Projekt 1 datasikkerhed og databaser

Gruppe 4: Muna, Cecilie, Kunuut og Jon.

Indholdsfortegnelse

Indledning

Denne rapport vil redegøre for udviklingen af en ”to-do” liste applikation. Kravene sat for udviklingen af projektet, er at ”to-do” listen skal have; en titel, en beskrivelse, en startdato samt en deadline. Når en opgave er blevet udført, skal det være muligt at kunne ændre statussen, så den står som fuldført eller opgivet. ”To-do” applikationen skal have en login og registrerings funktion for at brugeren kan oprette sig og dertil kunne lave sin egen “to-do” liste. Kravet for registrerings funktionen er at når en bruger registrere sig, så bliver deres bruger-id og password gemt i en database. Når et password bliver gemt i databasen, skal det sikres mod SQL injections. Dette gøres ved at passwordet bliver hashet og saltet inden det gemmes.

Dette projekt vil blive udviklet ved hjælp af Node.JS, Express.JS, CSS, samt eventuelt MySQL eller MongoDB.

Databaser er et vigtigt redskab til at gemme data, som senere kan blive kaldt tilbage. Der findes mange forskellige databaser med forskellige metoder og tilgangsmåder. I denne rapport vil der forklares hvordan udarbejdelsen af back end-databaser som Node.JS, Express.JS, og MongoDB fungere.

MongoDB er en open source-dokumentdatabase der bliver brugt til datalagring i store mængder. MongoDB bliver brugt i mange webapplikationer og er betegnet som en NoSQL-database, fordi MongoDB ikke er afhængig af en struktureret tabel database ligesom mySQL også kaldt ”My structured query Language”. Når man gør brug af MongoDB har man ikke brug for at lave en tabel før man tilføjer data til databasen og man ændrer tabeller når som helst uden at oprette en ny database.

mySQL er en open source-software som tillader at gemme og opbevare data fra information website. SQL bliver brugt til at oprette, ændre og udtrække data og kontrollere brugeradgang til databasen. MySQL bliver typisk bruget med PHP. PHP står for Hypertext Pre-processor og er en open source, server side script- og programmeringssprog som primært bruges på en serverside for webudvikling.

Express.JS er en server-side applikation og back-end framework af Node.JS. Express bliver brugt til at lave informations websites og mobile applikationer.

Problemformulering

Hvordan kan man ved brug af Node.js, express samt databaser oprette en simpel og sikker web applikationer til oprettelse og opbevaring af to-do liste elementer.

Metodeovervejelser

Valg af database

I dette projekt blev der tildelt valgmuligheden mellem MongoDB eller en MySQL database. I overvejelsen blev der diskuterede fordele og ulemper ved brug af begge database typer, for at kunne bedømme hvilken database der skulle benyttes til projektet. I løbet af beslutningsprocessen blev der truffet valget om at bruge MongoDB i stedet for MySQL da interessen for at arbejde med MongoDB var større. Derudover vil det styrke vores kompetencer at arbejde med MongoDB da det er et mindre bekendt redskab.

Opbygning af applikation

Selve applikationen blev valgt at bygges op ad express. Der er blevet valgt forskellige dependency pakker ud fra de krav der bliver sat ved starten af udviklingen af applikationen.

Først og vigtigst skal følsomme data kunne krypteres før de bliver gemt i databasen og applikationen skal kunne autentificere brugere før de gemmer nye data.

For at gemme adgangskoder forsvarligt, blev der brugt krypterings pakke til at envejs kryptere adgangskoder. Og hver gang en bruger logger ind, bliver den indtastede adgangskode sammenlignet med den der ligger i databasen.

Dernæst for at autentificere brugeren, bliver der gemt en cookie i klientens enhed, som kun kan verificeres af serveren. Og dermed opnå en sikker verificering af brugeren. Denne cookie indeholder også en unikt brugernavn som bruges til at hente todo elementer der blev lavet af brugeren.

I alle de følsomme routes der blev lavet i applikationen, bliver der sat en middleware som verificere brugeren.

