Labbrapport

Laboration 3: Ordersystem för Restaurang

Anders Ullström
940803
17 maj 2022

Introduktion

I denna laboration var syftet att skapa en applikation som hanterar beställningar av varor hos en pizzeria. Detta har inneburit att skapa ett grafiskt användargränssnitt. Denna laboration innehöll följande kravspecifikation:

- Servitören ska kunna registrera vilka maträtter ett bord önskar.
- Servitören ska lätt kunna se vilka eventuella allergier och annan information om varje rätt.
- Servitören ska dessutom kunna lägga till noteringar om specialbeställningar vid varje order.
- Programmet ska ha en knapp som visar ett kvitto, med priset för varje maträtt/dryck var för sig och även summan för hela bordet.
- Applikationen ska vara skriven med HTML (eventuellt Bootstrap) och Javascript.

Grafisk design

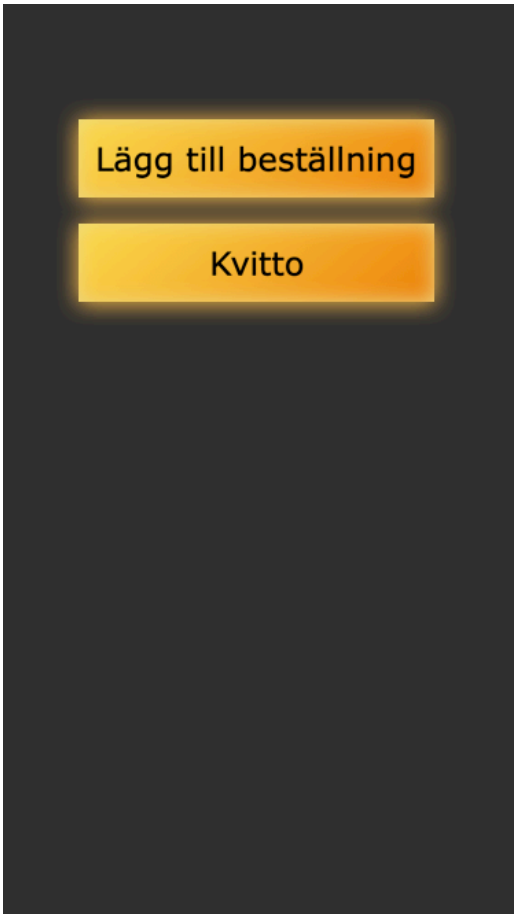
Den grafiska designen är baserad på att det ska vara lättanvänt, lättförstått och lättmanövrerat. Projektet började med att göra skisser för att få en ungefärlig uppskattning hur programmet skulle se ut (se Figur 1 och 2).
Den visuella belastningen försöktes minskas så mycket som möjligt genom att inte klottra ner användargränssnittet med onödig grafik. Målet var att det skulle se professionellt och städat ut. Inspiration togs ifrån liknande existerande applikationer såsom Foodora. Det var viktigt att programmet skulle vara snyggt rent visuellt och detta uppnåddes genom att ha ett snyggt färgschema på alla komponenter som passade

Figur 1: Första skiss på beställningssektionen.

bra tillsammans. Programmet är lent och gemytligt för ögat. Det är även anpassat för färgblinda.

Den kognitiva belastningen förebyggdes genom att göra en så pedagogisk meny som möjligt. En sak som gjordes var att använda så kallade ”collapse-menyer” i Bootstrap-biblioteket så att det skulle vara lätt att välja olika kategorier (se Figur 3). Det lades även ner tid på att minska antalet steg användaren behövde genomgå för att utföra en beställning. Tanken var att det skulle vara snabbt och effektivt att lägga en order.

Den motoriska belastningen minskades genom att sätta rimliga marginaler mellan knapparna och tydliga förklarande texter vad de olika knapparna gjorde. Likasom den kognitiva belastningen så lades tid ner på att minska antalet knapptryck och steg som användaren behövde genomgå för att utföra olika moment.



Figur 2: Första skiss av förstasidan på appen.

Ett koncept som har följt med under hela utvecklingen är Steve Krug’s citat ”Don’t make me think” och ”Get rid of half of the words on each page, then get rid of the half of what’s left” vilket sammanfattar applikationens grafiska design ganska bra.

Programdesign

Programdesignens byggstenar är en HTML-fil vid namn index.html, en Javascript-fil vid namn script.js och ännu en Javascript-fil vid namn labb3.js som har samma funktion som en JSON-fil skulle ha tillfört.

HTML-filens kärna består av Bootstrap-biblioteket för att undvika tidsödslande på att skapa en separat CSS-fil.

HTML-filen är egentligen bara ett statiskt tillstånd av applikationens visuella grundpelare. Den behöver någon slags logisk kontroll för att kunna manövreras. Detta uppnås med hjälp av Javascript-filen vid namn script.js. Denna fil ser till att knappar, menyer och andra element i HTML-filen går att använda till det syfte de blev skapade för.

