Haganäsgymnasiet

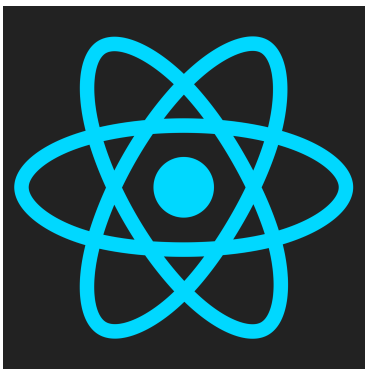
Teknikprogrammet

Gymnasiearbete 100p

Läsåret: 2023/2024

Digital försäljning

Utforska tekniken bakom modern E-handel



Balazs Hevesi

TE21

Annica Wassby

Sammanfattning

I detta gymnasiearbete presenteras processen för att skapa en e-handelssida, där jag har utforskat och använt mig av moderna webbteknologier inom både front-end-end och back-end utveckling. Arbetet inkluderar en översikt över relevanta JavaScript-ramverk, databasval mellan SQL och NoSQL, samt en diskussion kring de tekniska beslut som fattats under projektets gång. Slutresultatet är en fungerande webbshop, med insikter och reflektioner kring de utmaningar och lärdomar projektet medfört.

Abstract

This thesis presents the process of creating an e-commerce site, exploring and utilizing modern web technologies within both front-end-end and back-end development. The work includes an overview of relevant JavaScript frameworks, database choices between SQL and NoSQL, and a discussion on the technical decisions made throughout the project. The final outcome is a functioning ecommerce-store, with insights and reflections on the challenges and lessons learned from the project.

Innehållsförteckning

1. Inledning

1.1 Bakgrund

Idag finns det fler sätt än någonsin att bygga hemsidor. Utvecklare har en lång rad av teknologier och verktyg välja mellan. I online-communityn brukar det ofta skämtas om att det släpps nya verktyg dagligen. Jag vill utforska vad skillnaden mellan de olika teknologierna egentligen är.

Jag har hört mycket om hur viktigt det är att en webbsida laddar snabbt, eftersom det kan göra stor skillnad i hur många besökare som bestämmer sig för att göra något på sidan, som att köpa något eller registrera sig. Man kanske kan justera sitt val av front-end bibliotek efter vilken som presterar bäst, men hur stor skillnad kommer det faktiskt att göra?

Jag har som mål att arbeta med webbutveckling direkt efter gymnasiet, därför valde jag att göra mitt gymnasiearbete om just det. En e-handelssida är ett bra projekt att bygga för att förbättra mina kunskaper, då det finns många rörliga delar som måste jobba ihop.

1.2 Syfte

Syftet med detta arbete är att undersöka hur valet av front-end-bibliotek påverkar prestandan och konverteringsgraden på en e-handelssida. Genom att utforska populära front-end bibliotek som React, Vue, Qwik, Solid och Svelte, har jag som mål att förstå hur dessa kan påverka användarupplevelsen och utvecklar-upplevelsen. Detta kommer också att hjälpa mig att utveckla mina kunskaper inom webbutveckling genom att bygga en e-handelssida

1.3 Frågeställningar

1. Vilket front-end JavaScript ramverk bör man använda i 2023/2024 för att bygga en webbshop?
2. Hur bör man lagra datan för en webbshop?

2. Metod

2.1 Forskning

Idag finns nästan all information online, och det är väldigt enkelt att komma åt. I stort sett har alla populära programmeringsspråk, bibliotek, ramverk, och databassystem offentliga dokumentationer. Dokumentationerna varierar i kvalitet, oftast finns det en korrelation mellan hur populär en teknologi eller verktyg är och hur bra dokumentationen är.

Dokumentationer är perfekta för att ta reda på specifika saker om en teknologi, men de brukar vara mindre bra på att ge en översiktlig bild. Youtube visade sig vara ett väldigt användbart verktyg när det gällde att få en snabb och översiktlig bild av hur en teknologi fungerar och vad vilka huvudidéerna som teknologin är skapad omkring.

Jag en del erfarenhet av programmering från min studietid på gymnasiet. Det visade sig vara till stor hjälp för mig för att förstå alla koncept och idéer. Majoriteten av det jag har programmerat har varit i JavaScript och TypeScript. Jag har också gjort färdigt ”The Odin Project” som är en onlinekurs och studieplan som är riktat mot personer med utan tidigare erfarenhet av programmering och som har som mål att bli webprogrammerare.

2.2 Min webbshop

Jag har också byggt en webbshop för att direkt omsätta mina teoretiska kunskaper. Jag hade redan idéer på hur jag skulle göra och vilka teknologier jag skulle använda.

Min tidigare erfarenhet av det populära JavaScript biblioteket, React, kom ifrån ”The Odin Project”, men under utvecklingsprocessen märkte jag att den inte riktigt räckte till. När jag först började utveckla hemsidan så var min kunskap om databaser inte så stor. Jag förstod jag inte fördelen som SQL hade över NoSQL, jag valde att använda NoSQL, men senare in i arbetet märkte jag att SQL hade passat mycket bättre.

När jag utförde min praktik på Kaxig AB pekade Jordan (Lundgren, 2023) ut bristerna i hur jag hade tänkt att databasen skulle se ut. Då hade inte min databas databasintegritet, alltså att datan som var lagrad i den kunde ibland vara falsk. Att en databas inte har databasintegritet kan till exempel betyda att data A refererar till data B, men data B finns inte. Det fel kan

ställa till det enormt, speciellt ifall man jobbar med större mängder data. Sådana fel kan väldigt enkelt förebyggas genom en bra databasdesign.

Jag la märke till att när Jordan jobbade på sitt konsultuppdrag så använde de SQL, så efter min praktik så bestämde jag att fördjupa mig mer inom det.

Jag såg “Did I Pick The Right Database??? av "Theo - t3.gg" på Youtube där han diskuterade olika databassystem. Videor som den hjälpte mig att förstå spektrumet av tillgängliga alternativ och hur väl de passar för olika typer av projekt. Jag insåg att SQL skulle vara den rätta typen av databas för mitt projekt.

För att fördjupa mig inom SQL så tog jag först en online-kurs av Maximilian Schwarzmüller på Udemy. Den gick igenom syntaxen vad SQL är, hur man sätter upp en SQL server och syntaxen som SQL använder, men den berörde inte databasdesign särskilt mycket. För att förstå mig på det lite mer så kollade jag på Caleb Currys YouTube video om databasdesign (”Database Design All-in-One Tutorial”).

3. Resultat

3.1 Hastighet och konverteringsgrad

Hastigheten på en webbplats refererar till hur snabbt sidan laddas och blir interaktiv. Detta inkluderar alltså tiden det tar att ladda allt innehåll på sidan, som text, bilder, videor och andra element.

Konverteringsgraden refererar till den procentandel av besökare som utför en önskad åtgärd på en webbplats, till exempel att göra ett köp eller registrera sig för ett nyhetsbrev.

Studie efter studie visar att det finns en direkt korrelation mellan hastighet och konverteringsgrad. Till exempel en studie av Google från 2016 (Google, 2016) visade att ju längre det tar för en sida att ladda, desto högre är sannolikheten att besökaren kommer att lämna sidan innan den ens har laddat klart. Studien visade att 53% av mobilanvändarna lämnar en sida om det tar längre än 3 sekunder att ladda. Detta har en direkt påverkan på konverteringsgraden eftersom besökare som lämnar sidan innan den har laddat klart inte kommer att kunna göra några köp. Walmart hittade att för varje sekund som hemsidan laddade snabbare så ökade konverteringsgraden upp till 2 procent (Green, 2016).

Detta visar att det är väldigt viktigt för webbplatsägare och utvecklaren att optimera sidans hastighet för att maximera konverteringsgraden. Dessa optimeringar kan inkludera saker som att göra filer mindre, göra bilder och videor mer effektiva, eller se till att bara det som besökaren faktiskt ser laddas.

JavaScript-bibliotek är samlingar av funktioner och objekt som används för att underlätta utvecklingen av webbapplikationer. Populära JavaScript-bibliotek som React, Vue, och Angular har blivit standardverktyg för många webbutvecklare på grund av deras förmåga att hjälpa till att organisera och strukturera kod, samt hantera komplexa användargränssnitt och interaktioner. Många jobbannonser kräver dessutom att sökaren har erfarenhet av de. (Toal, u.d.)

Samtidigt som dessa bibliotek kan göra det enklare att bygga robusta och interaktiva webbplatser, kan de också ha en påverkan på sidans hastighet. Många JavaScript-bibliotek är ganska stora och kan öka mängden data som måste laddas ner när en användare besöker sidan. Det är därför viktigt att noggrant överväga valet av JavaScript-bibliotek och hur de används på en webbplats. Att använda mindre och mer effektiva bibliotek, ladda ner biblioteket synkront eller endast ladda ner de delar av biblioteket som faktiskt används, kan hjälpa till att minimera påverkan på sidans hastighet. Dessutom kan verktyg som ”tree shaking” och ”code splitting” hjälpa till att minska den slutliga storleken på JavaScript-filerna som skickas till användarens webbläsare. (MDN, 2024)

3.2 Dagens JavaScript-ekosystem

Inom programmering så används termen "ekosystem" till att referera till samlingen programvaruverktyg, bibliotek, ramverk och andra resurser som är sammanlänkade runt ett specifikt programmeringsspråk eller teknologi.

JavaScript har kommit en lång väg sedan den skapades 1992. Det har utvecklats från ett enkelt skript-språk för att manipulera webbsidor till att bli ett av de mest populära programmeringsspråken i världen. JavaScript definieras av ECMA Script standarden.

JavaScript är ett interpreterat programmeringsspråk, vilket betyder att koden exekveras rad för rad av en ”interpreter” i webbläsaren eller i en runtime-miljö, snarare än att kompileras till maskinkod innan den körs. Detta betyder att JavaScript-kod kan fungera olika beroende på vilken webbläsare som kör den, och vilken version användaren använder. På grund av att det

aldrig kompileras till maskinkod så betyder det att när JavaScript inkluderas på en webbplats så skickas källkoden till användaren.

ECMA Script har genomgått flera revisioner sedan dess ursprungliga publicering 1994. Till exempel, ES6 (också känd som ES2015) introducerade funktioner som klasser, moduler, och pilfunktioner, medan senare versioner har lagt till funktionalitet som async/await.

En annan viktig utveckling i JavaScript-ekosystemet har varit utvecklingen av TypeScript. TypeScript är en överbyggnad till JavaScript som lägger till statisk typning till språket. ”Statisk typning”, eller ”hård typning” betyder att variabler och funktioner har en definierad struktur som resten av programmet kan anta.

TypeScript är ett så kallat ”transpilerat” programmeringsspråk. En transpilering syftar på processen av att omvandla ett programmeringsspråk till ett annat för att kunna köras. TypeScript transpileras till JavaScript.

TypeScript möjliggör fångandet av typ-relaterade fel vid kompileringstid snarare än vid körningstid, vilket innebär att det hjälper till med att hitta och åtgärda fel tidigare i utvecklingsprocessen. TypeScript har blivit mycket populärt bland utvecklare, särskilt de som bygger stora och komplexa applikationer, eftersom det kan bidra till att göra koden mer robust och enklare att underhålla.

Node.js är en annan viktig del av dagens JavaScript-ekosystem. Det är en runtime-miljö som gör det möjligt att köra JavaScript på servern, vilket gör JavaScript till ett fullständigt full-stack språk. Node.js har också lett till skapandet av NPM (node package manager), som idag är världens största programvarubibliotek och gör det enkelt att dela och använda kod från andra utvecklare.

