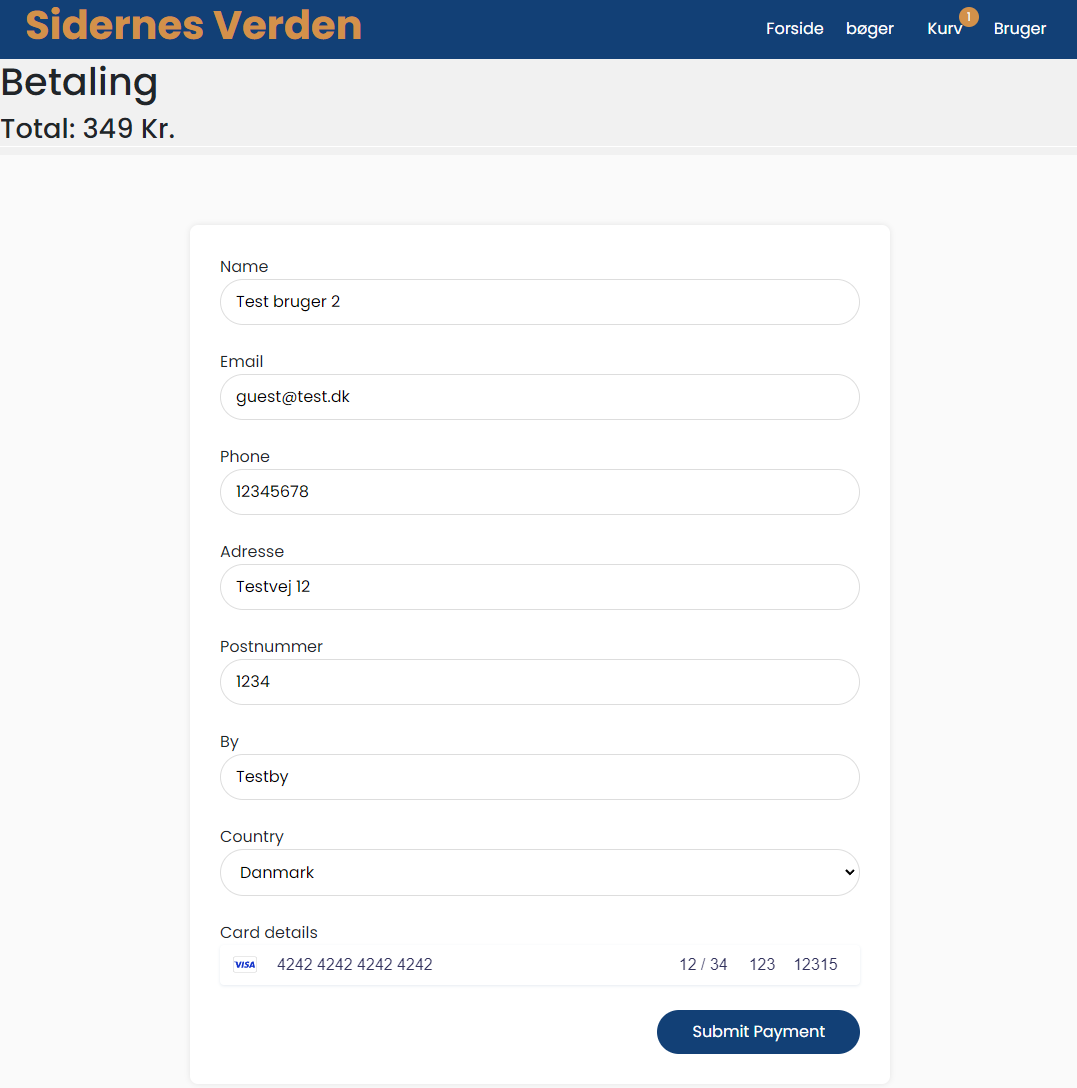
Ved klik på bog er det muligt at se detaljer om den specifikke bog, samt vælge antal og tilføje til kurv:



Ved tilføjelse til kurv, kommer en pop-up boks frem som giver mulighed for at fortsætte køb, eller gå til kurven, hvis man senere selv vil tilgå kurven er det muligt fra navigationen i øverste højre hjørne:



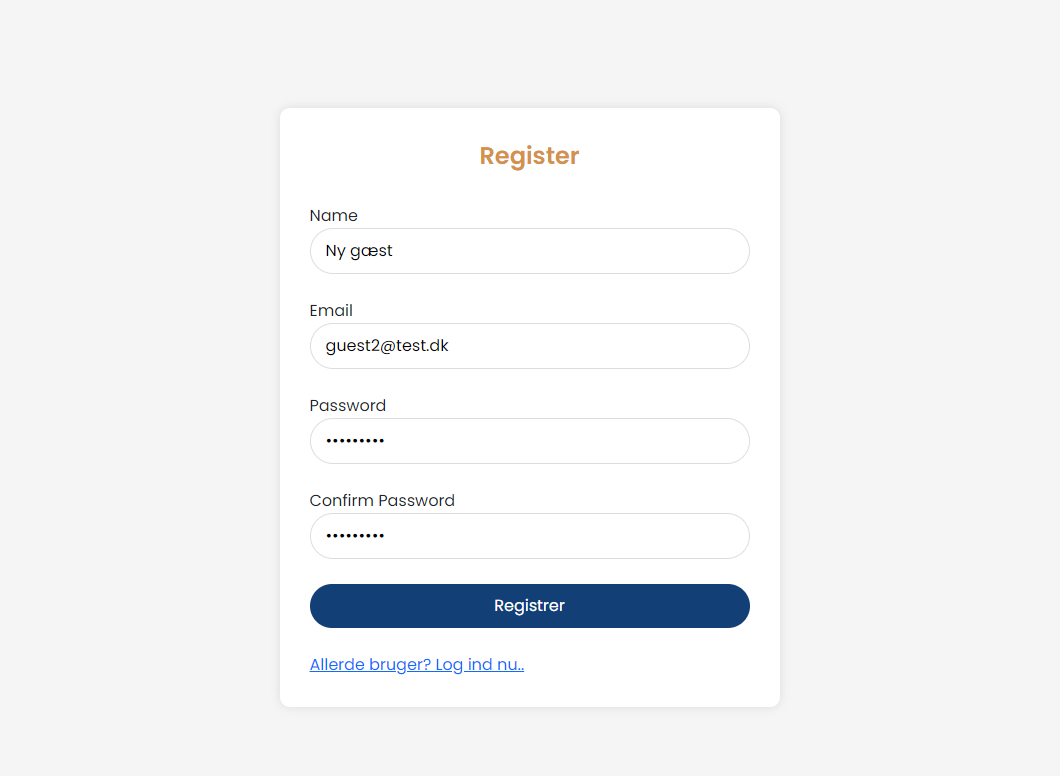
Ved klik på ”Til betaling”, føres kunden nu til betalingssiden hvor man kan udfylde oplysninger samt anvende test kortet ”4242 4242 4242 4242 – 12 / 34 - 12315” som vist på følgende billede, hvis man er logget ind på siden med en oprettet bruger, vil felterne delvist være auto-udfyldt ud fra brugeroplysninger:

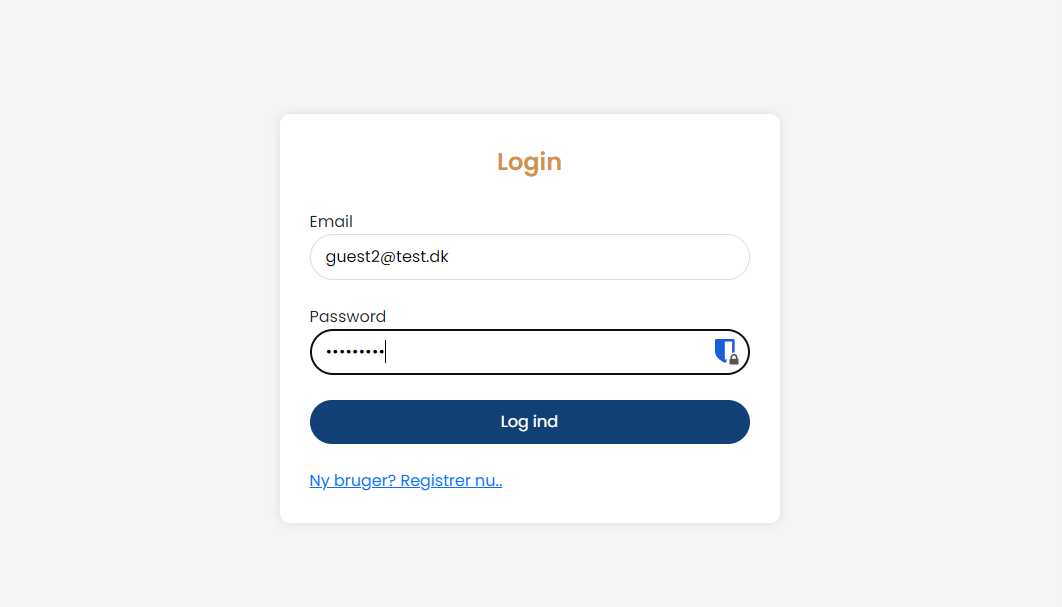


Ved gennemført betaling vil kunden nu modtage en ordrebekræftelses side med ordrenr.:

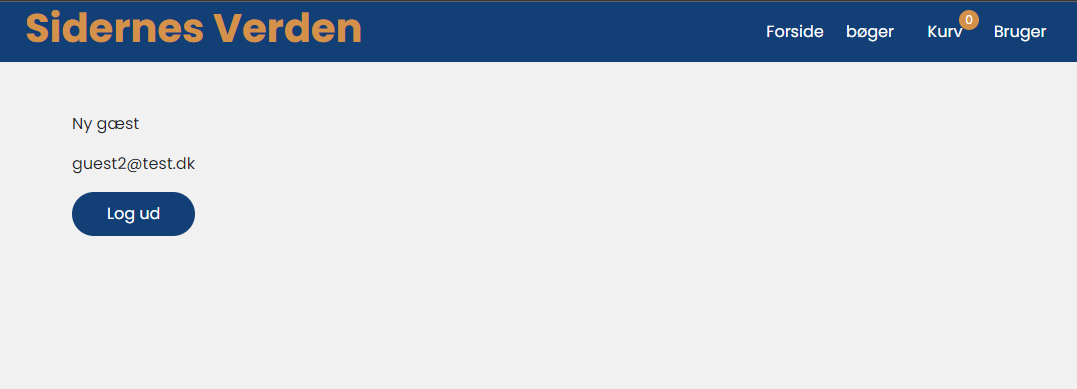


For oprettelse af bruger, klik på ”Bruger” i navigationen i øverste højre hjørne, herefter klik ”Ny bruger? Registrer nu.” og udfyld informationer, klik herefter registrer:



Efter registrering logges kunden automatisk ind på nyoprettede bruger, ellers er det muligt fra øverste højre hjørne ved klik på ”Bruger”:  


Efterfølgende kan man ved klik på ”Bruger” i øverste højre hjørne, logge ud eller se login oplysninger:



Det er blandt andet muligt at logge ind på følgende test bruger som forbruger af siden:

Brugernavn: ***guest@test.dk***

Adgangskode: ***12345***

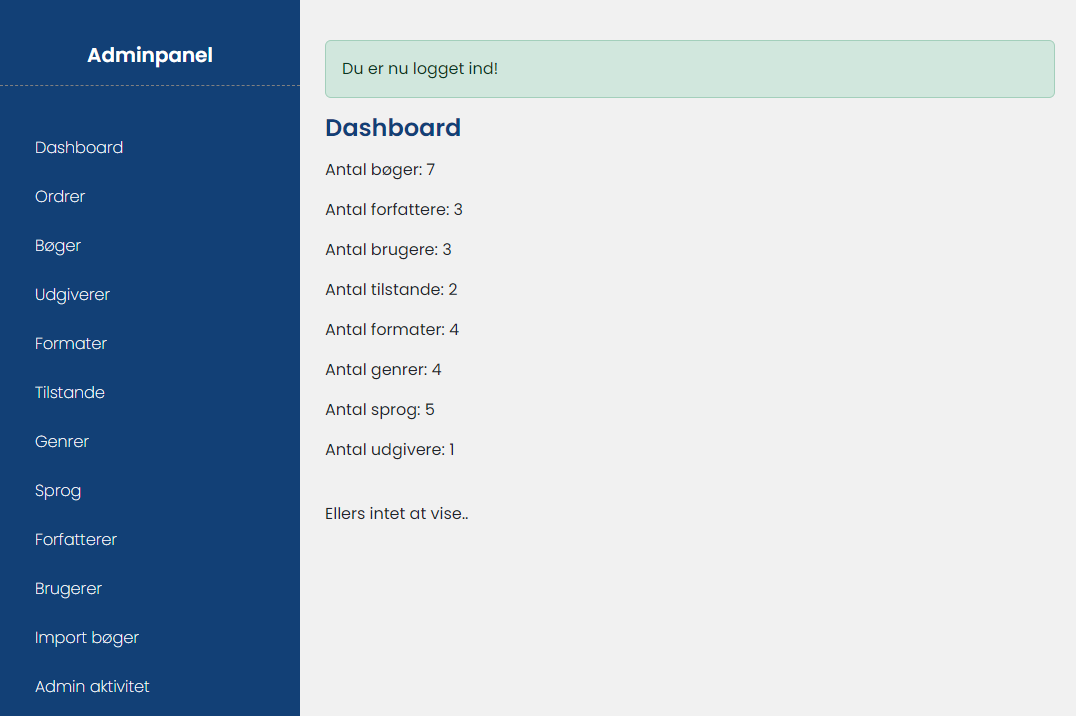
## Administrator manual

Logges der ind på en bruger med administrator adgange, som eksempelvis følgende test admin bruger:

Brugernavn: ***admin@test.dk***

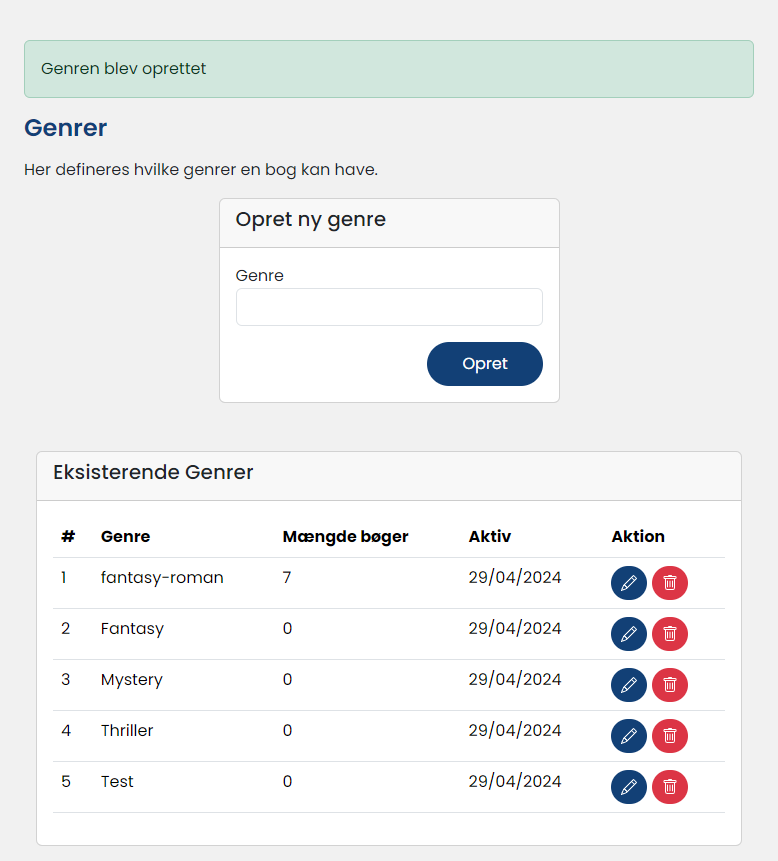
Adgangskode: ***12345***

Føres administratoren automatisk til det såkaldte Adminpanel, her vises som forside et dashboard med oversigt over antal af entiteter på siden:

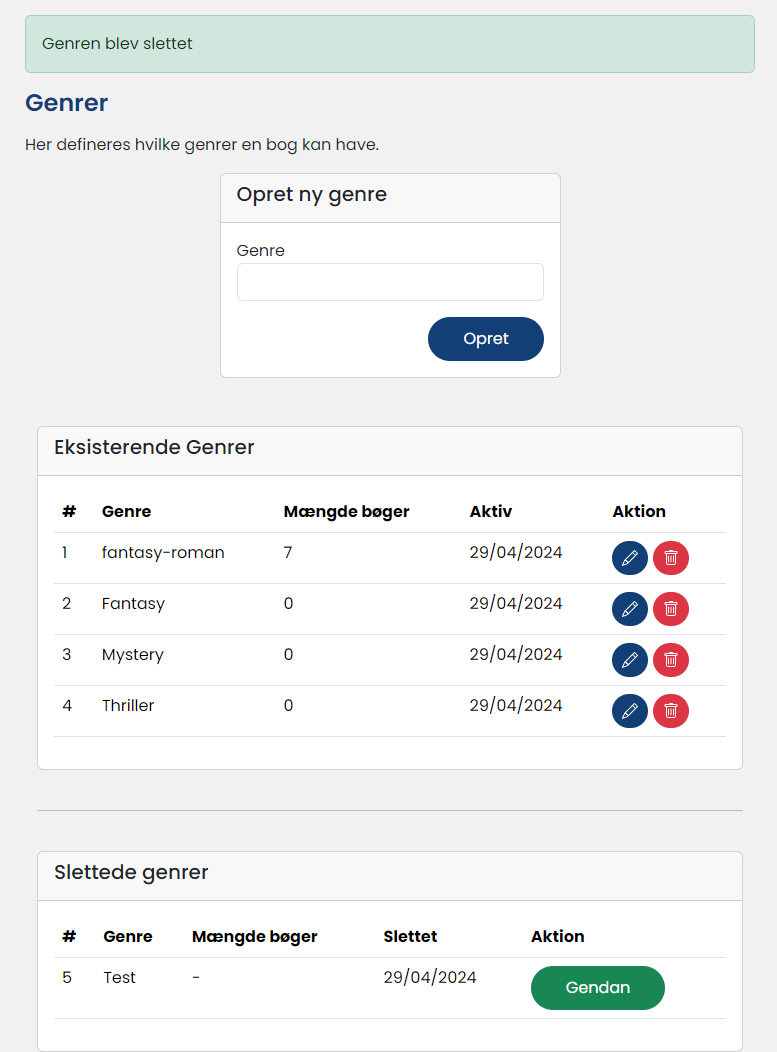


På siden er det muligt at få overblik, oprette, redigere eller slette samt gendanne alle entiteter på siden, i følgende eksempel vises det hvordan en ny genre bliver tilføjet til siden.

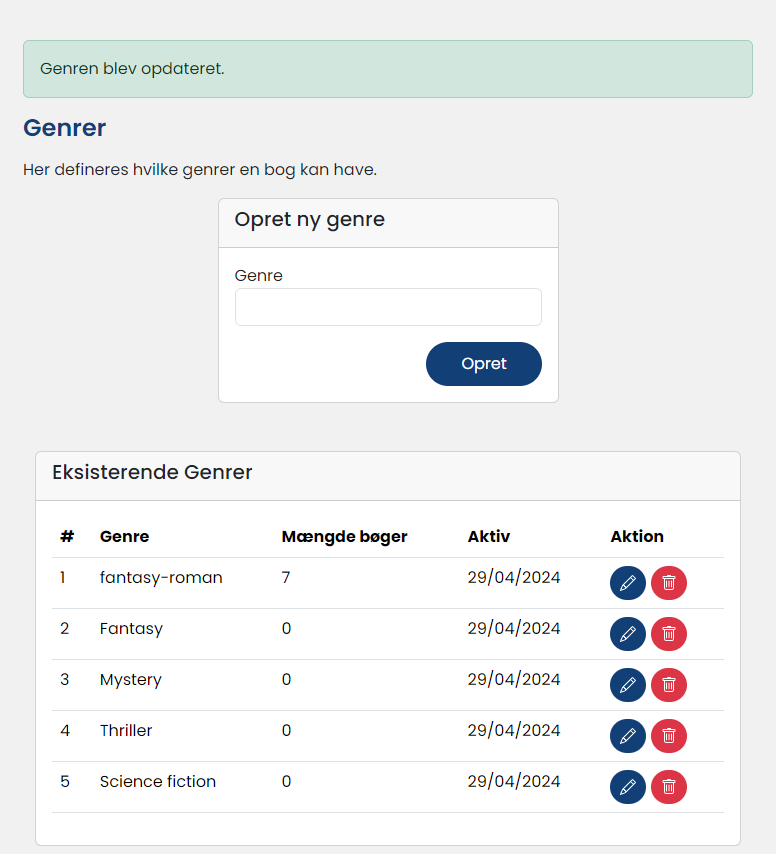
Klik på ”Genrer” i venstre navigationsmenu, her vises et opret felt samt alle eksisterende generer på siden, opret nu en ny genre kaldet ”Test”:

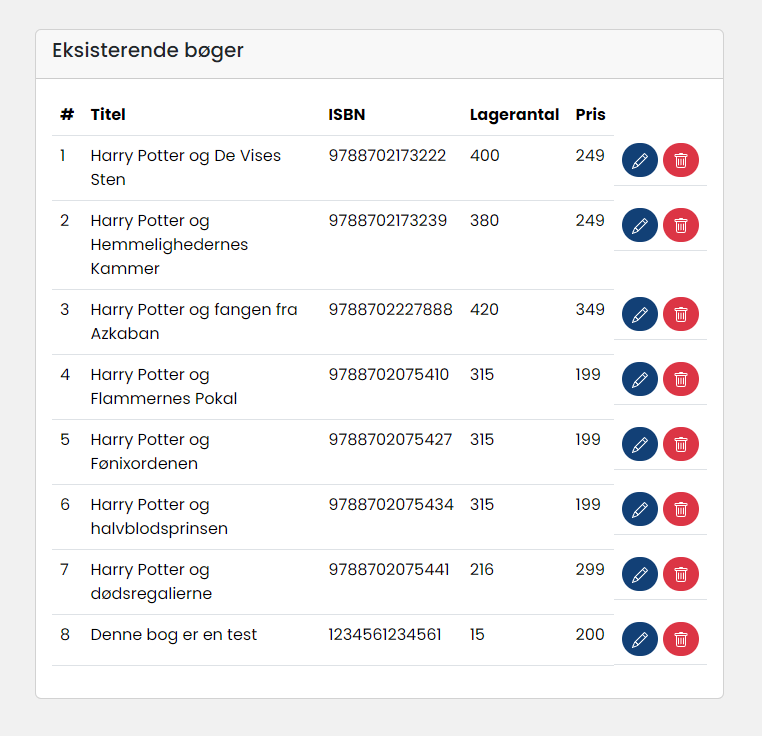


Slet herefter ”Test” ved at klikke på den røde skraldespand til højre for navnet, herefter fremgår slettede genrer i en ny sektion under de eksisterende genrer:

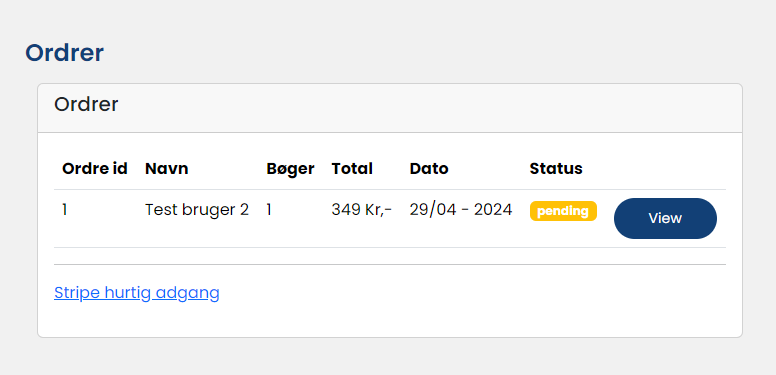


Gendan Test genren ved klik på Gendan, derefter rediger Test genre til et mere korrekt navn ved at klikke på den blå blyant til højre for genren, gem ved at klikke ”Gem Redigering”:

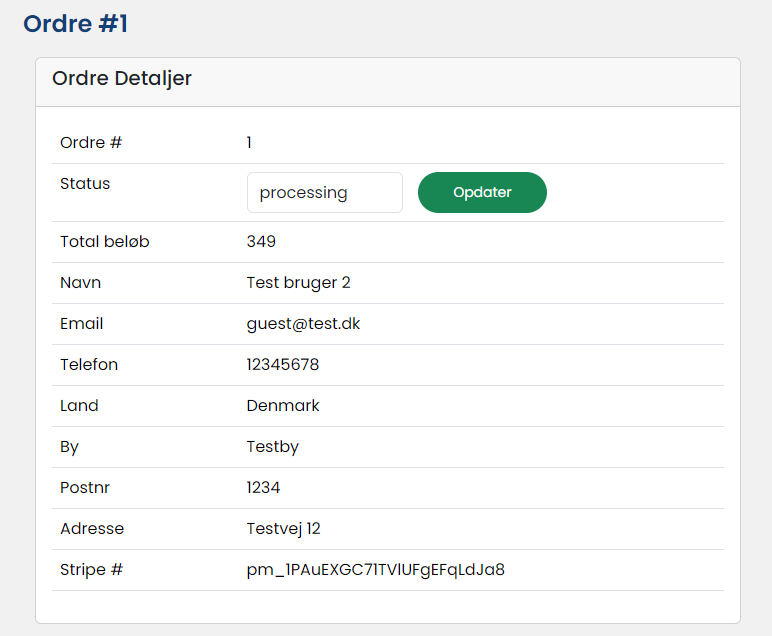


Lav nu en ny bog med denne genre, klik på menupunktet ”Bøger” og udfyld informationerne, der er blandt andet muligt at tilføje flere genrer samt forfattere til bogen, gem bogen og se at den nu ligger under eksisterende bøger:  


Naviger til menupunktet ”Ordrer” for at se ordrer samt ændre ordre status for en specifik ordre:



Klik på ”View” til højre for en specifik ordrer og ændre nu status til Processing, gem derefter ved at klikke opdater:



Klik på linket under ordrer oversigten for at gå til Stripe platformen, eller tilgå den via følgende url: <https://dashboard.stripe.com/test/developers>

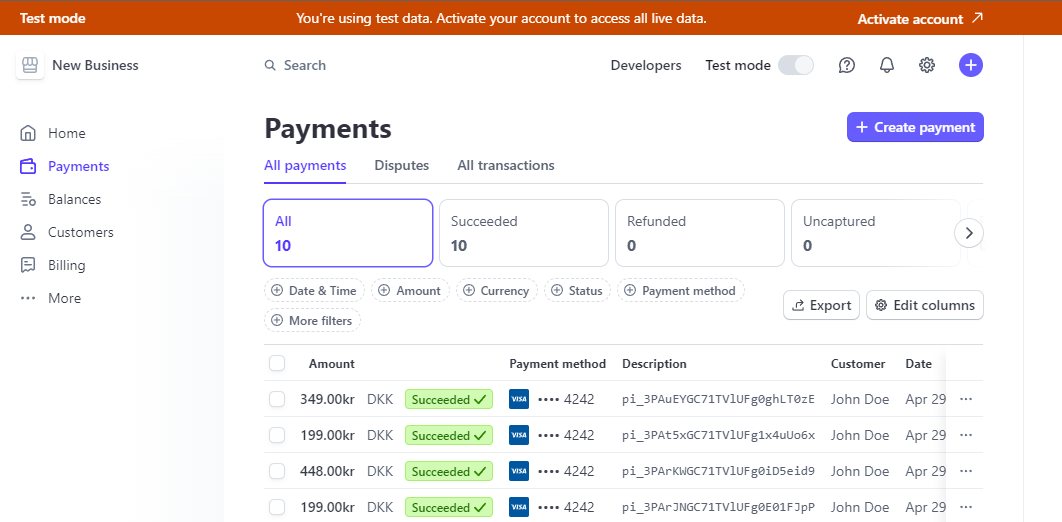
Log ind med følgende oplysninger:

Brugernavn: ***92aggerholm@gmail.com***

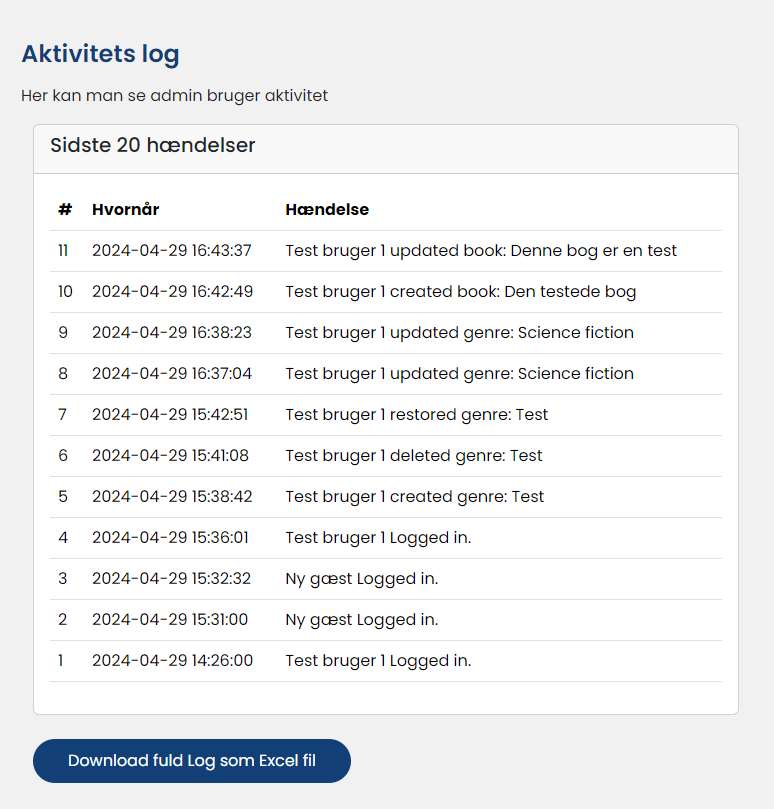
Adgangskode: ***!TestBruger1970***

Ovenstående er selvfølgelig en test bruger og udelukkende anvendt til dette projekt, efterfølgende vil adgangen blive lukket.

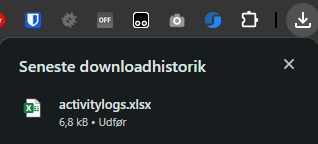
Naviger nu til Payments i venstre menu og se at sidste betaling er gået ind med succes:



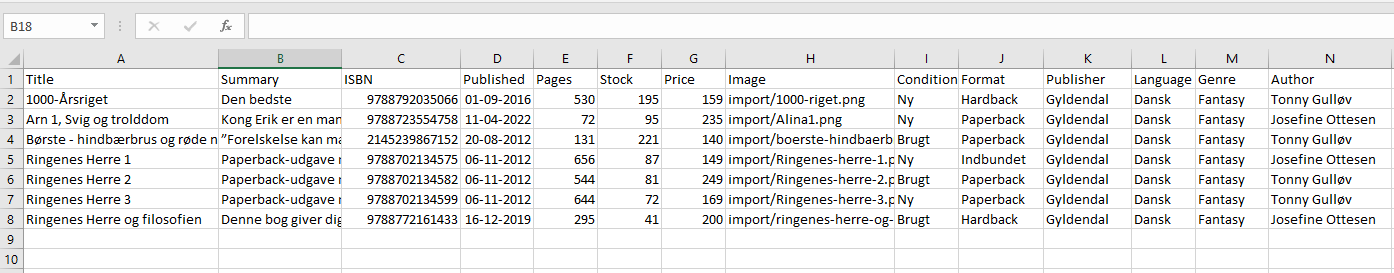
Nu hvor ordrer og entitets styring på siden er gennemgået, er det tid til at se på hændelseslog, denne del er yderst nødvendig når der er tale om en administration med flere adgange, da det giver mulighed for øjeblikkelig fejlretning samt fejlfinding ud fra sidste aktivitet. Naviger til menupunktet ”Admin Aktivitet” i venstre menu i Adminpanel, her vises de sidste 20 hændelser, samt hvem der har udført en aktivitet:



Ønsket en komplet liste af alle aktiviteter downloadet i Excel format, er det muligt ved klik på ”Download ful log som Excel fil” knappen som findes under listen af hændelser:



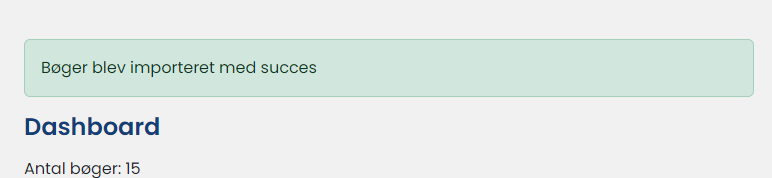
Sidste punkt i administrationen er import af bøger fra Excel ark, denne import er ikke så dynamisk, da test filen ligger statisk på serveren og ikke giver brugeren mulighed for at tilføje bøger til importen, et udkast af filen kan ses her:



Den kræver følgende kolonner og er ikke afhængig af en given rækkefølge:

