Webbutveckling i CMS

Föreläsning 4

NORTH LINK

Projekt!

- Idag ska vi börja arbeta med inlämningsuppgiften
- Er uppgift är att bygga en webbshop med Cockpit som backend
- Det finns en PDF på PingPong
- Uppgiften ska lämnas in senast kl 08:00 onsdag 18 september
- Precis som på labben får ni använda React, Vue eller Angular
- Ni är fria att själva bestämma vilken typ av produkter webbshoppen säljer!

Muntlig redovisning

- Projektet ska redovisas muntligt onsdag 18 september
- Jag återkommer med redovisningsschema vid ett senare tillfälle
- Under redovisningen ska ni presentera applikationen ni har byggt

Implementation

- Ni ska bygga en SPA i antingen Vue, React eller Angular som använder Cockpit som backend.
- Applikationen ska vara en webbshop som innehåller
  - en huvudsida där produkterna listas
  - en produktsida som visar information om en produkt och där användaren kan lägga till produkten i en varukorg
  - en sida som visar innehållet i varukorgen och där användaren kan gå vidare till utcheckning
  - en utcheckningssida där användaren skriver in namn och adress
  - en sida som visas när en beställning är klar
- Det finns ett par extra krav för betyget VG som kommer presenteras senare

Huvudsida

- Huvudsidan ska visa en lista av produkter med namn, pris och en bild på varje produkt
- Det ska gå att söka på produkternas namn
- Det ska finnas en kryssruta för att endast visa produkter som finns i lager

Produktsida

- Produktsidan ska visa information om en produkt och ska visa minst
  - namn på produkten
  - en beskrivning
  - pris
  - lagersaldo
  - ett bildgalleri
- Det ska gå att lägga till produkter i varukorgen på den här sidan och det ska gå att välja hur många produkter man vill lägga till
- Sidan ska visa en lista av recensioner. En recension ska ha en titel, ett innehåll och ett betyg mellan 1 och 5

Varukorgen

- Den här sidan ska visa en tabell över produkter som ligger i varukorgen. Inkludera namn, antal och pris.
- Skriv ut det totala priset under tabellen.
- Om det ligger produkter i varukorgen ska det finnas något sätt att gå vidare till en sida där man kan beställa varorna
- Varukorgen ska implementeras med localStorage så att den inte töms när sidan laddas om
- Innehållet i varukorgen ska inte sparas i Cockpit

Utcheckning

- Utcheckningssidan ska ha ett formulär där användaren kan skriva in namn och adress
- När en order har lagts ska användaren skickas vidare till en enkel sida som berättar att ordern är färdig och att produkterna är på väg
- Att spara ordern i Cockpit krävs endast för betyget VG

Collections i Cockpit

- För att kunna lägga in produkter och recensioner behöver ni skapa två collections i Cockpit
- Produkterna ska minst innehålla
  - Namn
  - Beskrivning
  - Pris
  - Bilder
  - Lagersaldo
- Recensionerna ska minst innehålla
  - Titel
  - Innehåll
  - Betyg

Extra krav för VG - Spara ordrar

- Det finns ett par extra krav för betyget VG
- För att få betyget VG behöver färdiga ordrar sparas i Cockpit. För att göra det behöver ni lägga till en extra collection och ni behöver minst spara
  - Namn
  - Adress
  - Totalt pris
  - Lista över produkter och hur många av varje produkt kunden köpte
- För att kunna skapa en lista över produkter i ordern kan ni använda typen “Repeater” i Cockpit https://github.com/agentejo/cockpit/wiki/Collection-Field-Types#nested-sets-within-repeaters

Extra krav för VG - Recensioner

- För att få betyget VG ska det gå att skriva recensioner på produktsidorna
- När en användare skriver en ny recension ska den sparas till Cockpit

