



# FENSTER ONLINE-SHOP

## Dokumentation

Exposee

Software Engineering, alles rum um das Projekt

Norbert Mosinski  
mosinski95@gmail.com

# Inhaltsverzeichnis

## Inhalt

Inhaltsverzeichnis .....	1
1. Einleitung.....	5
2. Grundlagen.....	0
2.1. Firma .....	0
2.2. Produkte .....	0
2.2.1. Fenster .....	0
2.2.1.1. TODO Fenstereigenschaften .....	0
2.2.1.2. Umfang des Produktangebots.....	0
2.2.2. Rolläden .....	1
2.2.3. Zubehör.....	1
2.2.4. Service .....	1
2.2.5. Lieferung .....	1
2.2.6. Montierung .....	1
2.3. Online-Shop-Anbieter .....	1
2.3.1. Wix .....	1
2.3.1.1. Grundlagen .....	1
2.3.1.2. Online-Shop-Plugin.....	2
2.3.1.2.1. Elemente .....	2
2.3.1.2.2. Einfacher Einkaufsprozess .....	3
2.3.1.2.3. Einschränkungen .....	7
2.3.1.3. Wix-Pay-Plugin .....	7
2.3.1.4. Wix Mail .....	8
2.3.1.5. Wix-users .....	8
2.3.1.6. Wix-Datenbanken .....	8
2.3.1.6.1. Wix-Data.....	8
2.3.1.6.2. Hooks .....	9
2.3.1.6.3. Kollektion .....	9
2.3.1.7. Wix-Dataset .....	9
2.3.1.8. Wix-Selectors.....	9
2.3.1.8.1. Global-Selector.....	9

2.3.1.8.2.	Scoped-Selector .....	9
2.3.1.9.	Pages und grafische Elemente .....	10
2.3.1.10.	Repeater .....	10
2.3.1.11.	Container .....	12
2.3.1.12.	Rasterlinien.....	12
2.3.1.13.	Fixierte Elemente.....	12
2.3.2.	WordPress .....	12
2.3.3.	Wix vs. WordPress .....	12
2.4.	Datenbanken vs. Session/Local-Storage.....	13
3.	Methodik.....	15
3.1.	Entwicklungsstil .....	15
3.2.	Projektmanagement .....	15
3.3.	Programmierrichtlinien.....	15
3.4.	Dokumentation im Code .....	15
3.4.1.	Funktionen und Klassen.....	15
3.4.2.	Kommentare.....	15
3.4.3.	Tags .....	16
3.4.4.	Tests .....	16
3.5.	Namenskonventionen.....	16
3.5.1.	Verzeichnisse.....	16
3.5.2.	Klassen .....	16
3.5.3.	Funktionen .....	16
3.5.4.	Tests .....	16
3.5.5.	Datenbanken.....	17
3.5.6.	Seiten .....	17
3.5.7.	Pakete .....	17
3.6.	Packaging.....	17
4.	Anforderungsanalyse.....	18
4.1.	Ist-Zustand .....	18
4.2.	Soll-Zustand .....	18
4.3.	Methodik der Anforderungserhebung .....	18
4.4.	Anforderungen.....	18
4.4.1.	Nichtfunktionale Anforderungen .....	18

4.4.2.	Funktionale Anforderungen .....	18
4.4.2.1.	Produkte.....	19
4.4.2.2.	Produktauswahl und -Konfiguration .....	19
4.4.2.3.	Warenkorb .....	20
4.4.2.4.	Nutzer-Login .....	21
4.4.2.5.	Kaufprozess .....	22
4.4.2.6.	Kaufbestätigung .....	22
4.4.2.7.	Bestellübersicht.....	22
4.4.2.8.	Rechtliche Bausteine .....	22
5.	Entwurf .....	23
5.1.	Plattformanalyse.....	23
5.1.1.	Wix mit dem Online-Shop-Plugin .....	23
5.1.1.1.	Herausforderungen .....	23
5.1.1.2.	Umsetzungsmöglichkeiten .....	23
5.1.1.3.	Fazit .....	24
5.1.2.	Wix ohne des Online-Shop-Plugins.....	24
5.1.2.1.	Herausforderungen .....	24
5.1.2.2.	Umsetzungsmöglichkeiten .....	25
5.1.2.3.	Fazit .....	25
5.1.3.	WordPress .....	25
5.1.3.1.	Herausforderungen .....	25
5.1.3.2.	Umsetzungsmöglichkeiten .....	26
5.1.3.3.	Fazit .....	26
5.1.4.	Plattformvergleich.....	26
5.1.5.	Fazit .....	28
5.2.	Konzeptioneller Entwurf.....	29
5.2.1.	Ablauf des Einkaufs im Online-Shop .....	29
5.2.2.	Produkte.....	30
5.2.2.1.	Produktaufbau.....	30
5.2.2.2.	Produkterstellung .....	30
5.2.2.3.	Bearbeitung/Konfiguration von Produkten .....	31
5.2.2.4.	Datenspeicherung.....	32
5.2.3.	Warenkorb.....	32

5.2.4.	Nutzerverwaltung .....	32
5.2.5.	Bezahlung .....	32
5.2.6.	Bestellungsaufnahme.....	32
5.2.7.	Speicher- und Datenverwaltung .....	33
5.2.7.1.	Datenbankstruktur.....	33
5.2.7.2.	Session-/Local-Storage-Struktur .....	33
5.3.	Technischer Entwurf.....	33
5.3.1.	Schichten-Übersicht .....	33
5.3.2.	Packages-Übersicht .....	33
6.	Implementierung.....	34
7.	Test .....	35
8.	Fazit.....	36
9.	Old TO REFACTOR .....	36
9.1.	Generelle Funktionsweise .....	36
9.2.	Vererbung.....	36
	Einfache Vererbung.....	37
	Abstrakte Klassen.....	37
	Interfaces.....	37
	Statische Methoden.....	38
	Abstrakte Methoden .....	38
9.3.	Komponenten .....	38
	typing.js .....	38
	storageManager.js.....	38
	debug.js.....	39
9.4.	Fehlerbehandlung.....	39
	Generelle Richtlinien bei Fehlerbehandlung.....	39
	Halterichtlinien für throw .....	39
	Halterichtlinien für catch .....	39
	Besonderheiten .....	40
	Tests.....	40
10.	Verweise .....	41

## 1. Einleitung

Viele Käufe geschehen heutzutage Online. Daher ist eine Online-Präsenz von Nöten. Es existieren bereits viele Verkaufsportale Online, die das Anbieten von Produkten vereinfachen. Zu den bekanntesten gehören Ebay und Amazon. Die Präsenz auf Ebay und Amazon soll in naher Zukunft ebenfalls umgesetzt werden. Das **Ziel** dieses Projekts ist jedoch die Erstellung einer eigenen Website, die als ein **unabhängiges Online-Shop** dient. Bei der Umsetzung und Konzipierung sind jedoch die zukünftigen Ebay- und Amazon-Projekte grob zu berücksichtigen, insbesondere im Bezug auf die Produktdarstellung, Preisverwaltung etc.

## 2. Grundlagen

### 2.1. Firma

Im Folgenden werden die Firmendaten tabellarisch dargestellt.

