Projekt Exam – EG

Indholdsfortegnelse

Indledning

Kort resume

I denne rapport gennemgår vi udviklingen af en webapplikation, som skal gøre det muligt for forbrugere af byggemarkeder, at scanne forskellige varer og blive mindet om hvilke varer de mangler at købe for at kunne bruge det givent produkt, samt vise andre relevante varer til det produkt. Denne rådgivning skal let kunne fjernes på en brugervenlig måde, hvis brugeren ved hvad de går efter. Løsningen udvikler vi i Node.js, med et REST interface og brug af MySQL som database.

Problemformulering

Problemstilling

ASPECT4 Trælast/DIY mobile salgs app har brug for en brugervenlig model løsning som kan hjælpe slutbrugeren af produkterne, igennem en indkøbs- og betalingsproces, der sikre at brugeren har husket alt der skal til, for at kunne gennemføre deres projekt og derved undgår gentagende indkøb. Der er brug for en løsning som hjælper både den private kunde, samt den professionelle kunde, og som også giver dem muligheden for helt at fravælge at få rådgivning og hjælp til indkøbslisten. App'en skal også hjælpe med valget af betalingsløsninger, som imødekommer de moderne løsninger der i dag er tilgængelige ved hånden, i form af dankort og mobilepay, samt fakturerings kørsel til de professionelle. Der forespørges desuden på en brugervenlig løsning med hensyn til produkt lokationer i byggemarkedet, til fordel for både de ansatte som skal scanne og registrere varen ved køb, og forbrugerne der ønsker at kigge efter varen på egen hånd.

Problemformulering

"Hvordan kan vi udvikle en brugervenlig og sikker mobil applikation til EG som skal bruges til forbedret varehåndtering og indkøb i byggemarkeder ved hjælp af node.js og mysql?"

- Hvordan kan vi skabe en betalingsproces som tager højde for alle nødvendige betalingsmuligheder, på en brugervenlig måde?
- Hvordan kan vi bygge et REST interface som kan stille data til rådighed for klienten?
- Hvordan kan vi skabe en database, som tager højde for brugeroplevelsen i forhold til varehåndtering og indkøb?

Anvendelse/opsætning af projekt

Installer databasen i mappen "data" med filen data.sql. Med dette følger databasen med krævede tabeller til brug af applikationen. Efter dette kan applikationen køres fra terminalen ved at køre "npm run nodemon" eller "npm run test".

Metodeovervejelser

Planlægning

For at planlægge vores fremgang og holde styr på, hvilke mål der skal opnås samt prioriteten derpå, bruger vi Trello til opsætning og overblik over projektet som helhed. På denne måde sikre vi os at komme i mål med så meget som muligt, og mindske risikoen for at overse de forskellige aspekter i projektet.

Teknologier

I dette projekt vil vi gøre brug af følgende teknologier.

- Node.js som vores web server løsning.
- Moduler:
    - Express til opbygning af REST interface.
    - MySQL for at forbinde vores server til vores MySQL database.
    - Nodemon bruger vi til at gøre udvikling af applikationen nemmere for os selv, da den automatisk genstarter vores server, hver gang en rettelse gemmes.
    - EJS bruger vi som vores template view engine, til at vise vores data for brugeren.
    - bcrypt som vi bruger til hashing af passwords før de bliver gemt i databasen, som et led af sikkerhed
- MySQL bruger vi som database, for at lagre de forskellige produkter og deres relation til hinanden.
- Git til lokal versionsstyring af vores webapplikation.
- GitHub bruger vi til at dele vores lokale filer og ændringer i projektet med hinanden.

Research

Sikkerhed

bcrypt

Vi vil bruge bcrypt[1] som øget sikkerhed i lagringen af passwords. Dette npm modul gemmer de indtastede passwords som hashed når der oprettes en bruger i vores applikation. Dette skaber en sikkerhed for den potentielle bruger ved ikke direkte at gemme deres password og i stedet sammenligne når der skal logges ind.