I tilfælde af brugeren ikke opnår autentificering, bliver brugeren sikkert redirected til en log ind side.

I applikationen bliver der brugt pugjs som laver html sider. Disse filer kan opbygges dynamisk så man kan inkludere forskellige sider sammen.

Der er 2 hoved routes filer der styre hele applikationen. En til brugere og en til todos. I brugers route er der en til at vise log ind siden og en til at vise tilmeldings side. De har også hver især respektive route til at modtage post data fra brugeren.

I todos routen er der route til at vise oversigten med todos lavet at den indlogeret bruger, og der er en route til at vise en formular til at lave en todo, samt en route til at modtage indtastede data fra formularen. Hele todo routen bliver beskyttet med en middleware der først verificerer brugerens authentifications cookie før der bliver sendt data tilbage til brugeren.

Research og analyse

For at kunne udforme et produkt er det først vigtigt at vide hvilke muligheder som der er til stede for udformningen af produktet.

Så for at kunne lave den bedst mulige sikkerhed og opsætning af applikationen blev der kigget på de forskellige muligheder.

Injection metoder

Siden at siden skulle være sikker så var en af de ting som man kunne kigge på var injections da selv om der ikke blev gjort brug en mySQL database findes der stadig mulige injection trusler i forhold til den valgte databasestruktur.

MongoDB Payloads

```
true, $where: '1 == 1'
, $where: '1 == 1'
$where: '1 == 1'
', $where: '1 == 1'
1, $where: '1 == 1'
{ $ne: 1 }
', $or: [ {}, { 'a':'a
' } ], $comment:'successful MongoDB injection'
db.injection.insert({success:1});
db.injection.insert({success:1});return 1;db.stores.mapReduce(function() { { emit(1,1
|| 1==1
' && this.password.match(/.*/)//+%00
' && this.passwordzz.match(/.*/)//+%00
'%20%26%26%20this.password.match(/.*/)//+%00
'%20%26%26%20this.passwordzz.match(/.*/)//+%00
{$gt: ''}
[$ne]=1
';return 'a'=='a' && ''=='
";return(true);var xyz='a
0;return true
```

(swisskyrepo 2022)

For bedre at kunne mod kæmpe mulige injection angreb blev der lavet research i hvordan injections ville kunne blive brugt imod databasen og dermed ville det være muligt at formulere en ide til hvordan at disse problemer kunne overkommes.

Til at hjælpe imod injections og andre uautoriseret adgang til databasen og dets indhold blev der inkorporeret DB modeller og cookie tokens i applikationen.

DB modeller

For at komme rundt om mulige injections i login feltet blev der gjort brug af modeller i databasen. En DB model er en model for hvordan at data, der skal sendes og integrere med databasen, skal opsættes med den korrekte information.

Dette hjælper mod injections i f.eks login feltet da man ikke kan bruge f.eks kommentare til at snyde login systemet da feltet kun kan tage strings og skal have indhold i sig.

Cookie tokens

Cookie tokens blev implementeret som en form for dobbelt validering der forhindrer uautoriseret adgang til selve todo listen af todo liste genereringens funktion gennem url'en.

Dette bliver gjort gennem genereringen af en token ved login.

Denne token skal eksistere for at en brugere overhoved ville kunne tilgå andre sider end login eller registrering.