<b>Firmenname</b>	Vision-Fenster
<b>Geschäftsführer</b>	Agnieszka Mosinska
<b>Firmenanschrift</b>	Friedrich-Ebert-Platz 2, 48153 Münster
<b>Anschrift des Geschäftsführers</b>	Gleich wie Firmenanschrift
<b>Tel.</b>	017632091782
<b>E-Mail</b>	info@vision-fenster.de
<b>IBAN</b>	DE36 40050150 0034 4462 45
<b>PayPal-E-Mail</b>	info@mosinski.net
<b>USt.-IdNr.</b>	DE 321584255
<b>Steuernummer</b>	337/5185/2915

### 2.2. Produkte

#### 2.2.1. Fenster

Fenster werden in verschiedenen Ausführungen angeboten. Jede bestimmte Ausführung führt zu einem anderen Preis. Der Preis wird teils **pauschal**, teils **prozentual** ermittelt. Zu verschiedenen Fensterausführungen gibt es **Preistabellen**. Eine Fensterausführung ergibt sich aus der Kombination verschiedener **Eigenschaften**. Zu diesen gehören *Material, Profil, Fenstertyp (zweiteilig, dreiteilig etc.), Ober-/Unterlicht, Öffnungsart, Farbe und Maß*. Anhand der Eigenschaften *Profil, Öffnungsart* und *Maß* lässt sich ein bestimmter Eintrag in einer bestimmten Preistabelle eindeutig identifizieren. Attribute wie *Fenstertyp* oder etwa *Ober-/Unterlicht* sind Eigenschaften, die sich ohnehin aus der *Öffnungsart* ergeben, jedoch erleichtern sie die Suche nach dem gewünschten Fenster über das Einsetzen von Filtern. Eigenschaften wie *Farbe* und einige andere bilden Zusatzeigenschaften, die den Preis oft prozentual beeinflussen. Einige Eigenschaften sind mehrfach vorhanden, da sie von verschiedenen Produzenten stammen. Diese unterscheiden sich grundsätzlich nur im Preis und deren Marke voneinander.

##### 2.2.1.1. TODO Fenstereigenschaften

##### 2.2.1.2. Umfang des Produktangebots

Die Anzahl der Produkte und ihrer Varianten hängt davon ab, was als Produkt angesehen wird. Beispielsweise besitzt eine bestimmte Fensterpreistabelle ca. 400 verschiedene Preise für die verschiedenen Breiten und Höhen der Fenster. Wenn eine Breite-Höhe-Kombination als ein Produkt angesehen wird, dann gibt es bei ca. 20 verschiedenen Fensterpreistabellen (KÖMMERLING) ca.  $20 \cdot 400 = 8000$  Fensterprodukte, von denen jedes Produkt weitere Varianten besitzt. Allein durch die breite Auswahl an Farben (ca. 36) und Sicherheitsverglasungen (ca. 17) würde jedes Produkt über  $36 \cdot 17 = 612$  Varianten haben. Wird eine Fensterausführung als ein Produkt mit Varianten angesehen, dann gibt es ca. 20 Produkte mit jeweils tausenden von Varianten. Einige dieser Varianten sind

zudem sehr variabel, z. B. die Verbreitung des Profils, das in Centimetern vom Kunden angegeben wird und dessen Aufpreis von anderen Faktoren abhängt.

#### 2.2.2. Rolläden

Zu lösen in ferner Zukunft. Im Allgemeinen sollten keine funktionellen Einschränkungen vorgenommen werden.

#### 2.2.3. Zubehör

Zu lösen in ferner Zukunft. Im Allgemeinen sollten keine funktionellen Einschränkungen vorgenommen werden.

#### 2.2.4. Service

Zu lösen in ferner Zukunft. Im Allgemeinen sollten keine funktionellen Einschränkungen vorgenommen werden.

#### 2.2.5. Lieferung

Geliefert wird i. d. R. selbstständig. **Pro 100 km fallen 10€-15€ Lieferkosten an.** Die Lieferkosten lassen sich auch je Region verallgemeinern und bestimmen.

#### 2.2.6. Montierung

Zu lösen in ferner Zukunft. Montierung wird ggf. in der Zukunft angeboten. Dies gilt es zu klären.

### 2.3. Online-Shop-Anbieter

Es gibt mehrere Online-Shop-Anbieter, die das Erstellen einer Website ermöglichen. Auf Grund des Zeitdrucks und der generellen Ausgangssituation kommen jedoch nur zwei Online-Shop-Anbieter infrage: **Wix** und **WordPress**. Diese werden im folgenden Kapitel beschrieben und anschließend miteinander verglichen.

#### 2.3.1. Wix

Wix ist ein Website-Baukasten mit eigenständiger API namens Corvid. Alle Funktionen, von Datenbankschnittstellen bis hin zu Listenern, werden abgekapselt. Dadurch ist der Umgang mit denen in der Regel einfacher, jedoch eingeschränkter. Eine bei Wix erstellte Website ist für immer an Wix gebunden. Der Online-Shop wird über ein von Wix gestelltes Plugin umgesetzt. Dieses wird im Folgendem näher erläutert.

##### 2.3.1.1. Grundlagen

Bei Wix gibt es Pages, Public-Code und Backend-Code.

Pages stellen die oberste Schicht dar, auf denen die grafischen Inhalte angezeigt werden. Von hier aus werden Events abgefeuert und weitere Codefragmente aufgerufen. Pages können Objekte halten. Diese Objekte halten ihren Zustand solange, bis die Page gewechselt wird.

Public-Code ist der Client-Code. Sichtbar für den Server und den Client.

Backend-Code ist der Server-Code. Sichtbar nur für den Server.



#### 2.3.1.2. Online-Shop-Plugin

Das Online-Shop-Plugin erlaubt den Verkauf von max. 50.000 Produkten, von denen jedes Produkt insg. Über max. 6 Optionen verfügen kann mit max. 30 Auswahlmöglichkeiten je Option. Die Gesamte Anzahl an Variationen, die sich durch die Optionen ergibt, kann jedoch 300 nicht überschreiten [1]. Physikalische Produkte lassen sich über eine CSV-Datei hochladen, die allerdings max. 5000 Produkte auf einmal verarbeiten kann. Das Online-Shop-Plugin ist prinzipiell die schnellste Variante (auch wegen der Vorarbeit und Erfahrung mit Wix), um einen Online-Shop aufzubauen, sofern die Limitierungen das Projekt nicht zu sehr beanspruchen.

##### 2.3.1.2.1. Elemente

Wix verfügt über viele Elemente, die zusammen das Online-Shop-Plugin bilden. Die zur Erstellung/Hinzufügung/Konfigurierung/Veränderung von Produkten wichtigsten Elemente sind aus der Bibliothek wix-stores-backend und lauten Product (Produkt), Products-Kollektion, createProduct(), deleteProduct(), Lineltem und Cart (Einkaufswagen). Im folgendem werden diese genauer erläutert. Anschließend wird die Zusammenarbeit dieser Elemente erklärt, die beim Online-Shopping getan wird.

