NACKADEMIN

WEBB20 Backend 2 PHP – Inlämningsuppgift 3 (Projektarbete) (G)

PHP Webbshop

Datum: 2021-05-28

Kurs: Backend 2 – PHP **Uppgift**: 3 (G)

Klass: WEBB20, grupp 8 (Inna, Viktoria, Mariia)

Student: MARIIA PARAKETSOVA

Introduktion

Målet av den uppgift var att skapa databasbaserad PHP-applikation för fake webbshop. Kund skall skapa konto. logga in, se sortiment, välja produkt och beställa det. Admin skal titta produkterna, uppdatera, ta borta en produkt och addera nya. Dessutom kan admin se alla beställningar och uppdatera status på en beställning.

Projects språk - PHP. Projekt baserar på Model View Controller Architecture och använder MySQL DB. Dessutom använder vi för styling: Bootstrap, CSS.

Webshops sortiment - inredning.

Projects repo:

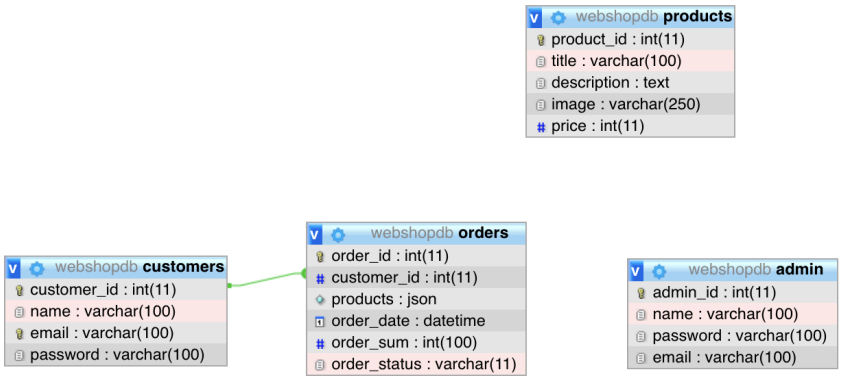
***https://github.com/paraketsova/fakestore_php* - GitHub**

Genomförande

Vi har börjat projekt med planläggning - utvärderat kraven för projektet, skapade ett socialt kontrakt för gruppen under hela projektets varaktighet, där vi bestämde interaktionen mellan gruppmedlemmarna, bestämde hur, när och under vilka förhållanden gruppen kommer att arbeta med projektet. Vidare har vi skapat TRELLO desk och definierat tasks. Ila uppgiften löstes av oss tillsammans, koden skrevs tillsammans och delades bara när ett problem uppstod för att försöka hitta den optimala lösningen individuellt och sedan välja den bästa.

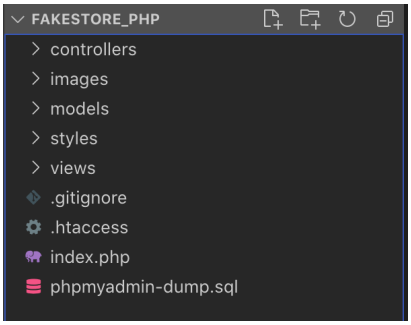
Vi har valt vilka produkter vi ska sälja i vår butik (inredning), har valt huvudfärger och mönster för front-end. Sedan bestämde vi oss för databasens struktur och huvudarkitekturen för PHP-koden.

Databasstrukturen ser ut så här:



Jag lade till en falsk lista med produkter, kunder och beställningar i databasen. Skapat ett konto för en administratör.

Repository struktur ser ut så här:



