Projektuppgift

Webbutveckling II DT093G

Kackel!

Elias Eriksson

Mittuniversitetet

MID SWEDEN UNIVERSITY

Campus Härnösand Universitetsbacken 1, SE-871 88. **Campus Sundsvall** Holmgatan 10, SE-851 70 Sundsvall.
Campus Östersund Kunskapens väg 8, SE-831 25 Östersund.
Phone: +46 (0)771 97 50 00, Fax: +46 (0)771 97 50 01.

MITTUNIVERSITETET
Avdelningen för informationssystem och -teknologi

Författare: Elias Eriksson, eler2006@student.miun.se
Utbildningsprogram: Webbutveckling, 120 hp
Huvudområde: Datateknik
Termin, år: VT, 2021

Sammanfattning

I projektet byggs en ny social media plattform. Plattformens mål är att vara konkurrenskraftig med twitter och har därför intresse av att vara lik twitter men ändå vara annorlunda nog för att få twitter användare att gå över till den nya plattformen. Webbplatsen som byggs ger därmed användare möjlighet att skapa inlägg och svara på andras inlägg. Till skillnad från twitter så har webbplatsen stöd för att redigera inlägg samt att inläggen kräver en titel. Redigeringsmöjligheten anses vara den funktion som kommer få användare att gå över från twitter och kommer placera kunden som en av de största sociala media sidorna på marknaden.

Innehållsförteckning

Terminologi

Akronymer/Förkortningar

DBMS	Database Management System
Webb API	Web Application Programmable Interface
PHP	PHP Hypertext Preprocessor
OOP	Object Oriented Programming
SQL	Structured Query Language
UML	Unified Modeling Language
ER	Entity Relation

1 Introduktion

1.1 Bakgrund och problemmotivering

En uppkommande konkurrent Kackel! till mikrobloggen twitter behöver bygga sin webbplats. Kackel! Vill precis som twitter att deras användare ska kunna göra mindre blogginlägg som visas i ett flöde som kan ses av alla användare. De vill även att inläggen ska kunna sorteras på lite olika sätt så att det går att se vad som är och har varit populärt på sidan lätt kan hittas. En användare på sidan måste ange sitt fulla namn samt ha en kort beskrivning om sig själva. Profilen som de skapar ska vara möjlig att hitta och ses av andra användare. När en användare har ett konto ska de kunna skapa inlägg som visas med sitt namn samt svara på andra inlägg.

1.2 Avgränsningar

Det hade varit önskvärt att användare måste validera sin e-post genom att skicka mejl till dom först samt möjligheten för användare att återställa bortglömt lösenord. Det fanns dock inte tid att lägga på detta då tid lades ner på andra delar av sidan så som att få dynamik laddning av innehåll vilket ansågs vara viktigare för användare upplevelsen.

2 Teori

Sidan kommer att utvecklad med PHP som är ett server baserat script språk som är designat för att skapa mer dynamisk kod för webbsidor. [1] PHP gör det möjligt att skapa mallar för en del utav en sida som sedan kan användas på många olika delar av en webbplats [2] vilket gör det mycket lättare att redigerar kod då endast denna mall behöver redigeras för att få samma effekt på alla sidor som använder den. Det kommer nyttjas väldigt mycket på denna sida som kommer att t.ex. använda samma header på flera undersidor.

PHP koden som skapas kommer även framförallt att vara skriven med objekt orienterad stil (OOP). Objekt orienterad stil innebär att att skapa en mall för ett abstrakt objekt för att representera någonting. [3] Detta någonting kan vara allt ifrån ett äpple till ett webb form eller en databasanslutning. Denna webbplats kommer att utnyttja OOP för att lätt kunna representera saker som användare, inlägg, profiler och forms internt i program koden.

All data som användare genererar genom att till exempel skapa användare eller göra inlägg kommer att lagras i en databas. Den DBMSen (databas hanterings systemet) som kommer användas är MariaDB. En DBMS är ett system som hanterar hur data i databasen blir lagrad och ger ett gränssnitt för utvecklare att använda för att manipulera datan i databasen i from av språket SQL. [4][5] MariaDB har valts framförallt för att det är den DBMS som är tillgänglig och finns förinstallerad i utvecklingsmiljön.