Login

For at sikre den bedst mulige login bruger vi nmls best practice opskrift til login-rækkefølgen.[2]

[1] Bcrypt dokumentation
https://www.npmjs.com/package/bcrypt
[2] NMLA login best practice
http://dkexit.eu/webdev/site/ch46s03.html

Helmet

Vi vil bruge helmet[3] til øget sikkerhed i vores express applikation. Helmet er middleware som skaber bedre sikkerhed ved at sætte forskellige HTTP headere.

- npm install helmet --save
    - (--save putter den direkte i dependencies)

Validator

Validator modulet bruges til både at validere og sanitize bruger inputs. Validator.js har blandt andet en dependency til Sanitizer, som gør det muligt at benytte en masse sanitizer metoder.

Express-session

Modulet gør det muligt at oprette session middleware. Session sikre at brugeren automatisk får slettet alle server cookies gemt i session ID, så snart brugeren lukker appen, eller at sessionen udløber.

Cookie

Cookie modulet er en basic HTTP cookie parser og serializer for HTTP servere. Modulet gør det muligt at oprette cookies samt parse dem med nogle options som deres eget objekt tilknyttet den enkelte cookie. Cookies gemmes lokalt, på klient siden, og gør det derfor muligt at vende tilbage til samme app, hvor den har husket de ting som bliver gemt i en cookie, eks. inputs og indkøbskurve.

Cookie parser

Cookie parseren tager på request, cookies fra headeren og opsætter dem som objekter. Objektets egenskaber bliver gemt i req.cookies, så hvert objekts key er cookiens navn.

Analyse

EG database og tabeller

For at forstå og opbygge den korrekte struktur af databasen med henblik på tabeller og dets rækker og kolonner, har vi analyseret det excel ark som Steen har udleveret “Varegruppe 4740 lager oversigt”, som repræsentere et eksempel på hvordan én varegruppe ser ud med produktoversigt. Vi har forsøgt at genskabe den samme database med lidt ekstra, ud fra vores forestilling om hvordan den ser ud når strukturen for et produkt er som i det ark. Tabellen for et produkt tager også højde for dets lokation, i form af nær- og fjern-lokation. Vores tabel repræsentere et udkast over hvordan en database kunne se ud, hvor der er sammenhæng/forhold mellem tabeller som leverandør, produkter, kategorier og lokationer. Databasen skulle dog udbygges med en del mere data, i form af generel lokationsdata ud fra et valgt varehus, samt data om lagerbeholdning, hvor der kan forbindes flere varehuse til det

