**Projektuppgift**

*DT207G – Backend-baserad webbutveckling*

**Moment 5**

Webbplats + CMS för restaurang Benvenuto Amico

**Emma Lorensson**

**MITTUNIVERSITETET  
Avdelningen för informationssystem och -teknologi**

**Författare:** Emma Lorensson, [emlo2302@student.miun.se](mailto:emlo2302@student.miun.se)  
**Utbildningsprogram:** Webbutveckling, 120 hp  
**Huvudområde:** Datateknik  
**Termin, år:** VT, 2024

# Sammanfattning

Projektet gick ut på att skapa en REST-webbtjänst med en mongoDB-databas för datalagring som hanterar restaurangen Benvenuto Amicos menyer, bordsbokning och användarinformation. En webbsida skapades sedan som konsumerar REST-webbtjänsten och möjliggör inloggning till ett administrationsgränssnitt, ett s.k headless CMS, där personalen kan logga in och redigera de olika menyerna och hantera bordsbokning.

Kravet för godkänt var att ha en meny i webbtjänsten som dynamiskt skrivs ut på webbplatsen och som personalen kan redigera. Jag har tre olika menyer och möjlighet för användare att boka bord. Personalen kan också redigera samtliga tre menyer och hantera bordsbokningen i administrationsgränssnittet.

Innehållsförteckning

[Sammanfattning iii](#_Toc167972692)

[Terminologi vi](#_Toc167972693)

[**Definition av termer och förkortningar** vi](#_Toc167972694)

[1 Introduktion 1](#_Toc167972695)

[1.1 Bakgrund och problemmotivering 1](#_Toc167972696)

[1.2 Övergripande syfte 1](#_Toc167972697)

[1.3 Konkreta och verifierbara mål 2](#_Toc167972698)

[2 Teori 3](#_Toc167972699)

[2.1 REST-webbtjänster 3](#_Toc167972700)

[2.2 NPM 3](#_Toc167972701)

[2.2.1 Mongoose 3](#_Toc167972702)

[2.2.2 Express 3](#_Toc167972703)

[2.2.3 Dotenv 3](#_Toc167972704)

[2.2.4 CORS 4](#_Toc167972705)

[2.2.5 Body-Parser 4](#_Toc167972706)

[2.2.6 Jsonwebtoken 4](#_Toc167972707)

[2.2.7 Nodemon 4](#_Toc167972708)

[3 Metod 5](#_Toc167972709)

[3.1 Moodboard 5](#_Toc167972710)

[3.2 Designskisser 5](#_Toc167972711)

[3.3 Utvecklingsmiljö och verktyg 5](#_Toc167972712)

[3.4 Webbtjänst 6](#_Toc167972713)

[3.5 Validering och tillgänglighet 6](#_Toc167972714)

[3.5.1 Validering av kod 6](#_Toc167972715)

[3.5.2 Validering enligt WCAG 2.0 6](#_Toc167972716)

[3.5.3 Prestanda 6](#_Toc167972717)

[4 Konstruktion 7](#_Toc167972718)

[4.1 Skapande av Moodboard och logotyp 7](#_Toc167972719)

[4.2 Skapande av designskisser 7](#_Toc167972720)

[4.3 Planering av webbtjänst 8](#_Toc167972721)

[4.4 Utveckling av webbtjänst 9](#_Toc167972722)

[4.4.1 Server 9](#_Toc167972723)

[4.4.2 Modeller 10](#_Toc167972724)

[4.4.3 Routes 10](#_Toc167972725)

[4.5 Utveckling av webbapplikation 11](#_Toc167972726)

[4.5.1 Struktur 11](#_Toc167972727)

[4.5.2 Responsivitet 11](#_Toc167972728)

[4.5.3 Globala element 12](#_Toc167972729)

[4.5.4 Startsida 12](#_Toc167972730)

[4.5.5 Om oss 14](#_Toc167972731)

[4.5.6 Menyer 14](#_Toc167972732)

[4.5.7 Administrationsgränssnitt 16](#_Toc167972733)

[5 Resultat 19](#_Toc167972734)

[6 Slutsatser 20](#_Toc167972735)

# Terminologi

Nedan presenteras de akronymer som används i rapporten.

|  |  |
| --- | --- |
| **Definition av termer och förkortningar** | |
| NPM | Node Package Manager |
|  |  |
|  |  |

# Introduktion

I den här rapporten kommer jag att gå igenom och motivera de tillvägagångssätt samt kod- och designval som använts för att skapa en webbapplikation åt den fiktiva restaurangen Benvenuto Amico Trattoria. I detta första kapitel behandlas *bakgrund och problemmotivering, syfte, avgränsningar* samt *konkreta och verifierbara mål.*

## Bakgrund och problemmotivering

Benvenuto Amico Trattoria är en finare italiensk restaurang belägen i en medelstor stad i Sverige. Många i staden anser att det är den bästa restaurangen som staden har och den är därför välbesökt. Då restaurangen funnits i många år och är väletablerad har den en befintlig webbplats idag, men restaurangens ägare är varken nöjda med utformningen eller funktionaliteten på webbplatsen. Dels tycker de att utformningen känns utdaterad, dels saknar de att kunna uppdatera sina menyer själva, då de i dagsläget behöver kontakta sin webbyrå inför varje menyuppdatering. Eftersom Benvenuto Amico serverar lunch blir det minst en uppdatering i veckan av lunchmenyn och därtill byter de emellanåt ut rätter i såväl middags- som takeaway-menyn. För att slippa kontakta webbyrån och betala pengar varje gång de behöver uppdatera menyn önskar de nu ett administrationsgränssnitt kopplat till webbplatsen, där de själva kan gå in och uppdatera utvalda delar av webbplatsens innehåll.