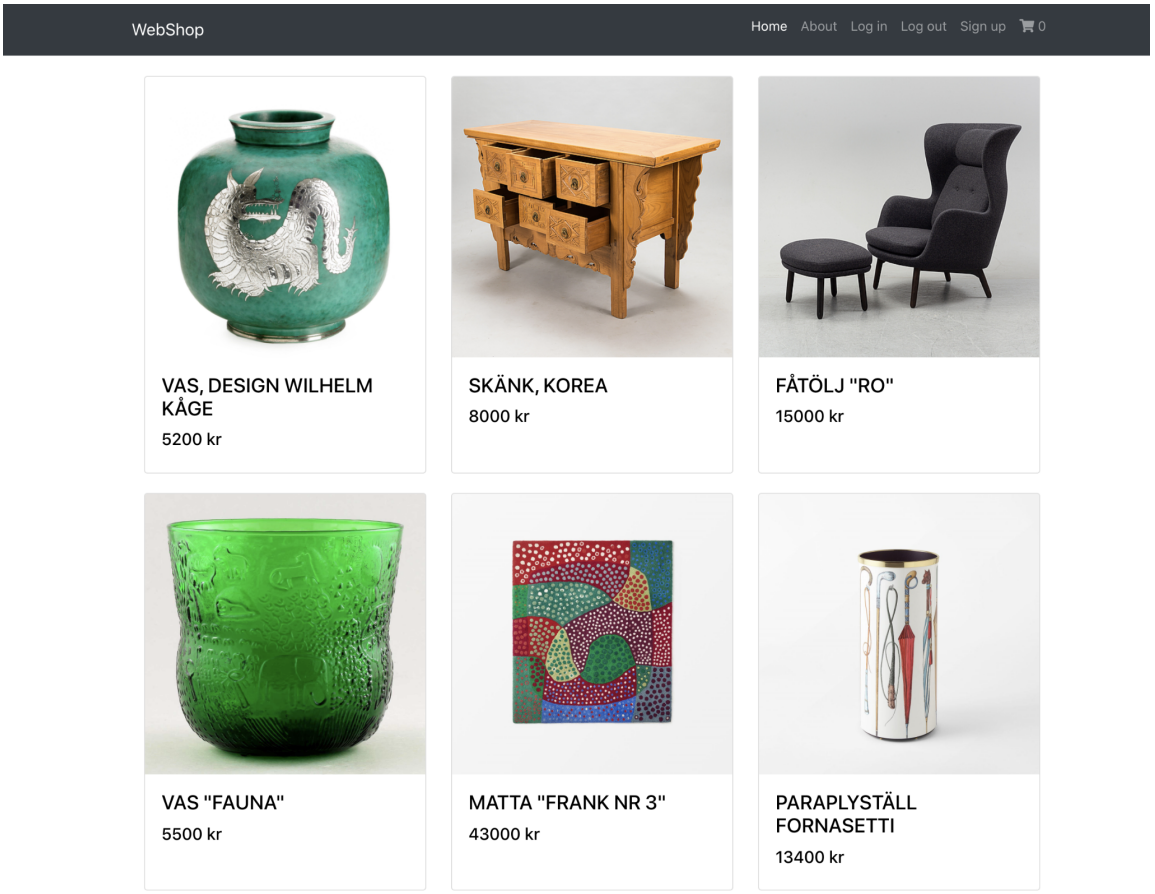
phpmyadmin-dump.sql - data som importera från MySQL DB, innehåller data för alla tabeller i databasen och förhållanden mellan tabeller.

Database.php - den här skapar jag class **Database** klass med metoder **__construct**, som skapa koppling till MySQL DB och kastar exception om det finns problem. Metod **execute** - en instansmetod som exekverar en PDO-sats. Metoder **select, insert, update, delete** definierar grundläggande operationer med databasdata.

index.php - definera urlroot for project, addera alla models, controllers och views. Skapar ett nytt objekt av klass Database. Skapar “case controlling” som läser querystring och fick namn för Controller och funktion för varje case. Om name för funktion är tom i querystring använder “index” case för controller. Således skapar vi för varje fall en “omkopplare” som hittar önskad styrenhet och möjliggör den önskade funktionen.

Vi tog denna idén (https://stackoverflow.com/questions/1737868/an-example-of-an-mvc-controller/1737903) som grund och använde den i vårt projekt. Till exempel, finns case “signup” som funkar med hjälp SignUpController, SignUppModel och SignUpView.

Default case som funkar för main sidan - **IndexController.php**, **Model.php** och **View.php** hjälpfiler.



Användaren kan här se alla produkter från sortimentet i butiken, se deras bild, namn, pris. Han kan gå från den här sidan till sidan i en separat beställning genom att klicka på produktkortet i huvudfältet på sidan. Han kan gå till informationssidan i butiken, till SignUpformuläret eller till inloggningsformuläret och också logga ut genom att gå till motsvarande länkar i navbaren.

klass IndexController med metod **__construct** initierar skapandet av objekt av motsvarande modell- och viewklasser. Method **index** aktiverar fetching alla produkter och rendering for header och footer.

