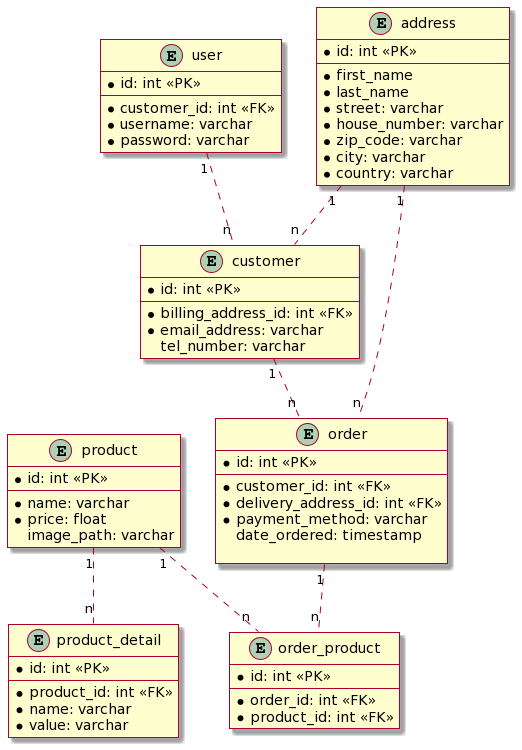
Doku

# Datenstruktur



# Vorbereitung

## Frontend

Um das Vue-Projekt starten zu können, müssen die notwendigen NPM-Packages installiert werden.

cd gui

npm install

## Backend

Um die Flask-App starten zu können, müssen die notwendigen PIP-Packages installiert werden. Man kann sie auch in einem virtuellen Environment installieren und dort die Flask-App starten.

cd api

pip install -r requirements.txt

# Verwendete Technologien

## Backend

### Python

Ich verwende Python, weil ich einiges an Erfahrung mit dieser Programmiersprache gesammelt habe.

### Python Flask

Flask ist ein flexibles Framework zum Entwickeln von APIs, wodurch es ideal für kleinere Projekte geeignet ist. Es gibt einige Plugins, die man je nach Bedarf verwenden kann. Zum Beispiel kann man ganz einfach mit Flask-Login eine Authentifizierung in das Programm einbauen. Außerdem hat Flask eine große Community, weshalb man schnell Antworten auf etwaige Probleme im Internet findet.

### SQLAlchemy

SQLAlchemy ist das größte (d.h. mit größter Community) ORM (Object-Relational-Mapping)-System für die Programmiersprache Python.

Es ermöglicht mir, objektorientiert mit den Daten aus der Datenbank zu arbeiten, da ich mit SQLAlchemy Python-Klassen erstelle, welche die Datenbankentitäten repräsentieren. So kann ich Abfragen machen, ohne selber SQL-Scripts zu erstellen. Solche SQLAlchemy-Abfragen sind auch gegen SQL-Injection geschützt.

## Frontend

### VueJS

Ich verwende das Javascript-Framework “Vue”, weil ich bereits Erfahrungen damit in der Arbeit gesammelt habe und mir das Entwickeln damit gefällt. Besonders mag ich die Erstellung von eigenen “Components”, die man weiterverwenden kann. Außerdem kann man öffentliche Component-Libraries verwenden, zum Beispiel PrimeVue.

### TailwindCSS

Ich benutze TailwindCSS, weil es mir das Styling vereinfacht. Ich spare mir Zeit, indem ich mir nicht immer Namen für neue CSS-Klassen ausdenken muss, und bin deshalb einfach ein Fan davon, wie schnell man mit TailwindCSS zu schönen Ergebnissen kommt. Auch die “Modifier” von Tailwind sind sehr nützlich, so muss man zum Beispiel nicht extra Media Queries in CSS schreiben.

### Typescript

Man kann Vue entweder mit Javascript oder mit Typescript verwenden. Ich habe zwar deutlich mehr Erfahrung in normalem Javascript, aber finde Typescript sehr interessant, weil man durch das Type-Checking schon während der Entwicklung auf mögliche Fehler aufmerksam wird und somit sauberer arbeitet.

## Datenbank

Als Datenbank verwende ich MySQL, da ich Erfahrung damit gesammelt habe und es via XAMPP schon vorinstalliert ist.

### MySQL starten

MySQL starten via XAMPP; es läuft dann auf Port 3306.