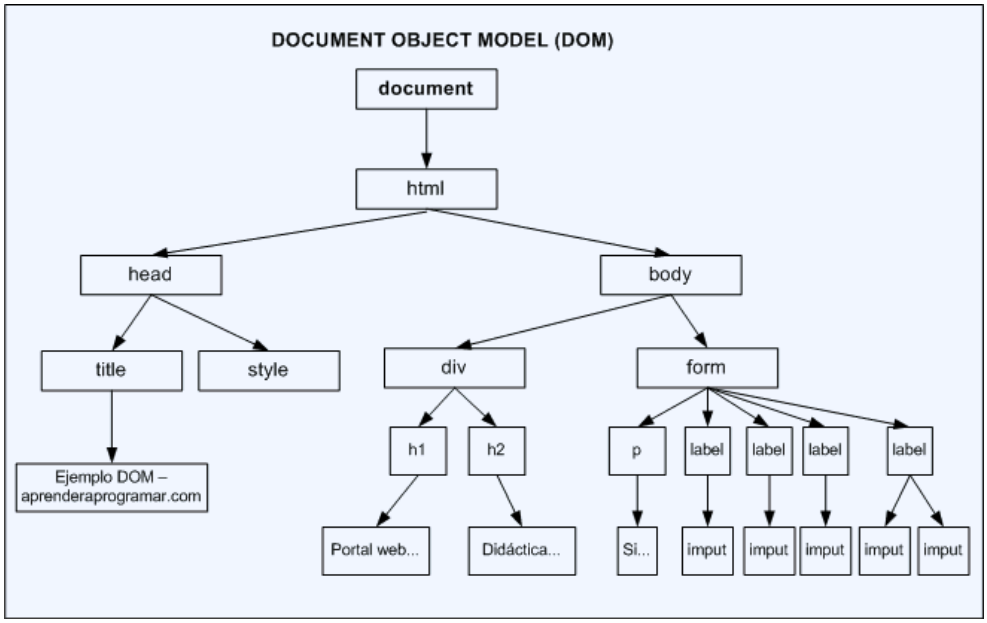
Konceptet med JavaScript bundling har blivit ett måste för moderna webb-appar. Verktyg som Webpack gjorde det möjligt att kombinera flera JavaScript-filer och moduler till en enda fil, vilket minskade antalet HTTP-förfrågningar som behövdes för att ladda en webbsida och därmed förbättrade sidans laddningstid och prestanda. (Carnes, 2018)

3.3 JavaScript front-end-end ramverk och bibliotek

Webbsidor och olika typer av användargränssnitt kan representeras med det så kallade “document object model”, som brukar förkortas med akronymet ”DOM”. Det representerar

strukturen på dokumentet som ett träd av objekt. Varje objekt motsvarar en del av dokumentet, som till exempel ett element (även kallat node) eller ett attribut. DOM ger ett sätt för program att manipulera strukturen, stilen och innehållet i webbdokument. Det kan till exempel användas för att lägga till, ta bort eller ändra element och attribut i dokumentet.

Front-end-end utvecklare brukar oftast vända sig till bibliotek eller ramverk för att underlätta manipulationen av DOM trädet.



Bilden visar en visualisering av ett DOM-träd (Krall, u.d.)

3.3.1 Ramverk och bibliotek

När man pratar om utvecklingen av mjukvara stöter man ofta på begreppen “ramverk” och “bibliotek”. Både bibliotek och ramverk finns för att underlätta utvecklandet av mjukvaran. Trots att termerna ofta används omväxlande, så finns det viktiga skillnader mellan dem.

Ett ramverk är en samling av kod som utgör en mall för hur en applikation ska struktureras och fungera. Ramverket bestämmer “reglerna” för applikationen, och utvecklaren fyller i de specifika detaljerna. Exempel på populära JavaScript-ramverk inkluderar Angular, Next, Nuxt.

Den huvudsakliga skillnaden mellan ett bibliotek och ett ramverk är kontrollen som utvecklaren har. Med ett bibliotek har utvecklaren mer kontroll och använder bibliotekets funktioner när och hur de vill. Med ett ramverk är det ramverket som har kontrollen och

kallar på utvecklarens kod vid specifika tidpunkter. En annan viktig skillnad är användningen. Ett bibliotek används för specifika uppgifter inom en större applikation, medan ett ramverk används som grunden för hela applikationen. (Hansa, 2022)

3.3.2 React

React.js, är ett öppen källkods JavaScript-bibliotek som används för att bygga användargränssnitt. Det utvecklades och underhålls av Facebook (numera känt som Meta) tillsammans med en community av enskilda utvecklare och företag. React är idag ett av de mest populära och använda biblioteken för att bygga webbapplikationer och har en stor och aktiv community som bidrar med verktyg, bibliotek och komponenter.

React bygger på konceptet med komponenter. Komponenter är återanvändbara och oberoende bitar av kod som representerar en del av användargränssnittet, t.ex. en header, eller footer. Varje komponent har sitt eget tillstånd och egenskaper, de kan även innehålla andra komponenter. Detta gör det möjligt att bygga komplexa användargränssnitt genom att kombinera enkla komponenter. I koden så kan komponenter skrivas med klasser, så kallade “class-based components”, men idag så är "functional components” föredragna, då de är enklare att jobba med.

React använder en virtuell DOM (Document Object Model) för att optimera prestandan. Den virtuella DOM:en är en kopia av den riktiga DOM:en som React använder för att hålla reda på ändringar som behöver göras. När en ändring görs i en komponent, uppdaterar React den virtuella DOM:en först och jämför sedan den nya virtuella DOM:en med den gamla för att avgöra vilka ändringar som behöver göras i den riktiga DOM:en. Detta minskar antalet ändringar som behöver göras i den riktiga DOM:en och ökar därmed prestandan.

Den stora fördelen med React jämfört med andra JavaScript-bibliotek är dess stora community, ifall man stöter på en bugg så finns det en hög probabilitet på att någon annan också har haft samma bug, därmed blir det enklare att de-bugga koden. Det är också relativt enkelt att lära sig, särskilt om man redan är bekant med JavaScript.

En annan stor fördel med React är att man kan använda det för att bygga mobil-appar också, inte bara hemsidor. Det finns något som kallas ”React Native” som är väldigt likt det vanliga React, men det är gjort för att bygga mobil-appar istället. Så, när man har lärt sig React, kan

man ganska enkelt lära sig React Native och börja bygga appar utan att behöva byta till ett helt nytt programmeringsspråk som till exempel Swift eller Kotlin.

Nackdelar med React inkluderar dess storlek (ca 100kb), det kan vara ganska stort jämfört med andra bibliotek. Det finns bibliotek som Preact som har som mål att vara ett mer effektivt alternativ till React. Preact är bara 3,5 kb och erbjuder en väldigt liknande utvecklar-upplevelse till React. Nackdelen med Preact är dock att communityt är mycket mindre och därmed blir det svårare att de-bugga koden, dessutom så finns det många React-tillägg som inte stödjer Preact. (Preact, 2024)

Dessutom är React bara ett bibliotek för att bygga användargränssnitt och inte ett fullständigt ramverk som till exempel. Angular. Dock så finns det Next.js som bygger på React och har mer fullständiga lösningar. (Meta Open Source, 2024) (Vercel, 2024)

3.3.2.1 Komponent livscykeln

Komponent Livscykeln någonting som inte är helt unikt för just React, men är ändå en av de grundläggande koncepten man behöver förstå om man vill använda React.

Det finns tre huvud-faser i en komponents livscykel:

1. **Mounting**: Detta är fasen när komponenten skapas och läggs till i DOM. De metoder som ingår i denna fas är `constructor()`, `static getDerivedStateFromProps()`, `render()` och `componentDidMount()`.

2. **Uppdatering**: Detta är fasen när komponenten uppdateras, antingen på grund av ändringar i dess state eller props, man kan också kalla det för rerender. De metoder som ingår i denna fas är `static getDerivedStateFromProps()`, `shouldComponentUpdate()`, `render()`, `getSnapshotBeforeUpdate()` och `componentDidUpdate()`.

3. **Unmounting**: Detta är fasen när komponenten tas bort från DOM. Den metod som ingår i denna fas är `componentWillUnmount()`.

Om man t.ex. vill göra en API anrop så måste man veta vart man ska lägga den, antagligen vill man inte göra API anropet när komponenten tas bort (unmounting), och antagligen vill man inte heller göra det varje gång komponenten uppdateras. (The Odin Project, 2024)

3.3.2.2 JSX

JSX, står för JavaScript XML, det är en syntaxförlängning för JavaScript som liknar XML, som HTML är byggt på. Det används tillsammans med React, men är inte begränsat till React, för att beskriva hur användargränssnittet ska se ut. JSX gör det möjligt att skriva HTML-element och komponenter i JavaScript-kod, vilket gör koden lättläst.

JSX syntaxen kan man säga är blandning mellan HTML och JavaScript. Till exempel kan man använda JavaScript-uttryck inuti HTML-taggar genom att sätta dem inom klamrar, som i följande exempel: <h1>Hello, {name}</h1>.

JSX behövs så kallat transpileras till vanlig JavaScript för att webbläsaren ska kunna tolka det. Transpilering är processen att konvertera källkod skriven i ett programmeringsspråk till en annan källkod i ett annat programmeringsspråk. Det är en typ av kompilering, men istället för att omvandla källkod till maskinkod (som en traditionell kompilator gör), omvandlar en transpiler källkod till källkod i ett annat programmeringsspråk eller en annan version av samma språk.

Till exempel, Babel är en populär transpiler som omvandlar modern JavaScript (ES6 och senare) till äldre versioner av JavaScript (ES5) som kan köras i äldre webbläsare som inte stöder de nyaste funktionerna i språket. Detta gör det möjligt för utvecklare att skriva kod med de senaste funktionerna och syntaxen, men fortfarande stödja äldre webbläsare.

Här är ett exempel på ett enkelt React-komponent innan det transpileras.

```
function App (){
return (<h1 className="hej">tjena</h1>)}
```

Så här kan det se ut efter att det har transpilerats till vanligt javascript. JSX:et omvandlas alltså till funktioner som hanteras av React och i sin tur renderas ut som DOM nodes så att webbläsaren ska kunna förstå.

```
import { createElement } from "react";
function App() {
  return createElement("h1", {
```

```
    className: "hej",
    children: "tjena"
  });
}
```

JSX skiljer sig från HTML på flera sätt. Till exempel, i JSX används className istället för “class” och “htmlFor” istället för “for”. Dessutom är alla element i JSX stängda, inklusive de som är självstängande i HTML, som <img> och <input>.

JSX är dessutom mycket striktare med att man skriver korrekt syntax än vad HTML är. Ifall man till exempel glömmer stänga en tagg i JSX så kommer hela appen sluta fungera, medan om man gör samma sak när man jobbar med HTML så kanske man inte ens märker felet förrän man kör koden i en HTML-validator. Detta är på grund av att webbläsare alltid försöker sitt bästa med att tolka koden, ändå är den i grunden fel. JSX-transpilatorn, å andra sidan, är inte så generös. (Babel, 2024)

En annan notabel skillnad mellan JSX och HTML är att eftersom JSX i grunden är JavaScript, så kan funktioner aldrig returnera fler än ett element. De andra elementen får helt enkelt finnas inuti föräldrars element, annars går det inte. JSX har ett speciellt element som HTML inte har som heter fragments de skrivs <> </>. Det går att tänka på de som tomma element, de används då man behöver returnera flera syskonelement. När React i sin tur renderar ut JSX:et så att webbläsaren ska kunna tolka det så kommer fragmenten inte att finnas där.

En viktig skillnad är även att många HTML-attributnamn är annorlunda. En vanlig missuppfattning om varför de är annorlunda är att det är reserverade ord i JavaScript och därför inte kan användas eftersom webbläsaren hade blivit förvirrad. Men i själva verket så används reserverade orden bara som nycklar i ett JavaScript-objekt, och sedan ES5 ersatte ES3 i 2009, kan reserverade ord faktiskt användas som objektegenskaper. När JSX (och React) skapades runt 2013, var det dock viktigt att vara kompatibel med Internet Explorer 8, som inte stödde ES5 och därmed inte tillät reserverade ord som objektegenskaper. Detta, tillsammans med historiska skäl, migrationskostnader och problem med de-strukturering, har lett till att alternativa namn, som 'className' istället för 'class', används i JSX, trots att det

tekniskt sett skulle vara möjligt att använda de ursprungliga HTML-attributnamnen idag. (Abramov, 2018)

3.3.2.3 React bakom kulisserna

När JSX har transpilerats så kallar den på createElement. I forum och online dokumentation så används ofta termen “virtual DOM”. "Virtual DOM" är helt enkelt bara ett JavaScript objekt som createElement skapar från vår JSX. Objektet beskriver vår DOM.

Innan man kan visa Virtual DOM på skärmen måste man göra om den till vanliga JavaScript DOM operationer. Denna procedur kallas för rendering. Render funktionen tar två argument, vår Virtual DOM och en vanlig DOM node. Funktionen klurar ut de lämpliga DOM node-operationerna baserat på Virtual DOM. Det är främst detta steg som avgör skillnaden mellan en React app och en React Native app. Vanliga React renderar till webbläsaren, medan React Native renderar till IOS och Android.

Vid det här laget så har vi inte lyckats med mycket, då det inte finns någon typ av så kallad “state management". Termen state-management syftar på hanteringen av olika tillstånd som en front-end-end kan ha. Det kan handla om mer komplexa ting som vilka produkter användaren har i sin varukorg, eller om mer enkla grejer som om en meny just nu är öppen eller stängd. När React skapades, 2013, ansåg skaparen att state-management var den största svårigheten i front-end-utveckling. Förr hanterades state-management lite annorlunda, men idag sker state-management i React oftast med så kallade “hooks”. Hooks är helt enkelt vanliga funktioner som är gjorda för att användas inuti funktionella komponenter. Den vanligaste är “useState”. Alla hooks returnerar två värden, i fallet av useState så returnerar den en oföränderlig variabel och en funktion. Variabeln bör man använda när man vill få värdet, och funktionen bör man använda när man vill ändra på värdet.