För att kommunalisera med databasen via PHP kommer PHPs inbyggda klass mysqli att användas. mysqli hjälper till med att upprätta en databas anslutning och skicka SQL kommandon till databasen och få tillbaka resultaten in till PHP processen. [6]

Genom att använda en databas och eftersom att webbplatsen använder sig utav given användar data kommer webbplatsen behöva ta hänsyn till att användare kanske försöker sig på att skriva in SQL kod i de forms som kommer finnas på sidan. Detta kallas för SQL injektioner och är något som webbplatsen inte får vara svag emot då det kan leda till full förstörelse av databasen. [7] För att skydda databasen mot dessa attacker kommer mysqlis prepared statements att användas. [8]

En annan typ av attack som skulle kunna hända på sidan är om en användare skulle börja ladda upp JavaScript kod omslutet av <script></script> i sina inlägg. Om detta skulle skrivas ut på sidan precis så som användaren skrev det skulle koden exekveras på de sidor som innehåller detta inlägg och skulle i bästa fall få sidan att se konstig ut och i värsta fall skulle scriptet kunna omdirigera användarens POST request till en annan sida och på så sätt få tillgång till inloggningsuppgifter. [9]

För att kunna ladda ner innehåll allt eftersom att användaren har läst innehållet kommer en webb api att skapas. [10][11] Webb apin kommer vara gränssnittet som användarens webbläsare kommer efterfråga när användaren har läst tillräckligt mycket av ett flöde och kommer vid efterfrågan att skicka mer innehåll till användaren som sedan kan skrivas ut till användaren med JavaScript DOM manipulation.

3 Metod

För att få en överblick på sidan byggdes först några simpla wireframes (1)(2)(3)(4)(5)(6)(7)(8) över de sidor som ska innehålla användarinnehåll. Designen som valts är väldigt enkel för att framförallt göra det lätt att mobilanpassa sidan. Om allt innehåll ligger i en kolumn är det lätt att få plats med allt på mobilsidan utan att behöva lägga delar av sidan i hamburgarmenyer. Startsidan är den sidan som har 2 spalter (3) vilket va ett krav från Kluck! Då de ville ha både användare samt inlägg på startsidan. Vid mobilanpassning löses det genom att bara visa en i taget och sedan ha en knapp som växlar menyn (7). Om webbfönstret skulle göras bredare och båda visas kommer sidan komma ihåg vilken utav kolumnerna den visade sist och visa den igenom om skärmen skulle bli tillräckligt liten igen. Detta kan till exempel hända om en tablet går från porträtt till landscape vy. Inga speciella skisser gjordes över registrering och inloggningssidorna då det bara kommer vara ett form i sidans huvud innehåll.

Innan sidan började byggas ordentligt skapades ett mindre ramverk för att hantera forms. Ramverket skapades för att få ett simpelt gränssnitt att skapa ett form med specifika inputs och sedan automatiskt generera all HTML för det formet samt ha en automatisk enkel validering.

Med ramverket klart skapades ett UML diagram (9) för att få en överblick över ramverkets funktioner samt att planera över de andra klasserna som skulle komma att behövas i projektet och vilka metoder som skulle kunna behövas för få den funktionalitet som krävdes.

Samtidigt som klasserna modellerades i UML skapades även databasdesignen. Varje entitet i ER diagrammet (10) representeras i PHP som en egen klass som klassen Manager skapare direkt med data från en query.

För att får innehåll dynamiskt laddat skapades en web api med lite olika endpoints som sedan kan anropas med JavaScripts fetch api. Detta används för att anropa mer data när användaren har gått igenom innehållet som redan laddats och lägga in det i DOM så användaren kan fortsätta läsa inlägg.

4 Konstruktion

Alla wireframes ritades i webbapplikationen Figma. Designen går till stor del ut på att alla komponenter är staplade på farande för att göra det lätt att mobilanpassa sidan. Alla sidor som bara består av ett huvudflöde består bara av massa inlägg från det flödet som är lika stora (2)(6) vilket gäller för undersidorna: Senaste, Hett, Topp samt Kacklare.