Benvenuto Amico har en väletablerad grafisk profil och det är viktigt för restaurangens ägare att den grafiska profilen bibehålls och genomsyrar den nya webbplatsen. Detta inkluderar framför allt färger och typsnitt.

Webbplatsen ska vara simpel att använda och det ska vara tydligt hur användaren bokar bord samt hittar restaurangens menyer. Den behöver vara väloptimerad och ladda snabbt så att potentiella gäster inte lämnar sidan.

## Övergripande syfte

Projektets övergripande syfte är att skapa en ny webbplats åt den fiktiva restaurangen Benvenuto Amica Trattoria. Webbplatsen ska representera restaurangen på ett tydligt och simpelt sätt, i enlighet med deras befintliga grafiska profil. Gäster ska kunna boka bord på webbplatsen. Webbplatsen ska också innehålla ett administrationsgränssnitt så att restaurangens medarbetare själva kan uppdatera utvalda delar av innehållet på webbplatsen.

## Konkreta och verifierbara mål

Projektets mål är att nedan punkter ska vara uppfyllda vid färdigställande:

1. Designskisser ska presenteras i enlighet med Benvenuto Amico Trattorias grafiska profil.
2. Det ska vara enkelt att boka bord och hitta restaurangens tre menyer via webbplatsen.
3. Ett lättanvänt administrationsgränssnitt ska skapas där restaurangens medarbetare ska kunna uppdatera de tre olika menyerna, samt hantera inkommande bordsbokningar.
4. Webbplatsen ska validera enligt standarden för w3c och WCAG 2.0 för att säkerställa en god tillgänglighet.

# Teori

## REST-webbtjänster

En webbtjänst är en typ av programvara som möjliggör kommunikation mellan olika system över internet eller ett nätverk. Enkelt förklarat används webbtjänster som mellanhand för att kommunicera mellan webbapplikationer och databaser [1, s. 4-7].

REST eller RESTful står för Representational State Transfer som är ett begrepp som förklarar hur kommunikation mellan system kan tillhandahållas via webben. En REST-webbtjänst använder http-metoder för att överföra data mellan en databas och en webbapplikation. De http-metoder som används är POST för att lägga till data, PUT för att redigera befintlig data, GET för att hämta data och DELETE för att radera data. Dessa motsvarar CRUD-operationerna INSERT, UPDATE, SELECT och DELETE [1, s. 13-15].

## NPM

NPM är världens största mjukvaruregister som tillhandahåller över 800 000 kodpaket, även kallat NPM-paket för såväl backend som frontend-applikationer skrivna i JavaScript. Samtliga kodpaket som tillhandahålls i NPM-registret har öppen källkod och är gratis att använda [2].

För att starta upp ett NPM-projekt och installera de paket som ska användas i projektet anges kommandot npm init i kommandotolken eller annan CLI. Därefter installeras de NPM-paket som ska användas med kommandot npm install ”npm-paket” [2].

### Mongoose

Mongoose är ett NPM-paket för Node.js som erbjuder en schemabaserad lösning för att modellera data i Node.js-applikationer som använder MongoDB som databas. Schemana gör det enklare att upprätthålla en tydlig struktur på datan som lagras i databasen [3].

### Express

Express.js är ett ramverk för Node.js-applikationer som tillhandahåller funktioner för att bygga webbapplikationer. Express.js förenklar bland annat för utvecklare att skapa routes och integrera med tredjepartstjänster som databaser [4].

### Dotenv

Dotenv är ett NPM-paket som möjliggör användandet av variabler från .env-filer i JavaScript-applikationer [5]. Det är fördelaktigt att använda .env-filer för att lagra till exempel [url:er](file:///C:\Users\Emma\kurser\DT207G\projekt\Rapport\er) eller nycklar till databaser, eller annan känslig information då du vid import av .env-filen sedan kan använda dessa variabler istället för att ange den känsliga informationen direkt i javascript-filen så att det är öppet för alla som granskar koden att se.

### CORS

CORS står för Cross-Origin Resource Sharing och används för att skydda och begränsa åtkomsten till en http-resurs, till exempel en webbtjänst. Som standard blockeras korsdomänförfrågningar i webbläsaren av CORS, men det går att styra hur strikt åtkomsten mellan olika domäner ska vara [1, s. 22].

NPM-paketet CORS kan installeras i ett Express-projekt för att möjliggöra styrning av CORS inställningar. Genom att använda CORS som middleware i en webbtjänst kan utvecklaren styra vilka specifika domäner som ska ha tillgång till webbtjänsten, eller om den ska vara öppen för alla domäner [6].

### Body-Parser

Body-Parser är en middleware för Node.js-applikationer som används för att tolka inkommande förfrågningar. Tack vare detta kan man enkelt extrahera data från till exempel webbplatsformulär och lagra den i req.body. Det finns dock ingen inbyggd validering i body-parser och informationen som skickas från användare behöver valideras manuellt innan den används för att undvika problem i applikationen [7].

### Jsonwebtoken

JSON web tokens, nedan förkortat JWT, används för att på ett säkert sätt överföra information mellan system som ett JSON-objekt. JWT används ofta för autentisering av användare som ska ha tillgång till exempelvis privata delar av en webbplats. Genom att signera tokens med ett nyckelpar för att säkerställa att informationen som skickats inte har manipulerats. JWT kan överföras via [URL:en](file:///C:\Users\Emma\kurser\DT207G\projekt\Rapport\en) till en webbplats, POST-parametrar i till exempel ett formulär eller http-headers [8].

### Nodemon

Nodemon är ett NPM-paket för node.js som underlättar utvecklingen av en server genom att övervaka ändringar i koden och då starta om servern per automatik [9].