**Product:** Product repräsentiert ein Produkt im Online-Shop. Er verfügt unter anderem über eine eindeutige Id, einen Namen, eine Beschreibung, einen Preis, Varianten und beliebige Zusatzinformationen.

**Products-Kollektion:** In der Products-Kollektion werden alle Produkte gespeichert oder dynamisch erstellten Produkte hinzugefügt. Um die Kollektion im Code zu referenzieren, muss als Name der Kollektion „Stores/Products“ angegeben werden.

**CreateProduct():** CreateProduct() dient zum Erstellen eines neuen Produktes. Die Funktion nimmt eine ProductInfo als Parameter auf. Dieser Parameter ähnelt einer Liste, die Informationen zu dem Produkt enthält, das erstellt werden soll. Die Funktion gibt ein Promise zurück, das in dem erstellten Product resultiert, wenn der Erstellungsprozess abgeschlossen ist. Das erstellte Produkt wird automatisch der Kollektion „Products“ hinzugefügt. Jeder Nutzer kann diese Funktion aufrufen.

**DeleteProduct():** DeleteProduct() dient zum Löschen eines Produktes. Die Funktion nimmt die Id des Produktes als Parameter auf, das gelöscht werden soll. Jeder Nutzer kann diese Funktion aufrufen.

**Lineltem:** Ein Lineltem repräsentiert ein Produkt im Cart. Es ist jedoch nicht das Produkt selbst und enthält ihn auch nicht. Stattdessen wird die Id des gegebenen Produktes gehalten, das in der DB hinterlegt ist (beide Ids sind gleich). Des Weiteren enthält ein Lineltem Informationen über die Anzahl der gewünschten Produkte der gegebenen Id, den Gesamtpreis und ein CustomTextField (beliebiger Text).

**Cart:** Cart ist ein Objekt, dass unter anderem Daten über die Versandadresse, die Währung, Lineltems und den Gesamtpreis, der sich im Cart befindlichen Produkte enthält.

#### 2.3.1.2.2. Einfacher Einkaufsprozess

Der übliche Einkaufsprozess sieht wie folgt aus:

Zuerst wird das passende Produkt dem **Warenkorb** hinzugefügt. Produkte werden standardmäßig über ein **I-Frame** dem Warenkorb hinzugefügt, der **statisch** auf ein aus der Produkt-Kollektion stammendes Produkt verweist. Der Verweis lässt sich dynamisch nicht ändern. Alternativ lassen sich die Produkte auch über das Element „**shoppingCartIcon**“ dem Warenkorb hinzufügen. Dieses Element besitzt die Funktion „**addToCart**“, die diese Funktionalität realisiert. Das Produkt muss sich jedoch weiterhin in der Produkt-Kollektion befinden.

Das ausgewählte Element wird dem Warenkorb automatisch hinzugefügt. Der Kunde kann an dieser Stelle den Einkauf fortsetzen, oder zum Warenkorb gehen. Im Warenkorb werden alle ausgewählten Produkte aufgelistet. Die im Warenkorb sichtbaren Informationen sind der Titel des Produktes, der Preis, alle ausgewählten Optionen (!), ein optional hinzugefügtes Textfeld, das bis zu 500 Zeichen umfassen kann und weiteres (vgl. Darstellung 1). Klickt der Kunde auf ein sich im Warenkorb befindliches Produkt, dann wird er auf die Produktseite weitergeleitet (vgl. Darstellung 2).

Nach dem Klicken des Buttons „Zur Kasse“ wird man nach seinen **Versandangaben** gefragt (sofern eingeloggt), sowie nach der **Versandmethode** (vgl. Darstellung 3). Bei einer erfolgreichen Bestellung werden die **Bestellungsdaten** in der Orders-Kollektion und in der Kontoverwaltung automatisch sichtbar (vgl. Darstellung 4). Zu diesen gehören der **Status** der Bestellung (ausgeführt/Nicht ausgeführt, bezahlt/nicht bezahlt), eine Auflistung der Kosten (Steuer, Brutto, Netto etc.), eine Lieferadresse, Rechnungsadresse, das bestellte Produkt und weiteres. Das nachträgliche Löschen eines zuvor bestellten Produktes aus der Product-Kollektion hat keinen Einfluss auf die Orders-Übersicht und das bestellte Produkt ist dort weiterhin sichtbar (allerdings ohne jeglicher Details). Des Weiteren stehen einem mehrere Aktionen zur Verfügung, wie z. B. das Senden einer **Versandbestätigung**, die **Stornierung/Rückerstattung** und weiteres. Schließlich bekommt der Verkäufer eine **Bestätigungsemail** der Bestellung (vgl. Darstellung 5, Darstellung 6).

## Mein Warenkorb



The created product

~~79.000,00 €~~ 78.995,00 €

78.995,00 €

X



Testprodukt Fenster

5,00 €

OptionA: WahlA1

OptionB: WahlB1

OptionC: WahlC1

OptionD: WahlD1

OptionE: WahlE1

OptionF: WahlF1

Fenstermaßen?: 500x400

5,00 €

X

## Bestellübersicht

Zwischensumme 79.000,00 €

Versand KOSTENLOS

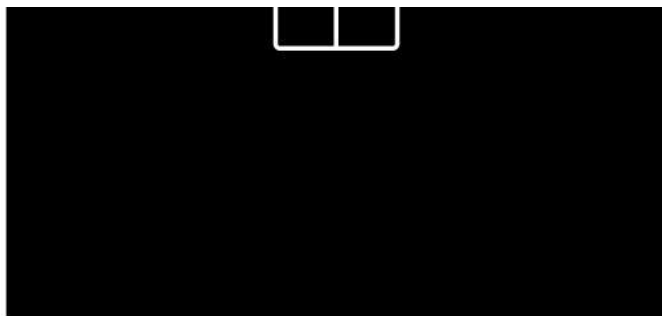
Nordrhein-Westfalen,  
Deutschland

Gesamtsumme 79.000,00 €

Zur Kasse

Bezahlen mit **PayPal**

Darstellung 1: Wix Online-Shop-Plugin, Warenkorb



Das ist die Beschreibung.  
Z.B. Farbe: Grün

WahlB1

OptionC

WahlC1

OptionD

WahlD1

OptionE

WahlE1

OptionF

WahlF1

Fenstermaßen? (optional)

500

Darstellung 2: Wix Online-Shop-Plugin, Produktansicht

Eingeloggt als vision.kundenservice@gmail.com
Abmelden

## 1 Versandangaben

\*Vorname

\*Nachname

\*Adresse

\*Stadt

### Bestellübersicht (1)

[Bestellung bearbeiten](#)

testprodukt  
Menge: 1

[Gutscheincode eingeben](#)

Zwischensumme	0,01 €
Versand	Kostenlos
USt / MwSt	0,00 €
<b>Gesamtsumme</b>	<b>0,01 €</b>

SSL SECURE SHOPPING  
Your data is safe and secure.

Darstellung 3: Wix Online-Shop-Plugin, nach "Zur Kasse Button".

WIX
Meine Websites
Entdecken
Hilfe
Partner von Wix engagieren
Suchen Sie Werkzeuge, Apps und mehr
Agnieszka Mosins...