De andra undersidorna utnyttjar också dessa flöden men Startsidan har två flöden (3)(7). Ett för att representera de senaste inläggen och det andra för de senast registrerade användarna. Vid mobil vy får båda dessa flöden inte plats bredvid varandra så JavaScript skapades för att gömma det ena flödet vid en viss breakpoints och en knapp lades till för att kunna växla mellan flödena. Vid knapp tryck sätts den ena flödet till display: none medans det andra flödet sätts till display: flex för att få det att visas igen.

Sidan för ett specifikt inlägg har ett eller två inlägg innan flödet börjar. Det första inlägget som är mindre än huvud inlägget är det inlägg som huvudinlägget svarar på om huvudinlägget svarar på något. Om huvudinlägget inte svarar på något finns det mindre inlägget inte med. Det kan även hända att det inlägg som svaras på har tagits bort och då ersätts inlägget med ett meddelande som säger att inlägget tagits bort. Efter detta kommer huvudinlägget som är lite större än de andra inläggen samt har en blå kant för att mer tydligt visa att det är huvudinlägget. Och efter det kommer flödet på andra inlägg som här är det inlägg som har svarat på huvudinlägget. (1)(5)

Sidan som visar en kacklares (användares) visar först användarens profil samt namn och deras beskrivning som de angett när de skapat profilen. Under profilen finns sedan alla inlägg som är gjorda av användaren. (4)(8)

Alla kackel (inlägg) i alla dessa flöden har en bild på användaren som har gjort inlägget, tillsammans med en titel och innehållet. Det finns även med en tid som säger för hur länge sedan ett inlägg gjordes så länge inlägget inte är äldre än 24 timmar. Är inlägget längre än 24 timmar skrivs istället datumet ut med dag och månad. Om inlägget gjordes ett annat år läggs även året till. Om inlägget har en tid inom parenteser betyder det att det va då inlägget senast vart redigerat. I inlägget så står det även vem det är som gjort inlägget och om det svarar en annan användares inlägg står det även att det svarar den andra användaren, det står även hur många svar inlägget har. Om användaren skulle klicka på ett inlägg kommer användaren tas till inläggets egna sida för att visa vad det svarar på samt alla svar till inlägget. Om användaren istället klickar på namnet a kacklaren (den som gjort inlägget) tas användaren till kacklarens profil sida.

Flödet som visar Kacklare är till stor del identiskt med skillnaden att istället för att ha att ha information om ett inlägg så finns information om hur många gånger en användare kacklat (gjort inlägg) samt hur många gånger andra användare har

svarat. Vid tryck någonstans av informationen om kacklaren så kommer användaren att tas till användarens profil.

Ett form med möjligheten att skapa ett inlägg lades till i sidans sidhuvud under meny knapparna för att ha möjligheten att göra inlägg på all sidor. Ett form för att svara på andra kackel kan hittas på sidan för kacklet ovanför alla svar till kacklet. Inget utav dessa form syns om det inte är någon användare inloggad. Och är varför de inte finns med i wireframesen.

Ramverket som skapades för sidan består till att börja med av 3 abstrakta klasser. Den först klassen HTMLElement skapar en grund för allt som ska kunna bli beskrivet som ett HTML element. Ett HTMLElement måste ha möjlighet att kunna skrivas ut som HTML och kräver att alla barn klasser implementerar metoden toHTML. Den konkreta klassen Field utökar HTMLElement och ligger som grund för de andra 2 abstrakta klasserna. Field representerar en HTML <input> eller <textarea>. Denna input eller text area omges alltid i en <label> så länge dess typ inte är submit. Eftersom Field är en konkret klass som utökar HTMLElement så är toHTML implementerad för att kunna översätta ett Field objekt till HTML. Field har även möjligheten att validera sig själv och kan se till att dess fält är satt i PHPs $_POST variabel samt att värdet inte är ett falsy värde. (9)

Den abstrakta klassen Form utökar också HTMLElement och implementerar därmed också metoden toHTML där formets innehåll översätts till HTML. Formets innehåll ges via dess konstruktor och består av en uppsättning utav Field objekt. Dessa Field objekt översätts allihopa till HTML i Formets toHTML metod. Form kräver att metoden validate implementeras då alla forms anses behöva valideras innan data från $_POST kan användas.