# Metod

I detta kapitel beskrivs vilka metoder som kommer att användas för att uppfylla de mål som angetts i kap. 1 Introduktion.

## Moodboard

Arbetet inleds med att ta fram en tydlig moodboard som återspeglar den grafiska profil Benvenuto Amico ska ha. Moodboarden visar vilka färger, typsnitt och teckenstorlekar som är tänkta att användas på webbsidan.

## Designskisser

När moodboarden är framtagen och klar tas enklare designskisser fram som visar hur ett par av webbplatsens sidor ska se ut och fungera. Detta görs för att underlätta konstruktionen av webbplatsen i ett senare skede.

## Utvecklingsmiljö och verktyg

De system som används för att utveckla webbutiken är följande:

* **Visual Studio Code**Den kodeditor som används är Visual Studio Code – i den utvecklas all HTML, CSS och JavaScript-kod.
* **DRAWIO**Drawio används som ett tillägg i Visual Studio Code för att skapa ett simpelt diagram över de tabeller som databasen ska innehålla.
* **Adobe Premiere Pro**Adobe Premiere Pro används för att klippa samman de videoklipp som används i filmen på startsidan.
* **Adobe Illustrator**Adobe Illustrator används för att ta fram logotyp och favicon till webbplatsen.
* **MongoDB och Atlas**Databasen som lagrar datan för webbtjänsten är skapad i MongoDB och MongoDB:s användargränssnitt Atlas.
* **Render**Webbtjänsten är publicerad hos Render.
* **Figma**Figma användes för att ta fram moodboard och designskiss.
* **Visual Studio Code**Den kodeditor som används är Visual Studio Code – i den utvecklas all HTML, CSS och JavaScript-kod.
* **Node.js och NPM**Flertalet NPM-paket installeras i projektet för att underlätta utvecklingsarbetet.
* **GitHub**Git och GitHub används för att versionshantera sidan.
* **Netlify**Webbapplikationen är publicerad hos Netlify.

## Webbtjänst

När designskisserna är klara skapas ett simpelt diagram för att planera de olika tabeller som ska användas i databasen till webbtjänsten. Därefter skapas och konfigureras databasen i Atlas innan den kopplas ihop med webbtjänsten.

## Validering och tillgänglighet

### Validering av kod

För att säkerställa att webbplatsens kod är skriven på ett tillfredsställande och välfungerande sätt så kommer validering av samtliga HTML- och CSS-filer utföras med W3C Web Validator i webbläsaren.

### Validering enligt WCAG 2.0

Webbplatsen kommer att valideras enligt WCAG 2.0 med hjälp av AChecker Web Accessibility Checker för att säkerställa en inkluderande webbplats med god tillgänglighet.

### Prestanda

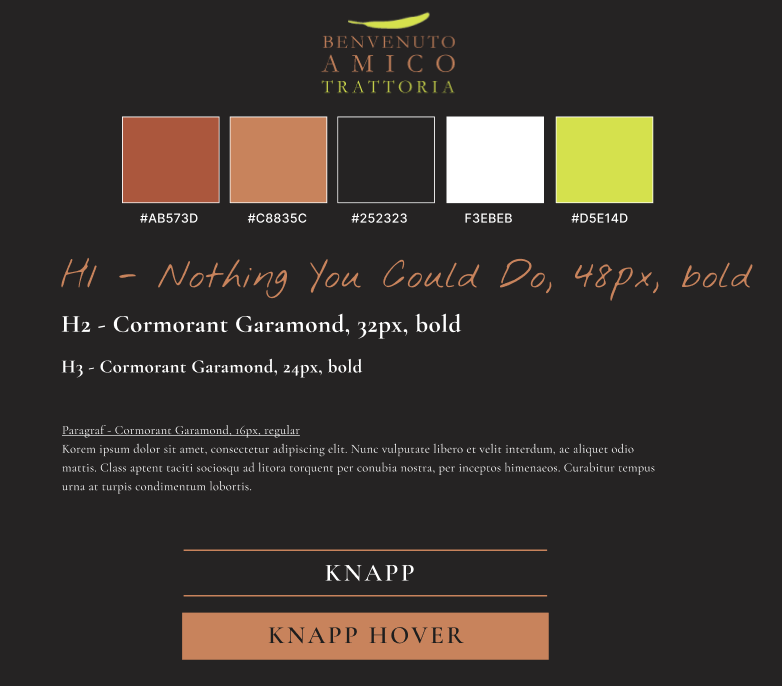
För att säkerställa att webbplatsen håller en genomgående god prestanda kommer den att testas med hjälp av Lighthouse.

# Konstruktion

## Skapande av Moodboard och logotyp

Konstruktionen av projektet inleddes med att experimentera fram en grafisk profil innehållandes färger, typsnitt och knapplayout för restaurangen Benvenuto Amico. Samtliga designval samlades i en moodboard för att tydliggöra layouten för den kommande webbplatsen (Se figur 1). För att komma fram till vilka färger webbplatsen skulle ha hämtades inspiration, efter inhämtat godkännande, från en väns restaurang. Färgvalen testades sedan i webaims contrast checker för att säkerställa att de uppfyller WCAG 2.0 AA (https://webaim.org/resources/contrastchecker/).

I samband med att moodboarden skapades togs även restaurangens logotyp fram i Adobe Illustrator.

  
*Figur 1. Moodboard för restaurangen Benvenuto Amico*

## Skapande av designskisser

Efter att moodboarden tagits fram i enlighet med en tydlig grafisk profilering lades fokus på att ta fram designskisser i enlighet med den grafiska profilen (se figur 2). Designskisserna skapades i prototypverktyget Figma. På grund av en känsla av tidspress skapades designskisser för endast tre av webbplatsens undersidor. Inga designskisser togs fram för administrationsgränssnittet.

