

# Projektuppgift

*DT208G – Programmering i TypeScript*

## **Moment 5 - Projekt**

Webbplats för det fiktiva universitetet Nordic Institute of Advanced Learning

**Emma Lorensen**

**MITTUNIVERSITETET**

**Avdelningen för informationssystem och -teknologi**

**Författare:** Emma Lorensen, [emlo2302@student.miun.se](mailto:emlo2302@student.miun.se)

**Utbildningsprogram:** Webbutveckling, 120 hp

**Huvudområde:** Datateknik

**Termin, år:** VT, 2024



**Mittuniversitetet**  
MID SWEDEN UNIVERSITY

## Sammanfattning

Projektets syfte var att med Angular som ramverk och TypeScript som programmeringsspråk skapa en webbplats åt det fiktiva universitetet Nordic Institute of Advanced Learning (NIAL). Webbplatsen skulle innehålla en startsida med allmän information om universitetet, en sida som listar samtliga av universitetets kurser och en sida som listar det ramschema studenten själv väljer utifrån universitetets kurssida.

Webbsidan skapades både med egenskapade komponenter och komponenter från Angular Materials komponentbibliotek. Dels för att snabba på utvecklingsarbetet och dels för att testa på att använda ett färdigt komponentbibliotek.

Arbetet inleddes med att planera webbapplikationen och ta fram en grafisk profil och designskisser i Figma. Därefter startades ett nytt Angular-projekt upp i Visual Studio Code och komponenterna skapades en efter en. Totalt består projektet av nio komponenter, två services och ett interface.

Samtliga av projektets mål uppnåddes, men en reflektion är att utvecklingsarbetet gjordes onödigt komplicerat då några av komponenterna hade kunnat slås ihop för att få en bättre struktur på projektet.

## Innehållsförteckning

Sammanfattning .....	ii
1 Introduktion.....	1
1.1 Bakgrund och problemmotivering .....	1
1.2 Övergripande syfte .....	1
1.3 Konkreta och verifierbara mål .....	2
2 Teori .....	3
2.1 TypeScript .....	3
2.2 Angular .....	3
2.2.1 Komponenter.....	3
2.2.2 Routing.....	4
2.2.3 Services .....	4
2.2.4 Data Binding .....	4
2.2.5 Dependency Injection .....	4
2.3 Angular Material .....	4
3 Metod .....	5
3.1 Moodboard.....	5
3.2 Designskisser .....	5
3.3 Utvecklingsmiljö och verktyg.....	5
3.4 Validering och tillgänglighet.....	6
3.4.1 Validering av kod .....	6
3.4.2 Validering enligt WCAG 2.0.....	6
3.4.3 Prestanda .....	6
4 Konstruktion .....	7
4.1 Planering av webbapplikation.....	7
4.2 Skapande av moodboard och logotyp .....	7
4.3 Skapande av designskisser.....	8
4.4 Utveckling av webbapplikation .....	8
4.4.1 Komponenter.....	8
4.4.2 Services .....	15
4.4.3 Responsivitet.....	15
5 Resultat .....	16
6 Slutsatser .....	17

# 1 Introduktion

Den här rapporten kommer att steg för steg gå igenom och motivera de tillvägagångssätt, kod och designval som gjorts för att skapa en webbplats åt det fiktiva universitetet Nordic Institute of Advanced Learning, nedan kallat NIAL. I detta första kapitel behandlas *bakgrund och problemmotivering*, *syfte*, *avgränsningar* samt *konkreta och verifierbara mål*.

## 1.1 Bakgrund och problemmotivering

NIAL är ett universitet beläget i en medelstor stad i Sverige som erbjuder befintliga och blivande studenter att bygga sitt eget ramschema. För att studenterna enkelt ska kunna göra detta vill NIAL ha en webbsida där samtliga av deras kurser listas i en sök- och filtreringsbar lista i tabellformat. Utifrån listan ska studenterna sedan snabbt kunna söka fram relevanta kurser och lägga till dem i sitt ramschema, som ska visas på en egen undersida.

Samtliga av universitetets kurser och information om dessa finns lagrat i ett API som universitetet administrerar. Detta API kommer att ligga till grund för de kurser som listas på webbplatsen. För att universitetet enkelt ska kunna kontrollera att samtliga kurser är tillgängliga i listan på webbplatsen önskar de att det i anslutning till listan står skrivet hur många kurser som finns tillgängliga i den. De önskar också att studenten, i sitt ramschema, ska kunna se det sammanlagda antalet högskolepoäng som de tillagda kurserna har.

Projektet ska byggas i TypeScript med Angular som ramverk då skolan har erfarenhet av att arbeta i detta sedan tidigare. För att snabba på utvecklingstiden och därigenom minimera kostnader önskar de att, i så hög grad som möjligt, ett komponentbibliotek används för webbplatsens olika komponenter.