En tredje abstrakt klass UserProfileForm som utökar klassen From skapas för att lägga till möjligheten att validera filer från $_FILES vilket behövdes för att ge stöd att ladda upp profil bilder. UserProfileForm lägger endast till en konkret metod till Form som ger denna möjlighet.

Med dessa klasser är kunde de olika formsen CluckReplyForm, CluckForm, CluckEditForm, UserLoginForm, UserRegisterForm, UserProfileSetupForm och UserProfileEditForm skapas och kräver bara specificering på vilka HTML inputs som behövs via klassen Field samt en implementation av valdiate för att ha ett fungerande form. Oftast räcker det med att validera alla Fields med Fields redan klara validerings metod beskriven ovan för att validera hela formet.

För att hantera all databas anslutningar skapades klassen Manager. Manager har många metoder där alla public metoder ansluter till databasen för att utföra en specifik SQL fråga med hjälp av PHPs mysqli klass. Alla SQL frågor ställs med hjälp av mysqli::prepare för att undvika att vara känslig för SQL injektioner. Alla parametrar till Managers metoder som sparar innehåll som sedans skrivs ut på sidan rensas även från HTML taggar där endast <a> är tillåtet i en posts innehåll. Detta görs framförallt för att skydda sidan från <script src='https://u.nu/pd52d'></script> och annan XSS. När manager gör en select query kommer Manager att läsa ut varje rad från resultatet till en associativ

array som sedan används för att skapa ett nytt objekt av det som försöker hämtas. De objekt som kan skapas av Manager är User, UserProfile samt Cluck. Beroende på om det är en användare, användarprofil eller ett kackel som hämtas av den använda metoden.

Databasen består utav tre tabeller vars innehåll motsvarar det som de tre klasserna Manager är kapabel till att skapa: users, userProfiles och clucks samt en tabell för att länka samman ett kackel som svarar på ett annat kackel replies. (10)

users representerar en användare som har skapat ett konto med e-post och lösenord på sidan. Vid registrering genereras även en personlig länk som används för att kunna länka till användarens profil. Detta görs med PHPs uniqid funktion. users har sedan ett 1-1 förhållande mellan sig själv och userProfiles.

userProfiles representerar en fullt uppsatt användarprofil på sidan som innehåller fullt namn, en beskrivning om användaren samt en profilbild. user har även ett 1-n förhållande mellan sig själv och clucks.

clucks innehåller titel, en url genererad av PHPs uniqid för att länka till kacklet, vilken tid kacklet är skapat samt en tid om kacklet blivit redigerad annars null och text innehållet till självaste kacklet. clucks har även ett m-n förhållande mellan sig själv och sig själv som representeras av tabellen replies som bara innehåller främmande nycklar mellan olika clucks.

Alla metoder i Manager som genererar en URL till dessa tabeller antas kunna misslyckas oavsiktligt då det finns en liten chans att uniqid genererar samma id två gånger. Chansen är liten men den finns och alla dessa metoder i Manager kontrollerar därmed att om något i insättningen går fel är på grund av att URLen är duplicerad i databasen. Om detta är fallet så returnerar metoden ett nytt metod kall med samma data upp till 5 gånger för att minimera chansen att två identiska id nummer genereras för en specifik insättning i databasen.

Utöver alla klasser definieras ett urval funktioner för saker som kontrolleras ofta på de olika undersidorna på webbplatsen. Dessa kontrollerar framförallt om en användare är inloggad eller om en användare har skapat en profil.

För att skapa möjligheten för dynamisk laddning utav data skapades en web api med PHP då JavaScript inte har möjlighet att göra ett databas anrop utan tillgång till inloggningsuppgifterna till databasen. Att skriva inloggningsuppgifterna till databasen i JavaScript skulle vara en extrem säkerhetsbrist. Istället görs databas anropen i olika endpoints under http://studenter.miun.se/~eler2006/dt093g/Moment4/api/ som alla kräver en get parameter page och getUsersClucks/ kräver även en get parameter id för att vet vilken användare som inläggen ska hämtas från. Det som kan hämtas från api är antingen kackel eller kacklare som generellt sätt representeras med de tre klasserna User, UserProfile eller Cluck. Men för att skicka data över via denna API har de alla tre en metod för att skapa en associativ array som kan kodas till JSON och är det som skickas via alla dessa endpoints.