*Figur 2. Designskisser för restaurangen Benvenuto Amico*

## Planering av webbtjänst

Projektet fortskred, efter skapande av designskisser, med att planera för den webbtjänst som är tänkt att hantera inloggning och de menyer som ska kunna redigeras av personalen. En enkel skiss togs fram över vilka routes API:et skulle innehålla och totalt planerades det inledningsvis för åtta routes, men en bit in i processen beslutades att lägga till ytterligare en route för att hantera bordsbokning. Se figur (3).

|  |  |
| --- | --- |
| Routes | Beskrivning |
| /api |  |
| /api/users | Route för att nå samtliga användare som är registrerade i API:et. |
| /api/users/user | Route för att nå en specifik användare som är registrerad i API:et |
| /api/register | Route för att skicka data för en ny användare till API:et. |
| /api/login | Route som hanterar inloggning med JWT-token till skyddade routes (se figur 4). |
| /api/dinners | Route för att nå middagsmenyn i API:et. |
| /api/lunches | Route för att nå lunchmenyn i API:et. |
| /api/takeaways | Route för att nå takeaway-menyn i API:et. |
| /api/bookings | Route för att nå alla bokningar i API:et. |

*Figur 3. Tabell över de routes webbtjänsten innehåller*

## Utveckling av webbtjänst

För att utveckla webbtjänsten användes Visual Studio Code för att skriva serverkoden. Utvecklingsarbetet inleddes med att skapa en ny katalog för projektet, därefter initerades ett nytt npm-projekt i terminalen med kommandot *npm init -y.* Npm-paketen Mongoose, Express, Dotenv, Cors, Body-parser, Jsonwebtoken och Router installerades därefter i projektet. Även Nodemon installerades för att automatisera utvecklingsmiljön något. Scriptet ”start”: ”nodemon server.js” lades in i package.json-filen för att enkelt kunna starta upp servern med Nodemon.

I nästa steg skapades en .env-fil för att hantera [url:en](file:///C:\Users\Emma\kurser\DT207G\projekt\Rapport\en) till databasen och en nyckel som används för att skapa en token via Jsonwebtoken. En .gitignore-fil skapades också där mappen node\_modules och .env-filen lades in.

### Server

En startfil för servern skapades med namnet server.js där samtliga npm-paket och två routingfiler importerades (se kap 4.2.3 Routes). Därefter initierades en express-app genom att anropa express()-funktionen vars resultat tilldelas variabeln ”app”.

Två stycken middleware-funktioner lades till med hjälp av npm-paketen body-parser och cors. Middlewaren som registrerar body-parser i applikationen gör så att inkommande request bodies tolkas som JSON och middlewaren för Cors begränsar vilka resurser som har åtkomst till webbtjänsten. I detta fall har endast restaurangens webbsida tillgång till webbtjänsten.

I nästa led länkas de två filerna för routing in i applikationen.

Som ett sista steg i applikationen definieras de routes som ska vara skyddade i admingränssnittet, vilket är en ”välkommen admin”-sida, en sida för att registrera och hantera användare, en sida vardera för att uppdatera lunch, takeaway och middagsmeny (Se figur X).

|  |  |
| --- | --- |
| Routes | Beskrivning |
| /api/admin | Skyddad route för att nå administrationsgränssnittets startsida. |
| /api/admin/bookings | Skyddad route för att nå sidan för bordsbokningar. |
| /api/admin/lunch | Skyddad route för att nå sidan som hanterar lunchmenyn. |
| /api/admin/dinner | Skyddad route för att nå sida som hanterar middagsmenyn. |
| /api/admin/takeaway | Skyddad route för att nå sidan som hanterar takeaway-menyn. |

*Figur 4. Tabell över skyddade routes*

### Modeller

Totalt skapades sex modeller som importerades i webbtjänsten. En modell som hanterar anslutningen till databasen, för att slippa repetera den koden i de olika filerna. En modell som hanterar schemat för användare inklusive metoder för att hasha lösenord, jämföra lösenord, registrera användare och logga in användare. En modell för att hantera schemat för bordsbokning. Utöver dessa tre modeller finns också en modell vardera för lunch, takeaway och middagsschemat.

### Routes

Två separata routingfiler skapades som därefter importerades i server-filen. Den första routingfilen hanterar samtliga routes för att hantera användare och heter authRoutes.js I denna fil importeras först de npm-paket som behövs i filen, därefter hämtas modellen för databasanslutningen och modellen för användarhanteringen. Fyra routes skapades därefter. En för att hämta alla användare, en för att hämta en användare, en för att lägga till ny användare och en för att logga in användare.

Slutligen exporterades samtliga router med kodraden module.exports = router;

I den andra routingfilen som heter menuRoutes.js hanteras samtliga routes för att hämta, lägga till, uppdatera och radera rätter ur de tre olika menyerna – middagsmeny, lunchmeny och takeaway-meny. Samt routes för att hämta, lägga till och radera bordsbokningar.

Även dessa routes exporteras i slutet med ovan nämnd kodrad.

## Utveckling av webbapplikation

För att utveckla webbapplikationen användes Visual Studio Code för att skriva HTML, SCSS och JavaScript-kod. Utvecklingsarbetet inleddes med att starta upp ett nytt npm-projekt. Npm-paketet Parcel installerades som en dev dependency för att skapa en automatiserad utvecklingsmiljö som minifierar och kompilerar de filer som publiceras för webbapplikationen. En katalog med namnet src lades till för att hanter samtliga webbplatsfiler i projektet.