Yderligere muligheder

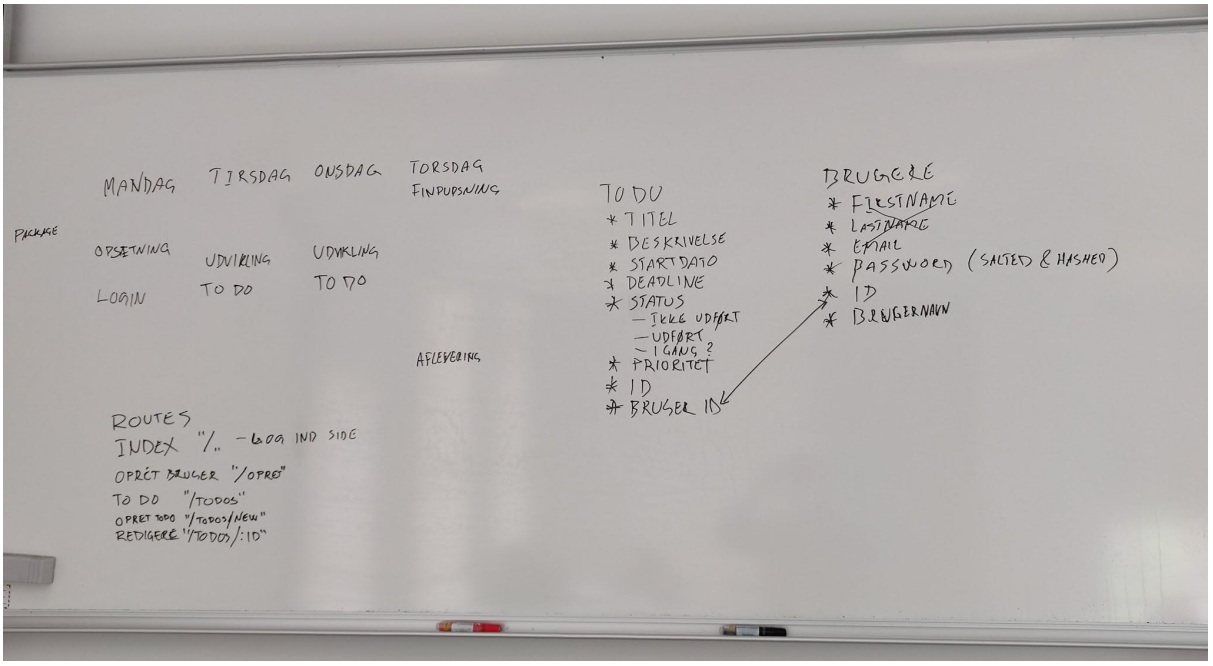
Udover de ovennævnte metoder, hvorved siden kan beskyttes mod injections og andre uautoriseret brug af applikationen og dens funktioner, findes der en masse andre metoder der kunne have været taget i brug for at sikre siden. Der er dog forskelle på hvor effektive de er til det.

Konstruktion

Produktet skulle laves i løbet af en uge, så det var vigtigt fra dag et, at have en klar ide over hvad der skulle laves og hvornår.

Mandag

Mandag bestod først i at komme frem til hvordan at processen skulle tages and hvad skulle laves hvornår og hvad skulle applikationen og databasen indeholde.



Efter at alt var på plads i forhold til organisering og valg af database gik selve arbejdet i gang. Startede med at lave en generel opsætning i express som der blev sat ind på github så hele gruppen havde adgang til den. Derefter begynde hvert gruppemedlem at arbejde på deres egen branch.

Tirsdag

Tirsdag blev brugt på at arbejde videre med applikationen

- Todo liste
- users

Onsdag

Onsdag blev der fortsat arbejde videre på applikationer

- Tilførsel af af validering af bruger

Torsdag

Torsdag bestod i at lave de sidste finjusteringer af applikationen så at alt virkede som påtænkt dette inkludere

- Validering af todo liste elementers datoer (er startdato mindre ind slut dato)
- styling

Efter at alt var færdigt blev applikationen merged med main branch i repository

Evaluering af process

Udviklingsprocessen af dette produkt har været meget lærerigt, men samtidig udfordrende. Da der ikke tidligere har været kendskab til databasen, hvilket betyder at før udviklingen af selve produktet skulle der være en form af viden og undersøgelse af databasen. Under processen af udviklingen opstod der tekniske fejl såsom problemer med terminalen.

Konklusion

Det kan konkluderes at det ønskede resultat som sat i problemformulering er blevet opfyldt og en todo liste genereret med mongodb som er sikret mod injections er blevet produceret og opbygget med express og Node.js.

Referencer/litteraturliste

swisskyrepo (2022) *PayloadsAllTheThings/NoSQL Injection at Master · Swisskyrepo/PayloadsAllTheThings* [online] available from <https://github.com/swisskyrepo/PayloadsAllTheThings> [14 February 2023]