I kod ser det ut så här:

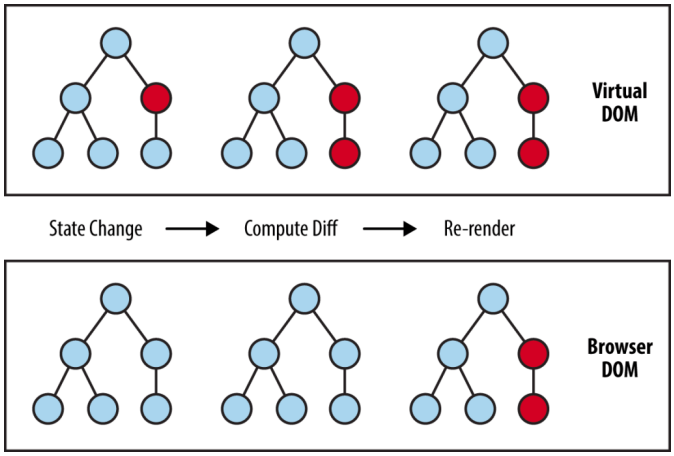
```
const [count, setCount] = useState(0);
```

useState kräver även ett argument. Argumentet fungerar som initiala värdet, i detta exemplet så får den 0.

När man kallar på funktionen som useState returnerar så kommer React först att kolla om det nya värdet över huvud taget är annorlunda än det förra. Ifall den är annorlunda så skapar

React en helt ny Virtual DOM. Den nya Virtual DOM:en jämförs sedan med det förra för att hitta ändringarna. Denna jämförelse kallas för “diff”, sedan renderas ändringarna. Hela proceduren kallas för reconciliation.

Reconciliation är en relativt komplext procedur som under åren har finjusterats. Utan den så hade React behövt rendera om hela DOM:et, och det hade varit väldigt ansträngande för webbläsaren.



När Reacts källkod släpptes 2013 så var reconciliation ett revolutionerande koncept, men idag så anser många JavaScript-experter att det är långsamt. (Shinde, 2021)

3.3.2.4 Next

Next.js är ett JavaScript-ramverk som bygger på React (som bara är ett bibliotek). Det utvecklades av företaget Vercel och är utformat för att underlätta utvecklingen av server-renderade och statiskt genererade React-applikationer.

Anledningen till varför enbart React ofta inte räcker-till är att React i grunden bara är JavaScript. Sökmotorer som Google kollar inte igenom JavaScript filer, och håller sig istället bara till HTML. Next.js erbjuder automatisk server-side rendering (SSR) och static site generation (SSG) för att lösa problemet, det betyder med andra ord att den skapar HTML-sidor från JavaScript filer.

SSR hjälper till med SEO, och även hastigheten på hemsidan. När användaren klickar in på hemsidan så behöver inte webbläsarens JavaScript-interpreter ha jobbat klart med React-koden för att det ska visas någonting på skärmen. Det är eftersom HTML-filen (som Next.js skapade) laddas in först, det är bara förrän efter HTML-filen har laddat klart som webbläsarens JavaScript-interpreter börjar tolka JavaScript-koden. När webbläsaren har

laddat ner och tolkat JavaScript-koden så sker en så kallad “hydration”. Hydration innebär att den statiska HTML-sidan blir numera interaktiv, med andra ord; den statiska HTML-sidan blir nu en till en interaktiv React app. Det finns dock en liten nackdel med denna typ av Hydration och det är att webbläsaren behöver rendera hemsidan två gånger, en gång för statiska HTML:en och en till gång när hemsidan blir interaktiv.

Next.js har en så kallad ”filbaserad routing”. Det innebär att filstrukturen i "app"-mappen automatiskt blir webbplatsens ruttstruktur. Till exempel blir en fil med sökvägen `app/about.js` tillgänglig på webbplatsen som `/about`.

Next.js är lämpligt för att bygga många olika typer av webbapplikationer, från enkla webbplatser till mer komplexa webbapplikationer. Det används av många stora företag, som till exempel Netflix, Uber och Airbnb. (Vercel, 2024)

3.3.3 Vue

Vue.js, eller helt enkelt Vue, är ett JavaScript-ramverk som används för att bygga användargränssnitt. Det skapades av Evan You ungefär två år efter React blev open-sourced och underhålls av honom tillsammans med communityn.

Vue bygger på konceptet med reaktiva data och komponenter likt React. Data i Vue är reaktiva, vilket betyder att när data ändras uppdateras vyn automatiskt. Komponenter i Vue är återanvändbara och oberoende bitar av kod som representerar en del av användargränssnittet.

Vue använder en HTML-baserad mall-syntax som gör det möjligt att binda den renderade DOM:en till underliggande data. Skillnaden i Reacts och Vues syntax beskrivs ofta genom att React är html-i-JavaScript, medan Vue är JavaScript-i-html

Vue används för att bygga många olika typer av webbapplikationer, från enkla webbplatser till komplexa. Det är särskilt populärt bland fristående utvecklare och små till medelstora företag på grund av dess enkelhet och flexibilitet, dessutom så är de alternativen React, Next, Angular skapat och underhållet av stora tech-företag, medan Vue bara underhålls huvudsakligen av en person. Företagen har ju som mål att dra in stora mängder pengar, vilket kan påverka utvecklandet av ramverket.

Vue är lättillgängligt för nya utvecklare och kan integreras med en rad olika bibliotek och verktyg. Dessutom är dess renderingssystem snabbare än React. Vue är även ett steg ifrån

stora tech-företag som Facebook eller Vercel som ligger bakom React respektive Next, och därmed är mer omtyckt av de som föredrar open-source-projekt som inte styrs av stora företag (notera dock att även React och Next är open-source-projekt). Detta kan vara en fördel för de som värderar oberoende och community-drivna projekt.

Nackdelar med Vue inkluderar att det inte har lika stort och aktivt community som React eller Angular, även om det växer snabbt. Det kan leda till att det blir svårare att felsöka buggar. Det finns också färre färdiga komponenter och bibliotek tillgängliga för Vue jämfört med React. (Vue, 2024)

3.3.3.1 Nuxt

Nuxt.js är ett ramverk som är starkt inspirerat av Next.js, vilket framgår av namnet och funktionaliteten. Det är som Next.js men för Vue, istället för React. Precis som Next.js underlättar utvecklingen av server-renderade och statiskt genererade webbapplikationer, gör Nuxt.js detsamma för applikationer byggda med Vue.js. Detta inkluderar funktioner som automatisk routing, server-side rendering, och state management, vilket gör det enklare att bygga komplexa applikationer med Vue.js.

Nuxt.js har många av samma funktioner som Next.js har. Dessa inkluderar:

Server-Side Rendering (SSR): Både Nuxt.js och Next.js erbjuder server-side rendering som hjälper till med SEO och prestanda.

Static Site Generation (SSG): Möjligheten att generera statiska sidor vid byggtid

Filbaserad Routing: Automatisk routing baserad på filstrukturen i "pages"-mappen.

Plugin System: Möjlighet att utöka funktionaliteten med plugin och moduler.

Code Splitting: Automatisk koduppdelning för att optimera laddningstider.

Nuxt.js är ett ramverk som tar det bästa från Next.js och anpassar det för Vue.js-ekosystemet. (Nuxt, 2023)

3.3.4 Angular

Angular är ett ramverk som är skapat av Google. Det släpptes 2010 och anses ibland därför vara föråldrad. Det är en 'allt-i-ett'-lösning som täcker allt från routing och formulärhantering till HTTP-klienten, det går jämföras med Nuxt eller Next.

Kärnan i Angular är dess komponenter och tjänster. Komponenterna representerar olika delar av användargränssnittet, och varje komponent består av en HTML-mall och en klass som styr logiken för den delen av gränssnittet.

Angular använder sig av tvåvägs databindning. Det innebär att när modellen ändras uppdateras vyn automatiskt, och tvärtom.

Det finns många fördelar med Angular, som dess omfattande funktionalitet, starka typning och stödet från Google. Det inkluderar en mängd funktioner och verktyg som verkligen underlättar utvecklingen av komplexa applikationer.

Angular också sina nackdelar. Dess inlärningskurva och storlek kan vara ganska överväldigande, särskilt för nya utvecklare. Det kan också vara lite mycket för mindre projekt som inte behöver alla dessa funktioner. (Angular, 2023)

3.4 JavaScript på back-end

Att JavaScript nu kan användas för back-end-utveckling är en utveckling vi främst har Node.js att tacka för. Node.js, en JavaScript-runtime, möjliggör körning av JavaScript-kod utanför webbläsarens miljö. Innan Node.js introducerades 2009 var JavaScripts användningsområde huvudsakligen begränsat till klient-sidans programmering i webbläsare.

Sedan Node.js lansering har ytterligare alternativ som Deno och Bun framträtt på scenen, vilka båda visar förbättrad prestanda jämfört med Node.js. Dessa teknologier erbjuder moderna utvecklare ett bredare spektrum av valmöjligheter när det kommer till att bygga robusta och effektiva back-end-system med JavaScript. (OpenJS Foundation, u.d.) (Deno, u.d.) (Bun, 2024)

3.5 Databaser

3.5.1 SQL

Under 1970-talet introducerades konceptet om relationsdatabaser av Edgar F. Codd, med syftet att adressera och förbättra de brister som fanns i dåtidens databassystem. Dessa brister inkluderade bland annat problematiken med duplicerade data och utmaningar relaterade till dataintegritet.

SQL står för ”Structured Query Language” och är det standardiserade språket som tillämpas för att hantera relationella databaser. För de som arbetar med databaser idag, är det närmast oumbärligt att ha en god förståelse för SQL, eftersom det är centralt för att kunna skapa, modifiera och interagera med databasdata.

Viktigt att notera är att SQL i sig inte är en databas, utan snarare ett språk som används för att arbeta med databaser som följer den relationella modellen. Bland de mest kända systemen som använder SQL kan nämnas MySQL, PostgreSQL och Microsoft SQL Server.

Ett av de mest grundläggande koncepten inom SQL-databaser är relationer, vilket beskriver hur data organiseras och relaterar till varandra över olika tabeller. Genom dessa relationer kan man återskapa och hantera komplexa datastrukturer på ett effektivt sätt. Det finns huvudsakligen tre typer av relationer i en relationsdatabas: ”one-to-one”, ”one-to-many ”och ”many-to-many”.

En one-to-one relation illustrerar en situation där exempelvis en användare är kopplad till exakt en kundvagn, vilken i sin tur endast är kopplad till just den användaren.

En one-to-many relation kan beskriva hur en kategori kan inkludera flera olika artiklar, vilket möjliggör en hierarkisk datastruktur.

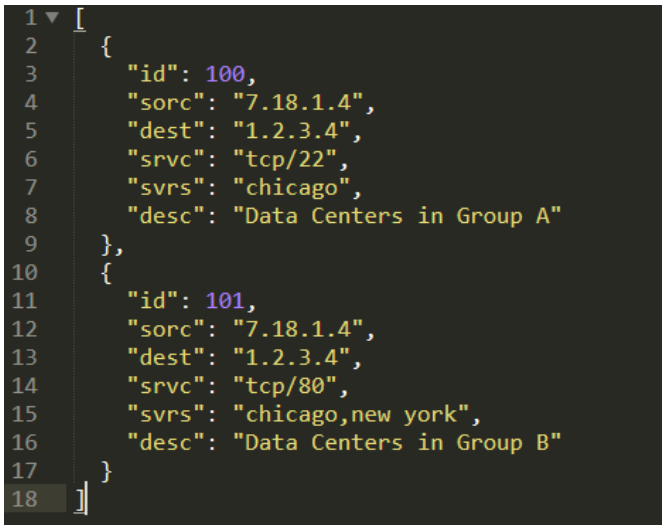
För att hantera situationer där många entiteter relaterar till många andra, som i fallet med kundvagnar som innehåller flera artiklar och vice versa, används many-to-many relationer. Dessa kräver skapandet av en speciell tabell för att korrekt representera relationerna.

I dagens affärsvärld är användningen av SQL-databaser utbredd bland stora organisationer. Exempelvis utnyttjar YouTube MySQL för sin datalagring. Även Netflix, som använder en kombination av teknologier inklusive MySQL och NoSQL-databasen Cassandra, illustrerar

hur företag kan integrera olika databaslösningar för att möta sina specifika behov. (Wikipedia, 2024) (Sarawagi, u.d.)

3.5.2 NoSQL

NoSQL-databaser, som avviker från den traditionella SQL-standarden, och representerar en annorlunda filosofi när det gäller hantering av data och dess strukturer. Dessa system brukar att organisera data på ett sätt som påminner om JSON-objekt, vilket ofta gör dem intuitivt förståeliga för programmerare van vid att hantera sådana dataformat.



Bilden visar ett exempel på hur data kan se ut i en NoSQL Databas (Ruby Sash Consulting, u.d.)