### Struktur

I src-katalogen skapades ytterligare tre kataloger, en som hanterar media-filer, en som hanterar SCSS-filer och en som hanterar JavaScriptfiler. Samtliga HTML-filer placerades direkt i huvudkatalogen src.

Totalt skapades 12 HTML-filer i projektet. Sju av sidorna går att nå från webbplatsens frontend och fem av sidorna tillhör administrationsgränssnittet och kräver att användaren är inloggad för att dessa ska kunna nås.

För frontend-delen skapades en startsida, en om oss-sida, en bokningssida, en login-sida och tre menysidor, en för respektive meny. För administrationsgränssnittet skapades en dashboard för att nå samtliga undersidor, samt tre olika sidor för att uppdatera de olika menyerna och en sida för att hantera bordsbokningar.

Respektive HTML-sida har en egen JavaScript-fil knuten till sig med all JavaScript-kod som behövs för den specifika sidan. Därtill skapades en main.js fil som innehåller den JavaScript-kod som är generisk för samtliga sidor på webbplatsen.

I samtliga HTML-filer infogades därtill en main.scss-fil som genereras från tre andra SCSS-filer, varav den ena innehåller samtliga variabler, den andra innehåller samtliga animationer och den tredje all övriga SCSS-kod.

### Responsivitet

Webbplatsen utvecklades med responsiviteten i åtanke genom användandet av flexbox och dynamiska måttenheter i de fall där detta passat. Dock behövde webbplatsen trots detta anpassas med en relativt många media querys för att få de olika sidornas element att se bra ut på samtliga skärmstorlekar.

### Globala element

#### Header

Headern i HTML-filerna består av en logotyp som är placerad i det övre vänstra hörnet genom att tilldela den egenskaperna top: 0, left: 0. Då logotypen också har en animering tilldelad sig är left satt till -700px på elementet och efter att animationen körts tilldelas left: 0 som standard. Den animation som används på logotypen är ”slideInLeft” (se bilaga 1).

Därtill ligger även huvudnavigeringen i headern. Huvudnavigeringen är placerad i webbplatsens högra hörn och har tilldelats en fixed positionering för att den ska följa med vid scroll på sidan. Även huvudnavigering har tilldelats en animering, och elementet har inledningsvis egenskapen right: -900px men efter att animeringen körts sätts right till 0. Animationen som används på huvudnavigeringen är ”slideInRight” (se bilaga 1).

#### Footer

Footern innehåller tre containrar där den ena innehåller restaurangens logotyp, den andra innehåller länkar till webbplatsens olika sidor och den tredje innehåller kontaktuppgifter till restaurangen. Textfärgen sattes till vit och bakgrunden till svart. De olika containrarna positionerades genom att sätta footer-elementet till display: flex och justify-content: space-evenly. Footern har inga animeringar kopplade till sig.

### Startsida

#### Video

Det första användaren möts av på startsidan är en video som startar automatiskt och spelas på repeat genom attributen loop autoplay. Videon är placerad i vänsterkant med egenskapen left: 0. Videon är ett montage av flera olika videoklipp hämtade från Pexels.com. Montaget klipptes ihop i Adobe Premiere Pro. Animeringen ”slideInLeft” har lagts till på videoelementet (se bilaga 1).

#### Introducerande text

Ovanpå videon ligger en rubriktext i storlek 1 som är positionerad med en relativ positionering och en top-margin satt till -120 pixlar för att den översta delen av texten ska ligga ovanpå videon. Animeringen ”fadeInSkew” har använts på rubriktexten (se bilaga 1).

#### Citat

Efter rubriktexten kommer en paragraf med ett citat från restaurangens ägare. Denna är positionerad med en top-margin på 200 pixlar från föregående element. Citationstecknet ligger inom en span-container och är deklarerad med en font-storlek på 1400% för att sticka ut från texten.

#### Menyintroduktion

Ytterligare 150 pixlar från föregående element kommer en sektion som beskriver restaurangens menyer. Sektionen är deklarerad som en flexbox och innehåller i sin tur två containrar, där den första består av en rubrik i storlek 1, tre paragrafer och en knapp och den andra består av en bild. Samtliga element i den första containern har tilldelats animationen ”fadeInUp” och bildelementet har tilldelats animationen ”increaseWidth” (se bilaga 1).

#### Instagramflöde

Nästa sektion skapades för att lista restaurangens senaste inlägg på instagram. Det skapades ingen autentisk koppling till Instagram, utan istället har olika element satts samman för att efterlikna ett instagramflöde så som de generellt visas på webbplatser. Sektionen är också en flexbox som består av en rubrik i storlek 1, en container innehållandes ytterligare två containrar som vardera innehåller två ”kort” som ska motsvara instagraminläggen och en knapp.

Sektionen som innehåller samtliga av ovan nämna containrar är satt till flex med column som flex-direction. Rubriken är positionerad till höger med float: right. De två containrarna som innehåller ”instagraminläggen” och dess föräldracontainer är satta till flex. På större skärmar har båda dessa row som flex-direction, men allt eftersom skärmstorleken minskas så får först föräldra-containern column som direction och slutligen ändrar även kort-containrar direction.

Knappen är positionerad till höger genom egenskapen align-self: flex-end.

Samtliga element i instagram-containern är animerade med animationen ”fadeInUp” (se bilaga 1).

#### Öppettider

Efter instagramsektionen skapades en sektion som listar restaurangens öppettider. Sektionen består av en h1-rubrik positionerad till vänster med egenskapen align-self: flex-start. Därefter följer en container med tre ytterligare containrar som lista öppettider för olika dagar på veckan. Dessa containrar tilldelades en border på 1px solid. Föräldracontainern är satt till flex och de inre containrarna positioneras därefter med justify-content: space-between.