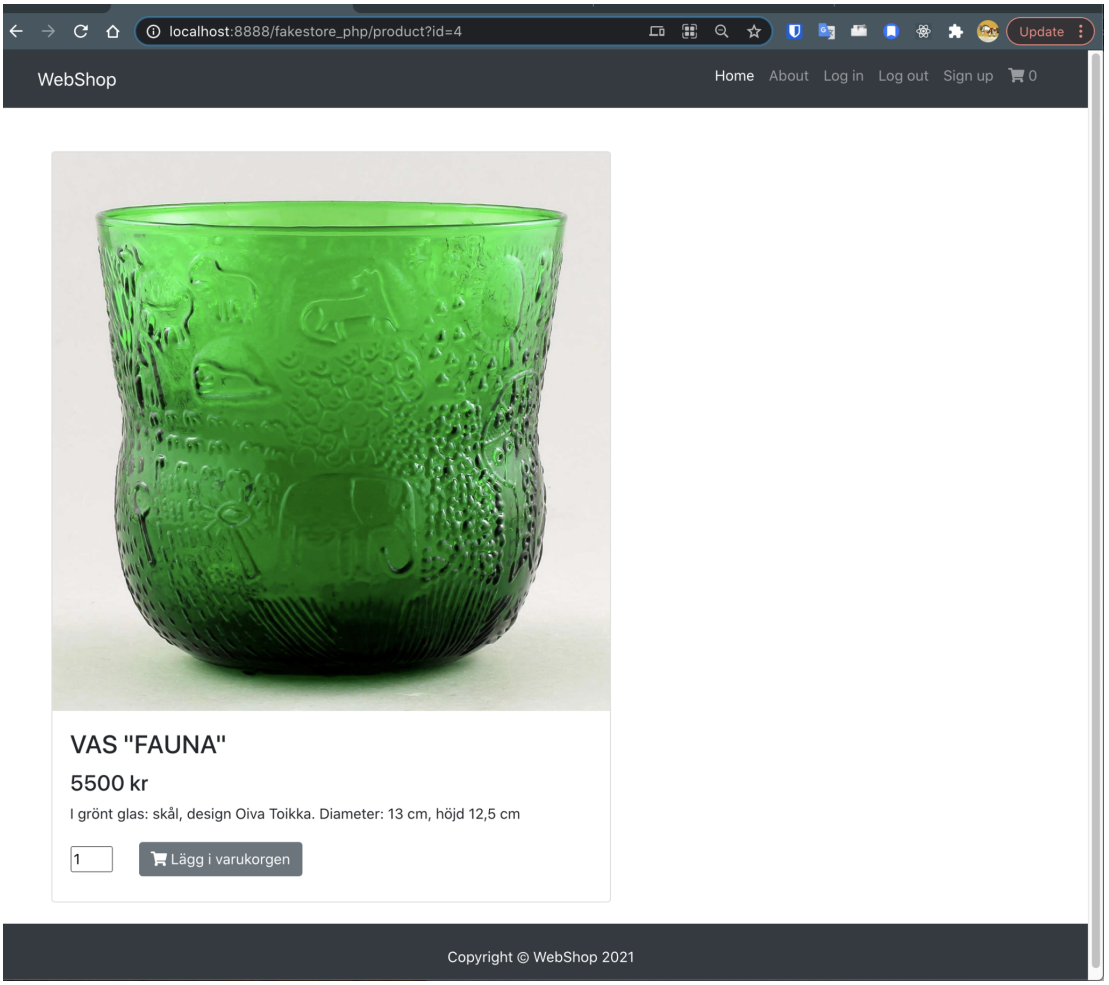
klass Model har metod **__construct** initierar skapandet av objekt av klass Database och metod **fetchAllProducts** läser och returnerar alla produkter från DB.

klass View har metod för addera rendering header och footer (includera filer från “include” mapp med HTML struktur), metod **viewAboutPage** som rendering sidan “Om oss”. Metod **viewAllProducts** som render data för varje produkt genom Bootstrap’s “product card”.

Product case som funkar för “one product’s” sidan - **ProductController.php**, **ProductModel.php** och **ProductView.php**.

Detta delen ansvarar för sidan för visning av en separat produkt, där användaren ser bilden, titeln, priset och beskrivningen av produkten. Här kan han lägga till en produkt i korgen genom att välja önskad mängd och klicka på knappen “lägga i varukorgen”.

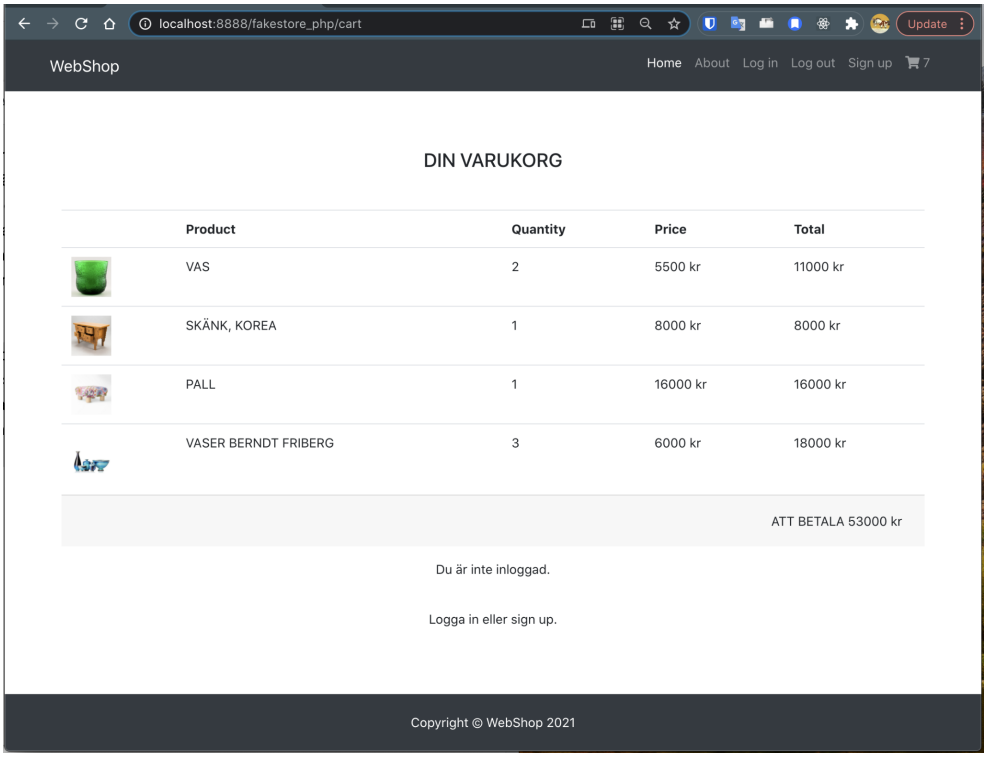
Därefter skapas en beställningssession, den skickade produkten visas i "korgen", som kan öppnas genom att klicka på korgikonen i navbaren, bredvid numret som visar antalet varor som redan har placerats där.



