

cultus

webprojekt von jordan & timo

Inhaltsverzeichnis

[1. Projektbeschreib 3](#_Toc106613306)

[1.1 Projektverzeichnis 3](#_Toc106613307)

[1.2 Projektbeschreib 3](#_Toc106613308)

[1.3 Zieldefinierung 3](#_Toc106613309)

[1.4 Meilensteine 3](#_Toc106613310)

[2. Architektur 3](#_Toc106613311)

[2.1 Backend 3](#_Toc106613312)

[2.2 Frontend 4](#_Toc106613313)

[2.3 CRUD / Datenbankanbindung 4](#_Toc106613314)

[3. Arbeitsjournale 4](#_Toc106613315)

[4. Datensicherheit 5](#_Toc106613316)

[4.1 Zwei-Faktor-Authentifizierung 5](#_Toc106613317)

[4.2 Password Encryption 6](#_Toc106613318)

# Projektbeschreib

## Projektverzeichnis

<https://github.com/seiemens/super-duper-octo-engine-exhaust-booster-extra-air-inlet-v2-max-mega-ultra-wqhd-4k-boost-enabled-devt>

## 1.2 Projektbeschreib

Mit unserem Projekt möchten wir einem Problem, dass die Welt schon lange hat, entgegenwirken. Der fehlende Komfort auf der Toilette. Jeder von uns weiss, wie unangenehm es sein kann, auf einem Thron zu sitzen, der einfach nicht passt. Deshalb haben wir eine Lösung entwickelt: ein Webshop, auf dem man Toiletten aus aller Welt kaufen kann. Ob dünn oder breit, ob bunt oder weiss. Wir haben alles! Mit unserer stolzen Auswahl von über 500 Exemplaren haben wir die besten Toilettentechnologien aus aller Welt gesammelt und verkaufen weltweit. Ungemütliche Geschäfte waren gestern!

## 1.3 Zieldefinierung

Als grundsätzliches Ziel haben wir einen Webshop. Folgende Funktionen sollte dieser haben:

* Benutzerverwaltung
* Ergonomisches Shop Design
* Datenbank für Benutzer und Produkte
* Zwei-Faktor-Authentifizierung
* Session Handling

## 1.4 Meilensteine

* Benutzer werden in der DB gespeichert und können bearbeitet werden
* Ein Benutzer kann sich einloggen
* Shop Page fertig (UI)
* Login Page fertig (UI)
* Produkte in DB gespeichert
* Dokumentation fertig
* Projekt fertig

# 2. Architektur

## 2.1 Backend

Für die Realisierung des Tools haben wir uns für Technologien / Frameworks entschieden, die gut aufeinander abgestimmt sind. Für das Backend setzten wir auf nodejs / expressjs, da wir bereits damit Erfahrung gemacht haben und express sich bewährt hat. Mit expressjs ist es auch einfach eine Session zu erstellen, da es dafür auch eine Library existiert. Für das Speichern von Daten verwendeten wir MongoDB, da es einer der einfachsten Wege ist SQL-Injections zu verhindern, da MongoDB eine NoSQL DB ist.

## 2.2 Frontend

Nuxtjs, ein Framework das auf Vue, basiert war unsere Wahl fürs Frontend. Es ist ein simples, zuverlässiges Framework das das Darstellen von dynamischen Daten (z.B. Shop – Items) extrem einfach macht, was mit nur HTML/CSS/JS nicht so einfach realisierbar ist.

## 2.3 CRUD / Datenbankanbindung

Für den Zugriff auf die Daten verwenden wir wie schon erwähnt expressjs. Mit expressjs ist es möglich, API’s und Backends einfach und übersichtlich zu erstellen. Auch gibt es bereits bestehende Libraries für den MongoDB – Zugriff, was uns die Arbeit nochmal einfacher macht.

# 3. Arbeitsjournale

23.05

**Jordan & Timo:** Zuerst haben wir nach einer Idee für unser Projekt gesucht. Als wir die Idee ausgearbeitet haben, haben wir angefangen zu planen. Wir haben uns überlegt, was muss alles getan werden und dann die Rollen verteilt. Schliesslich haben wir uns dann aufgeteilt und angefangen zu arbeiten.