#### Boka-animering

Den sista sektionen som skapades för startsidan innehåller en animerad text som säger ”Boka”. I HTML-filen skapades en container som innehåller en SVG, som i sin tur innehåller ett länkelement med ett textelement inuti, där x är satt till ”0” och y ”80%”. Teckenstorleken på denna text är satt till 300 pixlar, vilket sedan minskas med hjälp av media querys för mindre skärmstorlekar. Animationen som används på texten är döpt till ”textAnimate” och använder egenskaperna stroke-dasharray och stroke-dashoffset för att få till den rörliga linjen run om. Vid animeringens start är stroke-dasharray satt till 0 50%, stroke-dashoffset är satt till 20% och fill är satt till den ljusare röda färgen i färgschemat. När animeringen avslutas är stroke-dasharray satt till 50% 0, stroke-dashoffset är satt till -20% och fill är satt till samma som bakgrundsfärgen på sidan (se bilaga 1). Animeringen körs sedan med attributen infinite alternate för att den ska köras hela tiden och alterera framlänges och bakländes.

Vid hover på texten ändras muspekaren till pointer och animeringen pausas, samtidigt som linjen runtom animeringen tilldelas den mörkare röda färgen i färgschemat.

### Om oss

”Om oss”-sidan liknar till stor del startsidan som gås igenom i kap 4.5.5 Startsida. Det första som skapades var en h1-rubrik med samma egenskaper som rubriken på startsidan. Därefter följer tre sektioner som samtliga är satta till display: flex och som positionerar versin sektion med en h2-rubrik och en paragraf på ena sidan och en bild på andra sidan. Dessa positioneras i föräldracontainern med justify-content: space-between.

Efter dessa tre sektioner följer sektionen för öppettider som nämns i kap. 4.5.5.6 Öpettider och sektionen för den animerade bokningsknappen som beskrivs i kap. 4.5.5.7 Boka-animering.

### Menyer

Vid klick på ”Meny”-länken i huvudnavigeringen landar användaren på sidan för middagsmenyn. Det första som dyker upp på sidan är en kompletterande navigering specifik för meny-sidorna. Där kan användaren klicka sig mellan middagsmeny, lunchmeny och takeaway-meny, som består av tre separata HTML-filer med samma kompletterande navigering.

Den kompletterande navigeringen består av en h1-rubrik som sedan fått en border-bottom på 1 px solid. Under denna ligger en oordnad lista innehållandes länkar till de olika menysidorna. Dessa är formaterade genom en förlängning (@extend) av egenskaperna från huvudnavigeringen, men för att de ska hållas på plats vid samtliga webbläsarstorlekar har de tilldelats en absolut positionering. Därtill har de row som flex-direction istället för column. Vid hover över någon av länkarna ändras textfärgen från vit till svart och bakgrundsfärgen från mörkgrå till den ljusare av de roströda färgerna i färgschemat. Denna bakgrundsfärg visar även vilken av länkarna som är aktiv, genom att en klass lagts på den aktiva länken och tilldelats samma egenskaper som vid hover.

Gundlayouten för menylistningen på de olika menysidorna är relativt simpel och består av dels en h1-rubrik som beskriver vilken av menysidorna användaren är inne på. Därefter följer två till fyra containrar som innehåller antingen information om menyn eller menyalternativen.

#### Middagsmeny

Middagsmenyn består av fyra containrar, en för respektive kategori på menyn. Den enda statiska texten som ligger i dessa containrar är rubrikerna smått, förrätter, huvudrätter och efterrätter. I övrigt läses all text in från webbtjänsten som beskrivs i kap. 4.4 Utveckling av webbtjänst enligt texten som följer.

Samtliga av de fyra containrarna hämtas in i JavaScript-filen för sidan och lagras i variabler.

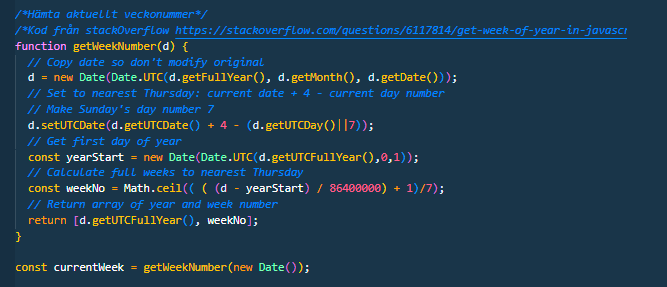
En funktion med try/catch skapades för att läsa in datan från api/dinners (se figur 4) genom att använda fetch. I funktionen kontrolleras det om svaret från API:et är okej och om det inte är det ”kastas” ett fel till catch-delen av funktionen som skriver ut felet i konsolen. Om svaret är okej omvandlas det till JSON-format och lagras i en variabel. Därefter anropas en funktion som loopar igenom middagsrätterna med datan från den lagrade variabeln.

Funktionen som loopar igenom datan är skapad för att ta emot ett argument. I funktionen deklareras först en tom variabel, därefter körs en forEach-loop som loopar igenom varje rätt i middagsmenyn. En switch-sats används för att kontrollera vilken kategori rätten tillhör för att kunna skriva ut den till rätt container i DOM. Det första caset i switch-satsen kollar om dinner.category är lika med ”Förrätt.”. Om så är fallet tilldelas den variabel som lagrat DOM-containern för förrätter till variabeln som deklarerades först i funktionen innan switch-satsen avbryts. Därefter hoppar funktionen vidare till en if-sats som kontrollerar variabeln som lagrar kategorin och skriver ut relevant information från API:t till rätt container via innerHTML.