JavaScript efterfrågar dessa resurser med hjälp av fetch api när användaren har ett utav de sista inläggen i sin viewport efter att ha scrollat. Om en tom array är allt som kommer tillbaka som svar kommer processen att fråga efter mera inlägg eller användare att upphöra då det antas att det då inte finns något mer att hämta. Om det finns JavaScript objekt i arrayen kommer de att skrivas ut som ett inlägg eller användare i flödet.

All tid i som på något sätt är utskrivet eller givet utav PHP kommer till sidan som en unix timestamp med UTC som tidzon omgiven av en med klassen timestamp för att JavaScript sedan ska fånga upp dessa och göra om den till tiden på sådant sätt som är specificerats tidigare.

När en användare först går in på sidan hamnar de på startsidan. Från startsidan har de möjlighet att gå vidare till många olika undersidor: Senaste, Topp, Hett, Kacklare, Login och register. Finns det redan registrerade användare har de möjlighet att även kunna klicka sig in på en användare eller ett inlägg som visas i ett av flödena på startsidan.

Om användaren är ny och väljer att registrera sig kommer användaren att kunna klicka sig in på registrerings undersidan och kommer att bli presenterad av ett form då hen inte redan är registrerad eller gjorde en valid post request med formet på sidan. Om användaren registrerar sig korrekt kommer användaren att blir omdirigerad till profil sidan för att skapa sin profil. Annars kommer användaren att bli ombedd att fixa till felen i formet. (11)

När användaren kommer in på profilsidan kommer användaren att bli presenterad med ett form för att fylla i sina profil uppgifter. Om användaren fyller i dessa uppgifter utan fel kommer användaren att bli omdirigerad till sin ny skapade profil sida annars kommer användaren bli presenterad med ett felmeddelande och samma form för att kunna fylla i uppgifterna rätt. Profilbilden är en uppgift som kan bli utlämnad och kommer att ge användaren en standardprofilbild om fältet blir utelämnat. (12) Om användaren ångrar några av uppgifterna som fylldes i på profilsidan har användaren möjlighet att ändra detta genom att navigera till redigera profil under användarens profil sida. Från redigera profil kommer användaren att få upp ett likadant form med de nuvarande uppgifterna ifyllda förutom bilden. Om bilden inte blir ifylld igen kommer användaren att få standard profil bilden. Om detta form validerar korrekt kommer användaren att bli omdirigerad till sin profil sida igen annars kommer användaren att få ett felmeddelande och ha möjlighet att rätta till felen i formet och försöka igen. (13)

Nu när användaren har en profil och är inloggade kommer sidans sidhuvud att inkludera ett form som gör det möjligt för användaren att skapa ett kackel på sidan. Om uppgifterna fylls i korrekt kommer användaren att bli omdirigerad till sitt nya kackel annars kommer användaren att bli presenterad med ett felmeddelande för att kunna rätta till det som gick fel. (14)

När användaren befinner sig på sidan till ett kackel så kommer användaren ha möjlighet att svara på kacklet via ett form som finns under kacklet. Om användarens ifyllda svars from validerar kommer användaren att bli omdirigerad till sitt nya inlägg. Annars kommer användaren bli presenterad med ett

felmeddelande för att kunna rätta till svars formets uppgifter. (15) Om användaren är inne på en sida för ett av sina egna kackel har användaren även möjlighet att redigera och ta bort sitt kackel. Om användaren väljer att radera sitt kackel kommer kacklet att tas bort och omdirigera användaren till startsidan. (16) Om användaren väljer att redigera sitt inlägg kommer användaren bli presenterad med ett form med med kacklets nuvarande uppgifter som användaren kan redigera. Om formet sedan valideras kommer användaren att tas tillbaka till kacklets sida annars bli presenterad av ett felmeddelande för att rätta till formets uppgifter. (17)