Trots deras flexibilitet och anpassningsbarhet, medför NoSQL-databaser även vissa begränsningar. En av de främsta utmaningarna är avsaknaden av en enhetlig standard, vilket innebär att de färdigheter och den kunskap man utvecklar inom en specifik NoSQL-teknologi, såsom Cassandra, inte nödvändigtvis är överförbara till en annan, såsom MongoDB. Denna variation kräver därför en bredare teknisk förståelse för att kunna navigera effektivt mellan olika NoSQL-baserade system. (Powell, 2023)

3.5.3 ORM

ORM står för "Object-Relational Mapping" och är en programmeringsteknik som används för att hantera data mellan objektorienterade programmeringsspråk och relationsdatabaser. En ORM fungerar som en bro mellan databasen och programmeringsspråket, där data i databasen representeras som objekt i programkoden. Detta gör det mycket enklare för utvecklare att interagera med databasen genom att använda ett vanligt programmeringsspråk,

utan att behöva skriva komplex SQL-kod. I JavaScript ekosystemet finns det flera populära ORM som t.ex. ”sequelize”, ” typeorm”, ”prisma orm”, eller ”drizzle orm”. (Wikipedia, 2024)

3.6 Tech-stack

En så kallad Tech-stack går att jämföra med en smörgås. Alla smörgås har olika lager, som bröd, pålägg, grönsaker, såser, osv., och varje lager har en specifik funktion och bidrar till den övergripande smaken och strukturen på smörgåsen. På samma sätt består en tech-stack av olika lager av teknologier, verktyg och ramverk som används för att bygga och driva en webbapplikation. Varje lager i tech-stacken har en specifik funktion, som att hantera databasen, servern, klienten, användargränssnittet, osv., och alla lager måste fungera tillsammans för att skapa en fungerande och effektiv applikation. Precis som att välja rätt ingredienser för en macka är avgörande för dess smak och konsistens, är valet av rätt teknologier för en tech-stack avgörande för prestanda, skalbarhet och underhållbarhet av en applikation. (Delaney, 2021)

Vilka teknologier man väljer för att bygga en mjukvara kan bero på flera olika saker. Huvudsakligen handlar det ju om vilka krav projektet har. Men det handlar ju också dels om vilka kunskaper man själv (eller sitt team) har, dels om hur bra dokumentationen och resurserna för den teknologin är, dels om vilken framtid man tror att teknologin har, dels även hur väl det kan integreras med andra teknologier.

3.4 Min webbshop

I den snabbt föränderliga världen av webbutveckling står utvecklare inför valet av det mest lämpliga front-end-ramverket för att skapa dynamiska och användarvänliga webbshoppar. Bland de många alternativ som finns tillgängliga, framstår React.js som ett framträdande val, mycket tack vare dess breda användning och rikedom på resurser. React.js, förstärkt genom meta-ramverket Next.js, erbjuder utökade möjligheter som utökar Reacts grundläggande förmågor, vilket gör det till ett lockande alternativ för många utvecklare.

Som ett annat framstående alternativ står Vue, känd för sin internationella användarbas och tillgängligheten av mångspråkig dokumentation. Vue, tillsammans med sitt meta-ramverk Nuxt, erbjuder en alternativ arkitektur som kan passa olika projektbehov.

Tillgången på omfattande och välstrukturerad dokumentation är avgörande för effektiv utveckling, eftersom den minimerar behovet av gissningar och förenklar problemlösningen. Utöver grundläggande dokumentation är ekosystemets mognad, exemplifierad genom kvaliteten och tillgängligheten av tredjepartsbibliotek, en viktig faktor att överväga.

React och Vue skiljer sig åt i sina filosofier och tillvägagångssätt för att bygga användargränssnitt. Medan React fokuserar på komponenter som enkla funktioner, adopterar Vue konceptet med ”single-file-components” som integrerar mall, script och stil i en enda fil. Denna skillnad i designfilosofi understryker att det inte finns ett universellt "rätt" eller "fel" val; det bästa ramverket för ett projekt beror på de specifika krav och preferenser som projektet och dess utvecklare har.

För att bygga en webbshop från grunden behöver man först bryta ner hela projektet i fler olika delar, och sedan välja de mest lämpligaste teknologierna för de individuella delarna. De olika delarna inkluderar:

1. En så kallad ”front-end”, det den delen av koden som användaren kommer att se och interagera med.
2. En så kallad ”back-end”, den delen av koden som står som en säkerhetsvakt mellan anropen som användaren skickar och databasen.
3. Och en databas, där all data om användare och produkter lagras.

Jag kom fram till att de mest lämpliga teknologierna att använda för att bygga mitt projekt (och för mig som utvecklare) var följande: TypeScript, React, Next, Tailwind, Hono, MySQL, Stripe, Bun, DrizzleORM, Docker.

Hemsida finns uppe just nu på följande adress: clothing-webshop-one.vercel.app

Källkoden ligger uppe på följande adress: github.com/balazshevesi/clothing-webshop

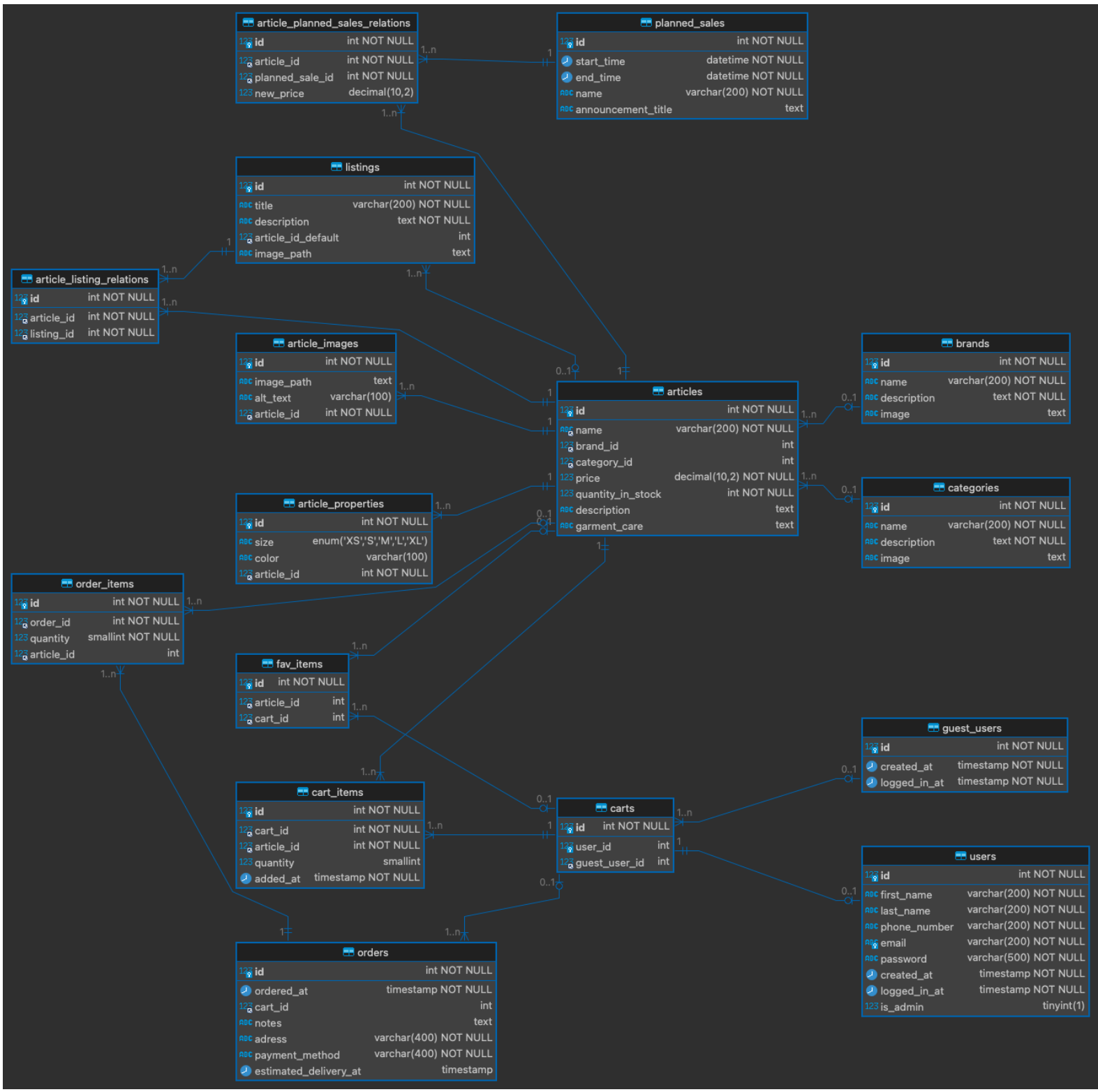
3.4.1 Databasen

Att välja rätt databas är inte enkelt. Idag finns det många exempel på företag som använder både SQL och NoSQL lösningar. När jag först började bygga min hemsida så använde jag MongoDB, en NoSQL databas. MongoDB är så kallad ”schemaless”, det betyder att den är väldigt flexibel med hur den låter klienten skriva data till den. Det låter först som en positiv egenskap, men det betyder samtidigt att det är väldigt lätt att göra fel. I min erfarenhet av

MongoDB så var just det ett upprepande mönster; det är väldigt lätt att göra fel, och oklart hur man ska utöka databasen ifall jag behöver mer data.

SQL är mycket mer sträng. Den kräver att man definierar datamodellen innan klienten kan skriva till den. I min erfarenhet så fungerar det mycket bättre, och det är mycket enklare att skapa databasintegritet med den.

Nedan är en bild på en visualisering ifrån programmet Dbeaver som visar hur databasdesignen för min hemsida ser ut:



Rutorna representerar tabeller. Texten som ligger till vänster i rutorna representerar de olika fälten som tabellen har. Texten till höger i rutorna förklarar datatypen som fältet har. Sträcken emellan tabellerna representerar relationerna mellan de.

4. Diskussion

Jag tycker att jag lyckades rätt bra med att förstå mig på de olika JavaScript ramverken som används idag, och jag tycker att min metod för att forska omkring det fungerade fint.

I utvecklingen av min egen webbshop så lyckades jag inte lika bra. En av de mest påtagliga utmaningarna var att följa den uppsatta tidsplanen, vilket visade sig vara en svår uppgift. Buggar och oväntade komplikationer uppstod kontinuerligt, vilket belyste mjukvaruutvecklings oförutsägbara natur. Detta blev särskilt tydligt när jag efter många timmars arbete insåg att kodbasen hade utvecklats till ett kaotiskt tillstånd, utan tydliga strukturer eller mönster. Lösningarna, särskilt de rörande hanteringen av tillstånd (state management), var svåra att följa och en enkel justering kunde oväntat kräva en timmes arbete.

Dessa erfarenheter ledde till beslutet att starta om projektet från grunden. I retrospektiv insåg jag att en noggrann kravspecifikation, särskilt för stora projekt som en e-handelsplattform, är avgörande innan tidsplanering påbörjas. Trots dessa hinder, var det slutgiltiga resultatet i stort sett i linje med min ursprungliga vision.

Projektet belyste också de begränsningar som följer med att använda en icke-relationell databas. Under min praktikperiod på Kaxig AB insåg jag att min produktdatabas saknade referensintegritet, vilket skulle medföra omfattande manuella uppdateringar vid ändringar i kategorinamn. Denna insikt motiverade mig att fördjupa mig i relationella databaser och SQL, vilket ytterligare stärkte beslutet att starta om projektet med en mer lämplig teknisk grund.

Sammanfattningsvis har detta projekt varit en praktisk övning i mjukvaruutveckling och samt ett lärorikt äventyr igenom de olika ramverken och teknologierna som används för att bygga hemsidor.

4.1 Det inte hann med

I och med att det var svårt att planera tiden så hann jag inte riktigt med att implementera alla funktioner i min webbshop som jag från början hade tänkt mig.