[3] Helmet middleware
https://helmetjs.github.io/

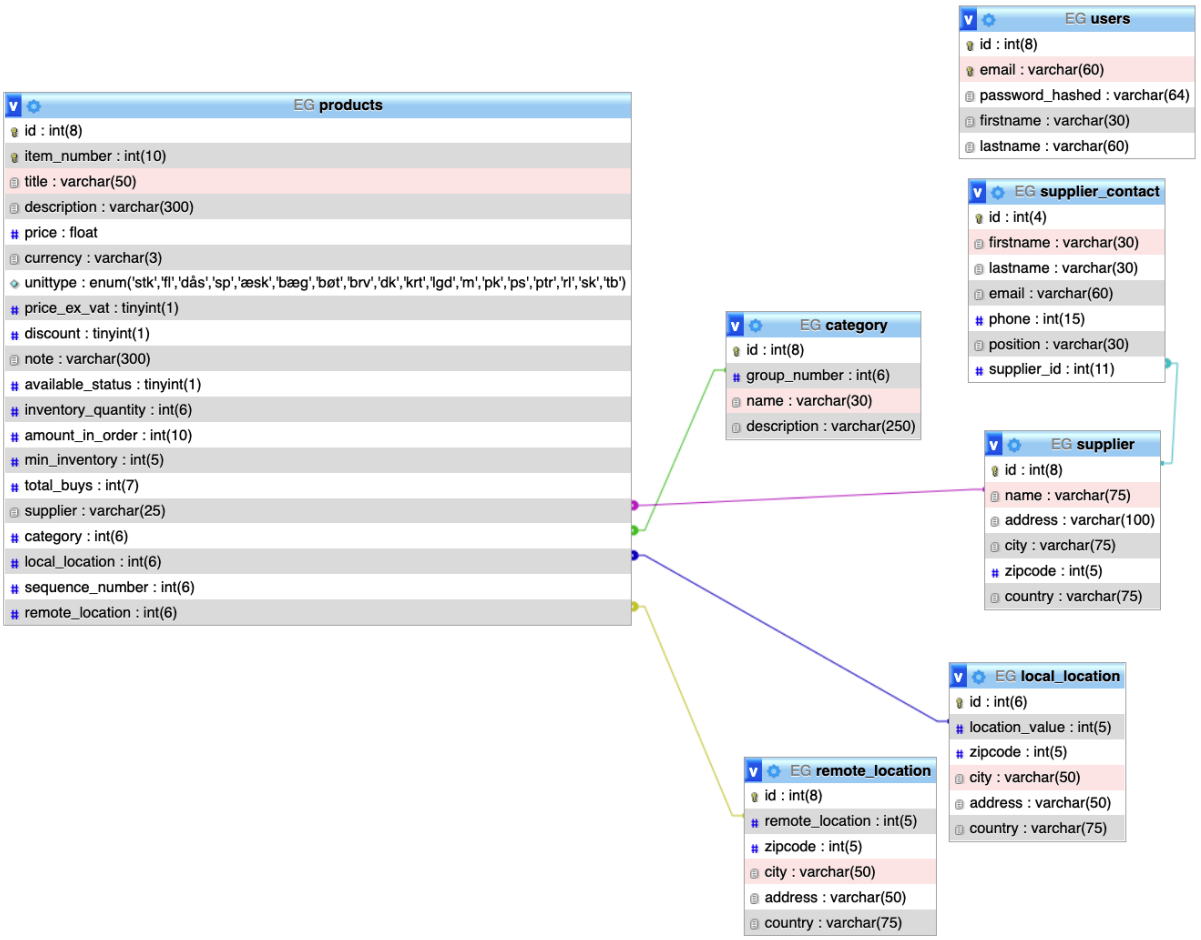
enkelte produkt, for at præsentere hvorhenne produktet er på lager, sammen med dets nøjagtige placering i varehuset.

“Påmind mig senere” cookie

I forbindelse med indkøbskurven i appen, har vi oprettet en cookie som brugeren sætter når de vælger “påmind mig senere”. Cookien er programmeret til at slette sig selv så snart brugeren har tilføjet mere end 5 produkter til deres kurv, fra brugeren har trykket “påmind mig senere” og dermed oprettet cookien. På den måde sikre vi os at brugeren stadigvæk bliver påmindet om rådgivning undervejs i handelsprocessen, og får den hjælp der skal til for at undgå gentagne indkøbsture. I et brugervenligt aspekt, ville det give mening at give brugeren mulighed for selv at vælge, hvornår de skulle præsenteres for rådgivningen igen, via et antal valgmuligheder, heriblandt først at se rådgivningen når de besøger siden med deres indkøbskurv.

Konstruktion

Database



Web

Pop-up

Vi har valgt at lave vores pop-up løsning til visning af supplerende og relevante produkter, i form af routes, hvor vi sender et produkt *id* med som vi bruger til at filtrerer dataen der vises i pop-uppen og renderer siden igen, med pop-uppen synlig.

```
const product = await queryController.handleQuery(`SELECT * FROM products WHERE id=${req.params.id}`);
const products = await queryController.handleQuery(`SELECT * FROM products`);
const othersBought = await queryController.handleQuery(`SELECT * FROM products WHERE id!=${req.params.id}
```

Vi bruger vores “query controller” til at håndtere hvilke data vi gerne vil hive ud fra databasen, ved at skrive vores SQL query i vores *handleQuery* metode. Vi gemmer denne data i de respektive konstanter så vi kan sende det videre til brugeren ved brug af render metoden.

```
res.render('products', {
    title: 'Alle Produkter',
    products: products,
    product: product[0],
    popUp: true,
    neededProducts: neededProducts,
    extraProduct: req.body.title,
    othersBought: othersBought
});
```

Denne data renderer vi på vores produktside, pop-uppen vises, når “popUp” key værdien er sat til “true”.