## 1.2 Övergripande syfte

Projektets övergripande syfte är att skapa en webbplats byggd i ramverket Angular åt det fiktiva universitetet NIAL. Studenter ska kunna söka och filtrera bland universitetets samtliga kurser och lägga till kurser till ett eget ramschema. Studenterna ska också kunna plocka bort kurser från ramschemat och se det totala antalet högskolepoäng som är tillagda i ramschemat.

### **1.3 Konkreta och verifierbara mål**

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

- Webbplatsen ska skapas med Angular och TypeScript och komponenter och routing ska användas.
- Webbplatsen ska bestå av en startsida och två undersidor, en för att söka och visa information om kurser - och en som visar skapat ramschema.
- Minst två services skall skapas, en för kursdata och en för hantering av ramschema.
- Skapat ramschema skall lagras med hjälp av localStorage, och läsas in vid inladdning av webbsidan.
- Kurserna i kurslistningen ska gå struktureras genom att filtrera på ämne, sorteras i samtliga kategorier och fritextsök.
- Det ska gå att se antal kurser när såväl samtliga kurser visas som i respektive sökträff.
- Listan med universitetets kurser ska vara paginerad för att förbättra användbarheten.
- Det sammanlagda antalet högskolepoäng för samtliga kurser i ramschemat ska visas.
- Webbplatsen skall vara snygg och pryddig och fungera väl på stora som små skärmar med bra responsiv design.

## 2 Teori

### 2.1 TypeScript

TypeScript är ett programmeringsspråk som är baserat på JavaScript, men lägger till statisk typning med möjlighet typannoteringar. När TypeScript körs transpileras det till JavaScript vilket innebär att det går att köra överallt där JavaScript går att köra [1]. Det är främst utvecklat för skapandet av stora applikationer, då den statiska typningen kan hjälpa till att minska antalet buggar och fel i koden, men kan även användas i mindre projekt [2].

Några av fördelarna med TypeScript är:

- **Statisk typning**  
Med statisk typning kan man definiera datatyper för variabler, funktioner och objekt. Det kan ge tidigare upptäckt av fel och förbättrar koden genom att göra den mer lättläst [3].
- **Klasser och arv**  
Objektorienterad programmering stöds genom klasser och arv, vilket ger ett mer strukturerat sätt att bygga och hantera kodbasen [3].
- **Gränssnitt (interfaces)**  
Det är möjligt att skapa gränssnitt som hjälper till att definiera kontrakt för olika delar av koden [3].
- **Moduler**  
Ett inbyggt modulstöd underlättar en tydligare organisering av koden och möjliggör effektiv återanvändning av kod [3].
- **Generics**  
Generics möjliggör skapande av flexibla och återanvändbara komponenter genom att tilldela specifika typer när komponenterna används, snarare än när de skapas [3].

### 2.2 Angular

Angular är ett ramverk för att bygga webbapplikationer som använder TypeScript som programmeringsspråk. Det utvecklades och underhålls av Google tillsammans med en community av utvecklare och företag [4].

#### 2.2.1 Komponenter

Angular använder en komponentbaserad arkitektur som gör det möjligt att utveckla modulära och återanvändbara delar i en applikation. Varje komponent inkapslar sin egen HTML-, CSS- och TypeScript-kod, vilket underlättar både underhåll och testning av de enskilda delarna av systemet [4].

### **2.2.2 Routing**

Angular kommer med en inkluderad router som möjliggör enklare hantering av navigationen i en single-page applikation [4].

### **2.2.3 Services**

I angular används services för att separera logiken från komponenterna, vilket gör det möjligt att återanvända funktionalitet. De används vanligtvis för att hantera serverkommunikation eller för att dela data och funktioner mellan olika delar av applikationen [5].

### **2.2.4 Data Binding**

Angular stödjer såväl en- som tvåvägs data binding, vilket innebär att ändringar i användargränssnittet automatiskt uppdaterar koden och ändringar i koden automatiskt uppdaterar användargränssnittet. Envägs data binding innebär att information bara skickas åt ena hållet, t. ex från koden till gränssnittet. Medan tvåvägs data binding innebär att information skickas åt båda hållen samtidigt [5].

### **2.2.5 Dependency Injection**

Angular har ett inbyggt system för dependency injections, vilket gör det enklare att hantera beroenden mellan komponenter och services. Detta skapar modularitet och förenklar testning av koden [4].

## **2.3 *Angular Material***

Angular Material är ett komponentbibliotek för Angular som är framtaget av Google. Material har en mängd olika tillgängliga komponenter, så som knappar, formulärfält, tabeller etcetera. Dessa komponenter är anpassningsbara och är lätta att integrera i angularapplikationer [4].

## 3 Metod

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

### 3.1 Moodboard

För att ha en tydlig bild av den grafiska profilen redan innan designskisser skapas inleds projektet med att ta fram en moodboard i Figma. Moodboarden representerar en övergripande bild av vilka färger, typsnitt, teckenstorlekar och hur knapplayout ska se ut.