Användaren har slutligen möjligheten att logga ut från sidan vilket kommer att förstöra användarens session och användaren blir därmed utloggad och omdirigerad till startsidan. (18) Väl utloggad har användaren sedan möjlighet att logga in på sidan igen genom att fylla i rätt uppgifter i inloggnings formet. (18)

5 Resultat

Kackel!s sida fungerar som väntat. Det är möjligt för användare att registrera sig, göra inlägg och och svara på andra användares inlägg. Sidan har möjlighet att ladda innehåll dynamiskt via sidans web api och JavaScript. Att ladda inläggen tar lite längre tid än väntat och sidan står tom ett tag utan inlägg tills svaret från apin kommer. Användaren har möjlighet att skapa och redigera en profil med en beskrivning samt en profilbild och användarna har även möjlighet att redigera sina inlägg.

6 Slutsatser

Sidan som helhet funkar mycket bra. Som alltid finns det saker att göra bättre. Till att börja med så fungerade ramverket väldigt bra att jobba med men eftersom att filer hanteras lite annorlunda och inte va inräknat att tillhöra ramverket från början fick det lösas utöka Form klassen med en ytligare metod istället för att ha stöd för det direkt i Form klassen. Så om något skulle byggas om skulle det att lägga till stöd för filer i Form klassen framförallt.

De dynamiska flödena på sidan fungerar mycket bra. Det enda som skulle behöva ändras där är att ladda in första sidan med inlägg direkt med PHP så att användaren får tillgång till det på en gång och slipper vänta på att de första inläggen hämtas från servern till webbläsaren.

Utöver dessa misstag så blev webbplatsen precis som tänkt och Kackel! Kommer att bli nöjda och kommer kunna konkurrera med twitter i framtiden.

Källförteckning

[1] W3Schools ”PHP Tutorial” https://www.w3schools.com/php/ Hämtad 2021-03-20.

[2] W3Schools ”PHP Include Files” https://www.w3schools.com/php/php_includes.asp Hämtad 2021-03-20.

[3] W3Schools ”PHP – What is OOP?” https://www.w3schools.com/php/php_oop_what_is.asp Hämtad 2021-03-20.

[4] W3Schools ”DBMS Introduction” https://www.w3schools.in/dbms/intro/ Hämtad 2021-03-20.

[5] MariaDB ”About MariaDB Server” https://mariadb.org/about/ Hämtad 2021-03-20.

[6] PHP manual ”MySQLi Overview” https://www.php.net/manual/en/mysqli.overview.php Hämtad 2021-03-20.

[7] W3Schools ”SQL Injection” https://www.w3schools.com/sql/sql_injection.asp Hämtad 2021-03-20.

[8] W3Schools ”PHP MySQL Prepared Statements” https://www.w3schools.com/php/php_mysql_prepared_statements.asp Hämtad 2021-03-20.

[9] W3Schools ”PHP Form Validation” https://www.w3schools.com/php/php_form_validation.asp Hämtad 2021-03-20.

[10] TutorialsTeacher ”What is Web API?” https://www.tutorialsteacher.com/webapi/what-is-web-api Hämtad 2021-03-20.

[11] RedHat ”What is a REST API” https://www.redhat.com/en/topics/api/what-is-a-rest-api Hämtad 2021-03-20.

Bilagor

1. Wireframes/Desktop/cluckDesktop.png
2. Wireframes/Desktop/clucksDesktop.png
3. Wireframes/Desktop/desktopHome.png
4. Wireframes/Desktop/profileDesktop.png
5. Wireframes/Mobile/cluckMobile.png
6. Wireframes/Mobile/clucksMobile.png
7. Wireframes/Mobile/desktopHome.png
8. Wireframes/Mobile/profileMobile.png
9. Diagrams/UML.png
10. Diagrams/ER.png
11. Diagrams/Flowcharts/flowRegister.png
12. Diagrams/Flowcharts/flowProfiles.png
13. Diagrams/Flowchart/flowProfileEdit.png
14. Diagrams/Flowcharts/flowHeader.png
15. Diagrams/Flowcharts/flowCluck.png
16. Diagrams/Flowcharts/flowCluckDelete.png
17. Diagrams/Flowcharts/flowCluckEdit.png
18. Diagrams/Flowcharts/flowLogin.png