För att få koden att köra anropas den första funktionen som hämtar datan och kör funktionen för att loopa igenom den.

#### Lunchmeny

Lunchmenyn består utav två containrar, en med statisk text som informerar om vad som gäller vid lunch och en som listar veckans luncher. JavaScript-koden som används för att skriva ut lunchmenyn till DOM liknar till viss del den som används för att skriva ut middagsmenyn. En skillnad är dock att personalen i administrationsgränssnittet kan lägga in flera lunchmenyer, en för respektive vecka på året, men endast den lunchmeny som är för innevarande vecka ska visas på webbplatsen. En funktion har därför lagts till för att hämta aktuellt veckonummer för att kunna kontrollera vilken veckas meny som ska skrivas ut till DOM. Koden för denna funktion är hämtad från Stack Overflow (se figur 5). Aktuell vecka lagras därefter i en variabel.

Figur 5. Bild av kodstycke som är hämtat från Stack Overflow.

Variabeln som lagrar den aktuella veckan används sedan forEach-loopen i funktionen som itererar datan genom att jämföra om den angivna lunchveckan i webbtjänsten är samma som aktuell kalendervecka, om de är samma, så loopas datan igenom och skrivs ut till DOM.

En annan skillnad i koden, jämfört med koden som skriver ut middagsmenyn, är att switch-satsen kontrollerar veckodagarna som är angivna för respektive lunch, istället för att kontrollera kategori.

#### Takeaway-meny

Koden som skriver ut Takeaway-menyn till DOM är i stort sett identisk med den som skriver ut middagsmenyn till DOM.   
Enda skillnaden är att det endast finns två Takeaway kategorier, till skillnad mot middagsmenyns fyra kategorier.

### Administrationsgränssnitt

Samtliga delar av administrationsgränssnittet är skyddade med jsonwebtoken och kan alltså endast nås om användare genererar en giltig token via inloggningsformuläret, eller om en giltig token redan finns lagrad i localStorage.

Samtliga sidor i administrationsgränssnittet har ett script lagt direkt i HTML-filen dör att verifiera om en giltig token finns eller inte. Anledningen till att scriptet ligger i HTML-filerna är för att koden ska köras direkt när sidan laddas. Under utvecklingen testades en variant där koden lades i JavaScript-filerna, men vid försök att nå någon av de skyddade routes laddades sidan i några millisekunder innan användaren omdirigeras. Genom att lägga koden direkt i HTML-filerna kunde detta problem undgås.

#### Dashboard administrationsgränssnitt

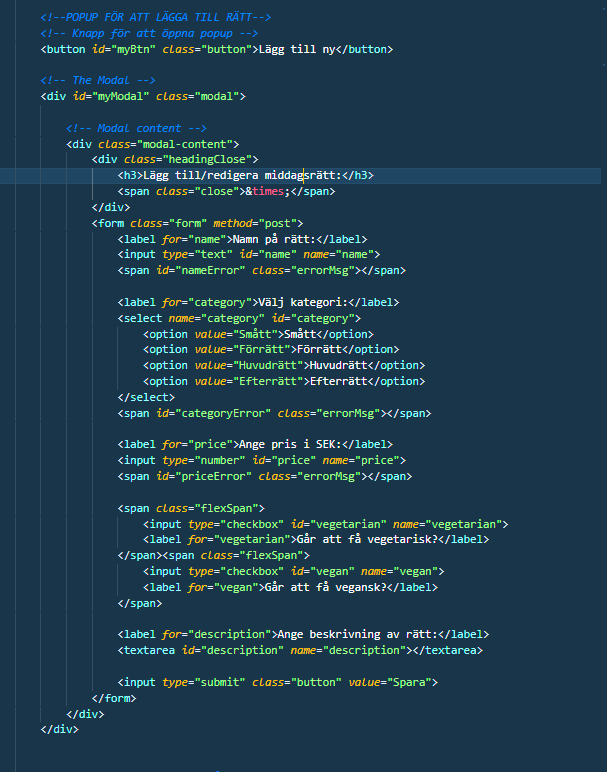
Dashboarden som är det första användaren kommer till efter inloggning i administrationsgränssnittet består av en HTML-fil som i princip endast innehåller fyra containrar innehållandes varsin h3-rubrik. En JavaScript-fil som hanterar omdirigering till skyddade routes vid klick på respektive container har importerats. I den filen hämtas de fyra containrarna och lagras i varsin variabel. Eventlyssnare har sedan lagts till för respektive container som vid klick omdirigerar användaren till korrekt sida. Från denna fil hanteras även koden för logga ut-knappen, som kort och gott är en eventlyssnare som hämtar elementet för logga ut-knappen och vid klick tar bort token från localStorage för att sedan omdirigera användaren till startsidan.

#### Uppdatera menyer

Det finns ett antal element som är gemensamma för samtliga av undersidorna i administrationsgränssnittet. De fyra containrarna som återfinns på dashboarden, finns också på samtliga undersidor, men det har fått en mindre storlek med SCSS och placeras högst upp i mitten av sidan. Detta för att användaren lätt ska kunna navigera mellan de olika undersidorna utan att behöva klicka sig tillbaka till dashboarden.

Därtill har samtliga sidor som uppdaterar menyer en modal innehållandes det formulär som används för att uppdatera eller lägga till en ny rätt. Modalen öppnas genom klick på antingen knappen för att lägga till, eller knappen för att uppdatera menyalternativ. Modalen är skapad med en mängd olika containrar och ett form-element innehållandes relevanta formulärfält för respektive databasschema (se figur 5).