### 3.2 Designskisser

Designskisser skapas i prototypverktyget Figma så snart moodboarden är färdigställd. Designskisserna ger en detaljerad bild av hur webbplatsen är tänkt att se ut och fungera. Samtliga beslut om hur sidorna ska se ut fattas under detta steg för att minimera att layoutbeslut behöver fattas vid konstruktionsmomentet.

### 3.3 Utvecklingsmiljö och verktyg

De system som används för att utveckla universitetets webbapplikation är följande:

- **Figma**  
I Figma skapades de designskisser som låg till grund för webbapplikationens layout.
- **Visual Studio Code**  
Den kodeditor som används är Visual Studio Code – i den utvecklas all HTML, CSS och TypeScript-kod.
- **Angular**  
Angular används som ramverk.
- **Angular Material**  
Flertalet av de komponenter som finns i webbapplikationen är hämtade från Angular Material.
- **Canva**  
Canva används för att ta fram universitets logotyp.
- **GitHub**  
Git och GitHub används för att versionshantera sidan.
- **Netlify**  
Webbapplikationen är publicerad hos Netlify.



### **3.4 Validering och tillgänglighet**

#### **3.4.1 Validering av kod**

För att säkerställa att webbplatsens kod uppfyller krav på kvalitet och funktionalitet kommer samtliga HTML- och CSS-filer att valideras med hjälp av W3C Web Validator [6] i webbläsaren.

#### **3.4.2 Validering enligt WCAG 2.0**

Webbplatsen kommer att granskas i enlighet med WCAG 2.0-standarderna med hjälp av AChecker Web Accessibility Checker [7], för att säkerställa en tillgänglig och inkluderande användarupplevelse.

#### **3.4.3 Prestanda**

För att säkerställa att webbplatsen upprätthåller hög prestanda kommer tester att genomföras med hjälp av Lighthouse [8].

## 4 Konstruktion

### 4.1 Planering av webbapplikation

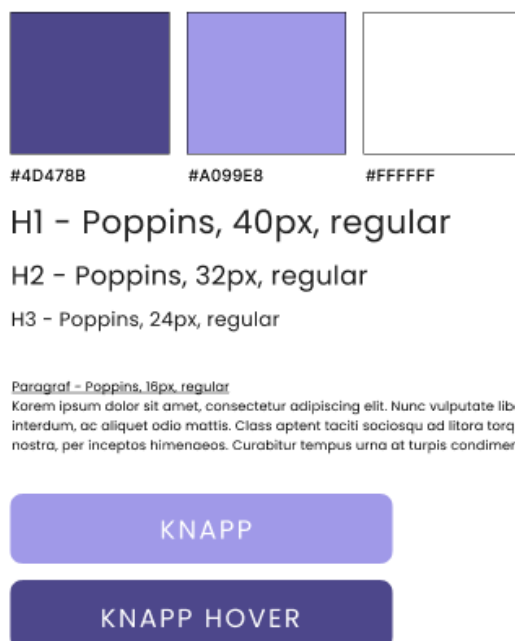
Projektet inleddes med att planera för de sidor och komponenter som behövde finnas på webbplatsen för att projektets mål skulle uppfyllas.

Det beslutades att webbplatsen skulle bestå av en startsida och två undersidor, varav den ena undersidan listar universitets samtliga kurser och den andra listar studentens valda ramschema.

### 4.2 Skapande av moodboard och logotyp

När strukturen på webbplatsen beslutats fortskred arbetet med att ta fram en grafisk profil till universitetet. Färgvalen gjordes utifrån en önskan om att framställa universitet dels seriöst, dels lekfullt och modernt. Den grafiska profilen inkorporerades sedan i en Moodboard där även typsnitt, textstorlekar och knapplayout beslutades (se figur 1).

I samband med att moodboarden skapades togs även universitets logotyp fram i verktyget Canva (se figur 2).