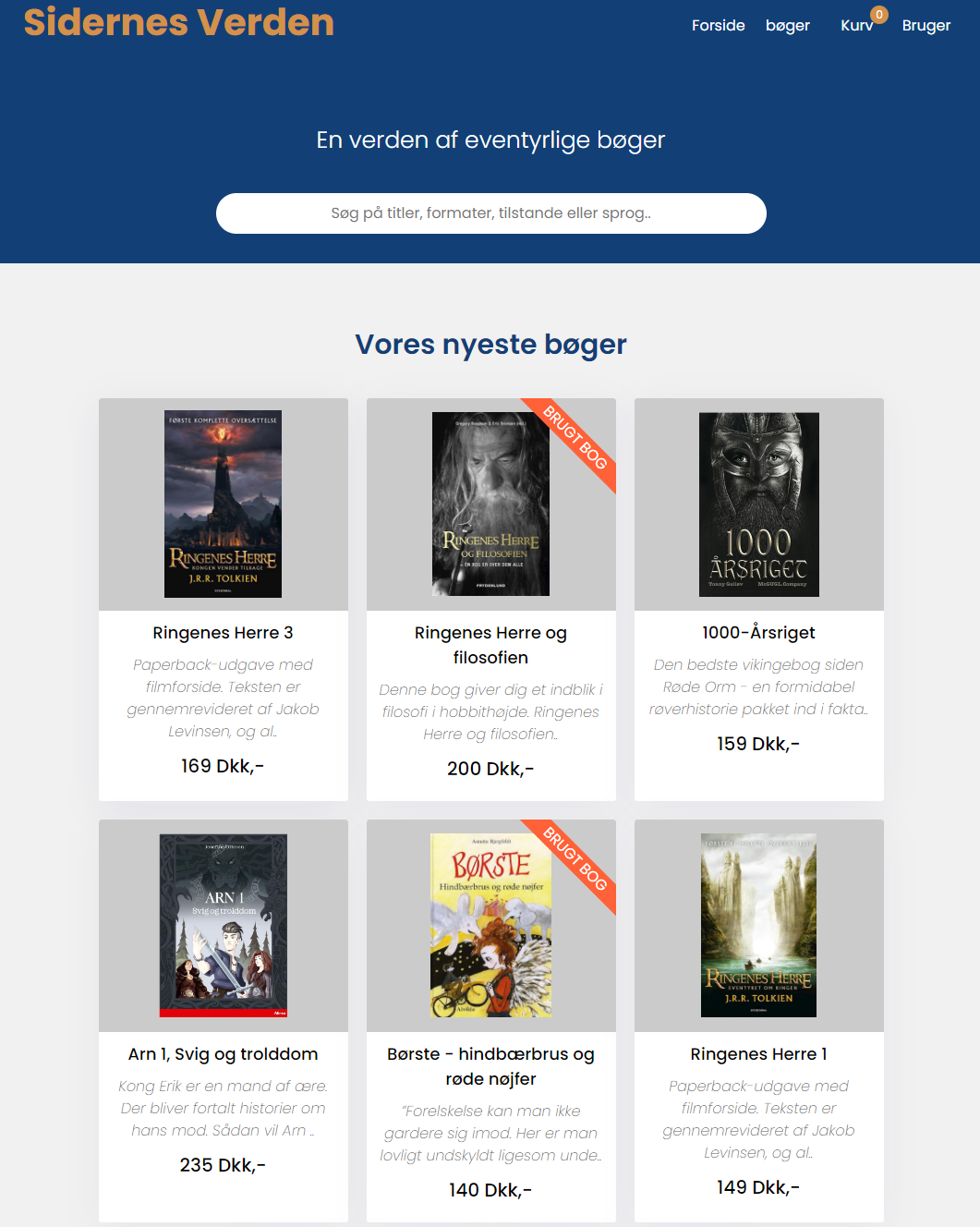
* Titel
* Summary
* ISBN
* Published
* Pages
* Stock
* Price
* Image
* Condition
* Format
* Publisher
* Genre
* Author

Såfremt alle data overholder kravene, vil bøgerne blive importeret ved klik på menupunktet ”Import bøger”:



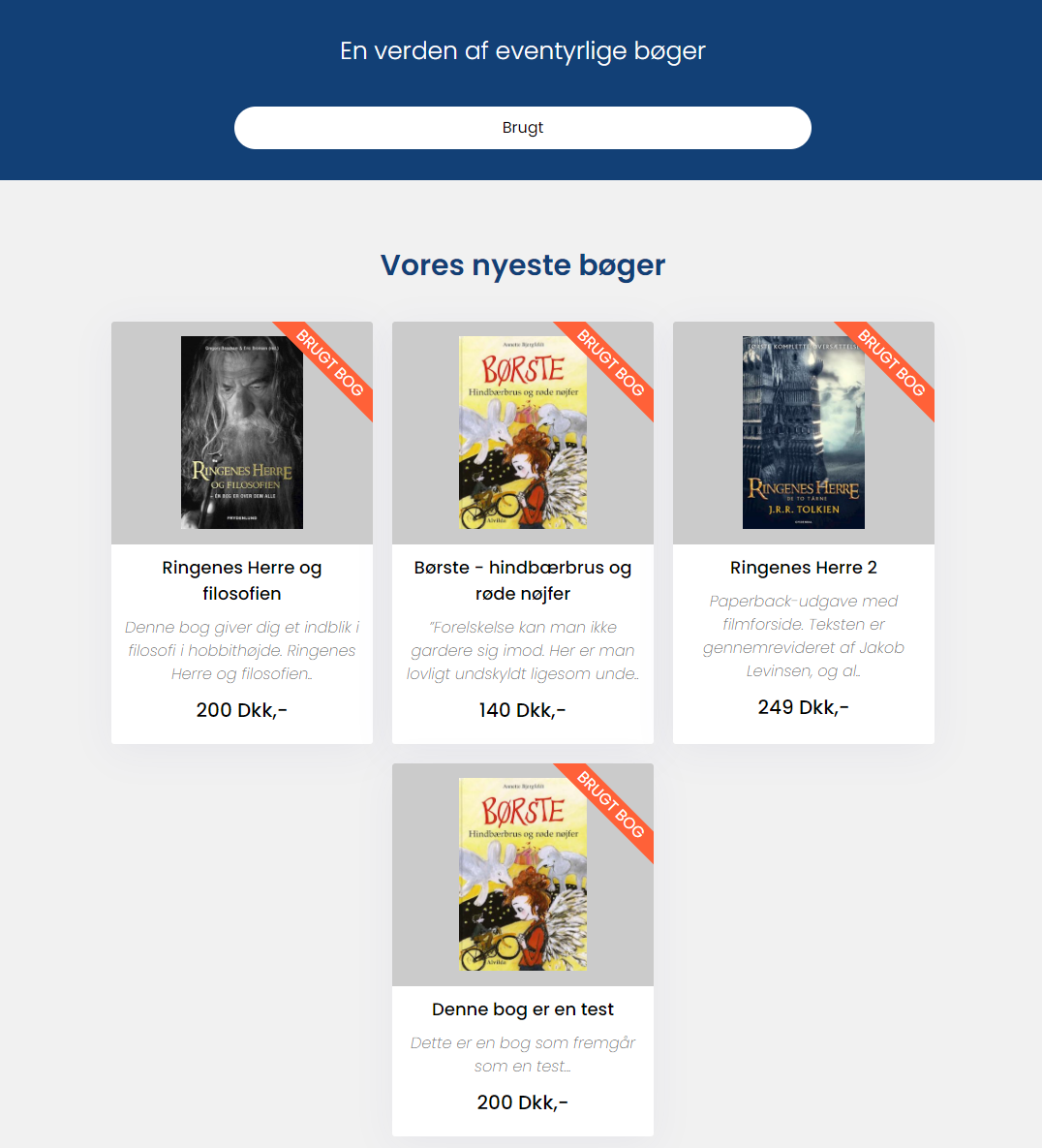
Efterfølgende er det muligt at gå til forsiden, enten ved at logge ud nederst til venstre, eller ved at gå til url: <http://128.199.39.171/>