### Datenbank erstellen

Die Datenbank „webshop-db“ habe ich gemäß dem ER-Diagramm erstellt und die SQL-Statements in der Datei db.sql gespeichert. Der Benutzer „webshop“ hat alle Berechtigungen (Passwort siehe api/config.py).

## Flask-App starten

### Option 1: Virtuelles Environment

Zuerst wird ein virtuelles Environment erstellt:

* cd api
* pip install virtualenv
* python.exe -m virtualenv venv
* venv\Scripts\activate

Dort werden die Dependencies installiert:

* pip install -r requirements.txt

Danach wird die Flask-App gestartet:

flask –app main run

Die App läuft dann auf Port 5000.

### Option 2: Ohne virtuelles Environment starten

Dependencies am lokalen Rechner installieren:

Pip install -r requirements.txt

Flask-App starten:

Python.exe -m flask –app main run

Die App läuft dann auf Port 5000.

Anmerkung: Die Flask App könnte man mit SSL starten, indem man die „cert“ und „key“ Parameter angibt, jedoch funktioniert dann das „fetch“ in Javascript nicht mehr, da es keine Zertifikate ohne valide CA akzeptiert.

Beispiel: Python.exe -m flask –app main run –cert domain.crt –key domain.key

## Den Vite-Webserver starten:

cd gui

npm run dev

Der Vite-Webserver läuft dann auf Port 8080.

# Zeitschätzung

Tabellen und Python ORM Models erstellen: 1 Stunde 8:20 – 9:20

Controller erstellen: 30 Minuten 9:30 – 10:00

Doku-Teil: 10 Minuten 10:00 – 10:10

Beispielwerte einpflegen: 15 Minuten 10:10 – 10:25

GUI Produktliste: 30 Minuten 10:30 – 11:00

Produkt Detailansicht: 15 Minuten 11:00 - 11:15

Warenkorb Ansicht: 45 Minuten 11:15 – 12:00

Bestell-Funktionalität: 2 Stunden 13:00 – 15:00

Statistiken API + View: 1 Stunde 15:00 – 16:00

Dokumentation + restliches Programm vervollständigen 1 Stunde

# Anforderungen an die Hostingumgebung

Um die Sicherheit der Website zu gewährleisten, gibt es folgende Anforderungen an die Hostingumgebung:

* Das Betriebssystem soll immer die neuesten Updates installiert haben, da neuere Versionen in der Regel sicherer sind (durch die Schließung von Sicherheitslücken)
* An dem physischen Gerät, auf dem die Hostingumgebung ist, darf man keine Geräte (z.B. USB-Sticks) anstecken, ohne sie vorher zu überprüfen
* Die Datenbank soll hinter einer Firewall sein
* Alle Verbindungen von und zur Hostingumgebung sollen verschlüsselt sein
* Nur berechtigte Personen sollen Zugriff zu dieser Hostingumgebung haben

# Über die Website

## Bilder

Bilder werden im Dateisystem (/api/images) gespeichert und der relative Pfad in der Datenbank (dieser Pfad ist eigentlich nur der Dateiname an sich, da es nicht vorgesehen ist, Unterordner für /api/images zu erstellen), zB wäre der Eintrag in der Datenbank „bild.webp“.

Über einen POST-Request können Bilder hochgeladen werden (es gibt dafür noch keine Funktionalität auf der GUI), die dann in das WEBP-Format umgewandelt und mit einem einzigartigen Filenamen abgespeichert werden.

## Warenkorb

Die Produkte, die der User über den Button „In den Warenkorb“ in den Warenkorb legt, werden mit einem Pinia-Store gespeichert (Pinia ist eine Library für VueJS, welche es ermöglicht, auf sehr einfache Weise Daten über mehrere „Pages“ (Navigation innerhalb der Vue App) hinweg zu speichern) und dann in der Warenkorb-Seite angezeigt. Der Nachteil an dieser Methode ist, dass die Produkte verschwinden, sobald man die Seite aktualisiert. In Zukunft könnte man die Produkte direkt in die Datenbank speichern.

## Produkteigenschaften

Jedes Produkt kann beliebig viele Eigenschaften haben, die mit „name“ und „value“ in der Datenbank (Tabelle product\_detail) abgespeichert werden. Diese Eigenschaften werden angezeigt, wenn man in der Produktübersichtsseite auf ein Produkt klickt.