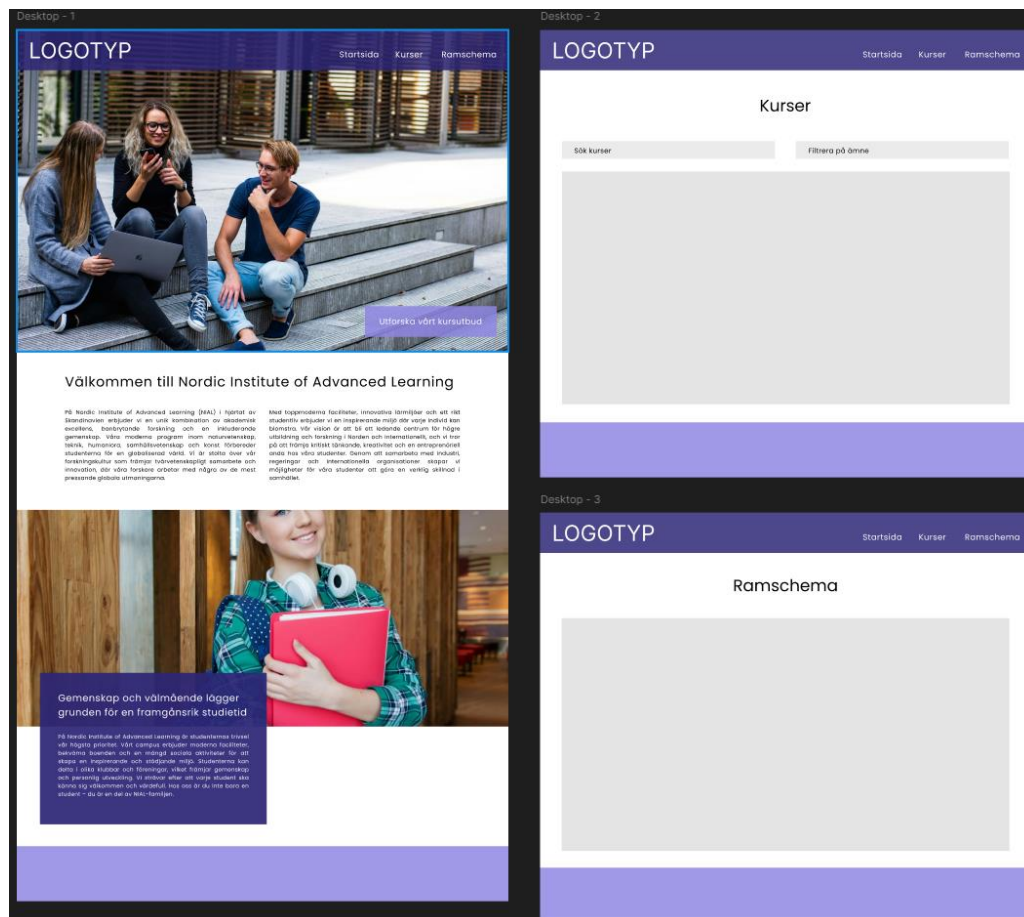
Figur 1. Moodboard för universitetet NIAL



Figur 2. Logotyp

### 4.3 Skapande av designskisser

I nästa steg skapades enkla designskisser över samtliga av webbplatsens sidor och komponenter i Figma (se figur 3). Designskisserna skapades endast för desktop-läge, främst på grund av tidsbrist. I designskisserna implementerades de designval som tagits i samband med att moodboarden togs fram och samtliga beslut om webbplatsens layout fattades.



Figur 3. Designskisser föreställande universitetet NIALs webbplats.

### 4.4 Utveckling av webbapplikation

För att utveckla webbapplikationen användes Visual Studio Code för att skriva HTML, SCSS och TypeScript-kod. Utvecklingsarbetet inleddes med att starta upp ett nytt angularprojekt genom kommandot `ng new project`. I och med att Angular själv installerar alla nödvändiga paket och sätter upp strukturen för projektet behövdes inget ytterligare göras i detta skede.

#### 4.4.1 Komponenter

Inledningsvis skapades tre komponenter, en för respektive startsida, genom kommandot `ng generate component 'component-name'`. Dessa döptes till `home`, `courses` och `myschedule`. Därefter skapades komponenter allt eftersom

sidorna byggdes. De övriga komponenter som skapades motsvarar elementen för header, footer, huvudmeny, text som placeras ovanpå header, tabell för kursvisning och tabell för ramschema. I slutändan blev det totalt nio komponenter som var och en gås igenom i detalj nedan.

#### 4.4.1.1 Komponent: home

Komponenten 'home' skapades för att utgöra startsidan (se figur 4) och består av komponenten 'heroimage', en container innehållande en h1-rubrik och en paragraf samt komponenten 'textonimage'.

Containern tilldelades genom CSS-egenskaper en bredd om 95% med marginaler satta till 0 auto och en inre sidopadding om 50 pixlar. Rubriken centrerades med egenskapen text-align: auto och paragrafen delades upp i kolumner med column-count: 2 och 30 pixlars mellanrum. Paragrafen tilldelades också en maxbredd om 1050 pixlar och centrerades med auto-margin.

Vid skärmstorlekar mindre än 900 pixlar sattes column-count till 1 istället för 2.



Figur 4. Komponent: home.

#### 4.4.1.2 Komponent: heroimage

Komponenten 'heroimage' skapades för att representera den övre bilden på startsidan (se figur 5) och består av endast en container och en knapp med ett routerLink-attribut som är kopplat till 'courses'-komponenten.

Containern tilldelades en bakgrundsbild via CSS och fick därtill en min-height om 90vh. Bakgrunden sattes till no-repeat och cover. Containern tilldelades också en flex-layout med justify-content: flex-end för att positionera knappen. För skärmar mindre än 850 pixlar sattes containerhöjden till 50vh.

Knappen tilldelades den ljuslila färgen som syns i moodboarden (se kap. 4.2) men med viss transparens. Vid hover ändras färgen till den mörka lila i moodboarden, även den med viss transparens. Knappens bredd sattes till 15em och med en padding på 1em åt samtliga håll. Den har relativ positionering och en border-radius på 10 pixlar.