Her ses at der nu er tilføjet flere bøger, samt at tilstanden ”Brugt” bog nu er kommet i spil:



Det er nu også muligt at vise funktionaliteten for at søge efter tilstande, både via søgefelt men også via filtrering på url: <http://128.199.39.171/all-books>

Dette gøres ved at skrive ”brugt” i søgefeltet, eller vælge brugt under filtrering:



Skulle det være nødvendigt undervejs, er der opsat nogle konfigurationsadresser som gør det muligt at teste websitet korrekt igennem, det er blandt andet følgende url:

Rydning af alle server caches:   
<http://128.199.39.171/clear-caches>

Link Storage, hvis manglende billeder efter redigering:   
<http://128.199.39.171/link-storage>

Migrering og Seeding af database på ny:   
<http://128.199.39.171/migrate-refresh-seed>

# Konklusion

Webshoppen er nu fuldt implementeret og fungerer som en integreret helhed bestående af en selvstændig backend, frontend og database. Dette design sikrer høj modularitet og skalerbarhed, hvilket betyder, at individuelle dele af projektet kan udvides eller tilpasses uden at påvirke de andre komponenter.

Administrationen tillader vedligeholdelse af alle entiteter. Det er muligt at søge efter bøger baseret på søgeord, genrer eller tilstande og at filtrere resultater på bogsiden. Brugere kan registrere sig, logge ind og gennemføre køb, mens administratorer kan tilgå et dedikeret Adminpanel, Her kan de overvåge Admin hændelser, se og tilpasse ordrestatusser. Admin kan desuden tilgå betalingsplatformen Stripe for detaljeret overvågning af transaktioner.

Import af bøger bør udvides til at kunne importere Udgivere, forfattere, genrer og lignende, der bør opsættes en import motor, hvor administratoren har mulighed for at indsætte fil i Adminpanel, samt vælge hvilken type import der er tale om, og mappe ”header” felterne direkte i admin Adminpanel, på denne måde giver det administratoren mere frihed til at berige siden, såfremt data er valid, med nuværende opsætning af fejlbeskeder ved uhåndterbar data, vil administratoren få klar besked om hvilke felt der giver problem, samt hvilken fejl der er opstået.

Til trods for disse avancerede funktioner er der visse mangler. Webshoppen er ikke fuldt responsiv på mobile enheder, og den understøtter ikke Cascade deleting med soft deletes. Disse funktioner er blevet nedprioriteret for at prioritere yderligere funktionalitet.

På den positive side er systemet designet med høj fejltolerance. Administratorer har mulighed for at slette og gendanne entiteter, hvilket forbedrer systemets robusthed. På frontend vises brugte bøger med et tydeligt skråbanner.

For at forbedre webshoppen yderligere og sikre en optimal brugeroplevelse er der nogle nøgleområder som bør forbedres i fremtiden, her er der blandt andet tale at implementere avanceret søgefunktionalitet i administrationen for at øje brugervenligheden og effektiviteten for administratorer, således de hurtigt kan lokalisere og redigere specifikke bøger. Ligeledes vil det gavne forbrugeren at få udvidet filtreringen af bøger til forfattere, udgivere, priser mm. På frontenden for at forstærke den samlede købsoplevelse for kunder.

Det nuværende design af webshoppen giver plads til forbedringer for at kunne tiltrække og fastholde kunder, der bør nok investeres i en professionel designer som kan levere et mere tiltalende og moderne layout.

Grundet projektets ridsramme har der heller ikke været fokuseret nok på den fordel som blade templating tilbyder med komponent baseret kodning og genanvendelse, her kunne en hel del af koden væres sparret ved at være defineret og kaldt som komponenter fremfor duplikeret kode.

# Bilag

Database diagram  
Kan også findes via GitHub url:  
<https://github.com/MichaelAggerholm/Apprentice_Test/blob/main/Assets/Documents/database_diagram.drawio.pdf>



Workflow runs  
Kan også findes via GitHub url:  
<https://github.com/MichaelAggerholm/Apprentice_Test/actions>



GitHub Repository  
Kan også tilgås direkte via url:  
<https://github.com/MichaelAggerholm/Apprentice_Test>