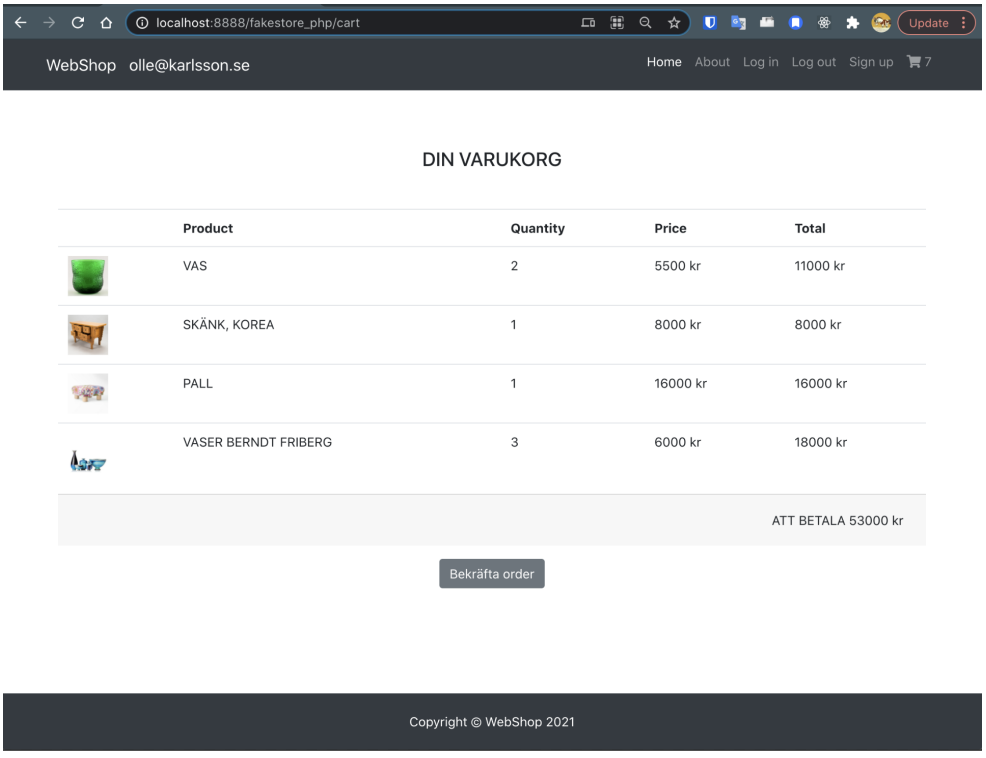
I **ProductController** finns (förutom **__construct** metod och **sanitize** hjälpfunktionen för produkt-ID) en **index** metod som bland annat skickar produktID till funktionen **fetchOneProductById**, som är en metod för **ProductModel-**klassen och hämtar produkten från databasen enligt passerat ID.

Case "cart": - CartModel, CartView, CartController.

Detta case är ansvarigt för användarens vagnvy, som visar de tidigare valda produkterna, deras kvantitet och priser. Dessutom beräknas det totala beloppet för ordern här. Knappen för att skicka en beställning blir endast aktiv om användaren är inloggad i systemet. “Email” sessionen skapas när användaren loggar in. “Cart” sessionen skapas när han lägger det första föremålet i kundvagnen.