Express-session modulet anvender vi til at starte en session når brugeren tilføjer noget til deres kurv. Sessionen bruges til at gemme data server-sided, eks. når brugeren tilføjer produkter til kurven og mødes af “har du husket” pop-uppen, så vil afvisningen af denne hjælp blive gemt i en session ID. Sessionen holder på denne data, indtil browseren lukkes eller der logges ud af appen, eller at sessionen udløber på tid.

```
res.cookie('isPopUpDisabled', 'true');
```

Vi bruger cookie parseren til at håndtere vores pop-up “har du husket” listen, da vi gemmer en key:value inde i den cookie, bestående af et navn og boolean (“isPopUpDisabled”: ”true”). Denne cookie kigger på om brugeren har trykket afvis til rådgivningen. I så fald vil cookien blive gemt i browseren, og starter en **counter** der kigger på hvor mange produkter brugeren har lagt i kurven igen efterfølgende, før den viser pop-uppen igen.

Login

Vi har valgt at lave et loginsystem til vores applikation, i forhold til at brugeren vil kunne have en profil, hvis ønsket, med øget sikkerhed i forbindelse med bcrypt, validator og helmet modulerne. Vi har opdelt login i signup og login.

Når brugeren submitter på signup formen, sker følgende:

1. Emailen bliver gemt i en variabel til senere at blive valideret med **isEmail()**
2. Kodeordet bliver hashed med bcrypt inden det gemmes i en variabel.
3. Inden navn og efternavn bliver gemt i deres respektive variabler, bruges metoden **toLowerCase** efterfulgt af validator metoden **whitelist()** med en RegEx der kigger efter alle lowercase bogstaver samt æøå (“a-zæøå”), for at fjerne alle andre uønskede tegn.
4. Så tjekkes der om alle input felter er blevet udfyldt, samt mailen tjekkes med **isEmail()** som, hvis den vender tilbage som “true”, indsætter informationen ind i databasen.

Når brugeren submitter på login formen, sker følgende:

1. Hvis emailen og kodeordet er tastet ind og **isEmail()** vender tilbage med “true”, vælges brugerens gemte kode fra databasen, ud fra emailen.
2. bcrypt.compare() tager det indtastede kodeord og vejer det op imod databasens kodeord hvis brugeren eksisterer.

3. Hvis koden er rigtig, oprettes en cookie med brugerens navn til senere brug (under profil siden) og brugeren sendes videre til forsiden.

NPM moduler

Validator

Vi benytter **whitelist** metoden som kræver et RegExp til at fjerne karakterer som ikke er på vores "whitelist", der er skrevet til kun at godtage "a-zæøå". Således kan brugeren ikke gemme navn og efternavn med et tal eller specialtegn i databasen. Inden vi tjekker for "a-zæøå", bruger vi metoden toLowerCase() på brugerens input, da vi gerne vil have det indsat i databasen med lowercase, så vi behøver kun at whitelist lowercase tegn.

Validatoren i sig selv benytter vi til at validere om inputtet er en email. **isEmail** er den metode vi anvender til at kigge på hvorvidt en string er en email adresse eller ej. Metoden har en del options man kan benytte, hvor vi benytter dens default da den opfylder vores behov om validering i UTF-8.

Cookie

Vi bruger cookie modulet til at sætte en cookie i klientens browser, ved brug af appen, da vi vil undgå at risikere at indkøbskurven bliver tømt hvis brugeren uheldigvis lukker vinduet, eller at vores session udløber fordi brugeren ikke interagere med appen i et givent stykke tid. Cookie modulet kan i princippet undværes, da vi benytter session modulet i express. Vi har dog valgt at beholde cookie modulet, fordi vi indtænker brugervenlighed med risikoer for at man lukker appen eller bliver forstyrret, og dermed ikke ønsker at få tømt hele sin kurv. Vi sætter to cookies, én for login, som validere at brugeren er logget ind og kan derfor se bruger-siden, og én for afvisning af "har du husket" rådgivningen, der popper op når en bruger tilføjer et produkt til kurven.

