36er Service Tool

# App Struktur

Neue Idee ist das wir nicht eine Mobile App für Android machen, sondern eine „Website“. Dort sollen sich alle „einloggen“. Drei Rollen sind möglich:

* Ein Master (Buffetteam) zum Anzeigen der Bestellungen
* Mehrere Kellner, die Bestellungen aufgeben können
* Mehrere Servierer, die sich eine hergerichtete Rechnung auswählen können und diese an den Tisch bringen können und abkassieren

Ein Kellner soll sich im Endeffekt auf die Website begeben (über IP und Port). Dort über den Login-Dialog Kellner wählen. Danach öffnet sich für ihn eine neue Ansicht. In der kann er einen Tisch auswählen und über Buttons das essen und trinken auswählen, das der tisch will. Über einen Knopf wird die bestellung an den Server geschickt und gespeichert. Zusätzlich kann er die letzte seiner bestellungen stornieren über einen knopf sofern notwendig.

Der Master logged sich ebenfalls auf der seite ein (mit user und password). Er list immer die letzten x bestellungen aus (getrennt nach Essen und trinken um es separat abwickel zu können). Und kann einzeln Bestellungen über einen knopf abfertigen. Diese wandern dann am server in rechnungen, und die nächste bestellung wird abgerufen.

Der Servierer bekommt nach dem login eine liste mit rechnungen die schon fertig sind und kann eine auswählen. Diese nimmt er sich dann vom buffet, sie wird aus der datenbank gelöscht, und der servierer geht mit bestellung und rechnung zum tisch. Dort hat er dann noch eine rechen hilfe angezeigt bei der er einzeln items der bestellung abrechnen kann.

# App Architektur

Sprache ist NodeJS, eine asynchrone serverseitige Javascript variante. Am server läuft das script zum handlen diverser requests. Beim erstmaligen aufrufen der seite, wird der login dialog geladen sowie alle anderen configurationen über .json files. Je nachdem als was der user sich dann einlogged, wird eine andere configuration aufgerufen und unterschiedliche elemente geladen. Das ganze wird über das Framework „VueJS“ laufen. In jeder config ist hinterlegt welche componenten geladen werden sollen, diese componenten müssen wir entwickeln. Die login componente ist schon fertig. Somit ersparen wir uns immer neue daten vom server laden zu müssen und reduzieren den traffic im netzwerk