## Bestellung Nr. 10001

Aufgegeben am 22. März 2020, 14:52

Bezahlt
Nicht ausgeführt

### Bestellübersicht (1 Artikel)

Versandbereit [Sendungsnummer hinzufügen](#)

testprodukt	Anzahl: 1	0,01 €
-------------	-----------	--------

### Zahlungsübersicht

Zwischensumme	0,01 €
Versand	0,00 €
Steuer	0,00 €
<b>Gesamtsumme</b>	<b>0,01 €</b>

### Bestellaktivität

### Kundeninfo

**MR** Maria R

**LIEFERADRESSE**  
Ramírez María  
Hansaring 38  
48155Münster, Nordrhein-Westfalen,  
Deutschland  
01621989443  
vision.kundenservice@gmail.com  
[Karte ansehen](#)

**RECHNUNGSADRESSE**  
Gleicht der Lieferadresse

**VERSANDMETHODE**  
Kostenloser Versand

Weitere Aktionen

- Als ausgeführt markieren
- Versandbestätigung per E-Mail senden
- Bestellung drucken
- Stornierung & Rückerstattung
- Rückerstatten
- Archivieren

Darstellung 4: Wix Online-Shop-Plugin - Übersicht einer Bestellung im Verkäuferkonto

Von WixStores <no-reply@mystore.wix.com> ☆

Betreff: Sie haben eine Bestellung erhalten (#10001)

An Mich <mosinski95@gmail.com> ☆

↩ Antwort

In Ihrem Online-Shop wurde eine Bestellung aufgegeben.

**Bestellung #10001**

Bestellung ansehen

**Bestellinformationen**

Bestellung #: 10001  
 Aufgegeben am: Mar 22, 2020  
 Gesamtpreis: 0,01 €  
 Zahlungsstatus: Beahlt  
 Zahlungsmethode: PayPal

Rechnung	Versand
María Ramírez Hansaring 38 Münster, Nordrhein-Westfalen, Deutschland 48155 01621989443 vision.kundenservice@gmail.com	María Ramírez Hansaring 38 Münster, Nordrhein-Westfalen, Deutschland 48155 01621989443 <b>Versandmethode:</b> Kostenloser Versand

Darstellung 5: Wix Online-Shop-Plugin - E-Mail-Bestätigung eines Zahlungseingangs, Verkäufer (1)

Von WixStores <no-reply@mystore.wix.com> ☆

Betreff: Sie haben eine Bestellung erhalten (#10001)

An Mich <mosinski95@gmail.com> ☆

↩ Antwort

vision.kundenservice@gmail.com

**Versandmethode:** Kostenloser  
Versand

**Bestelldetails**

Artikel	Menge	Gesamtsumme
testprodukt Preis: 0,01 €	1	0,01 €

Zwischensumme	0,01 €
Versand	0,00 €
MwSt.	0,00 €
<b>Gesamtsumme</b>	<b>0,01 €</b>

Darstellung 6: Wix Online-Shop-Plugin - E-Mail-Bestätigung eines Zahlungseingangs, Verkäufer (2)

#### 2.3.1.2.3. Einschränkungen

#### 2.3.1.3. Wix-Pay-Plugin

Wix-Pay ist ein Plugin für Zahlungen. Die Zahlungsmethoden reichen über PayPal, Kreditkarte, Banküberweisung etc. Es lassen sich mehrere Produkte zu einer Zahlung hinzufügen, die jeweils eine kurze **Beschreibung**, den **Preis** und die **Stückanzahl** besitzen. Des Weiteren werden der **Gesamtbetrag** und die **Währung** angegeben. Außerdem können Informationen zum aktuellen User angegeben werden, die dann bei der Zahlung im Voraus ausgefüllt werden. Löst man eine Zahlung über einen entsprechenden Button aus, dann wird man weitergeleitet, wo man nach seinen **persönlichen Daten** gefragt wird, die zur Vollendung der Zahlung nötig sind (vgl. Darstellung 7). Bei einer erfolgreichen Zahlung bekommen der Verkäufer und der Käufer eine **Bestätigungsemail**. Diese kann sich je nach Zahlungsmethode leicht von anderen Bestätigungsemails unterscheiden, beinhaltet jedoch in allen Fällen die gleichen Informationen. Nach einer Zahlung ist an dem Rückgabewert der Funktion, die für die Ingangsetzung der Zahlung zuständig ist (startPayment()) ersichtlich, ob die Zahlung erfolgreich war. Dabei gibt es 9 verschiedene Payment-Resultate [6]:

- "Successful": Payment was successfully received.
- "Pending": Payment is pending payment provider approval.
- "Failed": Payment has failed.
- "Chargeback": Payment is chargeback.
- "Refunded": Payment was fully refunded.
- "Offline": Payment will be executed offline.
- "PartiallyRefunded": Payment was partially refunded.
- "Cancelled": Payment was cancelled and was not processed.
- "Undefined": Payment status is pending payment provider input.

Wenn sich der Status des Payments geändert hat, dann wird ein Event gefeuert. onPaymentUpdate() bietet die Möglichkeit, diesen abzufangen [7]. Die startPayment()-Funktion nimmt außerdem optionale Parameter entgegen, die einige Zusatzoptionen erlauben. Diese sind beispielsweise das Einblenden einer „Thank You-Page“ am Ende der Zahlung oder der rechtlichen Informationen. Die zu der Bestellung zugehörigen Daten, inkl. der bestellten **Produktnamen**, werden in der Transaktionsansicht im Wix-Dashboard angezeigt.

Zur Kasse

0,11 € ✓

## 1. Ihre Details

\* E-Mail-Adresse

\* Vorname

\* Nachname

\* Land

Telefonnummer

Weiter

🔒 Sichere Zahlung mit SSL

Darstellung 7: Wix Payment-Plugin, nach "zur Kasse Button"

### 2.3.1.4. Wix Mail

Wix verfügt über eine Bibliothek, mithilfe derer sich eine beliebige E-Mail versenden lässt.

### 2.3.1.5. Wix-users

Wix verfügt über eine Bibliothek, mithilfe der sich User registrieren und einloggen können. Die von einem registrierten User eingegebenen Informationen lassen sich dann jederzeit abrufen.

### 2.3.1.6. Wix-Datenbanken

#### 2.3.1.6.1. Wix-Data

Wix-Data ist eine Bibliothek, die den Zugriff zu den Datenbanken von Wix ermöglicht. Sie verfügt über alle nötigen Bibliotheken zur Änderung, Abrufung, Löschung etc. von Daten.

#### 2.3.1.6.2. Hooks

**Hooks** sind Funktionen, die ausgeführt werden, bevor Daten hinzugefügt/geändert etc. ausgeführt werden. Sie lassen sich für jede Kollektion beliebig implementieren.

#### 2.3.1.6.3. Kollektion

Die einzelnen **Tabellen** einer Wix-Datenbank werden Kollektionen genannt.

Wix-Kollektionen verfügen über mehrere **Datentypen**, unter anderem Text, Nummer, Bild, Referenz etc.

Jeder **Eintrag** in einer Kollektion verfügt über eine Id, die sich nachträglich nicht ändern lässt. Sie lässt sich beim Einfügen eines Eintrages in die Kollektion festlegen.

