1.SEMESTERPROJEKT

University College Nordjylland,

Webudvikling

Sidetal: 6

Normalsidetal: 3.6

Anslag: 8757

Contents

1. Indledning

Denne rapport tager udgankgspunkt i at beskrive og præsentere hvilke overvejelser og teknologi valg der er blevet truffet, og med udgangspunkt i disse er der blevet udviklet et webshop.

2. Metode og teknologi valg

Dette afsnit omhandler hvilke metoder der har været tilgangelig, samt blevet undersøgt før et valg om, hvilken metode der skulle bruges til at lave databasen, serveren og frontenden.

2.1. Database

Dette afsnit tager udgangspunkt i at beskrive hvilke typer for databaser der blevet taget i betragtning før en beslutning om hvilken at bruge er blevet truffet.

2.1.1. Relationel

En relationel database er en samling af data med relation imellem dataet. Dataet bliver repræsenteret i tabeller, og hver tabel har kolonner og rækker som beskriver hvilken data tabellen indeholder.

Hvert kolonne kan indeholde en specifik type af data beskrevet ved oprettelsen af tabellen.

En række beskriver et samling af data som er relateret til hinanden og kan ses som værende et objekt.

Et eksempel vil være at tabellen hedder Person og den indeholder kolonnerne id, fornavn, efternavn og alder.

En række vil så repræsentere en Person hvor id, fornavn, efternavn og alder alle er relateret til hinanden.

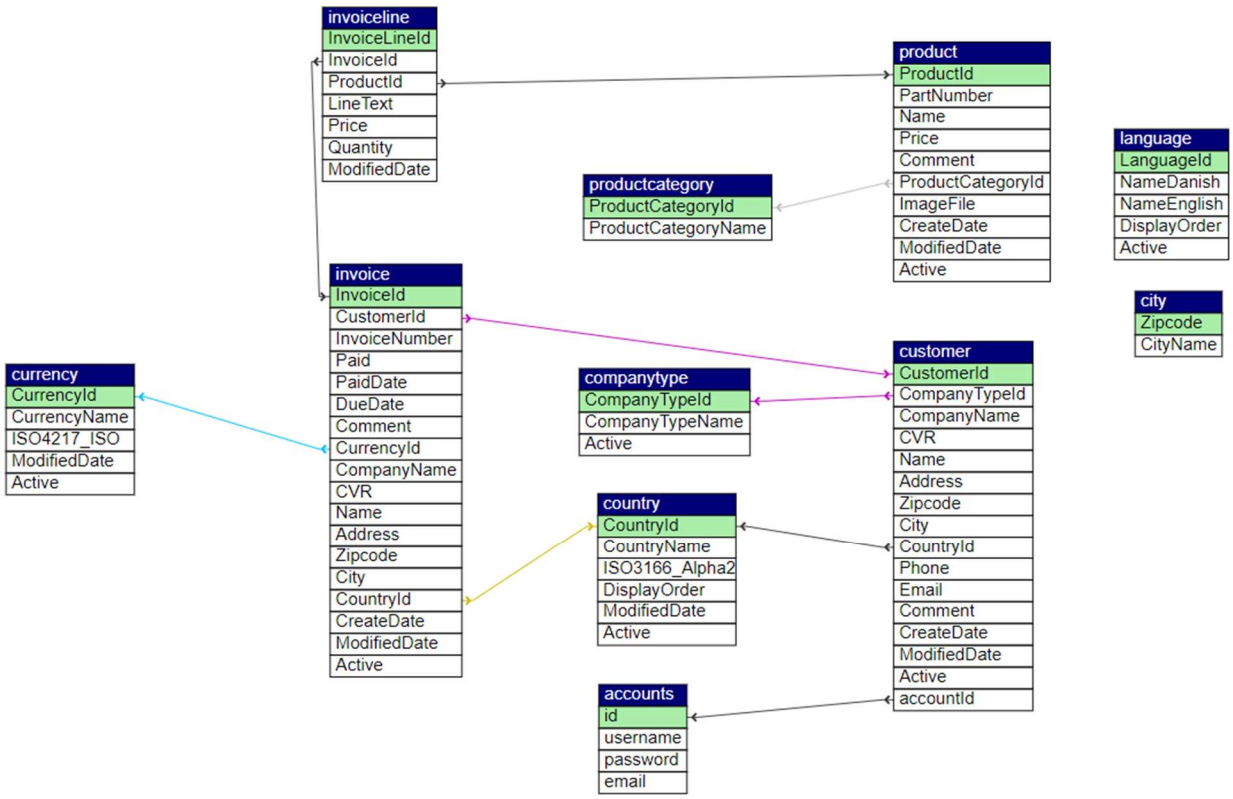
Med relationel databaser kan der også oprettes noget der hedder en primær nøgle. Denne nøgle bruges til at beskrive et specifikt objekt i tabellen. Den skal have en unik værdi som inkrementalt stiger når nyt data bliver importeret i tabellen.