Krav

- Applikationen ska vara en SPA som använder React, Vue eller Angular
- Cockpit ska användas som backend
- Ni ska använda korrekt routing
- Huvudsidan ska lista produkter med namn, pris och en bild
- Huvudsidan ska ha pagination
- Det ska gå att söka efter produkter och filtrera produkter så att endast de som finns i lager visas
- Produktsidorna ska visa namn, beskrivning, pris, lagersaldo och en bildgalleri
- Produktsidorna ska visa en lista av recensioner
- Det ska gå att lägga till nya produkter i kundkorgen
- Kundvagnen ska implementeras med localStorage
- Utcheckningssidan ska ha ett formulär där kunden kan skriva in namn och adress
- Kundvagnen ska tömmas när användaren skickar in formuläret
- Använd typen “Collectionlink” i Cockpit för att länka recensioner till produkter

Krav (VG)

- Användarna ska kunna lägga till nya recensioner på produktsidorna
- När användaren lägger en order ska den sparas till Cockpit och datan som sparas ska innehålla minst
    - Namn
    - Adress
    - Totalt pris
    - Lista över produkter och hur många av varje produkt som kunden köpte

Generera statiska sidor

- Det finns många sidor där allt innehåll kan genereras i förväg och där vi inte behöver något dynamiskt innehåll
- Detta brukar kallas “statisk generering” och det finns många olika verktyg vi kan använda oss av
- Några vanliga exempel på typer av sidor som kan vara statiska är
  - Dokumentation
  - Nyhetssidor och bloggar
  - e-böcker
  - Landing pages

Här finns en lista på några olika verktyg

https://www.staticgen.com/

Fördelar med statiska sidor

- En statisk sida behöver inte hämta något innehåll från en databas så vi behöver ingen backend
- Det är enkelt att cacha innehållet eftersom alla användare kommer få samma filer
- Statiska sidor är ofta snabbare att ladda eftersom vi inte behöver kommunicera med en backend

Gatsby

- Gatsby är ett verktyg som används att generera statiska sidor med hjälp av React
- Med Gatsby kan vi skriva vår sida som en vanlig React-applikation och sedan bygga den som en statisk sida.
- Gatsby kommer då generera HTML-filer och CSS-filer som vi kan ladda upp till en webbserver

https://www.gatsbyjs.org/

Installera Gatsby

- Gatsby är skrivet i JavaScript och går att installera med npm

```
npm install -g gatsby
```

- Vi kan skapa ett nytt projekt med gatsby genom att skriva

```
gatsby new NAME
```

Köra en utvecklingsserver

- Precis som med create-react-app kan starta en utvecklingsserver medan vi utvecklar sidan
- Utvecklingsservern startas genom att köra

```
gatsby develop
```

- Om vi gör några ändringar i koden kommer sidan automatiskt laddas om

Bygga en produktionsversion

- När man har byggt en sida med Gatsby kan man bygga statiska filer genom att köra

```
gatsby build
```

Sidor i Gatsby

- Det går att lägga till nya sidor i Gatsby genom att lägga till nya JS-filer i “pages”-katalogen
- Om ni startar en ny sida med Gatsby kommer ni se att det redan finns två sidor i katalogen
- Det går att länka till sidor med Link-komponenten

Övning - Gatsby

1. Installera Gatsby och skapa en ny sida
2. Starta utvecklingsservern och öppna sidan i en webbläsare
3. Utforska strukturen för exempelsidan som skapats och bygg en företagssida med en huvudsida och två undersidor. Undersidorna kan t ex vara “Om företaget” och “Lediga jobb”
4. Lägg till länkar till undersidorna i headern
5. Bygg en produktionsversion och publicera den med GitHub Pages

GraphQL

- I Gatsby används ett språk som heter GraphQL för att efterfråga data som visas på sidan
- I exempelsidan som skapas när ni kör “gatsby new” finns redan lite GraphQL-kod som hämtar information om sidan

```
graphql`
  query SiteTitleQuery {
    site {
      siteMetadata {
        title
      }
    }
  }
`
```