#### 2.3.1.7. Wix-Dataset

Ein Dataset ist eine Art **Brücke zwischen einer Kollektion und den grafischen Elementen einer Page**. Über ein Dataset lassen sich Page-Elemente mit Kollektionen verbinden. Es wird eine Abbildung zwischen den Spalten einer Kollektion und den Elementen in einer Kollektion erstellt. Datasets lassen sich nur auf Pages anwenden (nicht im Backend). [11]

Ein Dataset besitzt drei Zugriffsrechtmodi, die programmiertechnisch nicht geändert werden können: Read & Write, Read-Only, Write-Only.

Die **Inhalte**, die über das Dataset angezeigt werden, lassen sich auf eine einfache Art über **Filter** filtern. Sie lassen sich ebenfalls einfach über eine Funktion des Datasets namens **getItems()** abrufen.

Items werden bei Datasets in Chunks angezeigt. Es lassen sich maximal 1000 Items auf einmal anzeigen lassen.

**Referenzierte Daten**, d. h. Daten, die in der Kollektion über Referenzfelder verfügbar sind, lassen sich nur unmittelbar über eine Ebene abrufen. D. h., wenn in der Kollektion A auf ein Feld b aus der Kollektion B verwiesen wird und in der Kollektion B eine Referenz c auf ein Feld aus der Kollektion C existiert, dann kann ein Dataset, das auf die Kollektion A zugreift, maximal auf das referenzierte Feld b aus Kollektion B verweisen, jedoch nicht auf das Feld c aus Kollektion c.

#### 2.3.1.8. Wix-Selectors

Ein Wix-Selector dient dazu, Items aus verschiedenen Scopes auszuwählen. Im Folgendem werden die verschiedenen Selectoren beschrieben. [12]

##### 2.3.1.8.1. Global-Selector

Der Global-Selector besitzt einen Page-Scope erlaubt den Zugriff auf alle Page-Elemente. Dieser Zugriff wird über die \$w() getätigt.

##### 2.3.1.8.2. Scoped-Selector

Der Scoped-Selector besitzt einen bestimmten Scope. Dieser lässt sich über die Funktion \$w.at() auswählen. Beispielsweise der folgende Code ändert den Text eines



bestimmten Repeater-Textelementes, nachdem ein Bild in demselben Repeater-Container angeklickt worden ist:

```
$w.onReady( function () {  
  $w("#myRepeatedImage").onClick( (event) => {  
    let $item = $w.at(event.context);  
    $item("#myRepeatedText").text = "Selected";  
  } );  
} )
```

Alle Elemente aus dem bestimmten Repeater-Container werden über `$item()` angesprochen. [13].

#### 2.3.1.9. Pages und grafische Elemente

Bei Wix lassen sich einzelne grafische Elemente einfach per Drag & Drop auf einer Page platzieren und anhand der grafischen Oberfläche visuell anpassen. Zudem lässt sich das Verhalten der Elemente bei verschiedenen Interaktionen oder Events anpassen. Beispielsweise lässt sich ein Listener leicht hinzufügen, für den Fall, dass beispielsweise ein Button angeklickt wird.

Besonders interessant und nützlich bei der Visualisierung von variablen Inhalten sind die Elemente **Repeater** und **Container**. Diese werden im Folgendem genauer erläutert.

#### 2.3.1.10. Repeater

Repeater dienen zum wiederholten Anzeigen von Elementen, die sich strukturell nicht voneinander unterscheiden. [13]

Es existieren zwei **Scopes** für Repeater: Global und Repeat. Der Global-Scope gleicht dem üblichen Page-Scope und wird wie gewohnt über `$w()` aufgerufen. Der Repeater-Scope umfasst nur die Elemente, die sich im gegebenen Repeat-Element befinden (Vgl. Kapitel 2.3.1.8).

Um einen Repeater mit Daten zu **initiiieren**, wird die Funktion `onItemReady($item, itemData, index)` mit den genannten Parametern aufgerufen. Die Daten werden nicht automatisch den Elementen zugewiesen. Über den Parameter `$item` kann auf die Elemente eines jeden Repeater-Containers zugegriffen und dessen Inhalte gesetzt werden. In dem folgenden Beispiel werden die Daten für einen Repeater aus einer Database-Query gesetzt:

```
$w.onReady(function () {  
  $w("#myRepeater").onItemReady( ($item, itemData, index) => {  
    $item("#bookTitle").text = itemData.title;  
    $item("#bookSubtitle").text = itemData.subtitle;  
  } )  
}
```

```

    $item("#bookCover").src = itemData.pic;
  });

wixData.query("Books")
  .find()
  .then( (results) => {
    $w("#myRepeater").data = results.items;
  });
});

```

Die **Daten** für einen Repeater werden in einem Array von Objekten namens `itemData` festgehalten. Die Anzahl der Elemente in diesem Array bestimmt die Anzahl der sich wiederholenden Repeater-Container. Jedes dieser Objekte muss eine eindeutige ID besitzen, hinterlegt als `_id`. Das Data-Array für den Repeater kann nicht in-place modifiziert werden. Um die Daten für den Repeater zu modifizieren, müssen die Daten in einer temporären Variable gespeichert, dort bearbeitet, und dann erneut gesetzt werden. Dies wird im folgenden Beispiel veranschaulicht:

```

// get current data array
let dataArray = $w("#myRepeater").data;

// change something in the data array
dataArray[0].somefield = "New value";

// reset repeater data
$w("#myRepeater").data = dataArray;

```

Die **Daten** für einen Repeater können aber auch über ein Dataset angebunden werden. Dies lässt sich sehr einfach über die grafische Oberfläche realisieren. Besonders gut funktioniert auf diese Weise die Filterung der Inhalte. Über das Setzen von Filtern für das gegebene Dataset, das mit dem Repeater verbunden ist, können Elemente ein- und ausgeblendet werden. Es sind keine weiteren Aktualisierungen der Elemente nötig. Intern werden die Items des Datasets dem Data-Feld des Repeaters zugewiesen. Ein

schreibender Zugriff auf den Repeater über das Dataset und ein zugleich lesender Zugriff auf den Repeater über die onReady-Funktion ist problemlos möglich.

Die Repeater verhalten sich sehr intelligent. Sie **positionieren** sich korrekt und fast vollkommen automatisch auf der Website und reagieren korrekt auf Änderungen. Beispielsweise „rutscht“ ein Element B, das sich unter dem Element A befindet, nach oben, wenn Element A zur Laufzeit ausgeblendet wird. Genauso rutscht Element B wieder nach unten, wenn Element A wieder eingeblendet wird.

#### 2.3.1.11. Container

In einem Container werden Repeater-Items gehalten. Er besitzt mehrere Funktionen. Unter anderem lässt sich der gesamte Container mithilfe der Funktionen collapse()/expand() aus- und einblenden lassen. [14]

#### 2.3.1.12. Rasterlinien

Die Rasterlinien kennzeichnen Bereiche, die beim Wechsel zur mobilen Ansicht möglicherweise nicht mehr sichtbar sind.