Der kan også oprettes relationer mellem forskellige tabeller, det vil sige at hvis en Person har et kæledyr, så bliver relationen mellem tabellerne beskrevet ved brug af en fremmed nøgle.

Denne nøgle beskriver at der er en relationel mellem to tabeller. Relationen bestemmes af hvem der holder på nøglen. Hvis vi tager udgangspunkt i eksemplet med Person og kæledyr, så fordi at Person ejer en Kæledyr, så vil fremmed nøglen tilhøre Person. Nøglen beskrives bedst ved at sige at hvert eneste relation i databasen skal have en fremmed nøgle.

2.1.2. Valg

Med henblik på at skabe en mere gnidningsfri brug af en database i kombination med erfaring objekter, blev der valgt at gøre brug af en relationel database. Der var selvfølgelig NoSQL som et alternativt, men da dette kræver flere ressourcer at kunne tilegne viden blev det ikke valgt.



Dette er en relationel diagram over databasen som bliver brugt i forbindelse med udviklingen af web programmet. De kolonner som er markeret med grøn er primære nøgler og generelt set så skal alle tabeller indeholde en primære nøgle. De pile som kan ses pege på de markeret kolonner er relationer mellem tabellerne. Den kolonne som pilen stammer fra er en fremmed nøgle. Denne relation beskriver at en kunde(Customer) har en relation til firma type(companyType). I en tabel kan der være kun en primære nøgle, men for fremmed nøgle kan der være flere.

Til at lave relationel databasen er MySQL i forbindelse XAMMP blevet brugt. Selve scripts af tabellerne og databsen er lavet ved brug af phpmyadmim. Databaser lavet ved brug af XAMMP og phpmyadmin er lokale database og kan derfor kun eksistere på den enkelte computert.

2.2. Back-end

Dette afsnit tager udgangspunkt i at beskrive hvilke valg og alternativer der blevet i betragtning før et endeligt valgt blev taget om hvilken back-end der skal bruges i projektet.

2.2.1. Node.JS

Node JS er et back-end framework til at udvikle server side applikationer. Node er bruger JavaScript med henblik på at give udvikleren muligheden for at udvikle web API'er ved brug af JavaScript. I kombination med front-end JavaScript vil ved brug af Node JS skabe en "JavaScript everywhere" et

term som bliver brugt til at beskrive at alle lag i udviklingsprocessen bliver skrevet ved brug af JavaScript. I kombination med Node JS blev der gjort brug af ExpressJS for at skabe forbindelse til databasen og kunne udvikle sql statement og køre disse på databasen ved brug af JavaScript. Herunder bruges ExpressJS til skrive Routes og Services som skal kunne dirigere et url fra front-end til den rigtig metode som så gør en handling. Denne handling kan være at tilføje et nyt produkt, hente informationer om et produkt, opdatere informationer på et product, slette et produkt mm. Alt dette bliver gjort ved brug af JavaScript.

2.2.2. MVC Asp.NET

Et alternativt til Node JS til udvikling af API'et kunne være MVC Asp.NET. Dette er et web API udviklet i C# og er kendt for at have en stegl læringskurve. I princippet er det også muligt at udvikle både back-end og front-end i MVC Asp.NET. I så fald gør den brug af Razor på de views der bliver lavet på front-end. Razor bruges til at skrive C# kode direkte i html filen.