Knappen tilldelades även en transition ease 0.6s för att få en lite mjukare övergång av bakgrundsfärgerna vid hovring.



Figur 5. Komponenten heroimage.

#### 4.4.1.3 Komponent: textonimage

Komponenten 'textonimage' skapades för att innehålla den nedre bilden på startsidan med en textruta placerade ovanpå (se figur 6). Komponenten består av en container innehållandes ytterligare två containrar. Den första containern lämnades tom för att endast användas till att tilldela en bakgrundsbild via CSS och den andra containern innehåller en h2-rubrik och en paragraf.

Containern som tilldelades en bakgrundsbild sattes till en höjd på 600 pixlar på större skärmar och 300 pixlar på skärmar mindre än 660 pixlar breda. Containern för textrutan tilldelades den mörklila bakgrundsfärgen, en maxbredd på 550 pixlar och en bredd på 100%. Därtill tilldelades en inre padding om 50 pixlar åt alla håll. På större skärmar definierades textrutans marginaler till -110px 50px 80px och på skärmar mindre än 660 pixlar blev sidomarginalerna istället 20px.



Figur 6. Komponenten textonimage.

#### **4.4.1.4 Komponent: courses**

Komponenten *'courses'* skapades för att utgöra undersidan "kurser", vilket är den sida där universitetets kurser listas. Komponentens innehåller en container som i sin tur innehåller en h2-rubrik och komponenten *'courselist'*. Containern tilldelades egenskaper som flexbox med column direction och justify-content samt align-items center.

#### **4.4.1.5 Komponent: courselist**

*'courselist'*-komponenten skapades för att innehålla tabellen som universitetets kurser listas i. Tabellen skapades med Angular Materials tabellkomponent *'mat-table'* för att visa en lista med universitetets kurser i tabellformat. Komponentens struktur består inledningsvis av en container som innehåller två *'mat-form-field'*-element och en tabell. Det första *'mat-form-field'*-elementet används för att hantera fritextsökning i tabellen och det andra *'mat-form-field'*-elementet används som en drop-down för att kunna filtrera kurserna baserat på ämneskategori.

Tabellen innehåller sex kolumner skapade med *'ng-container'*-element, respektive kolumn innehåller ett element för table header och ett element för att lista table data. Kolumnerna visar kurskod, ämne, poäng och länk till kursplan. Varje rad innehåller även en knapp för att lägga till kursen i användarens schema. Tabellen är sorteringsbar och paginerad för att hantera större dataset.

Containern som innesluter hela tabellen tilldelades en maxbredd på 1370 pixlar och en marginal på 10 pixlar ovanför och 50 pixlar under innehållet. Själva tabellen tilldelades en maxhöjd på 600 pixlar och en maxbredd på 1000 pixlar med dess horisontella overflow dold.

Fritextsökningfältet och ämnesfilterfältet definierades båda med en bredd på 50%. På skärmar mindre än 1140 pixlar döljs kolumnen för ämne och på skärmar mindre än 812 pixlar döljs även kolumnen för kurskod. Teckenstorleken för både text och knappar justeras också för att förbättra läsbarheten på mindre enheter. Knapparna för att lägga till kurser använder moodboardens mörklila färg som bakgrund och den ljuslila färgen vid hover.

I TypeScript-koden deklarerades först vilka kolumner som ska visas i tabellen i en array. Denna array innehåller namnen på de sex kolumnerna som angetts ovan. Tabellens data hanteras av *'MatTableDataSource'*, där kurserna laddas in dynamiskt från tjänsten *'CourseDataService'*. När komponenten initieras hämtas kursdata från tjänsten och tilldelas datakällan för tabellen.

Sortering och paginering implementerades med Angular Materials sorterings- och pagineringskomponenter – *'MatSort'* och *'MatPaginator'*.

För att hantera filtrering av tabellens data skapades två huvudfunktioner – en för fritextsökning och en för ämnesfiltrering. Den kombinerade

filtreringen skapades med hjälp av *'filterPredicate'* som använder både fritext och ämne för att bestämma vilken data som visas i tabellen.

När användaren klickar på knappen "lägg till kurs" anropas en metod som heter *addCourse* och som lägger till kursen i användarens ramschema via servicen *'MyScheduleService'* (se kap 4.4.2.1).