#### 2.3.1.13. Fixierte Elemente

Das Fixieren eines Elements bewirkt, dass die Position des Elementes relativ zum Viewport betrachtet wird. D. h., dass sich das Element immer an der gegebenen Position befinden wird, selbst wenn gescrollt wird. Um ein Element zu fixieren, muss das Element mit einem Rechtsklick angeklickt werden und die Option „fixieren“ muss ausgewählt werden. Die Elemente sind bei der mobilen Ansicht nicht fixiert.

### 2.3.2. WordPress

Die meisten Websites werden mit WordPress erstellt. Die Erstellung bedarf mehr Aufwand und Vorwissen (mit der Tatsache, dass der Entwickler kein PHP o. Ä. kennt), gibt jedoch i. d. R. alle nötigen Möglichkeiten. Eine mit WordPress erstellte Seite lässt sich auf einen beliebigen Server übertragen. Bei WordPress wird für die Verwaltung der Produkte WooCommerce (Plugin) verwendet. Die Produkte lassen sich über eine CSV-Datei erstellen, verwalten und importieren. Die Erstellung verschiedener Varianten wird sehr einfach gehalten [3]. Zu WooCommerce existiert zudem ein Plugin namens „Extra Product Options“, mithilfe dessen sich die Preise dynamisch und abhängig von anderen Optionen und Formeln berechnen lassen [4].

### 2.3.3. Wix vs. WordPress

Im Folgenden werden die Unterschiede von WordPress und Wix tabellarisch gegenübergestellt anhand der für ein Projekt und Webhosting üblicher Kriterien.

Kriterium	Wix (Business Basic)	WordPress (Bei 1&1, Business und WooCommerce)
Kosten (pro Jahr)	Ca. 200€ + Domain pro Jahr	Ab 100€ pro Jahr + einmalig ~200€
Design	Großartige und einfache Design-Möglichkeiten	Viele Design-Optionen

Technische Einschränkungen (Anz. Produkte, prozentuelle Preise, CSV-Import)	<ul style="list-style-type: none"> <li>- Max. 50.000 Produkte</li> <li>- Max 6 Optionen pro Produkt</li> <li>- Max 30 Auswahlmöglichkeiten pro Option</li> <li>- Max 300 Variationen pro Produkt</li> <li>- Keine prozentuellen Preise</li> </ul>	<ul style="list-style-type: none"> <li>- Theoretisch ist alles möglich</li> <li>- Keine Einschränkungen für die max. Anz. Der Produkte und ihrer Variationen</li> <li>- Prozentuelle Preise für einmalig 120€ zu erwerben</li> </ul>
Aufwand bzgl. des aktuellen Vorwissens und getätigter Vorarbeit	<ul style="list-style-type: none"> <li>- Viel Vorwissen vorhanden</li> <li>- Ein großer Teil der Arbeit bereits getan (code, DB etc.)</li> </ul>	<ul style="list-style-type: none"> <li>- Wissen in PHP erforderlich</li> <li>- Wissen in WordPress erforderlich</li> </ul>
Serverkapazität	20GB (ohne E-Mail)	25 GB
E-Mail	Wird über G-Suite geregelt (mind. 5€ pro Monat Kosten, 30GB)	<ul style="list-style-type: none"> <li>- Max. 10 GB</li> </ul>
Zahlungsmöglichkeiten	Alles nötige	Alles nötige

## 2.4. Datenbanken vs. Session/Local-Storage

Datenbanken und Session/Local-Storage sind die populärsten Methoden der Datenspeicherung in Web-basierten Applikationen. Im folgendem werden diese beiden Methoden gegenübergestellt und die Pro und Kontra dieser tabellarisch dargestellt und anschließend erläutert:

Kriterium	Datenbank	Session-/Local-Storage
<b>Speicherungsdauer</b>	Dauerhaft, bis zur Löschung.	Grundsätzlich für die Lebenszeit einer Session, höchstens jedoch bis die Session-/Local-Storage vom Client gereinigt wird.
<b>Zugriffsgeschwindigkeit</b>	Langsam, da externer Zugriff nötig.	Schnell, nicht wahrnehmbar in einem üblichen Use-Case.
<b>Rechtliche Folgen</b>	Daten müssen gepflegt und geschützt werden. Hinweis definitiv nötig.	Hinweis auf Datenerhebung reicht aus.
<b>Wartbarkeit</b>	Muss ggf. regelmäßig aufgeräumt werden, was durch einen Nutzer geschehen muss.	Muss nicht aufgeräumt werden, da die Daten im Regelfall automatisch nach einiger Zeit gelöscht werden oder lassen sich sehr einfach, automatisch

		löschen. Müssen jedoch verwaltet werden.
<b>Zugriffsscope</b>	Alle Nutzer und Admins, die Zugriff auf die Db haben. Zugriffsscope lässt sich gezielt steuern.	Zugriff hat grundsätzlich nur der Client.

Grundsätzlich gilt, dass jegliche in den Datenbanken gespeicherten Inhalte zum einen von den Regelungen der DSGVO besonders stark beeinflusst sind, da diese grundsätzlich dauerhaft und nicht kurzzeitig, wie im Falle der lokalen Speicherung beim Client, gespeichert werden. Des Weiteren kosten Datenbankzugriffe mehr Zeit als die Zugriffe auf den lokalen Speicher. Auf die Session-/Local-Storage hat grundsätzlich nur der Client Zugriff, während der Zugriff auf die Datenbank sich gezielt bestimmen lässt. Außerdem sind die Datenbanken die einzige Methode, Daten dauerhaft und zuverlässig zu speichern. Die Folge davon ist, dass diese regelmäßig gepflegt werden müssen.

Zusammenfassend gilt, dass in einer Datenbank alle Daten gespeichert werden sollten, die dauerhaft gespeichert werden und/oder auf die Daten von anderen Personen als vom Client selbst, zu dem die Daten gehören, zugegriffen werden soll.

## 3. Methodik

### 3.1. Entwicklungsstil

Es wird **agil** entwickelt. Die Vorgehensweise folgt dem **Kanban**-Stil. Jede Iteration ist eine bessere Version des vorherigen Endproduktes. Es wird angestrebt, dass der Detaillierungsgrad mit der Zeit auf allen Ebenen zunimmt.

### 3.2. Projektmanagement

Diese Dokumentation wird parallel zum Fortschritt aktualisiert und fortgeführt und soll ein klares Verständnis über das gesamte Projekt liefern. Die Anforderungen werden zu Beginn grob in dieser Dokumentation erhoben. Weitere Anforderungen werden zum späteren Zeitpunkt aufgenommen und über die Applikation **Zenkit** gepflegt, aktualisiert, erweitert und verwaltet. Größere Diagramme, Tabellen etc. werden der Übersicht halber im selben **Verzeichnis** in getrennten Dateien gespeichert. In manchen Verzeichnissen ist eine README-Datei zu finden, die die Struktur der sich in dem Verzeichnis befindlichen Dateien erläutert. Auf diese Dateien wird bei Bedarf im Text verwiesen. Insbesondere die einzelnen Eigenschaften der Produkte sowie die Produkte selbst werden in Excel-Dateien gespeichert, um eine einfache Weiterverarbeitung oder etwa CSV-Imports zu ermöglichen. Daten für die Website werden zunächst **Lokal**, anschließend auf dem ausgewählten **Server** gespeichert. **Rechtstexte** sind online auf der Trustedshops-Plattform [5] abrufbar mit der E-Mail, die zur Domain gehört.

