Solution Design

1. Standards und Technologien

1.1 Rücksprache mit Auftraggeber

Der Auftraggeber stellte die Anforderung, dass das Backend in ASP.NET und das Frontend in VUE.JS v3.2.30 entwickelt werden muss.
Nach Rücksprache mit dem Auftraggeber dürfen wir nun selbst entscheiden, welche Standards wir verwenden, weil keine enge Integration in bereits bestehende Applikationen notwendig ist.

1.2 Backend

Das Backend wird mit Go Version 1.17.6 umgesetzt.
Das Backend wird als JSON-REST-API umgesetzt.
Für das Backend sind die Implementierungsrichtlinien der Waterbyte-CLEAN-Architecture (https://github.com/Engelbyte-s-Waterbyte/waterbyte-clean-architecture) einzuhalten.
Für die Datenbank-Interaktion wird ein ORM (https://gorm.io) verwendet v1.22.5 -> Database first. Für Gorm verwenden wir den SQLLite Connector https://github.com/go-gorm/sqlite v1.2.6

1.2.1 Datenbank

Für die Datenbank werden wir SQLite v3.37.2 verwenden.

1.3 Frontend

Für das Frontend haben wir uns für VUE.JS Version 3.2.28 entschieden.

1.3.1 UI/UX

Das User-Interface wird gemäß den Mockups in der Anforderungsspezifikation erstellt. Dabei wurden bereits alle Anmerkungen des Kunden berücksichtigt.

1.4 Allgemein (Docker)

Die Applikation soll als docker-containerized deployed werden. (Docker Version: 20.10.12)

2. Dokumentation

Die Dokumentation beschränkt sich auf die im Rahmen des SYP-Unterrichts angefertigten Dokumente (Anforderungsspezifikation, Solution-Design).

Für die grafische Modellierung verwenden wir draw.io.

Dokumente und Projekte werden über Git bzw. Github verwaltet.

2.1 Aufteilung der Arbeit und Code-Verwaltung

Die Aufteilung der Arbeit erfolgt über Clickup bzw Github.
Die Versionsverwaltung des Projektes erfolgt über Github bzw Git.

3. Solution-Modellierung

3.1. Klassen (UI, Business Logic, DB, API, ...)

Bei der Modellierung der Klassen wurden die SOLID-Prinzipien berücksichtigt.

4 Sequenzen

5 Software Architektur

Das System wird als Server-Client-Architektur umgesetzt. Die verwendeten Technologien wurden in 1. festgelegt.
Das gesamte System soll in einem Docker-Container funktionieren, auf Port 80 des Docker-Containers soll die Website aufzurufen sein.
Docker-Container:

- Image: Nginx

Im Docker Container läuft:

- das Backend:
  - binäre Datei, die mit `go build` erzeugt wurde
  - stellt API unter Port 3333 zur Verfügung
- NGINX Server
  - Port 80
  - statischer Fileserver, der die mit `vue build` generierten Dateien hostet
  - proxy: Requests, deren Route mit /api beginnt, werden an den Backend Server weitergeleitet

6 Daten

Die Daten werden in einer SqlLite-Datenbank (Version 3) gespeichert.

7 Deployment

Der Docker Container wird auf dem Server ausgeführt. Der Docker Container erfüllt alle Anforderungen, außer https.
Um https zu gewährleisten, wird direkt auf dem Server nginx laufen und mithilfe von Letsencrypt Zertifikat wird die Verbindung verschlüsselt. Die Requests werden einfach auf den Docker Container weitergeleitet (proxy_pass).

8 Security

Die finale Website soll nur über https erreichbar sein.
Requests auf HTTP werden auf HTTPS umgeleitet.

9 Technologie-Stack

Siehe Punkt 1.

10 Persistierung

Siehe Punkt 1.

11 Anhang

11.1 Umsetzungskonzept NF-Anforderungen

Das User-Interface wird gemäß den Mockups in der Anforderungsspezifikation erstellt.
Um eine möglichst schnelle Reaktion des Backends zu gewährleisten wurde Go gewählt.
Die http-Library https://github.com/julienschmidt/httprouter v1.3 soll möglichst schnelle http-Kommunikation ermöglichen.

11.2 Standard-Beschreibung

11.3 Qualitätssicherung

11.3.1 Klassen

11.3.2 Testen

Modul-Tests und Unit-Tests werden im Anschluss an das Solution-Design erstellt, danach werden Entwicklungspakete inkl. Verifikation (Tests) auf die Projektmitarbeiter aufgeteilt. Getestet werden muss nur die Business-Logic (Package Charge). (Test-Driven-Development)