Samtliga HTML-sidor har också containrar med en tunn border runt om som är positionerade centrerat med flexbox oxh justify-content: center. Respektive container innehåller en h3-rubrik som talar om vilken menykategori som återfinns i containern. På samma sätt som menyalternativen skrivs ut till DOM på webbplatsens menysidor skrivs de även ut i administrationsgränssnittet. Skillnaden är dock att för respektive menyalternativ tillkommer också två knappar, en för att uppdatera och en för att radera menyalternativet.



Figur 5. Modal i HTML-filen för att uppdatera middagsmeny

# Resultat

Projektets övergripande mål var att skapa en ny webbplats år den fiktiva restaurangen Benvenuto Amico Trattoria. Webbplatsen skulle representera restaurangen på ett tydligt och lättanvänt sätt och skapas i enlighet med en tydlig grafisk profil. Funktionalitet för att boka bord skulle finnas på webbplatsen och den skulle också innehålla ett administrationsgränssnitt där restaurangens personal kan logga in och själva uppdatera och redigera valda delar av innehållet.

Detta mål bröts sedan ner i mer konkreta delmål som skulle vara uppfyllda vid projektslut.

Det första konkreta delmålet var att designskisser ska presenteras i enlighet med Benvenuto Amico Trattorias grafiska profil. Detta mål skulle jag säga uppfylldes delvis, med tanke på att designskisser för endast ett fåtal sidor togs fram.

Nästkommande mål var att det ska vara enkelt att boka bord och hitta restaurangens menyer via webbplatsen. Detta mål tycker jag blivit uppfyllet. Ett av de första element användaren ser på webbplatsen är boka-knappen som ligger i huvudnavigeringen och har en annan färg än övriga knappar. Därtill har en stor boka-länk lagts till i slutet av flertalet sidor, denna leder också till sidans bordsbokning.

Det tredje målet var att ett lättanvänt administrationsgränssnitt skulle skapas där restaurangens medarbetare ska kunna uppdatera de tre olika menyerna, samt hantera inkommande bordsbokningar. Detta mål har uppfyllts och personligen anser jag att det är lättanvänt. Det är tydligt för användare hur de ska gå tillväga för att komma till de olika menyernas uppdateringssidor och knapparna för att ändra och redigera är också tydliga.

Det sista målet var att webbplatsen ska validera enligt standarden för w3c och WCAG 2.0 för att säkerställa en god tillgänglighet och även detta mål har blivit uppfyllt.

# Slutsatser

Det är många aspekter av det här projektet som jag hade gjort annorlunda om jag hade fått börja om. Dels hade jag lagt ner mer tid på det planerande arbetet när det kommer både till webbtjänsten och webbplatsen. Databasen för webbtjänsten hade troligtvis kunnat läggas upp mer effektivt med lite mer planering. Webbplatsen i sin tur hade troligtvis gått betydligt snabbare att utveckla om jag hade haft tydliga designskisser för samtliga sidor. Jag önskar därtill att jag hade planerat mina JavaScript-funktioner bättre från start. När jag nu i efterhand går igenom koden tycker jag att jag har väldigt mycket kod som är avsedd för att göra samma sak. Om jag planerat webbtjänsten bättre i kombination med en tanke kring hur jag skulle lägga upp mina JavaScript-funktioner hade jag troligtvis kunnat återanvända funktioner på ett annat sätt än vad jag gör nu. Så som det här projektet har utvecklats har jag tagit ett steg åt gången, utan att planera för hela applikationen och dess funktioner.

Jag har också känt att jag har komplicerat det här projektet onödigt mycket, med alla olika animationer och liknande, vilket har gjort att jag känt en otrolig tidspress och därmed inte kunnat hålla den kvalitet jag önskat på projektet. Med tanke på att det är ett stort projekt i flera delar redan från start, borde jag ha hållit mig till en mer enkel layout och därefter lagt på extra funktionalitet om det hunnits med.

Jag är relativt nöjd med sidans prestanda och är nöjd över att den ser bra ut på samtliga skärmstorlekar och i de olika webbläsare jag testat. Jag är också mycket nöjd med sidans layout som jag personligen tycker känns lyxig och lite annorlunda mot en vanlig restaurang.

Allt som allt finns det många lärdomar som jag tar med mig från det här projektet.

Källförteckning

1. Moodle: Log in to the site [Internet]. elearn20.miun.se. [cited 2024 May 24]. Available from: <https://elearn20.miun.se/moodle/pluginfile.php/1616610/mod_resource/content/0/dt207g_fo%CC%88rela%CC%88sning_webbtja%CC%88nster.pdf>
2. About npm | npm Docs [Internet]. docs.npmjs.com. [cited 2024 May 24]. Available from: <https://docs.npmjs.com/about-npm>
3. Mongoose. Mongoose ODM v5.8.2 [Internet]. Mongoosejs.com. 2011. [cited 2024 May 24]. Available from: <https://mongoosejs.com/>
4. Express 5.x - API Reference [Internet]. expressjs.com. [cited 2024 May 24]. Available from: https://expressjs.com/en/5x/api.html
5. ‌ dotenv [Internet]. npm. 2024 [cited 2024 May 24]. Available from: https://www.npmjs.com/package/dotenv#%EF%B8%8F-usage
6. ‌cors [Internet]. npm. [cited 2024 May 25]. Available from: <https://www.npmjs.com/package/cors>
7. body-parser [Internet]. npm. 2019. [cited 2024 May 25]. Available from: <https://www.npmjs.com/package/body-parser>
8. jwt. JWT.IO - JSON Web Tokens Introduction [Internet]. jwt.io. [cited 2024 May 25]. Available from: <https://jwt.io/introduction>
9. nodemon [Internet]. nodemon.io. [cited 2024 May 25]. Available from: https://nodemon.io/