Helmet

Vi bruger helmet modulet til at skabe øget sikkerhed på vores website. Dette er et middleware modul som sættes på applikationen, og øger sikkerheden ved at sætte forskellige HTTP headers. Vi har valgt at anvende modulet, da det ikke kræver mere end 2 linjer kode, men skaber et ekstra lag sikkerhed. Modulet sætter 15 HTTP headers, som bla. har følgende funktioner:

- Content-Security-Policy
    - Hælper med at afbøde cross-site scripting-angreb
- Cross-Origin-Opener-Policy
    - Mulighed for at sikre, at et dokument på øverste niveau ikke deler en browserkontekstgruppe med dokumenter med krydsoprindelse
- X-Content-Type-Options - nosniff
    - Giver dig mulighed for at undgå MIME-type-sniffing ved at sige, at MIME-typerne er bevidst konfigureret
- Origin-Agent-Cluster
    - Giver en mekanisme til at tillade webapplikationer at isolere deres oprindelse
- Osv.

Wireframe

Udarbejdelsen af vores wireframes er foregået i forlængelse af Steens ønsker samt inspiration fra EGs egen ASPECT4 hjemmeside, hvor vi har draget inspiration fra.

Vi har valgt den lyserøde farve fra ASPECT4 og genereret en farvepalette vha. https://mycolor.space. Vi har anvendt farverne ved at bruge indikerede knapper som skaber genskær igennem hele applikationen, og holder farverne på andre kasser, tekst og billeder simple, da det ikke skal være mere end det er, en applikation til handel i byggemarkeder.

Evaluering af proces

Vi begyndte hele projektet med en overordnet planlægning og struktur for hvordan forløbet skulle foregå, heriblandt uddelegering af opgaver, så vi kunne nå så langt som muligt, på så kort tid som muligt. Det lykkedes at dele opgaverne op så alle fik noget de ønskede at beskæftige sig med, og på samme tid mens alle arbejdede på deres eget område, sparede vi også flittigt med hinanden så snart der var spørgsmål eller noget som ikke virkede. Alle 4 medlemmer af gruppen har været engageret og interesseret i at arbejde og udvikle deres kompetencer inden for alle fagligheder som projektet indebar.

Konklusion

"Hvordan kan vi udvikle en brugervenlig og sikker mobil applikation til EG som skal bruges til forbedret varehåndtering og indkøb i byggemarkeder ved hjælp af node.js og mysql?"

Vi har i fællesskab udviklet et brugervenligt mobil wireframe af applikationen som EG forespørger, med fokus på at hjælpe brugerne til at huske at få alle de nødvendige supplerende produkter med, så de undgår flergangs indkøb og tilhørende frustrationer ved at mangle er supplerende produkt, som de måske ikke havde tænkt over.

Dertil er også et udkast på betalingsprocessen blevet designet, der tager højde for brugere samt virksomheder der handler via applikationen i byggemarkedet. I forbindelse med udviklingen af applikationen som webapp, har vi både udviklet et REST interface som gør det muligt at trække data ud på hhv. brugerne i databasen, samt alle produkter der er oprettet i databasen, sammen med et login system til de brugere som ønsker at oprette en profil til at gemme deres indkøb, eller virksomheder som vil registrere deres CVR til fakturabetalinger.

Applikationen er tilknyttet et middleware modul som hedder Helmet. Dette modul tilføjer og ændrer samtlige HTTP headers, sådan at de ikke er synlige, og skaber sikkerhed ved f.eks at fjerne oplysningerne i headeren om at applikationen er drevet af express. Disse ekstra lag af beskyttelse i headeren skaber øget sikkerhed for både brugeren af applikationen og ejerne.
.
På samme tid er login systemet også tilkoblet en validator der verificere login op mod databasens brugerprofiler, således at der ikke kan oprettes ukyndige profiler eller logges ind på dem.