**Jordan**: Meine erste Priorität war das Backend unseres Webservers. Da unser Zeitfenster einigermassen begrenzt ist wollte ich dies mit einem Tool machen, womit ich bereits Erfahrungen habe. Deshalb habe ich mich dafür entschieden das Backend mit nodejs und express zu machen. Den Rest der Zeit habe ich dann dafür genutzt, wieder in dieses Thema einzusteigen und auszuprobieren und den Grundsatz für den Webserver zu starten. Es gab von mir an diesem Tag keinen Commit, da es noch nicht so viel Sinnvolles gab zum pushen und ich nicht etwas unnötiges Hochladen möchte.

**Timo**:

30.05

**Jordan:** Heute habe ich wieder am Backend gearbeitet. Ich konnte einen Sinnvollen Webserver zum Laufen bringen. Dieser kann nun wie einen API funktionieren, sodass man vom Frontend Anfragen über eine URL schicken kann und dann eine Antwort bekommt. Danach habe ich angefangen die Sessions zu regeln. Dass ein User sich anmelden kann, dies validiert wird und dann eine Session entsteht. Diese Änderungen wurden aufs GitHub gepusht.

**Timo**:

13.06

**Jordan**: Da der Code bis jetzt auf einem guten Stand ist, habe ich heute an der Dokumentation gearbeitet, da diese bis her vernachlässigt wurde. Ich habe da eine Struktur erstellt und auch diverse Erklärungen dokumentiert etc. Diese kann man auch in einem Commit von mir nachvollziehen.

**Timo**:

20.06

**Jordan**: Heute ist der Abschlusstag. Das heisst ich habe die Punkte abgeschlossen, die noch offen waren. In diesem Fall bedeutete das, das Backend fertig stellen und noch die Dokumentation Final abschliessen. Dafür habe ich mehrere Commits und einen Merge heute gemacht.

**Timo**:

# 4. Datensicherheit

## 4.1 Unsere Schutzmassnahmen

### 4.1.1 Zwei-Faktor-Authentifizierung

*2FA ist eine hervorragende Technologie, um die meisten Angriffsmethoden zu verhindern.*

Multi-Faktor-Authentifizierung (MFA; Multi-Faktor Authentifizierung oder 2FA, neben den anderen Begriffen) ist ein elektronisches Authentifizierungssystem, bei dem ein Benutzer nur dann Zugang zu einer Internetseite oder Anwendung erhält, wenn er einem Authentifizierungsmechanismus zwei oder mehr "Beweise" (oder Faktoren) vorlegt: Wissen (etwas, das nur der Benutzer weiß), Besitz (etwas, das nur der Benutzer hat) und Inhärenz (etwas, das nur der Benutzer ist). MFA bewahrt Benutzerdaten - zu denen private Identifikationsdaten oder finanzielle Vermögenswerte gehören können - davor, dass ein unbefugter Dritter, der z. B. ein einzelnes Passwort herausfinden könnte, darauf Zugriff hat.

Oft wird dies über eine dritt Anbieter App geregelt. Es wäre aber auch möglich über beispielsweise SMS oder E-Mail. Wir haben uns für letzteres entschieden. Wenn ein User sich einloggen möchte, generieren wir ein Token, dass aus Zufälligen Zeichen besteht und schicken dies an die vom Nutzer hinterlegte E-Mail Adresse. Dieses Token muss der Nutzer dann einfügen und erst wenn es übereinstimmt, kann der Nutzer sich einloggen.

Durch diese Methode schützen wir uns vor diversen Angriffen. Vor allem aber sind es Angriffe die den User betreffen. Auch falls der Nutzer aus irgendwelchen Gründen sein Passwort freigeben sollte, bräuchte man immer noch Zugriff auf den Mail Account des Nutzers. Damit der Mailaccount nicht an Angreifer freigegeben wird liegt nicht in unserer Verantwortung und wir können auch nichts dagegen machen.

### 4.1.2 Password Encryption

Password Encryption ist ein Vorgang, bei dem das Passwort verschlüsselt wird. So wird ein «Hash» erstellt. Als Hash oder Hashwert bezeichnet die Informatik die Ausgabe einer Hashfunktion, als Hash aber auch einen listenartigen Datentyp, bei dem der Zugriff auf die Elemente über deren Hashwert erfolgt, die Hashtabelle. Einfach gesagt: es entsteht eine Zeichenfolge, die man nicht entschlüsseln kann. Dadurch schützen wir uns auf unerwünschten Zugriff, auf unserer Seite. Mit dieser Schutzmethode schützen wir uns vor allem gegen Angriffe gegen unser System, also den Server. Wenn ein Angreifer Zugriff auf unser Netzwerk bekommt, so kann er mit den Passwörtern der User nicht wirklich etwas anfangen.

## 4.2 Angriffe