Fritextsökning		Filtrera på ämne			
Kurskod	Kursnamn	Ämne	Poäng	Kursplan	Lägg till
AK001A	Teori och metod	Arkiv- och informationsvetenskap	7.5	<a href="#">Kursplan</a>	+
AK001G	Information, dokumentation och arkiv	Arkiv- och informationsvetenskap	7.5	<a href="#">Kursplan</a>	+
AK001U	Grundläggande dokumenthantering för registratorer	Arkiv- och informationsvetenskap	7.5	<a href="#">Kursplan</a>	+
AK002A	Uppsats	Arkiv- och informationsvetenskap	15	<a href="#">Kursplan</a>	+
AK002G	Handskriftsläsning II	Arkiv- och informationsvetenskap	7.5	<a href="#">Kursplan</a>	+
AK002U	Grundläggande dokumenthantering	Arkiv- och informationsvetenskap	7.5	<a href="#">Kursplan</a>	+
AK003A	Arkivsystem	Arkiv- och informationsvetenskap	7.5	<a href="#">Kursplan</a>	+
AK003G	Processbaserad dokumenthantering och arkivredovisning	Arkiv- och informationsvetenskap	15	<a href="#">Kursplan</a>	+
AK003U	Informationsförvaltning och registratur	Arkiv- och informationsvetenskap	7.5	<a href="#">Kursplan</a>	+
AK004A	Tillgängliggörande och användning	Arkiv- och informationsvetenskap	7.5	<a href="#">Kursplan</a>	+

Items per page: 10 1 – 10 of 4328 < >

Figur 7. Komponenten *Courselist*

#### 4.4.1.6 Komponent: *myschedule*

Komponenten *'myschedule'* skapades för att utgöra undersidan "Ramschema" och innehåller en container som i sin tur innehåller en h2-rubrik och komponenten *'myschedulelist'*. Containern tilldelades egenskaper som flexbox med column direction och justify-content samt align-items center.

#### 4.4.1.7 Komponent: *myschedulelist*

Komponenten *'myschedulelist'* representerar den tabell där de sparade kurserna från kurslistan visas och är precis som komponenten *'courselist'*, beskriven i kap. 2.2.1.5, skapad med Angular Materials tabellkomponent *'mat-table'*. I tabellen skrivs de kurser som användaren sparar till sitt ramschema ut under tabellrubrikerna kurskod, kursnamn, ämne, poäng och kursplan. Varje rad innehåller även en knapp för att ta bort respektive tillagd kurs.

Komponentens struktur består av en container innehållandes sex *'ng-container'*-element. Var och en av dessa innehåller i sin tur ett table header- och två table data-element. Det första table-data elementet är en mat-cell där datan från de sparade kurserna skrivs ut och det andra table-data



elementet är en mat-footer-cell. Det är endast mat-footer-cellen i kolumnen för poäng som nyttjas, detta för att skriva ut totalpoängen för de sparade kurserna.

Containern som innesluter tabellen tilldelades en maxbredd på 1370 pixlar och en marginal på 10 pixlar ovanför och 50 pixlar under innehållet. Tabellen har en maxhöjd på 600 pixlar och en maxbredd på 1000 pixlar med dess horisontella overflow dold.



På skärmar mindre än 1140 pixlar döljs kolumnen för ämne och på skärmar mindre än 812 pixlar döljs kolumnen för kurskod.

Knapparna för att ta bort kurser har en röd bakgrundfärg som blir något mörkare vid hover.

I TypeScript-koden för komponenten deklarerades först vilka kolumner som skulle visas i tabellen med en array. En *'MatTableDataSource'* används för att hantera datan i tabellen och kurslistan laddas in via *'MyScheduleService'*.

När komponenten initieras anropas en funktion som laddar in de sparade kurserna i tabellen. Funktionen hämtar kurserna med hjälp av en skapad metod som hämtar lagrade kurser. En funktion för att ta bort kurser skapades också.

För att räkna ut totalpoängen för alla sparade kurser användes en metod som summerar poängen för varje kurs i *'datasource'* och returnerar totalsumman.

Kurskod	Kursnamn	Ämne	Poäng	Kursplan	Ta bort
AK001U	Grundläggande dokumenthantering för registratorer	Arkiv- och informationsvetenskap	7.5	<a href="#">Kursplan</a>	
AK003G	Processbaserad dokumenthantering och arkivredovisning	Arkiv- och informationsvetenskap	15	<a href="#">Kursplan</a>	
			Totalt:	22.5 hp	

Figur 8. Komponentens Myschedulelist

#### 4.4.1.8 Komponent: mainmenu

Komponenten *'mainmenu'* utgör huvudmenyn på webbplatsen och är importerad till projektets *app.component.html*-fil.

Strukturen i komponenten är uppbyggd med en container som omsluter allt övrigt innehåll. Högst upp i containern placerades en länk som leder till komponenten *'home'* och som innehåller universitets logotyp. Därefter placerades ett nav-element innehållandes dels en oordnad lista för navigering på stora skärmar, dels en oordnad lista för navigering på små skärmar. Listelementen använder sig av Angulars RouterLink och RouterLinkActive för att möjliggöra intern navigering. Ovanför listelementet för små skärmar har



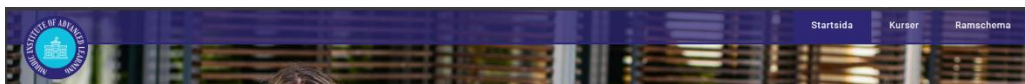
två span- och ett input-element av typen checkbox placerats för att skapa strecken till hamburgermenyn.