Detta är (bara en del av) listan på det som jag inte hann med:

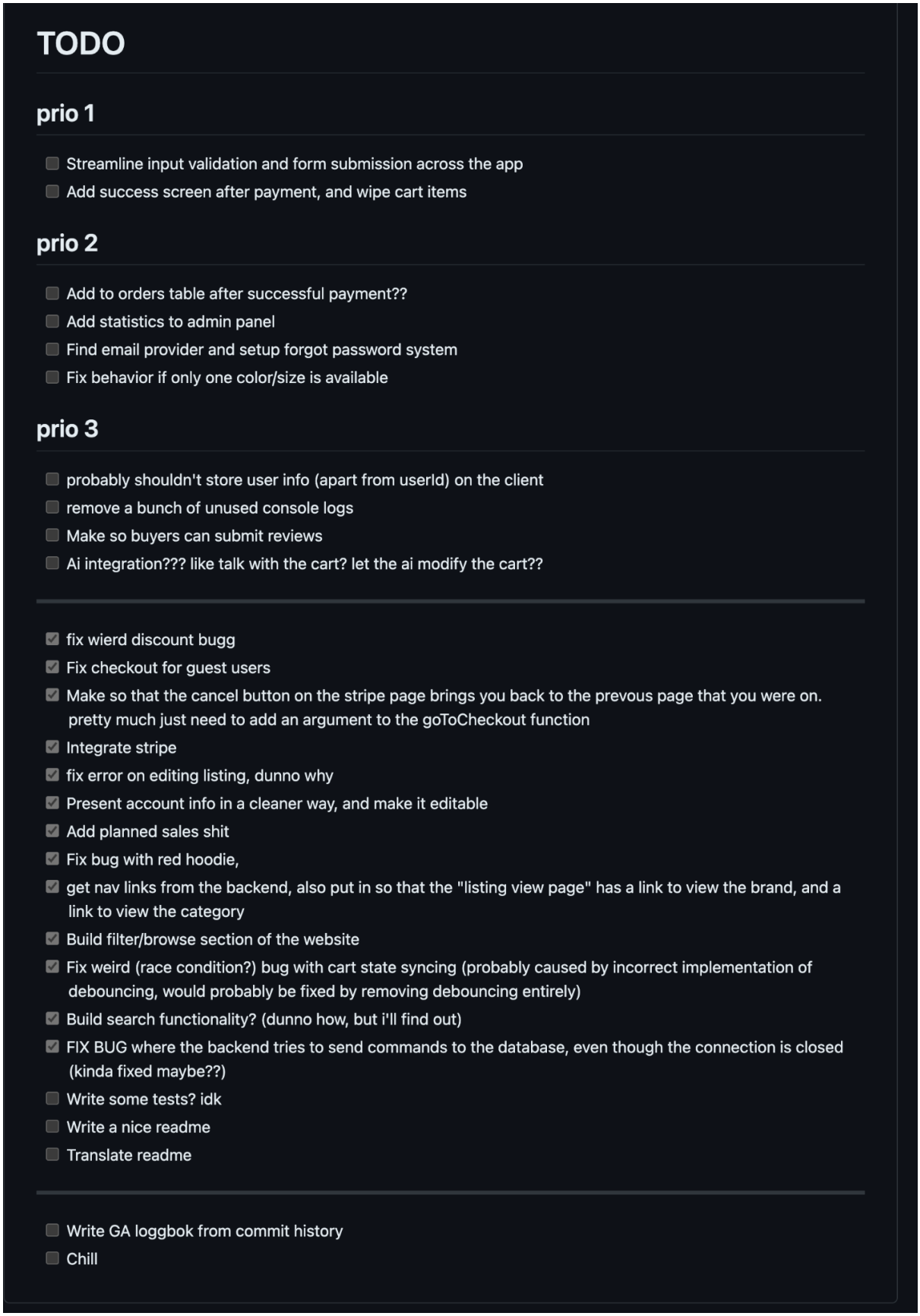
- Det finns en del buggar med hur rabatterna hämtas från databasen.
- Jag hann inte med att bygga ut logiken för hur beställningar ska fungera.
- Jag hann inte med att bygga ut ett system som skulle låta användarna byta lösenord ifall de glömde bort det
- "Most Popular" delen av hemsidan är egentligen falsk, ordningen av produkterna är i verkligheten baserade på ordningen som de blev inlagda i databasen, alltså är det inte alls baserat på hur populära de är.
- Produktens recensioner är falska, jag hann inte med att bygga ut systemet som skulle låta köparna skriva recensioner själva.
- Databasdesignen rymmer inte recensioner över huvud taget.
- Vissa delar av hemsidan är på Engelska, och vissa är på Svenska
- (Att i koden) Använda ”end-to-end type safety”

4.2 Om jag hade byggt min webbshop ifrån början igen

Om jag hade gjort om projektet så hade jag definitivt utnyttjat TypeScript mer. Både min front-end och back-end är skrivna med TypeScript, och det innebär att det är möjligt att dela typer mellan dem, s.k. “end-to-end type safety”. Att använda sig av det hade underlättat utvecklingen extremt mycket.

Jag hade också gärna jobbat ihop eller tagit hjälp av en webbdesigner eller bara kopierat någon annans design helt och hållet. Trots att jag har en känsla för estetik (om jag får säga det själv), så tog det rätt mycket av min tid att fixa så att hemsidan skulle se bra ut, och det var ju inte det som projektet skulle handla om.

Jag hade också använt mig av något “project management” system för att hålla koll på hur jag egentligen låg till med hemsidans utveckling. Den mest populära är ju “kanban”. Hur jag gjorde istället var att jag hade en textfil i samma mapp som jag hade koden i (root directory) och i den hade jag en att-göra-lista.



Detta är en bild på att-göra-listan. Jag skrev den bara för mig själv, det är därför det ser lite kaotiskt ut. Som man ser längst ner på bilden så fick jag aldrig chilla.

5. Källförteckning

Anon., n.d. *Mobile conversion rate statistics.* [Online]
Available at: https://www.thinkwithgoogle.com/marketing-resources/data-measurement/mobile-page-speed-new-industry-benchmarks/
[Accessed 26 August 2023].

Krall, C., n.d. *DOM o Document Object Model JavaScript ¿Qué es, para qué sirve? W3C. Nodos. Child. Ejemplos (CU01123E).* [Online]
Available at: https://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=801:dom-o-document-object-model-javascript-ique-es-para-que-sirve-w3c-nodos-child-ejemplos-cu01123e&catid=78&Itemid=206
[Accessed 27 Augusti 2023].

Hansa, U., 2022. *The difference between JavaScript libraries and frameworks.* [Online]
Available at: https://web.dev/articles/js-libraries-vs-frameworks
[Accessed 27 Augusti 2023].

Abramov, D., 2018. *React Fire: Modernizing React DOM.* [Online]
Available at: https://github.com/facebook/react/issues/13525#issuecomment-417818906
[Accessed 27 Augusti 2028].

Anon., n.d. [Online]
Available at: https://www.youtube.com/watch?v=f2mMOiCSj5c&t=938s&ab_channel=CodingTech
[Accessed 27 Augusti 2023].

Shinde, N., 2021. *React Reconciliation.* [Online]
Available at: https://nikshindeblogs.medium.com/react-reconciliation-547024dda9fc
[Accessed 27 Augusti 2023].

Powell, R., 2023. *SQL vs NoSQL databases.* [Online]
Available at: https://circleci.com/blog/sql-vs-nosql-databases/
[Accessed 18 Februari 2024].

Google, 2016. *of visits are abandoned if a mobile site takes longer than 3 seconds to load..* [Online]
Available at: https://www.thinkwithgoogle.com/consumer-insights/consumer-trends/mobile-site-load-time-statistics/
[Accessed 27 Augusti 2023].

Google, 2016. *of visits are abandoned if a mobile site takes longer than 3 seconds to load..* [Online]
Available at: https://www.thinkwithgoogle.com/consumer-insights/consumer-trends/mobile-site-load-time-statistics/
[Accessed 27 Augusti 2023].

Green, V., 2016. *Impact of slow page load time on website performance.* [Online]
Available at: https://medium.com/@vikigreen/impact-of-slow-page-load-time-on-website-performance-40d5c9ce568a
[Accessed 27 Augusti 2023].

Toal, R., n.d. *What Is JavaScript Framework?.* [Online]
Available at: https://codeinstitute.net/se/blog/javascript-framework/#:~:text=In%20brief%2C%20JavaScript%20frameworks%20are,saving%20time%20for%20the%20developer.
[Accessed 28 Augusti 2023].

MDN, 2024. [Online]
Available at: https://developer.mozilla.org/en-US/docs/Learn/Performance/JavaScript
[Accessed 10 April 2024].

Delaney, J., 2021. [Online]
Available at: https://www.youtube.com/watch?v=Sxxw3qtb3_g
[Accessed 25 Mars 2024].

Wikipedia, 2024. *Object–relational mapping.* [Online]
Available at: https://en.wikipedia.org/wiki/Object%E2%80%93relational_mapping
[Accessed 16 April 2024].

Lundgren, J, Webutveckling & programmering, Kaxig AB, 2023

Carnes, B., 2018. *The Timeline of JavaScript's Evolution.* [Online]
Available at: https://www.freecodecamp.org/news/timeline-of-javascript-evolution/
[Accessed 28 Augusti 2023].

Preact, 2024. *preactjs.* [Online]
Available at: https://preactjs.com/
[Accessed 27 Augusti 2024].

Vue, 2024. *The Progressive JavaScript Framework.* [Online]
Available at: https://vuejs.org/
[Accessed 27 Augusti 2023].

Nuxt, 2023. *The Intuitive Vue Framework.* [Online]
Available at: https://nuxt.com/
[Accessed 27 Augusti 2023].

Bun, 2024. *A fast all-in-one JavaScript runtime.* [Online]
Available at: https://bun.sh/
[Accessed 27 Augusti 2023].

OpenJS Foundation, u.d. *About Node.js.* [Online]
Available at: https://nodejs.org/en/about
[Använd 27 Augusti 2023].

Deno, u.d. *Next-generation JavaScript runtime.* [Online]
Available at: https://deno.com/
[Använd 27 Augusti 2023].

Angular, 2023. *Deliver web apps with confidence.* [Online]
Available at: https://angular.io/
[Accessed 27 Augusti 2023].

Meta Open Source, 2024. *React.* [Online]
Available at: https://react.dev/
[Använd 27 Augusti 2024].

Vercel, 2024. *The React Framework for the Web.* [Online]
Available at: https://nextjs.org/
[Använd 27 Augusti 2024].

Wikipedia, 2024. *Relational database.* [Online]
Available at: https://en.wikipedia.org/wiki/Relational_database
[Accessed 27 Augusti 2024].

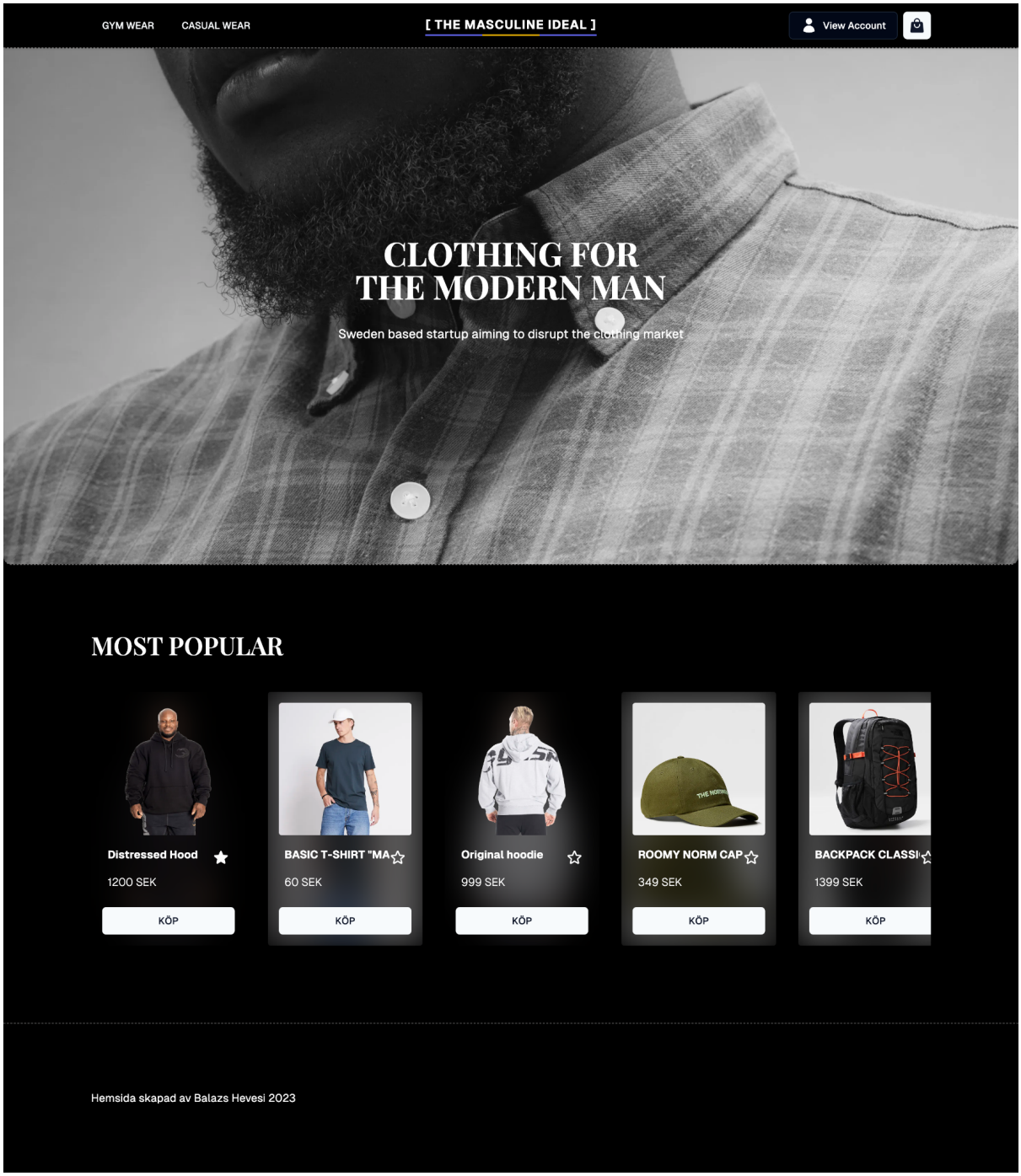
Sarawagi, S., n.d. *YouTube database – How does it store so many videos without running out of storage space?.* [Online]
Available at: https://scaleyourapp.com/youtube-database-how-does-it-store-so-many-videos-without-running-out-of-storage-space/
[Accessed 18 Februari 2024].