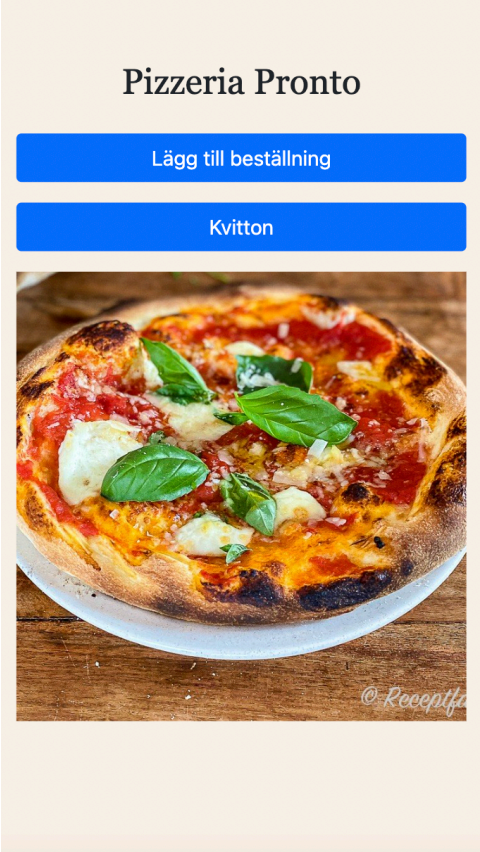
Figur 3: Slutgiltig produkt av beställningssidan i programmet.

Javascript-filen script.js har ett beroende av den andra Javascript-filen labb3.js för att kunna fungera korrekt. Filen script.js måste nämligen hämta den lagrade menyn från labb3.js. Utan denna meny med alla varor så skulle script.js-filen helt enkelt inte kunna fungera.

Denna programdesign är väldigt nära MVC (Model-View-Controller) på ett sätt. HTML-filen är den grafiska vyn av programmet, script.js är filen som kontrollerar programmet och uppdaterar dess logik och labb3.js är modellen som innehåller all essentiell data.

Implementation

Implementeringen påbörjades genom att skapa en HTML-fil och lägga in alla grundelement samt att länka in Bootstrap och script-filerna. JQuery länkades även in i HTML-filen för att kunna underlätta kodandet inom vissa moment i script.js.



Figur 4: Slutgiltig produkt av förstasidan i programmet.

HTML-filen:

HTML-filen består av en head och en body. Head används för att länka in olika bibliotek och för att sätta grundmallen för programmet. Body innehåller allting som är visuellt i applikationen när användaren tittar på skärmen. Den innehåller tre stycken huvudelement som är av typen div. Det första huvudelementet är för förstasidan av applikationen, det andra huvudelementet är för kvittosidan och det tredje är för beställningssidan (se kommentarer i HTML-filen). Med hjälp av Bootstrap så sätts klassen för beställningssidan och kvittosidan till d-none vid start av programmet så att dessa element blir osynliga tills användaren väljer att gå in på någon av dessa meny-knappar.

Filen script.js och labb3.js:

HTML-filen får ytterligare HTML-kod inläst när programmet startar ifrån script.js och labb3.js. Då läggs hela restaurangens meny till i ”collapse-menyn” på beställningssidan med hjälp av nästlade for-loopar i script.js. ”Collapse-menyn” består av 5 olika sektioner som representerar rubriken för varje huvudmeny i menyn. JQuery användes för mesta del för att lägga till denna HTML-kod ifrån

script.js som i sin tur hämtade meny-datan ifrån labb3.js (se rad 86-113 i script.js-filen). Den första for-loopen lägger alltså till titeln på varje sektion i "collapse-menyn" i ett li-element, den andra for-loopen lägger till alla varor som tillhör varje sektion i ett ytterligare li-element och den tredje for-loopen lägger till ingredienserna för varje enskild vara i samma li-element som skapades i andra for-loopen.

Sedan läggs ett nytt li-element till i varje existerande li-element som skapades i den andra for-loopen (de som innehåller alla separata varor). Detta li-element skapas med klassen d-none och ska endast bli synligt om användaren trycker på en vara. Då kommer li-elementets innehåll som består av en plus-knapp som lägger till den valda varan i varukorgen och ett textfält för att skriva särskilda önskemål ifrån kund att bli synligt. För att detta skulle kunna ske så behövdes det en lyssnare för varje enskild plusknapp (se rad 122-155 i script.js-filen). Det skapades även en meny uppe i högra hörnet av beställningsmenyn för att välja vilket bord ordern skulle läggas på. Detta fält fick lov att vara ifyllt med ett bord för att användaren skulle kunna lägga till en vara (se rad 140-151 i script.js-filen).

Figur 5: Slutgiltig produkt av kvittosidan i programmet.

Ytterligare lyssnare sätts på resterande knappar i programmet i script.js (se rad 4-71). De flesta av dessa knappar utför endast två uppgifter:

- Ta bort klassen d-none på sidan knappen leder till.
- Lägg till klassen d-none på sidan knappen blev tryckt på.