Containern som omsluter allt annat har tilldelats den mörkare lila färgen med viss transparens som bakgrundsfärg. Den har en minsta höjd om 4.5em och en nedre border, utan transparens, på 4 pixlar. Positionen är satt till fixed och den har en bredd på 100%.

Logotypen definierades med en absolut positionering och en border-radius på 50%. Maxbredden sattes till 120 pixlar. Nav-elementet tilldelades en flex layout och innehållet justerades till flex-end. Bredden sattes även till 100%. På stora skärmar är hamburgermenyn dold.

På skärmar mindre än 680 pixlar visas hamburgermenyn och den horisontella navigeringen döljs. Hamburgermenyn animerades och vid klick roteras de tre strecken till att visa ett kors istället. Animeringen och interaktiviteten för hamburgermenyn styrs med hjälp av transitioner som aktiveras när användaren klickar på menyns checkbox.

I TypeScript-filen för komponenten hanteras öppning och stängning av hamburgermeny vid klick. Genom en funktion lades klickhändelser till på varje menylänk så att när en länk klickas stängs menyn automatiskt genom att checkboxen för hamburgermenyn avmarkeras.



Figur 9. Komponent mainmenu

#### 4.4.1.9 Komponent: footer

Komponenten 'footer' finns med på samtliga av webbplatsens sidor och utgör som namnet antyder webbplatsens footer. Footerns struktur är uppbyggd med ett footer-element som i sin tur innehåller en container. Containern inuti footer-elementet innehåller ytterligare tre stycken containrar med textinnehåll i var och en.

Footer-elementet har tilldelats moodboardens ljuslila färg som bakgrundsfärg och satt till en bredd på 100%. Containern inuti footern har positionerats relativt med en flex layout. Innehållet i containern positioneras med justify-content: space-between för att få en jämn fördelning av mellanrum mellan containrarna.

På skärmar mindre än 600 pixlar ändras footerns flex-direction till column istället för row.



Figur 10. Komponent footer

#### 4.4.2 Services

Två stycken services skapades i projektet, en för att hämta och hantera kursdata och en för att hantera kurserna i ramschemat som sparas till localStorage. De metoder som finns tillagda för att hantera kurserna i ramschemat är en för att lägga till kurs till localStorage, en för att hämta kurser från localStorage, en för att radera enstaka kurs från localStorage och en för att rensa hela localStorage.

##### 4.4.2.1 Service: *CourseDataService*

Service *'CourseDataService'* skapades för att hämta kursdata från den JSON-fil som innehåller kursobjekten som visas i kurstabellen. Service är markerad som injectable vilket innebär att den kan injiceras i andra komponenter i hela applikationen.

I servicens konstruktor injiceras Angulars *'HttpClient'* vilket möjliggör http-anrop. Service innehåller en funktion som returnerar en ström av kurser från den URL där JSON-filen är publicerad.

##### 4.4.2.2 Service: *MyScheduleService*

Service *'MyScheduleService'* skapades för att hantera användarens sparade kurser i ramschemat. Service inkluderar flera olika funktioner. Den första funktionen kontrollerar om localStorage är tillgängligt och fungerar på användarens dator. Den andra funktionen lägger till en kurs till schemat genom att först hämta de redan tillagda kurserna från localStorage för att därefter lägga till den nya kursen och sedan spara den uppdaterade listan. Den tredje funktionen hämtar och returnerar alla sparade kurser från schemat. Den fjärde funktionen tar bort en kurs från schemat baserat på dess kurskod och den sista funktionen rensar samtliga kurser ifrån schemat och localStorage.

#### 4.4.3 Responsivitet

Webbplatsens responsivitet fanns i åtanke under hela utvecklingsarbetet genom att placera elementen på ett sätt så att dessa ser bra ut på olika skärmstorlekar och användande av flexbox där det var möjligt. I många av komponenterna användes också media queries för att få sidan att se bra ut på olika skärmstorlekar. Inga specifika brytpunkter för media queries användes utan istället granskades respektive sida i webbläsaren Edge:s DevTools för att upptäcka var media queries behövdes för de olika elementen. Mer information om var och när media queries användes går att läsa om under respektive komponentrubrik.

## 5 Resultat

Projektets övergripande mål var att skapa en webbplats i ramverket Angular åt det fiktiva universitetet NIAL. Webbplatsen skulle innehålla en startsida med information om universitetet samt två undersidor, varav den ena skulle innehålla en tabell med universitetets kurser och den andra skulle innehålla en lista med kurser som användaren sparar till sitt ramschema.

Detta mål bröts ned i mindre och mer konkreta delmål som går att läsa om under kap. 1.3 konkreta och verifierbara mål.

Webbplatsen består av tre sidor - en startsida, en kurssida och en sida för ramschema. Den är byggd i Angular, med nio olika komponenter, och navigeringen sköts med hjälp av routing. Två services skapades, en för att hantera kursdata och en som hanterar metoderna för att lägga till, spara och ta bort kurser i ramschemat som lagras i localStorage.