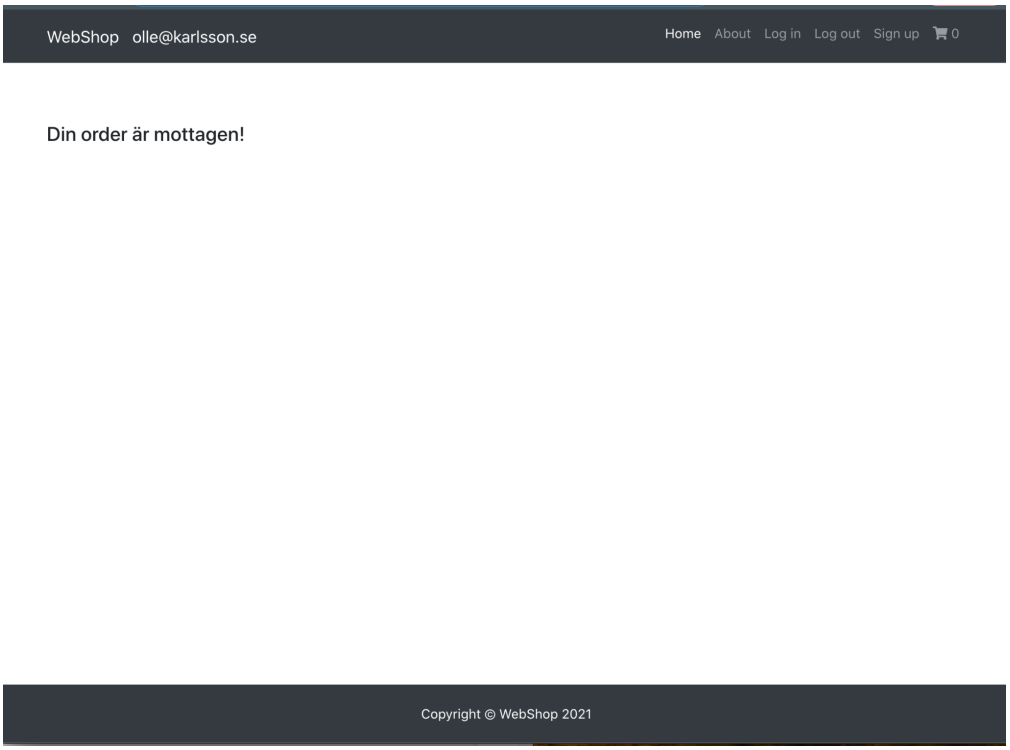
Om användaren är inloggad eller skapat ett nytt konto, då han kan, genom att klicka på knappen “Bekräfta order” på den här sidan, skicka en beställning, faktiskt göra ett köp, varefter orderdata skickas till databasen i tabellen "orders" med default "motagen" -status, order datum och customer-ID.



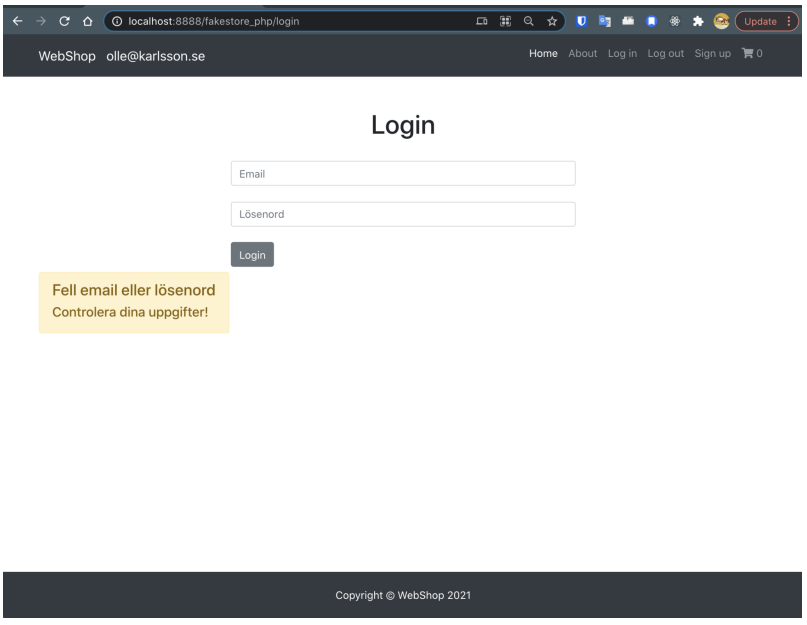
ProductControllers klass metod **index** rendera tom sidan om finns inte produkt i varukorg eller rendera cart genom viewCartPage metod av CartView klass.

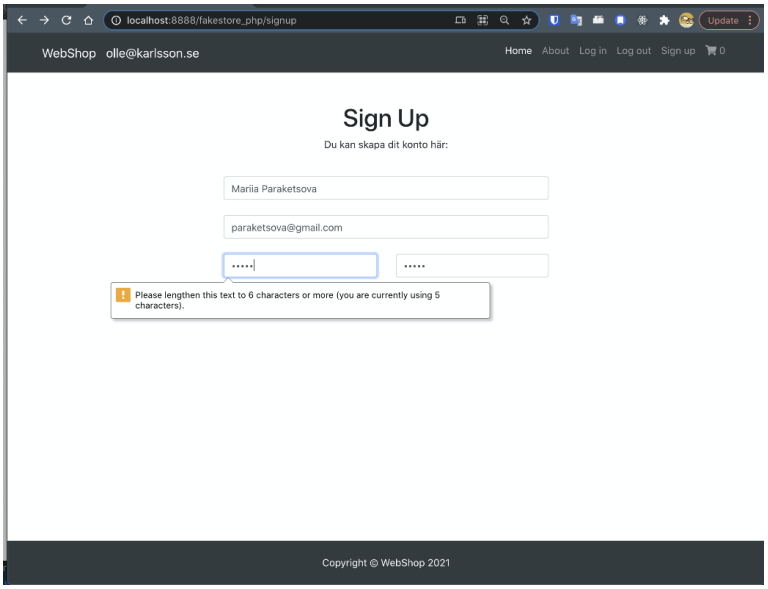
“Email” sessionen skapas när användaren loggar in. “Cart” sessionen skapas när han lägger det första produkt i kundvagnen.