### 3.3. Programmierrichtlinien

Beim Hinzufügen von neuen Funktionalitäten oder beim Erweitern der alten Funktionalitäten werden folgende Schritte durchlaufen:

- Struktur festlegen
- Tests schreiben
- Ggf. Backup anlegen
- Implementierung
- Korrektur
- Dokumentation

### 3.4. Dokumentation im Code

Dokumentiert wird im Code auf Englisch.

#### 3.4.1. Funktionen und Klassen

- Vor jedem Konstruktor wird eine generelle Beschreibung der Klasse eingefügt und deren Parameter beschrieben.
- Vor jeder Methode/Funktion wird eine generelle Beschreibung der Methode/Funktion eingefügt und deren Parameter beschrieben.
- Vor jeder Methode/Funktion werden alle Errors aufgeführt, die die Methode/Funktion schmeißen kann und die behandelt werden müssen.

#### 3.4.2. Kommentare

- Einfache Kommentare werden mit // eingefügt. Beispiel: // yourmum

### 3.4.3. Tags

Tags werden ggf. als Kommentare unmittelbar vor dem Funktionsanfang, hinter der Funktionsdokumentation eingefügt.

- `@abstract`: Abstrakte Methode. Wirft immer „MissingImplementationError“. Muss in unteren Klassen überschrieben werden.
- `@override`: Symbolisiert das Überschreiben einer bestimmten Methode von einem Parent oder Interface.

### 3.4.4. Tests

Jeder Test beschreibt einen Zweck, das zu erwartende Ergebnis und das tatsächliche Ergebnis.

## 3.5. Namenskonventionen

### 3.5.1. Verzeichnisse

Alle Source-Dateien befinden sich im `src`-Ordner. Labels und options sind in einem separaten Ordner getrennt. Classtests befinden sich in einem Extraordner Classtests in Tests. Als Name wird immer der klarste Ausdruck des Zwecks der Datei gewählt.

### 3.5.2. Klassen

Klassennamen beginnen mit einem Großbuchstaben.

Klassen, die Items aus einer Datenbanktabelle repräsentieren, heißen genauso wie die Datenbanktabelle.

Abstrakte Klassen haben immer `Abstract` am Anfang des Klassennamens.

Interfaces haben immer `Interface` am Ende des Klassennamens als Schlüsselwort.

Errors haben immer `Error` am Ende des Klassennamens als Schlüsselwort.

Nach Möglichkeit wird der Name der Elternklasse in der Kindklasse erweitert. Z. b.: `Product` -> `SimpleProduct`.

Interface-Klassennamen ergeben sich aus der Funktionalität, um die das Interface das jeweilige Objekt erweitert. Z.B `PrintableInterface`.

### 3.5.3. Funktionen

Boolsche Funktionen werden mit den Hintergedanken an `if`-Bedingungen benannt. Z. b. „`inherits()`“.

### 3.5.4. Tests

Tests heißen genauso wie die Funktion, die getestet wird – nur wird am Anfang des Namens „`Test`“ hinzugefügt. Z.B: `testInherits()`.

Gibt es mehrere Tests zu einer Funktion, so gibt es eine Testfunktion, die alle Untertestfunktionen aufruft. Die Untertestfunktionen werden dann dem Alphabet nach durchgenannt. Z.B: `testInherits()` ruft `testInheritsA()`, `testInheritsB()` ... auf.

Weitere Namenskonventionen sind den bereits vorhandenen Tests zu entnehmen.

#### 3.5.5. Datenbanken

Namen der Kollektionen und deren Spalten werden klein geschrieben und einzelne Wörter werden durch ‚\_‘ getrennt.

#### 3.5.6. Seiten

Wird eine bestimmte Komponente aus einer bestimmten Kollektion auf der jeweiligen Seite gewählt, dann heißt die Seite genauso wie die Kollektion.

#### 3.5.7. Pakete

Namen werden klein geschrieben und einzelne Wörter durch ‚\_‘ voneinander getrennt.

### 3.6. Packaging

Die Packages werden nach dem Prinzip „layered feature“ strukturiert, um die beste Übersicht zu behalten.



## 4. Anforderungsanalyse

### 4.1. Ist-Zustand

Jegliche Programmierarbeit ist bisher ausschließlich in Wix getan worden. Eine Grundidee ist vorhanden darüber, wie der Shop auszusehen hat. Der Entwicklungsstand ist mit einer Pause von 5 Monaten stehengeblieben. Das Datenbankmodell ist bereits erstellt und in der Praxis umgesetzt worden. Rechtliche Bausteine wie AGB, DSGVO etc. sind bereits einmal generiert worden. Bildmaterialien sind teilweise vorhanden.

### 4.2. Soll-Zustand

Der Online-Shop soll **bis Ende Mai** soweit ausgebaut werden, dass er in Betrieb genommen werden kann und zumindest der Verkauf von Fenstern möglich ist. Das bedeutet unter anderem, dass der Online-Shop über ein ausreichend ausgebautes Design verfügen soll. Der Kaufvorgang muss fehlerfrei ablaufen. Rechtlich soll sich die Seite auf einem Stand befinden, der einem Online-Shop gerecht ist. Eingeschränkt wird die Produktbreite zunächst auf Fensterprofile der Firma KÖMMERLING, insbesondere auf das Profil 76 AD. Eine spätere Erweiterung soll jedoch weiterhin einfach sein.

### 4.3. Methodik der Anforderungserhebung

Die Anforderungen werden grob vom Stakeholder aufgenommen, zum größten Teil werden sie jedoch vom Entwickler ausformuliert und erweitert. Dies geschieht in Form von Interviews.

### 4.4. Anforderungen

Der folgende Stand der Anforderungen ist die Ausgangslage für das Projekt. Die gleichen Anforderungen sind im Ticket-System Zenkit zu finden und werden dort erweitert und ergänzt.

#### 4.4.1. Nichtfunktionale Anforderungen

Der Online-Shop soll **bis Ende Mai** betriebsfähig sein (Vgl. Kapitel 4.2). Diese Anforderung besitzt eine hohe Priorität, da ab Anfang Mai aufgrund einer Vollzeitbeschäftigung voraussichtlich keine Zeit mehr überbleiben wird, um den Online-Shop weiterentwickeln zu können. Der Verwaltungsprozess muss so einfach sein, sodass diese Arbeit auch neben der Vollzeitbeschäftigung problemlos erledigt werden kann oder der Admin das ggf. selbst machen kann.

Der Einkaufsprozess soll grundlegend möglichst **schnell und korrekt** ablaufen. Die benötigten **Wartezeiten** sollten nach Möglichkeit **zusammengefasst** werden und an den Stellen auftreten, an denen es zu erwarten ist (beispielsweise beim Löschen eines Produkts aus dem Warenkorb, aber nicht bei der Auswahl einer Produkteigenschaft im Konfigurator).