De kurser som hämtas från JSON-filen är strukturerad i en mat-table och det går att sortera kurserna i kolumnerna kurskod, kursnamn, ämne och poäng. Det går inte att sortera kurserna efter kursplan eller lägg till eftersom en sådan sortering inte skulle ändra ordningen på kurserna. Det går att söka bland kurserna genom fritextsökning och det går att filtrera kurserna efter ämneskategori. Det går också att kombinera filtreringen med fritextsökning för att minska sökträffarna ytterligare. Tabellen är paginerad och användaren kan själv välja hur många sökträffar denne vill visa per sida. Bredvid pagineringen går det även att utläsa hur många kurser som visas.

På sidan för ramschema struktureras kurserna också i en mat-table. Under kurslistningen kan användaren utläsa det totala antalet poäng som summan av kurserna består av.

Webbplatsen är responsiv och har noggrant kontrollerats för att den ska se snygg ut på alla olika typer av skärmar.

Samtliga av målen som angavs i kap. 1.3 konkreta och verifierbara mål har därmed uppfyllts.

## 6 Slutsatser

Det har varit lärorikt att skapa ett projekt med Angular, men också svårt, då jag har fått kämpa en hel del med hur jag på bästa sätt skulle strukturera applikationen. I slutändan känner jag mig inte riktigt nöjd och tycker nog att det var onödigt att skapa så många olika komponenter med tanke på hur lite innehåll sidorna har. Om jag fått göra om projektet hade jag nog nöjt mig med att skapa en komponent för menyn, en för footern och en för respektive sida på webbplatsen. Min känsla är att jag mina Angular-kunskaper är för grundläggande för att på bästa sätt utnyttja hela potentialen i ramverket.

I början av projektet lade jag mycket tid på att utforska olika komponentbibliotek till Angular. Jag landade till slut i Angular Material då jag tyckte att dokumentationen till biblioteket var omfattande och bra. Jag känner att mina förväntningar på dokumentationen uppfylldes och jag är glad att jag valde Material då det har underlättat utvecklingsarbetet väldigt mycket. Jag gissar att konstruktionen av webbplatsen skulle tagit mycket längre tid om jag valt att skapa tabellkomponenterna själv från grunden.

En annan del av projektet som däremot varit desto mer tidskrävande är utvecklandet av TypeScript-kod. Jag känner mig fortfarande inte riktigt hemma när det kommer till statisk typning och användandet av interfaces, vilket har gjort att jag behövt läsa mig till mycket. I slutändan känner jag mig dock nöjd över logiken. Lite extra nöjd är jag över att ha fått fritextsökningen och ämnesfiltreringen i kurstabellen att fungera tillsammans och inte bara enskilt.

Allt som allt har det varit ett roligt projekt att arbeta med, men jag känner också att jag hade kunnat vidareutveckla det mycket mer om mer tid funnits. Det uppfyller grundkraven och har extra funktionalitet som paginering och en kombinerad filtrerings- och sökfunktion.

## Källförteckning

- [1] Microsoft. TypeScript - JavaScript that scales. [Internet]. Typescriptlang.org. 2015 [cited 2024 Sep 15]. Available from: <https://www.typescriptlang.org/>
- [2] TypeScript [Internet]. Wikipedia. 2020 [cited 2024 Sep 15]. Available from: <https://en.wikipedia.org/wiki/TypeScript>
- [3] Moment 1 - Teori [Internet]. Github.io. 2024 [cited 2024 Sep 15]. Available from: [https://matdah.github.io/DT208G---Programming-i-TypeScript/Moment%201/moment\\_1\\_teor.html](https://matdah.github.io/DT208G---Programming-i-TypeScript/Moment%201/moment_1_teor.html)
- [4] Wikipedia Contributors. Angular (web framework) [Internet]. Wikipedia. Wikimedia Foundation; 2019 [cited 2024 Sep 15]. Available from: [https://en.wikipedia.org/wiki/Angular\\_\(web\\_framework\)](https://en.wikipedia.org/wiki/Angular_(web_framework))
- [5] Moment 3 - Angular I - Teori [Internet]. Github.io. 2016 [cited 2024 Sep 15]. Available from: [https://matdah.github.io/DT208G---Programming-i-TypeScript/Moment%203/moment\\_3\\_teor.html](https://matdah.github.io/DT208G---Programming-i-TypeScript/Moment%203/moment_3_teor.html)
- [6] W3C. The W3C Markup Validation Service [Internet]. W3.org. 2013 [cited 2024 Oct 15]. Available from: <https://validator.w3.org/>
- [7] AChecker+ Web Accessibility Checker | ACHECKS [Internet]. achecks.org [cited 2024 Oct 15]. Available from: <https://achecks.org/achecker>
- [8] Lighthouse – Chrome Web Store [Internet]. Chromewebstore.google.com [cited 2024 Oct 15]. Available from: <https://chromewebstore.google.com/detail/lighthouse/blipmdconlkpinefehnmjammfjpmpbjk?hl=sv&pli=1>