Metod **add** addera produkt från produktsidan till varukorg. Metod **chekout** skickar data till DB om bestallning är skickas. Metod addOrder avslutar “cart”-sessionen och ger ett bekräftelsemeddelande för användaren (viewCheckoutPage metod CartView-klass).

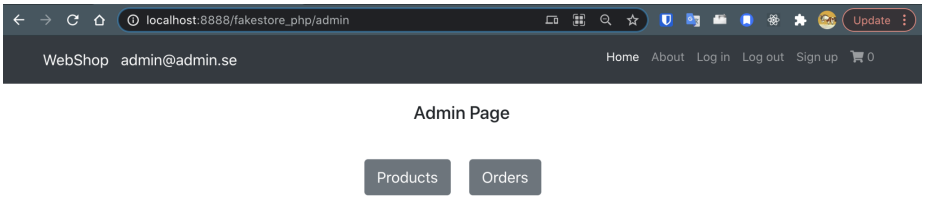


“Login”, “Signup” cases - löses på ett ganska standardiserat sätt. Klassmetoder läser och skickar data till / från databasen för en enskild användare. Det finns kontroller av unika e-postmeddelanden från databasanvändare och standardkontroller när du fyller i ett formulär (lösenordslängd, rätt upprepar för lösenord, förekomst av främmande tecken etc.

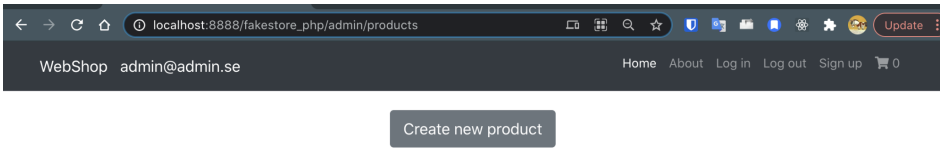




Admin case **-AdminModel, AdminView, AdminController.** Den case ger den inloggade administratören möjlighet att visa produkter och beställningar från databasen.

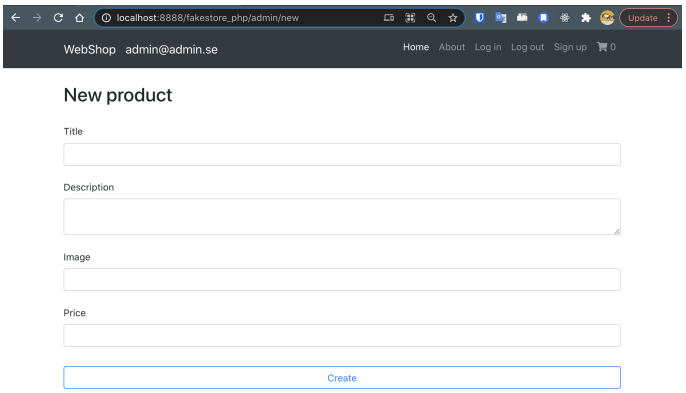


om admin väljer Produkten:

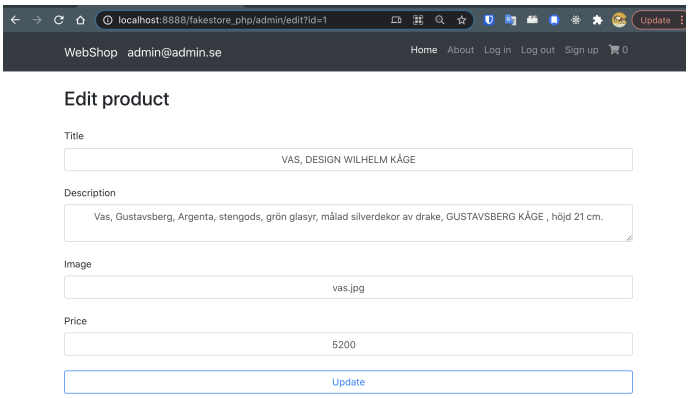


Title	Description	Image	Price		
VAS, DESIGN WILHELM KÅGE	Vas, Gustavsberg, Argenta, stengods, grön glasyr, målad silverdekor av drake, GUSTAVSBERG KÅGE , höjd 21 cm.	vas.jpg	5200	Edit	Delete
SKÄNK, KOREA	Skuren dekor, sex lådor med dragringar i mässing. Bredd 141 cm, djup 55 cm, höjd 85 cm.	chest.jpeg	8000	Edit	Delete
FÅTÖLJ "RO"	Jaime Hayon, fåtölj med fotpall, "Ro", Fritz Hansen, Danmark, 2018. Mörk yllekläsel. Lösa dynor. Ben av mörkbetsad ek.	ottoman.jpeg	15000	Edit	Delete
VAS "FAUNA"	I grönt glas: skål, design Oiva Toikka. Diameter: 13 cm, höjd 12,5 cm	vas-fauna.jpeg	5500	Edit	Delete
MATTA "FRANK NR 3"	Ett för Josef Frank mycket serent mönster och färgschema har matta nr 3 fått. Med de mustiga färgerna för den tankarna till mossbeklädd mark och till de klassiska färgerna i orientmattor. 120x140 cm, ull.	matta-frank.jpeg	43000	Edit	Delete
PARAPLYSTÄLL FORNASETTI	Ett dekorativt paraplyställ i lackerad plåt med Fornasettis karaktäristiska mönster Ombrelli e Bastoni. Den utsökta formen, färgerna och estetiken omvandlar med lätthet en regnig dag till en konstnärlig upplevelse. Samtliga Fornasettis kreationer, från möbler till inredningsdetaljer, tillverkas för hand i Italien.	paraplystall.jpeg	13400	Edit	Delete
PALL "FAMNA"	Pall Famna 2020 är formgiven av den Stockholmsbaserade design- och arkitekturstudion TAF, grundad av Gabriella Gustafson och Mattias Ståhlbom. 75X75X40 CM, bok ben.	pall_fanna.jpeg	16000	Edit	Delete
NATTLAMPA "BLOMBUKETT"	Bordslampa, "nattlampa", pâte-de-verre, Frankrike. En blombukett i rött, mörkgrönt och närmast svart glas, signerad i glasmassan G.	nattlampa.jpeg	16000	Edit	Delete

den här finns det möjligt att lägga till en ny produkt via formuläret:

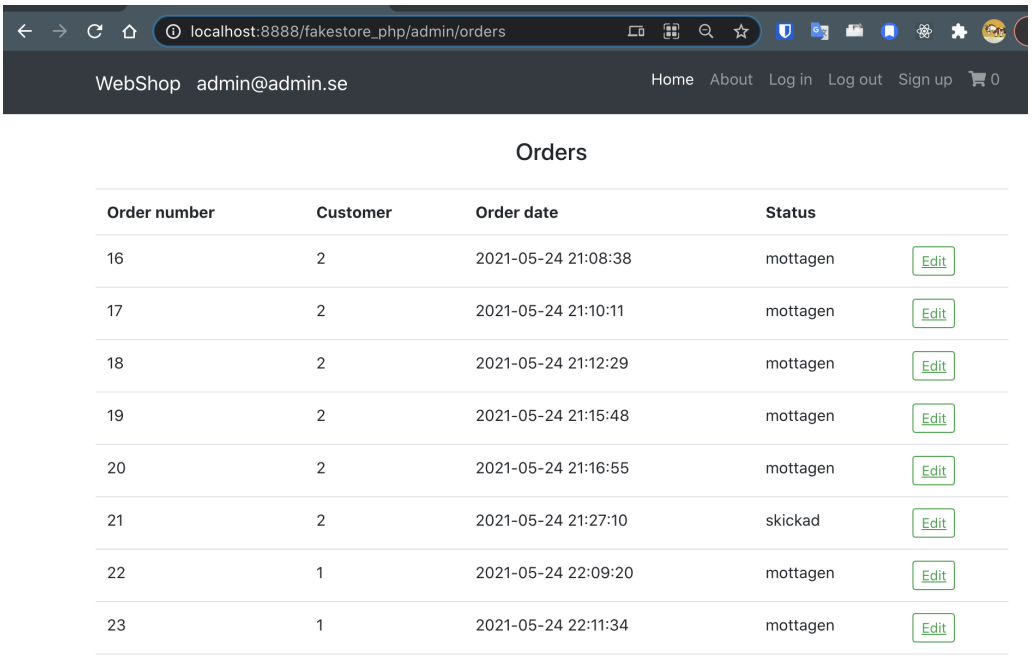


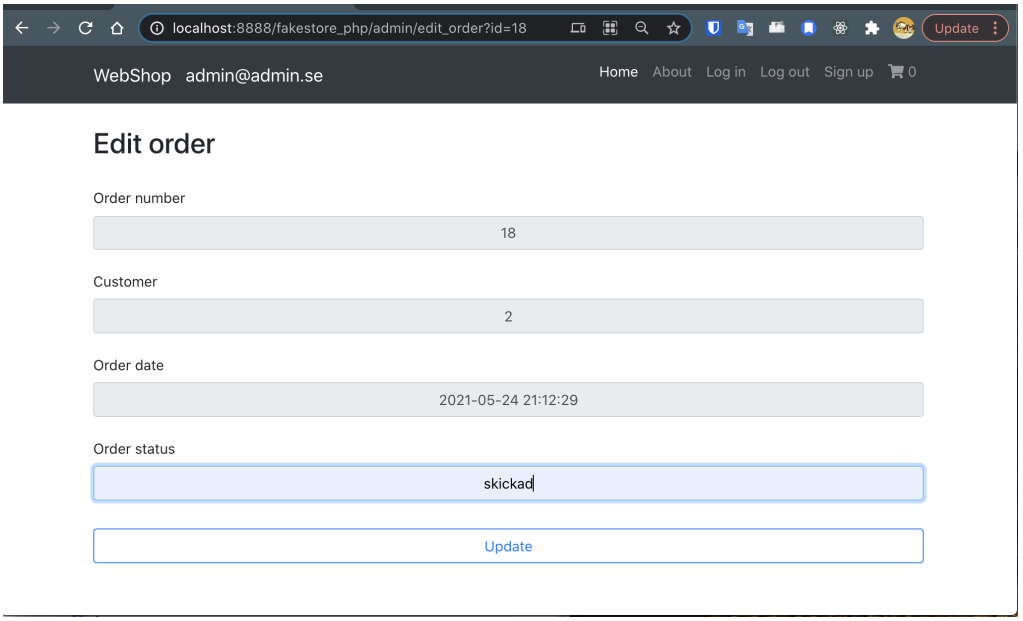
och kan också redigera befintliga produkter i databasen och radera dem:



Den fungerar tack vare metoder **create, update, delete** av klass **AdminController.** Och andra metoder som hämtar från DB och rendera resultat.

Liknande funktioner fungerar här för visar alla orders och redigeringsfunktionen för orderstatus:





Utvärdering

Generellt sett är jag nöjd med resultatet. Projektet uppfyller alla krav som ställs på det, det tar hänsyn till säkerhetsaspekter och har ett ganska trevligt och användarvänligt interface. Men vi hade verkligen inte tillräckligt med tid och ytterligare en teammedlem för att göra projektet så som vi skulle vilja se det.

Koden är långt ifrån perfekt, den innehåller repetitioner av små tilläggsfunktioner, själva projektets struktur kan göras enklare och tydligare genom att ställa in mer funktionalitet i indexfilen i fallkontrollen genom ytterligare variabler.

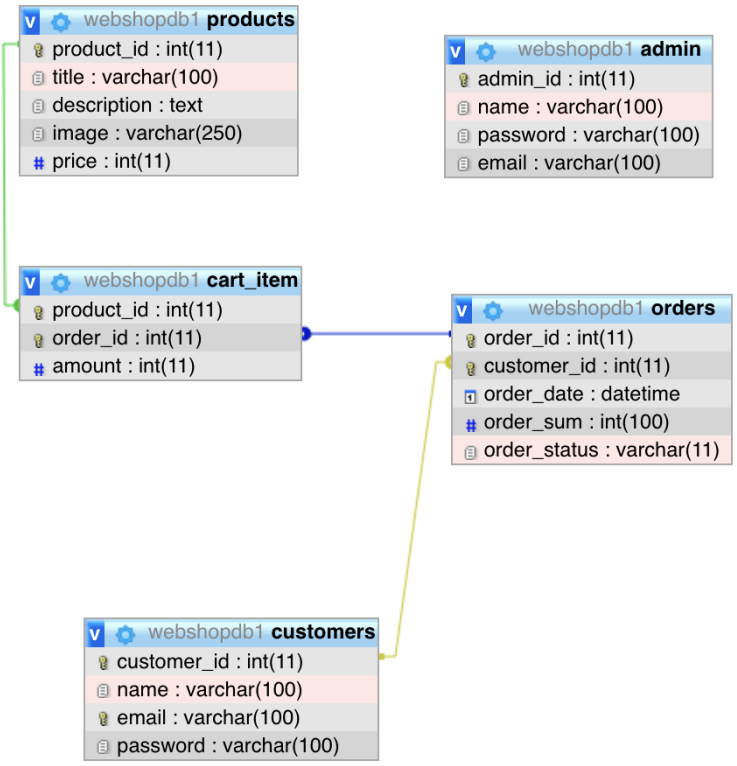
Projektet har några inte särskilt bra aspekter som jag skulle ändra:

- Innehållet i navigeringsfältet i rubriken beror inte på om användaren är inloggad och om användaren är administratör. Panelen innehåller alltid de "Log in, Sign Up" objekten som inte behöver ses av den inloggade användaren. Och för dem som inte är inloggade är “Log out” synligt.

- För att arbeta med adminpanelen finns det ingen separat knapp alls och administratören måste manuellt skriva till query sting "/admin".

- Jag skulle ha slutfört utseendet på admin-sidan, som visar information om beställningar. I stället för customer_ID måste vi naturligtvis visa customers namn/email, för detta måste vi hitta motsvarande data i tabellen customer_id och visa dem.

- Jag är inte så imponerad av gruppens beslut att spara orderdata (produkt-ID, antal produkter) som en JSON-fil. Därför ser databasstrukturen inte så bra. Jag skulle ändra strukturen i databasen och göra den så här:



Således skulle det för varje beställning finnas en koppling med flera rader i tabellen "cart_item". För detta skulle det naturligtvis vara nödvändigt att skriva om koden som är ansvarig för att skicka ordern till databasen och den del av koden som visar för admin sidan information om beställningarna.

Slutsatser

Bet var ett ganska stort och intressant projekt för mig.
Uppgiften verkade genomförbar och intressant, jag provade olika saker med nöje, försökte bygga klasser och experimenterade med PHP-funktionerna. Det är synd att det inte fanns tillräckligt med tid och energi för att göra projektet renare och genomföra alla våra idéer.