Produktrapport

Daniel Skriver Hansen

H6PD100120

Indholdsfortegnelse

[Titelblad 2](#_Toc56508569)

[Forord 3](#_Toc56508570)

[Læsevejledning 3](#_Toc56508571)

[Problemformulering 4](#_Toc56508572)

[Kravspecifikation 5](#_Toc56508573)

[Database 5](#_Toc56508574)

[REST-API 5](#_Toc56508575)

[User-Interface 6](#_Toc56508576)

[Testspecifikation 7](#_Toc56508577)

[Database 7](#_Toc56508578)

[REST-API 7](#_Toc56508579)

|  |
| --- |
| **Elev:**  Daniel Skriver Hansen |
| **Firma:**  Bang & Olufsen A/S |
| **Projekt:**  Bestillings Organiserings System |
| **Uddannelse:**  Datatekniker m. Speciale i Programmering |
| **Projektperiode:**  26/10/2020 – 26/11/2020 |
| **Afleveringsdato:**  18/11/2020 |
| **Fremlæggelsesdato:**  27/11/2020 |
| **Vejledere:**  Lars Thise Pedersen &  Lærke Brandhøj Kristensen |
|  |

# Titelblad

Tech College Aalborg,

Struervej 70,

9220 Aalborg Ø

# Forord

Denne produktrapport, som er en ud af to skrevet rapporter, udarbejdet i forbindelse med svendeprøveforløbet 2020, for datatekniker m. speciale i programmering.

Rapporten er skrevet og udarbejdet af Daniel Skriver Hansen.

# Læsevejledning

Denne produktrapport beskriver de tekniske aspekter af dette projekt. Den vil beskrive produktet i helhed, opsætning, workflow samt vise nogle kodeeksempler.

For at få mest muligt ud af denne rapport, anbefales det at læse procesrapport først, da den giver indblik i, hvilke tanker der er gjort før og under udvikling af projektet. Denne rapport vil efterfølgende uddybe den endelige opsætning.

# Problemformulering

Med udgangspunkt i den beskrevne case og problemformulering fra procesrapport, vil den tekniske vinkel på problemformuleringen være som følgende.

Vi skal formå at lave et system, som strukturerer måden, hvorpå at kunden kan bestille sin mad, tjeneren ikke glemmer / skriver kundens ordrer forkert og køkkenet overholder den tidshorisont, som de har sat.

Til dette skal der udvikles en app, til både kunder og køkkenpersonale, som snakker med sammen via et REST-API.

REST-API’et vil oprette en forbindelse ind til Databasen, der ligger i skyen.

# Kravspecifikation

Dette afsnit vil yderligere konkretisere de krav, som er til projektet. Jeg vil synliggøre hvilke krav som blev opfyldt, hvilke som mangler og hvad kravene reelt er.

Hvert krav er opbygget med følgende:

* **Krav Id**: Denne søjle viser det ID, som er blevet givet til kravet. Dette er gjort for bedre at kunne referere kravet igennem rapporten.
* **Krav**: Dette er en kort beskrivelse af kravet.
* **Ikke opfyldt**: Udfyldt, hvis kravet ikke er blevet opfyldt.
* **Opfyldt**: Udfyldt, hvis kravet er opfyldt.

Derudover har jeg valgt at dele kravene op i tre kategorier, for at simplificere overblikket over kravene.

De tre kategorier er:

* **Database**
  + Kravene til, hvordan databasen skal være.
* **REST-API**
  + Kravene til, hvordan REST-API’et skal agere.
* **User-Interface**
  + Kravene til, hvordan User-interfacet skal formeres og bruges.

## Database

|  |  |  |  |
| --- | --- | --- | --- |
| **DB1** | Databasen skal kunne modtage og afgive forudbestemte data. |  | X |
| **DB2** | Databasen skal kunne tilgås fra alle steder, hvor der er en internetforbindelse til stede. |  | X |

## REST-API

|  |  |  |  |
| --- | --- | --- | --- |
| **RA1** | Rest API’et skal kunne håndtere at modtage en ordre og sende informationerne de respektive steder hen. |  | X |
| **RA2** | REST API’et skal kunne håndtere at snakke om skrive til Databasen. |  | X |
| **RA3** | REST API’et skal kunne køres i en Docker Container fra en raspberry Pi | X |  |

## User-Interface

|  |  |  |  |
| --- | --- | --- | --- |
| **UI1** | UI’et skal være intuitivt og være let at bruge. | X |  |
| **UI2** | UI’et skal kunne bruges til at aflægge en ordre |  | X |
| **UI3** | UI’et skal kunne give information omkring, hvad der er forkskellige ting på menuen. |  | X |
| **UI4** | Brugeren af UI’et skal kunne se, hvad da, indtil videre, har valgt fra menuen, via en scrollable ude i siden med de valgte menuer. |  | X |
| **UI5** | Brugeren skal kunne bekræfte at det er de valgte menuer, som brugeren har valgt. |  | X |

# Testspecifikation

Under denne sektion vil der blive beskrevet test specifikationer ift. projektet. Test specifikationerne vil hovedsageligt bestå af Unit-Test, men det vil også kunne indføres i CI værktøjer som f.eks. CircleCi eller Jenkins.

Disse krav er opbygget på samme måde som i Kravspecifikationen.

## Database

|  |  |  |  |
| --- | --- | --- | --- |
| **TDB1** | Databasen skal kunne modtage et bestemt sæt data, og sende samme data tilbage igen. | X |  |
| **TDB2** | Databasen skal kunne ændre på data, og sende det ændrede data tilbage. | X |  |
| **TDB3** | Databasen skal kunne kontaktes via REST-API’et |  | X |

## REST-API

|  |  |  |  |
| --- | --- | --- | --- |
| **TRA1** | Rest-API’et skal kunne tilgås på ”/” routen. |  | X |
| **TRA2** | Rest-API’et skal kunne modtage data via ”/putOrder”, med data via en POST. |  | X |
| **TRA3** | Rest-API’et skal kunne sende data baseret på, om der er en ordrer eller ej. |  | X |
| **TRA4** | Rest-API’et skal kunne håndtere at modtage data i JSON-format. |  | X |
| **TRA5** | Rest-API’et skal kunne snakke med databasen. |  | X |

## User-Interface

|  |  |  |  |
| --- | --- | --- | --- |
| **TUI1** | Via UI’et skal man kunne tilføje menuer til ens ordrer. |  | X |
| **TUI2** | Via UI’et skal man kunne vælge mellem om det skal være køkken eller kunde, i FTS. |  | X |
| **TUI3** | Via UI’et skal man kunne sende en ordrer op til REST-API’et. |  | X |
| **TUI4** | Man skal kunne tilføje og slette menuer dynamisk. |  | X |

# Brugervejledning

Denne brugervejledning vil give indblik i, hvordan produktet bruges fra kundens side af, da køkkensiden ikke har et funktionelt design.