Det finns en varukorgs-knapp uppe i högra hörnet på beställningssidan som öppnar en modal som också finns skriven i HTML-filen (se rad 88-111 i

index.html). Denna knapp behöver ingen lyssnare i script-js för den finns inbyggd i Bootstrap när man skriver HTML-koden för en modal. Däremot när modalen öppnas så visas alla varor som har lagts till i varukorgen och där finns även olika knappar som kräver lyssnare. En av dessa knappar är för att skicka ett kvitto till kvittosidan med alla varor och det totala priset. Sedan finns det en minusknapp för varje enskild tillagd vara i varukorgen utifall att användaren skulle vilja ta bort en vara. Lyssnarna för dessa minusknappar har en hel del logik för att varan ska kunna tas bort ifrån varukorgen som bygger på att använda själva eventet där klicket skedde och sedan hämta ut parentNode och firstChild till elementet som blev klickat för att kunna tillslut träffa elementet som ska tas bort ifrån HTML-koden (se rad 233-276 i script.js-filen).

När beställningen har skickats med skicka-knappen i varukorgen så läggs kundvagnens innehåll in i kvittosidans div-element. Då när användaren går in på kvittosidan så kommer det att finnas en ny knapp med bordsnamnet som skickades ifrån varukorgen på beställningssidan (se Figur 5). Denna knapp skapades när skicka-knappen klickades (se rad 50-71). Denna knapp kontrollerar en ny "collapse-meny" som innehåller alla varor och den totala summan samt en knapp som ska skriva ut kvittot. När användaren klickat på valfri knapp med bordsnamn på så kommer alltså "collapse-menyn" att uppenbara sig under knappen som blev tryckt. Sedan, sist av allt, om användaren klickar på "Skriv ut kvitto"-knappen så kommer alla förfäder till den knappen att raderas ända upp till bordsnamnets-element. Detta uppnåddes återigen genom att använda event.target och hämta ut parent, grandparent och så vidare (se rad 64-71 i script.js).

Problem

Denna laboration har flutit på ganska bra, men några enstaka problem har uppstått som har löst ganska fort.
Ett problem var att Bootstrap-ikonerna man lade in med lyssnare på kunde ha olika event.target beroende på vart på ikonen man klickade. Det uppenbarade sig snabbt att det var på grund av att ikonerna hade lagts inuti en knapp och knappen var elementet som hade själva lyssnaren. Det innebar att ikonen även fick samma lyssnare men event.target blev inte densamma vilket ställde till problem när man ville radera en specifik förälder till exempel. Detta löstes med if-satser av två olika slag. En if-sats för button-elementet skapades och en if-sats för i-elementet där ikonen låg skapades (se rad 195 och 278 i script.js exempelvis). Dessa två if-satser hade nästan identisk kod förutom att i-elementet gick ett steg högre upp i släktträdet för att träffa rätt element att ta bort/lägga till.

Slutsats

Denna laboration har varit underhållande att hålla på med. Det märktes hur användbart JQuery är med tanke på att det går att konkatenera extremt långa strängar med varandra och lägga in det direkt i HTML-koden. Detta sparade många rader med kod. Det var lärorikt att få använda lite Bootstrap också eftersom det är något som används mycket ute i arbetslivet och det sparar enormt mycket tid under programmets grafiska designprocess. I det stora hela så har denna laboration varit väldigt bra och detta är något som kommer att vara användbart i framtiden att ha koll på.

Referenser

- www.stackoverflow.com
- https://www.extensis.com/blog/how-to-design-for-color-blindness *För att skapa ett färgschema anpassat för färgblinda.*
- www.google.com
- Krug, S. (2014). *Don't Make Me Think: A Common Sense Approach to Web Usability. 3 uppl.,* Förlag: New Riders Publishing
- Karlstads Universitet. (2022). *Föreläsning 10: Webb och HTML.* https://kau.instructure.com/courses/13875/pages/forelasning-10-web-och-html?module_item_id=386745 Kurs: Grafiska Användargränssnitt
- Karlstads Universitet. (2022). *Föreläsning 11: Bootstrap och Responsiv Design* https://kau.instructure.com/courses/13875/modules/items/390970 Kurs: Grafiska Användargränssnitt
- Karlstads Universitet. (2022). *Föreläsning 12: Javascript och JQuery* https://kau.instructure.com/courses/13875/modules/items/386746 Kurs: Grafiska Användargränssnitt
- Karlstads Universitet. (2022). *Föreläsning 13: Förberedelse Labb 3 och Uppgift 2* https://kau.instructure.com/courses/13875/modules/items/386747 Kurs: Grafiska Användargränssnitt
- Karlstads Universitet. (2022). *Föreläsning 14: Repetition* https://kau.instructure.com/courses/13875/modules/items/390972 Kurs: Grafiska Användargränssnitt