2.2.3. Valg

Med udgangspunkt i Node JS og MVC Asp.NET blev der trukket valget om at fortsætte med Node JS. Grunden til dette er at kunne forstå og lære at udvikle et web API udelukkende med JavaScript, da dette vil bruges i kombination med front-end som også vil blive skrevet i JavaScript. For at kunne skabe en bedre gnidningsfri process af dette, er Node JS blevet valgt. Selvfølgelig i andre tilfælde ville MVC Asp.NET være et bedre valg, da den i sig selv er meget godt skalarbar og nem at udvikle i såfremt udvikleren er lidt erfaren med frameworket. I forbindelse med Node JS API'et blev der også gjort brug af Postman for at sikre URL'en til et API kald fungerer som den skal før front-enden udvikles, da dette vil mindske risici for fejl.

2.3. Front-end

Dette afsnit vil tage udgangspunkt i hvilke teknologier der er på front-enden og hvilke beslutninger der draget. Disse beslutninger vil tage udgangspunkt i hvilke teknologier der findes indenfor HTML, CSS og JavaScript.

2.3.1. HTML

HTML står for HyperText Markup Language og er det "sprog" som bliver internettet forstår og kan processere for at vise indholdet i HTML filen. Da HTML er essentielt for internettet blev det, det naturlig valg at bruge. Men der findes også andre måder at skrive en HTML kode på. Dette er ved at gøre brug af HTML templates som introduceres af ExpressJS. Ved brug af ExpressJS er det muligt af vælge et HTML template engine som simplicere HTML'en og gør at det er muligt at lave JavaScripting i HTML'en.

På baggrund af disse blev der trukket et valg om at gøre brug af HTML filer på front-end uden brug af Template Engines. Grunden til dette er at skabe en fundamentalt viden om hvordan HTML skrives før at der tages fat på Template Engines.

2.3.2. CSS

Cascading Style Sheets er en teknik at tilføje layout og til HTML for at skabe en bedre visuelt layout af det udvikleren gerne vil have. Dette bliver anvendt for at separere hvad indholdet(HTML) skal være fra hvordan indholdet skal præsenteres(CSS). Herunder er der også forskellige mulige teknologier at tage fat på, men ligesom HTML er der behov for at skabe en bedre viden om hvordan CSS skrives før der gøres brug af andre teknologier. Responsive Design er en teknik indenfor CSS for at skabe et design som kan naturligt tilpasse sig enheden som siden bliver vist på.

2.3.3. JavaScript

JavaScript er et front-end fortolket programmeringssprog. Sproget bliver brugt i combination med HTML og CSS for at skabe en side. Det er ikke altid der er behov for JavaScript, men den er ofte brugt. Et fortolket sprog er et programmeringssprog som ikke compileres. Det vil sige at sproget på run-time bliver valideret og kørt når den skal bruges. JavaScript er også et dymanisk sprog, hvilket vil sige at den ikke har strikse typer som skal forholdes til når der oprettes data. Af denne grund er den også meget hurtig til at blive eksekveret.

JavaScript har mange frameworks som kan gøre det meget nemmere at udvikle web programmer. Herunder Laravel, Vue og React, samt indenfor mobiludvikling Angular, Ionic og ReactNative. Disse er alle frameworks som tager udgangspunkt i at udvikle ved brug af JavaScript. Det kræver dog også at udvikleren er vel bekendt med JavaScript for at kunne udvikle i disse. Mange af ovennævnte frameworks gør også brug af supersettet TypeScript som i bund og grund er JavaScript, men som er et stærkt typet sprog, i samme stil med C# og Java. TypeScript bliver på run-time transpileret ned til et JavaScript fil som så bliver kørt på siden.

I forbindelse med dette blev der trukket en beslutning om at skrive JavaScript uden nogle former for frameworks, da dette vil give et bedre udbytte i fremtiden når der engang skal bruges et framework, da udvikleren har en forståelse for hvordan JavaScript er at skrive kode i.

3. Konklusion

På baggrund af de valg der er blevet truffet er der blevet udviklet et program ved brug af et Node JS som server back-end API, samt HTML, CSS og JavaScript til front-end udvikling og MySQL som relationel database. Indenfor Node JS er der blevet brugt ExpressJS til at koble til MySQL. Databasen er blevet lavet i phpmyadmin i forbindelse med XAMPP.