Ruby Sash Consulting, u.d. *Rubysash.* [Online]
Available at: https://rubysash.com/programming/python/read-write-json-file-python-3/
[Använd 18 Februari 2024].

Babel, 2024. *The Compiler For Next Generation Javascript.* [Online]
Available at:
https://babeljs.io/repl#?browsers=defaults%2C%20not%20ie%2011%2C%20not%20ie_mob%2011&build=&builtIns=false&corejs=3.21&spec=false&loose=false&code_lz=GYVwdgxgLglg9mABAQQA6sQCgJQG8BQATgKZQiFKYA8AFgIyIQA2AhgM5sByLAtsQLwAiGsQBWggHxRRxMCyoB6ehOwBffPiA&debug=f
[Använd 27 Augusti 2023].

The Odin Project, 2024. *Component Lifecycle Methods.* [Online]
Available at: https://www.theodinproject.com/lessons/node-path-react-new-component-lifecycle-methods
[Accessed 27 Augusti 2023].

6. Bilagor



Hur hemskärmen på webshoppen ser ut.

ROOMY NORM CAP

The North Face Casual Wear

349 SEK

The Roomy Norm Cap offers slightly more depth than the classic Norm Cap but has the same great looks

Green Blue

In Stock: 50

ADD TO CART

ARTICLE DESCRIPTION

GARMENT CARE

SHIPPING

REVIEWS

★★★★★
Fantastic Experience
This was beyond my expectations! The quality and service were top-notch. Highly recommend to anyone looking for excellence.
Alex Johnson 2024/3/7

★★★★★
Absolutely Loved It
I'm thoroughly impressed with the level of detail and care put into this. It definitely deserves a five-star rating.
Samantha Lee 2024/3/7

★★★★★
Outstanding Quality
From start to finish, the experience was flawless. The attention to detail is evident, and I couldn't be happier.
Michael Brown 2024/3/7

★★
Rem
Every
above
expe
Emma

MOST POPULAR

Distressed Hood ★
1200 SEK
KÖP

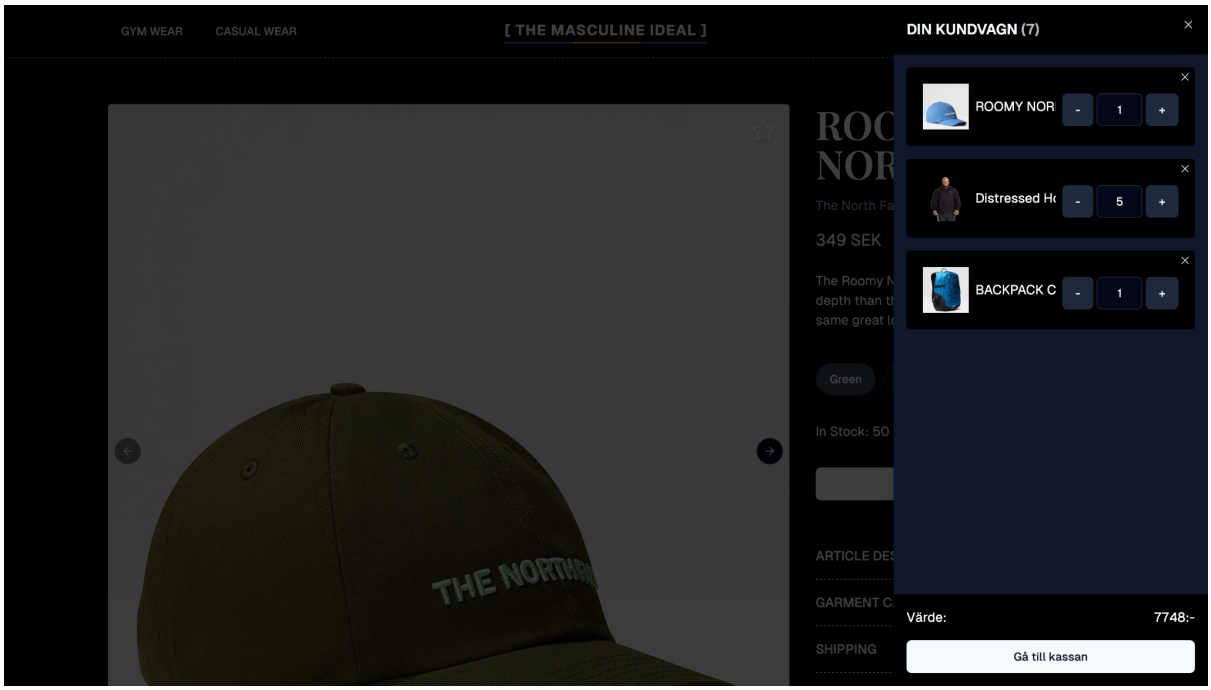
BASIC T-SHIRT "MA ☆
60 SEK
KÖP

Original hoodie ☆
999 SEK
KÖP

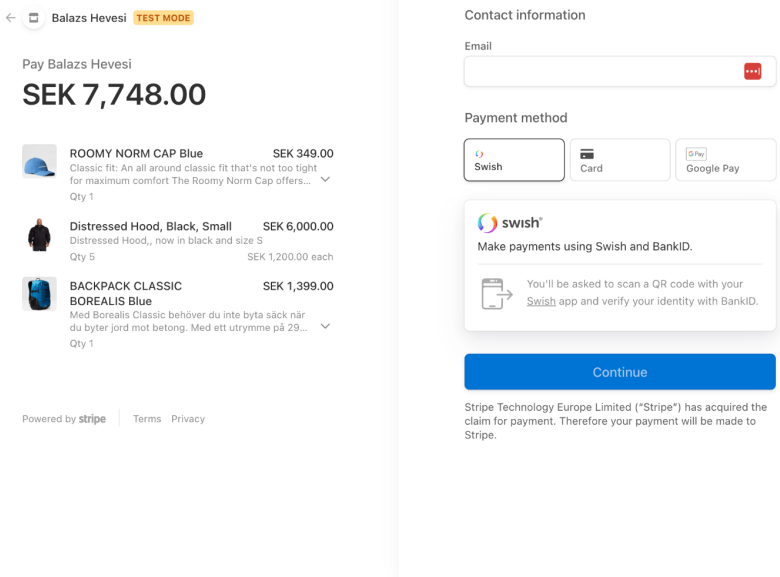
ROOMY NORM CAP ☆
349 SEK
KÖP

BACKPACK CLASSI ☆
1399 SEK
KÖP

Hur produktsidan ser ut (rent tekniskt är det väl egentligen annons-sidan).



Hur kundvagnen ser ut när den är öppen



Hur betalningssidan ser ut (Stripe).

GYM WEAR CASUAL WEAR [THE MASCULINE IDEAL] View Account

CASUAL WEAR

Casual wear for everyday use

Search 29 articles

CONFIGURE FILTER

Price:

0 4000

Categories:

Gym Wear ✓ Casual Wear

Brands:

Gasp HM Lager 157 The North Face

Show only if in stock:

☐ Show only if in stock

Show listings instead of articles:

☑ Show listings instead of articles

ORDER BY name

‹ Previous 1 2 3 ··· Next ›

BASIC T-SHIRT "MARLON" ☆

60 SEK

KÖP

ROOMY NORM CAP ☆

349 SEK

KÖP

BACKPACK CLASSIC ☆

1399 SEK

KÖP

LINNESKJORTA "LUCAS STRIPE ☆

200 SEK

KÖP

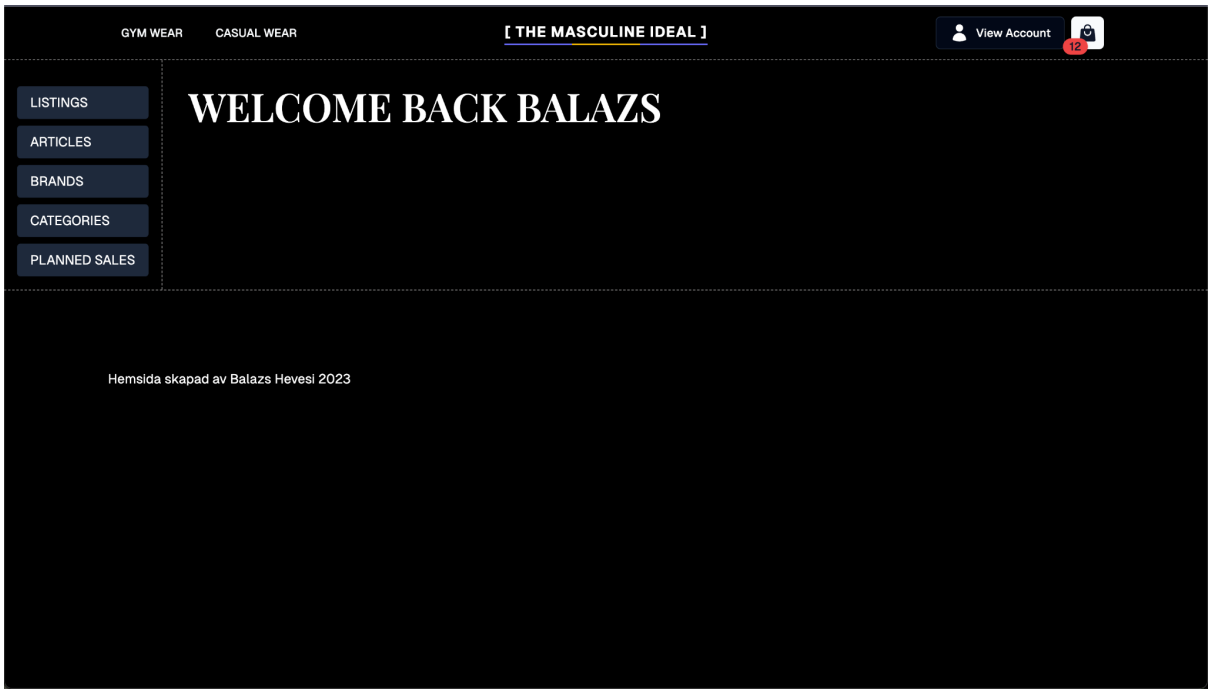
Skjorta i linmix Regular Fit ☆

299 SEK

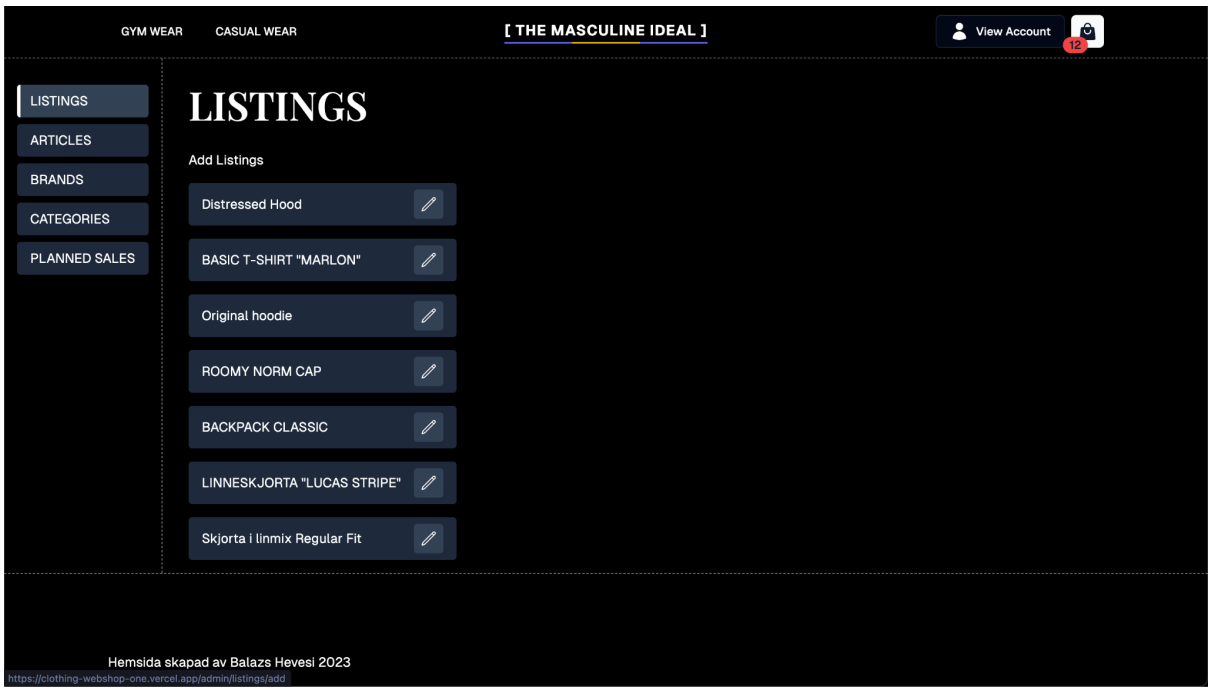
KÖP

‹ Previous 1 2 3 ··· Next ›

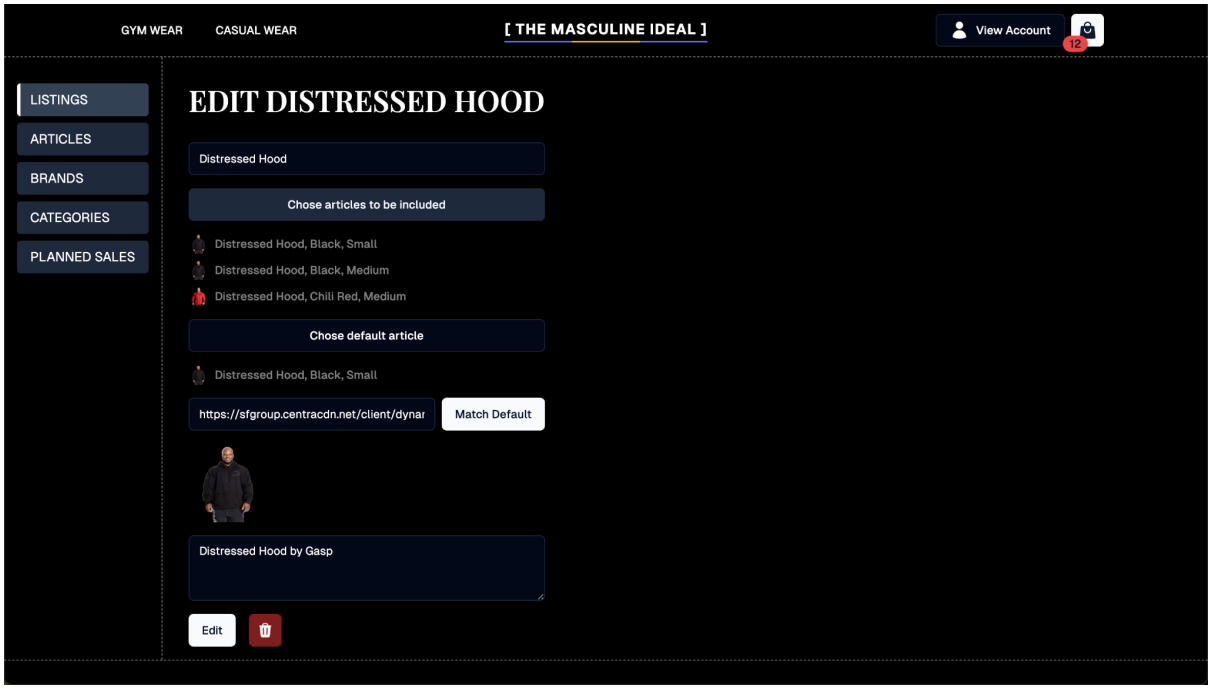
Hur kategori-eller-märke-sidan ser ut.



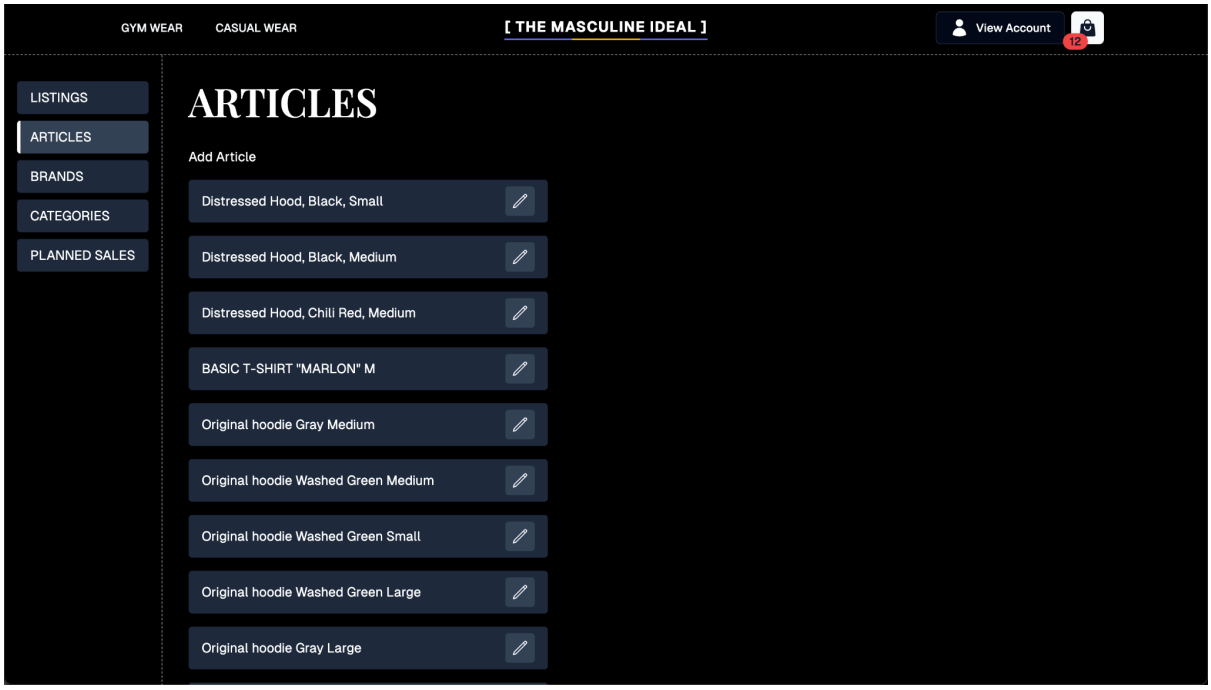
Hur hemskärmen i admin panelen ser ut



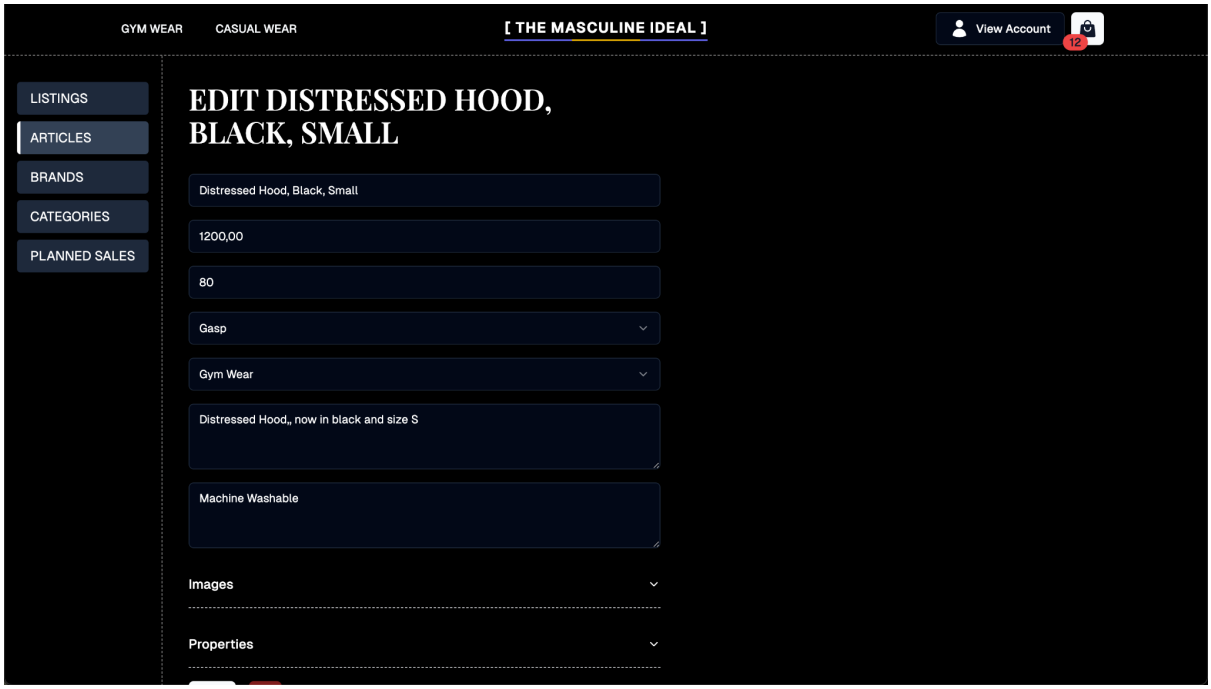
Hur annonssidan i adminpanelen ser ut



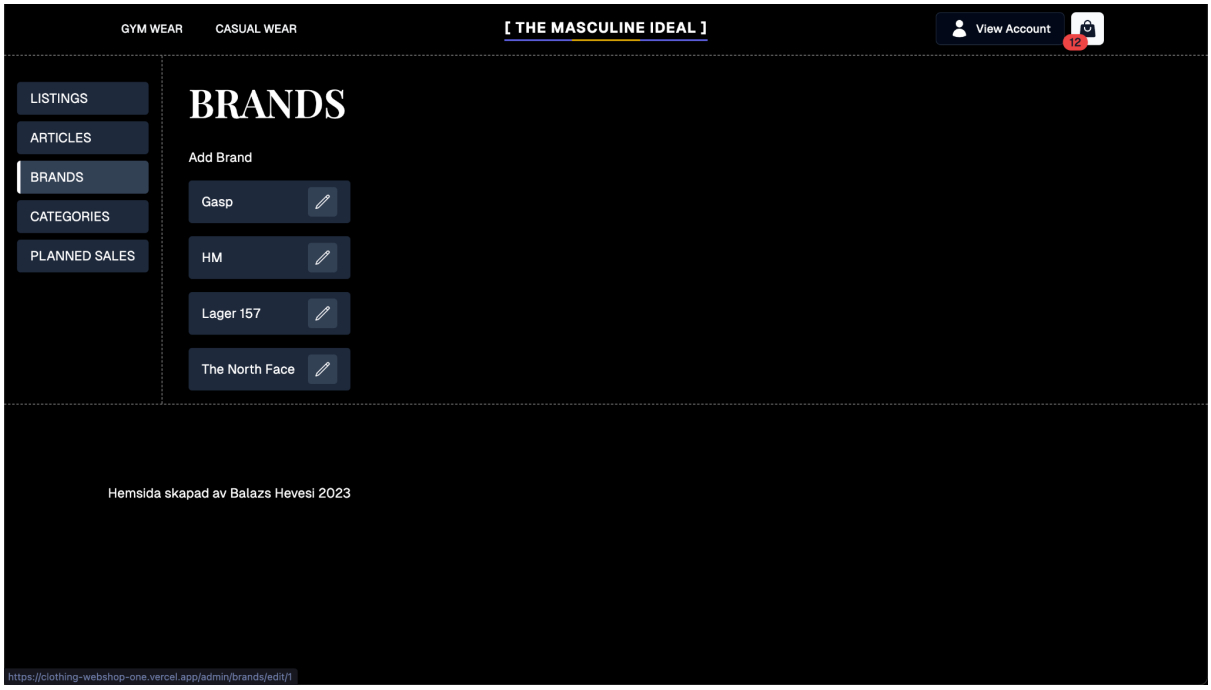
Hur redigera-annons-sidan i adminpanelen ser ut