GraphQL

- GraphQL-koden på förra sliden hämtar ut sidans titel ur sidans metadata
- Ni kan ändra sidans metadata i filen “gatsby-config.js”
- I det här fallet används GraphQL för att hämta data ur ett JavaScript-objekt men det går att använda GraphQL för att hämta data ur väldigt många olika källor
- GraphQL är ett språk som är utvecklat specifikt för frontendapplikationer

Ni kan läsa mer om GraphQL här
https://graphql.org/learn/

GraphiQL

- GraphiQL är en applikation som inkluderas när vi kör Gatsbys utvecklingsserver
- Den är bra för att utforska datan som går att efterfråga med GraphQL och för att lära sig GraphQL

När ni har startat utvecklingsservern kan ni komma åt GraphiQL på

http://localhost:8000/___graphql

Gatsby plugins

- Det finns väldigt många plugins som man kan använda i Gatsby
- På Gatsbys officiella hemsida kan ni hitta en lista på de mest populära https://www.gatsbyjs.org/plugins/
- Plugins installeras med npm och används genom att lägga till dem i “gatsby-config.js”

Läsa in data med Gatsby

- För att kunna använda Gatsby på ett bra sätt behöver vi kunna läsa in data så att vi har någonting att generera våra sidor med
- I Gatsby läses data in med s.k. “sources”. De är separata moduler på npm som vi kan inkludera för att kunna läsa in data från all möjliga källor
- Det finns moduler för att läsa in data från filsystemet, olika Headless CMS, osv.

https://www.gatsbyjs.org/tutorial/part-five/

gatsby-source-filesystem

- Den enklaste källan att läsa ifrån är från filsystemet
- gatsby-source-filesystem läser in filer som vi sedan kan efterfråga med GraphQL

https://www.gatsbyjs.org/packages/gatsby-source-filesystem/

gatsby-transformer-json

- När filer läses in med “gatsby-source-filesystem” behöver de ofta transformeras på något vis
- Pluginet “gatsby-transform-json” tar JSON-filer som läses in och konverterar dem till JavaScript-objekt

Om vi lägger till följande till “gatsby-config.js” kommer vi kunna lägga JSON-filter i katalogen “src/data” och efterfråga innehållet med GraphQL.

```
`gatsby-transformer-json`,
{
  resolve: `gatsby-source-filesystem`,
  options: {
    name: `data`,
    path: `${__dirname}/src/data`,
  },
},
```

NORTH LINK

Övning - Lista över frukter

1. Skapa ett nytt projekt med Gatsby
2. Lägg till “gatsby-source-filesystem” och “gatsby-transformer-json” på samma vis som i föregående slide
3. Skapa en katalog som heter “src/data/fruits/” och skapa några JSON-filer i den med information om frukter. T ex i följande format

```
{
  “name”: “Orange”,
  “description”: “Oranges grow on trees.”
}
```

4. Testa att efterfråga datan med GraphiQL
5. Ändra “src/pages/index.js” så att den efterfrågar en lista av frukter med GraphQL och renderer ut dem på sidan

Skapa nya sidor från data

- Vi kan enkelt lägga till nya sidor genom att skapa en ny fil i “src/pages”-katalogen
- Men hur gör vi om vi vill skapa sidor baserat på någon data som vi kan efterfråga med GraphQL? T ex skulle vi kanske vilja skapa en ny sida för varje frukt som finns definierad i våra JSON-filer

På den här sidan beskrivs hur vi kan lägga till nya sidor.

https://www.gatsbyjs.org/tutorial/part-seven/

Övning - Mer frukt

1. Utgå från sidan med frukt som vi skapade i förra övningen
2. Nu vill vi istället för att visa informationen på huvudsidan, skapa en sida för varje frukt och att huvudsidan länkar till varje sida
   Nya sidor läggs till med funktionen “createPage” som beskrivs i guiden på förra sliden.