Databasen er udviklet i SQL med fokus på at give et repræsentativt billede af hvordan et udsnit af en database til et varehus kunne se ud, med relationer der forbinder tabellerne med hinanden hvor de er nødvendige. Databasen er desuden designet ud fra den data vi har fået stillet til rådighed, kombineret med et udkast på hvordan det kunne være relevant for virksomheden at opstille deres database i forhold til lagring af leverandører, kontaktpersoner og produkt lokationer.

Referencer

new Promise metode [7min30sek]:
https://www.youtube.com/watch?v=VtepSMk__Ho

Password hashing via bcrypt
https://www.npmjs.com/package/bcrypt

NMLA best practise login
https://docs.google.com/document/d/1gUd6M59Xu_5iIZKP0UEp062NP_WEg8EWWxUK1B4IcT4/edit#

Helmet middleware
https://helmetjs.github.io/

Login system
https://medium.com/@prashantramnyc/a-simple-registration-and-login-backend-using-nodejs-and-mysql-967811509a64

Bilag

Spørgsmål til spørgetime (Steen):

- Hvad er det konkret Steen godt kunne tænke sig at vi hjælper med?
    - Er det et udkast på et brugervenligt UI design af appen med fokus på de sider der viser brugeren relevante produkter ved scanning af en vare, så de husker at få alt med – og at den ansatte der scanner varer ind husker at påminde brugeren?
    - Er der ønsket et udkast på hvordan betalings layoutet ser ud, eller er det blot niels ønske?

- Samarbejder appen med nogle af jeres mængde beregnere for både professionelle og private?
    - Kan en privat angive størrelsen på en terrasse der skal bygges, og blive vejledt til minimum-bestilling af brædder?

- Administrationsdelen i appen / den ansattes muligheder i appen:
    - Er administrator tænkt som værende den ansattes login, og hvilke behov/rettigheder har han?
        - Korrigering af priser on the fly? (eks. hvis der kan ses at et produkt har en skade men stadig kan bruges og derfor er 2. sortering?)
        - Kan han andet?

Noter til spørgetime:

- Design-guide/manual
    - Vil skaffe det hvis muligt
    - Vil checke om der bliver brugt forskellige design til apps såmænd som web
- Vil gerne have et helt brugervenligt UI design
- Ift. Excel dokumentet
    - varegruppenummer oversættes til: Overkategori eller underkategori
    - bedre varegruppestruktur vil blive fundet frem til os
- Privat/prof brug af mængdeberegninger
    - Private ville kunne bruge beregningsmuligen så lige som professionelle
- Hvordan skal vi relatere produkterne til hinanden?
    - Har du husket eksempel
        - Skal det være underkategorier?
            - Køb maling/pensler eksempler
        - Skal være specifikke pensler til den valgte maling
    - Det vil være underkategorier som pensel eller maling
        - Det vil være forkert at tilbyde specifikke værktøjer eller enheder da det aldrig er sikkert på hvilke der er behov for
        - Kvalitetsniveau vil altid variere og kan være svært at arbejde rundt om
- Administrator

    - Korrigere priser on the fly
        - Som udgangspunkt, ja
            - ret pris osv
            - tildeling af rettigheder ved brug af applikation
            - de vil ikke
    - Login er ikke til appen
        - Produkter ændres og oprettes gennem backend
            - Gennem website
    - Indkøber lave backenden
    - Det skal være datastrukturen og hvordan den kan præsenteres
    - De skal have den ret som vi synes de skal have i applikationen(medarbejdere)
- Varehuse varierer(ift fysisk lokation af varer)
    - Placeringer varierer
        - Varesortiment varierer
            - Grundsortiment
                - Altid varehus-specifikt placeringer pga fysisk fremtoning i bygningerne
            - Tillægsvarer
                - Tilvælgningsmuligheder
- Powerpoint bliver tilsendt
    - Til Niels