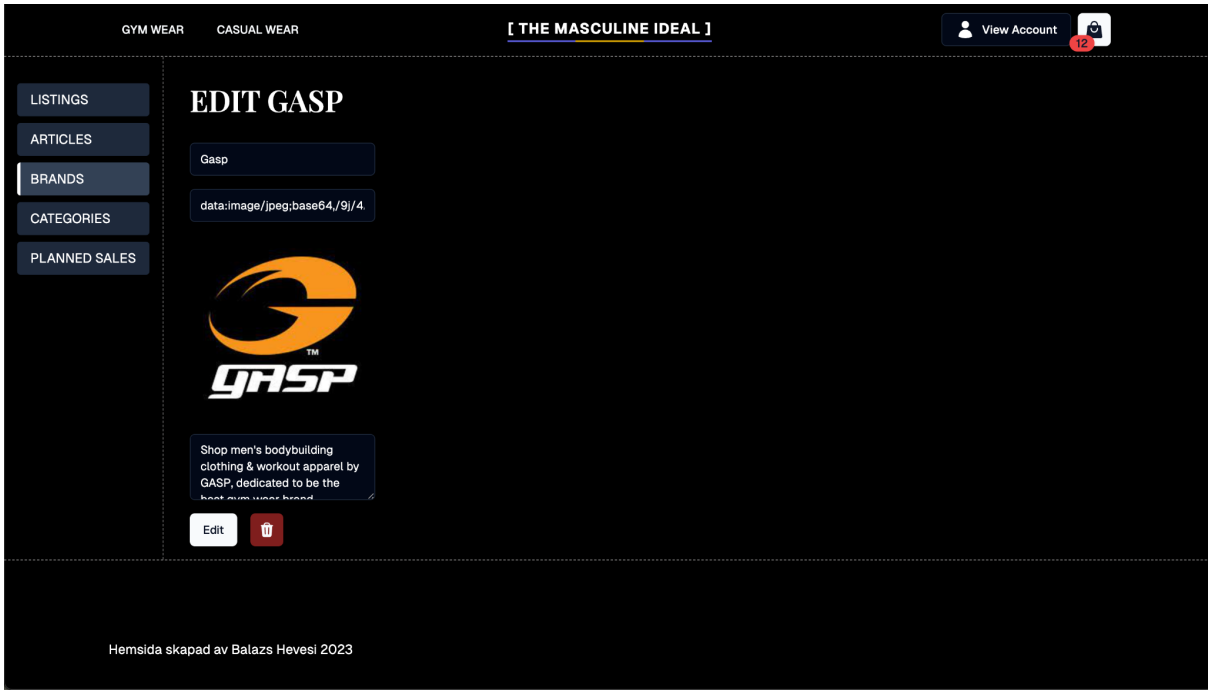
Hur artikelsidan i adminpanelen ser ut



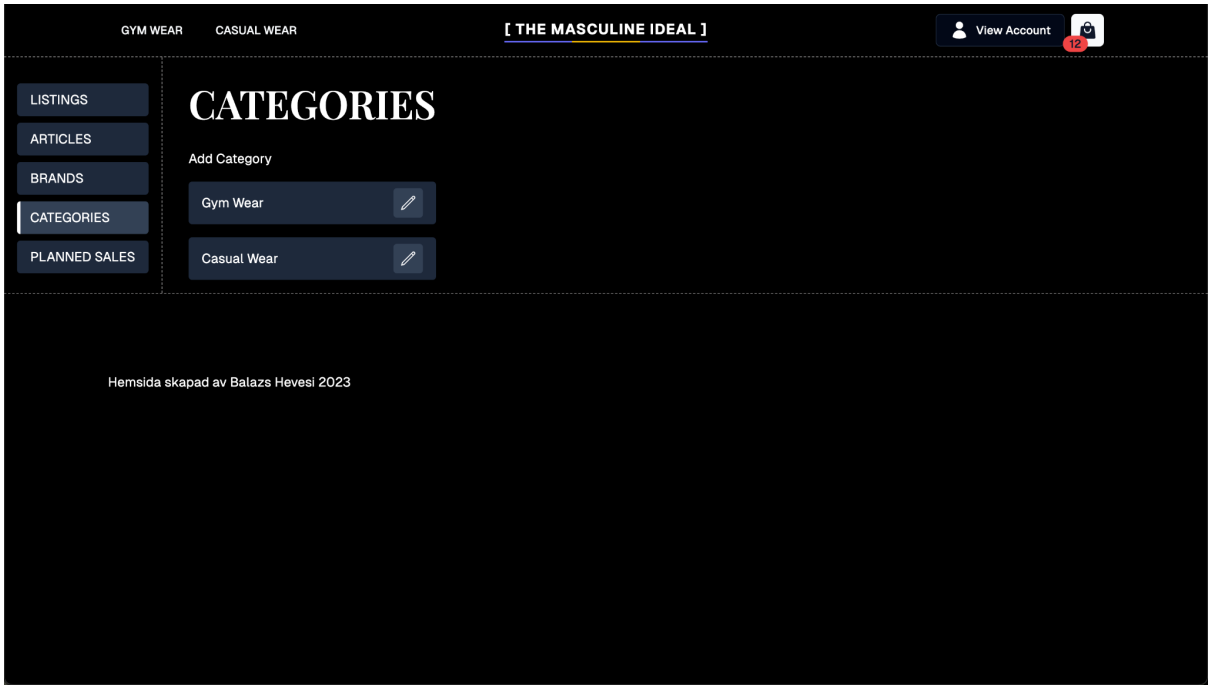
Hur redigera-artikel-sidan i adminpanelen ser ut



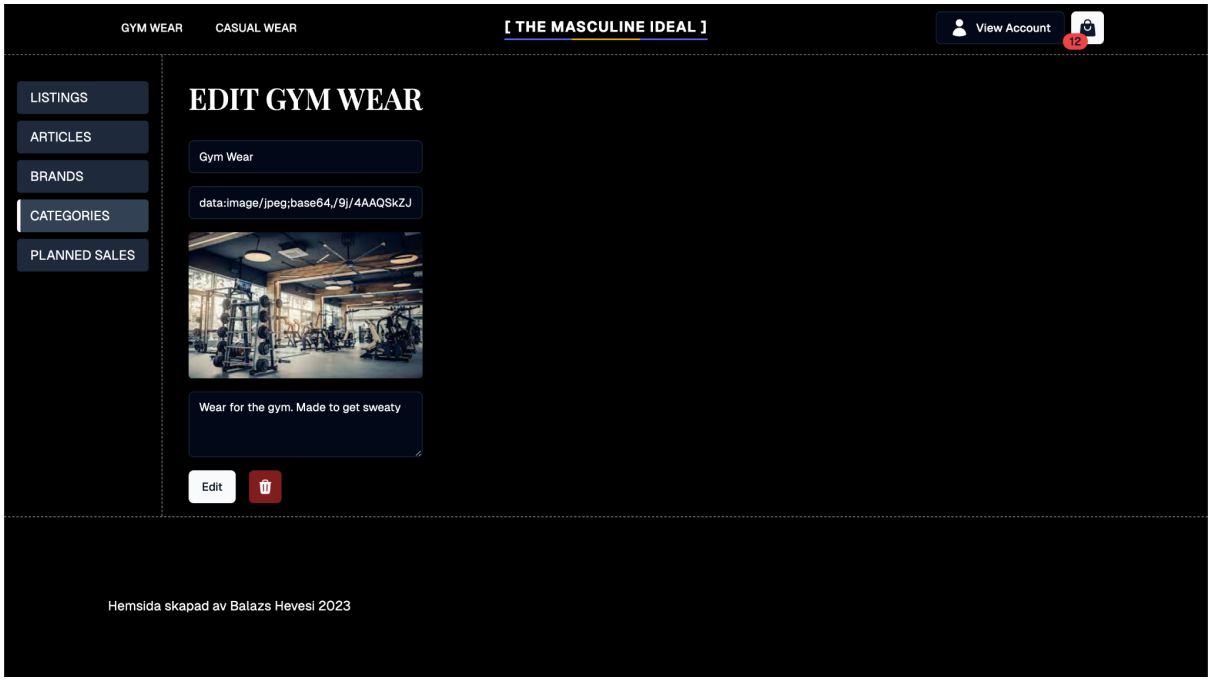
Hur märke-sidan i adminpanelen ser ut



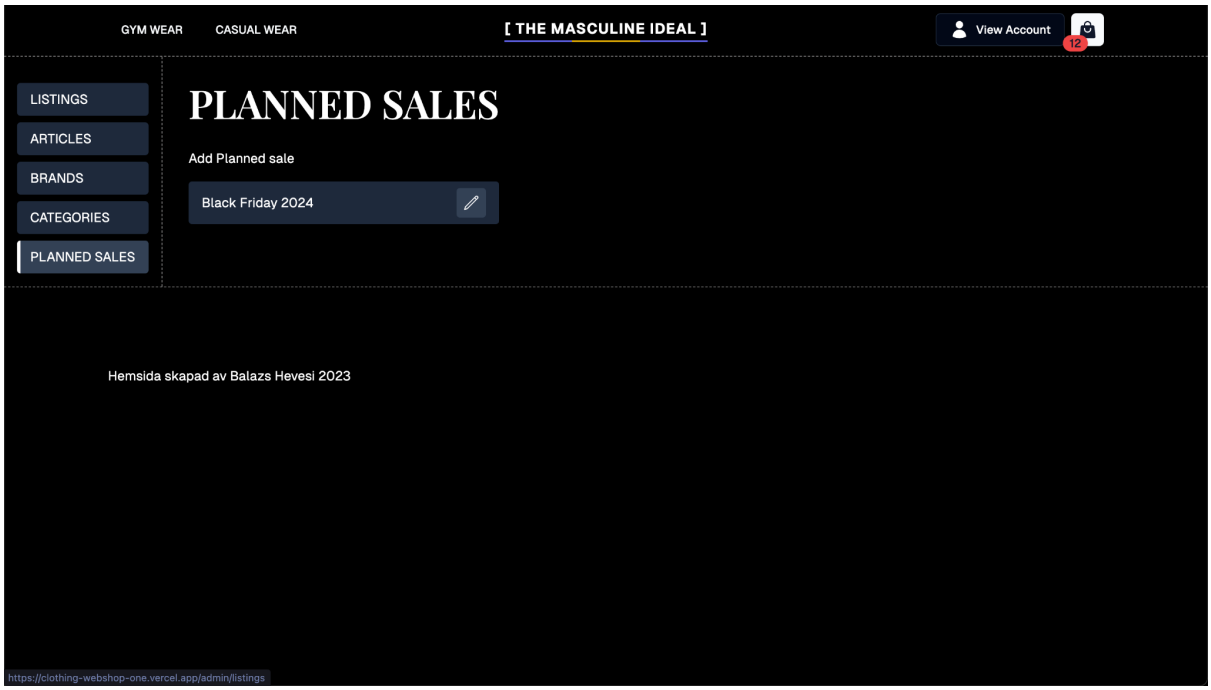
Hur redigera-märke-sidan i adminpanelen ser ut



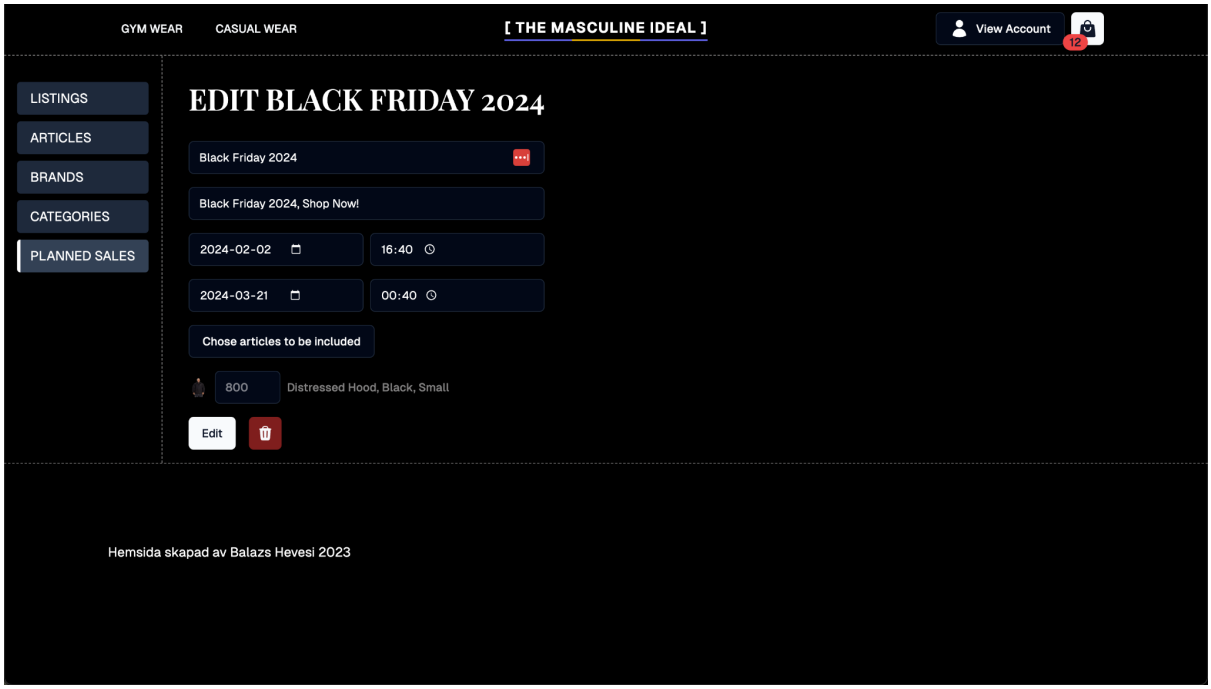
Hur kategorisidan i adminpanelen ser ut



Hur redigera-kategori-sidan i adminpanelen ser ut



Hur rabatt-sidan i adminpanelen ser ut.



Hur redigerar-rabatt-sidan i adminpanelen ser ut.