#### 4.4.2. Funktionale Anforderungen

Ein üblicher Einkauf im Online-Shop soll möglich sein, ähnlich wie unter <https://www.fensterversand.com/?cid=25&t=fenster-kunststoff-ideal-4000> zu sehen. Die speziellen Anforderungen an einzelne Komponenten des Online-Shops werden in den folgenden Unterkapiteln erläutert.

#### 4.4.2.1. Produkte

**Alle Produkte** müssen sich im Online-Shop aufnehmen lassen. Zu beachten ist, dass es einerseits ggf. viele Produkte oder Varianten geben kann und andererseits werden manche Preise pauschal, andere prozentual bestimmt. Bei der Lösungsbestimmung kann auch ein Workaround als Möglichkeit angesehen werden, bei dem die Produkte und ihre Preise über ein Excel-Skript aktualisiert werden und dann über einen CSV-Import im Online-Shop aktualisiert werden. Eine **einfache Aktualisierung** der Produktpreise soll jedenfalls möglich sein.

#### 4.4.2.2. Produktauswahl und -Konfiguration

Die Produkte sollen sich inkl. aller vorhandener Optionen beliebig konfigurieren lassen, i. A. ohne Einschränkungen. Preise müssen sich somit sowohl pauschal als auch prozentual bestimmen lassen. Der Konfigurator soll **nur gültige Optionen** anzeigen, basierend auf der bisher vorgenommenen Konfiguration eines Produkts (vgl. Darstellung 8). Bei der **Größeneingabe** des Fensters ist der gültige Wertebereich textuell anzuzeigen und die Eingabe zu überprüfen.

FENSTERTYP ? mehr Details

Einteilig ✓ Zweiteilig Dreiteilig Sondertypen

OBER-/UNTERLICHT

ohne Ober-/ Unterlicht mit Oberlicht ✓ mit Unterlicht

Fragen zu unseren Produkten?

Experten kontaktieren

Darstellung 8: Fensterversand.com - Fensterkonfiguration

Eine **Übersicht des aktuell bearbeiteten Fensters** soll an der Seite des Konfigurationsmenüs sichtbar sein. Damit ein Preis schon zu Beginn der Konfiguration sichtbar ist, soll die billigste Konfiguration bereits vorausgewählt sein. Die Übersicht soll den Preis des konfigurierten Produktes und einige Konfigurationsdetails anzeigen, die das Produkt grundlegend definieren. Zu diesen Details gehören das Material, Profil, die Farbe, der Fenstertyp, die Öffnungsart und die Gesamtgröße des Fensters. (Vgl. Darstellung 9).

IHRE KONFIGURATION

Innenansicht:

+

+

Regulärer Preis

326,83 €

Aktions-Rabatt

- 32,68 €

Neuer Preis

294,15 €

Sie sparen 10%

DETAILS

Material: Kunststoff

Profil: IDEAL 4000

Farbe und Dekore: Weiß

Fenstertyp: Einteilig

Öffnungsart: 1-teilig festverglast  
Oberlicht

Größe: 555 mm - 1555 mm

Darstellung 9: Fensterversand.com - Übersicht der aktuellen Fensterkonfiguration



#### 4.4.2.3. Warenkorb

Die Produkte sollen sich bequem und fehlerfrei auswählen und **zum Warenkorb hinzufügen** lassen. Diese sollen dort mindestens solange bestehen bleiben, bis der Kunde den Tab schließt. Im Warenkorb soll man die Möglichkeit haben zuvor ausgewählte **Produkte zu löschen, duplizieren oder zu bearbeiten**. Außerdem befinden sich Buttons im Warenkorb, die den Kunden zur Kasse bringen oder ihn weiter einkaufen lassen. Eine Auflistung der Lieferkosten soll zu sehen sein und diese sollen in den Preis hineinberechnet werden. Die zum Warenkorb hinzugefügten Produkte müssen im Warenkorb verständlich und **übersichtlich angezeigt** werden. Zu jedem Produkt werden die folgenden Informationen angegeben:

- Die Position (abgekürzt Pos.), die die Produkte im Warenkorb durchnummeriert
- Die Anzahl, die sich übrigens auch dynamisch ändern lassen soll
- Eine Beschreibung. Grundlegend, mit der Möglichkeit des Aufklappens eines Menüs, das eine vollständige Beschreibung des Produktes liefert.
- Der Einzelpreis des Produktes

- Der Gesamtpreis des Produktes unter Berücksichtigung der Anzahl der Produkte
- Eine Preisübersicht, die die Summe im Netto, die MwSt. und die Gesamtsumme angibt.

Ein Orientierungsbeispiel kann Darstellung 10 entnommen werden.

Pos.	Anzahl	Produkt	Beschreibung	Einzelpreis	Gesamtpreis												
1	1		<div>Positionsbezeichnung (z.B. Küche; keine technischen Anmerkungen)</div> <div> <b>Fenster</b> 189,00 € <ul style="list-style-type: none"> <li>Material: Kunststoff</li> <li>Profil: IDEAL 4000 (5-Kammer-System mit 2 Dichtungen)</li> <li>Farbe und Dekore: Weiß</li> <li>Fenstertyp: Zweiteilig</li> <li>Ober-/Unterlicht: ohne Ober-/ Unterlicht</li> <li>Öffnungsart: 2-teilig fest fest</li> <li>Gesamtbreite: 885 mm</li> <li>Gesamthöhe: 1555 mm</li> </ul> MEHR DETAILS </div> <div> BEARBEITEN DUPLIZIEREN ENTFERNEN </div>	331,27 €	331,27 €												
<div> <div>  <div> Käuferschutz inklusive Sehr gut 4.7/5.00 4,4 ★★★★★ Google Kundenrezensionen </div> </div> <table> <tr> <th>Voraussichtliche Lieferung</th><th>Lieferkosten</th><th>Mengenrabatt</th><th>Technische Daten</th><th>SUMME NETTO</th><th>331,27 €</th></tr> <tr> <td>Ihre Bestellung wird voraussichtlich zwischen <b>11.05.2020 - 15.05.2020</b> geliefert.</td><td>           Ab einem Kaufpreis von 1.000 € liefern wir <b>versandkostenfrei</b>            Unter 1.000 € berechnen wir:  <b>49,00 €</b> für Fenster, Türen und Vordächer  <b>16,90 €</b> für Rollläden und         </td><td>           Preisvorteil ab netto Bestellwert:            2,0% 5.000 €            4,0% 10.000 €            5,0% 15.000 €            6,0% 20.000 €            7,0% 25.000 €            8,0% 30.000 €            10,0% 40.000 €            12,0% 50.000 €         </td><td>           Wichtige Eckdaten zu Ihrer Bestellung:             Fläche gesamt: 1,38 m²            Laufmeter: 4,88 m            Stückzahl Fenster gesamt: 1            Stückzahl Türen gesamt: 0            Rabatt: 0%         </td><td> <b>MENGENRABATT (0%)</b> 0,00 €  <b>Summe netto neu</b> 331,27 €  <b>RABATTAKTION (-10%)</b> -33,13 €  <b>Summe netto neu</b> 298,14 €            Versandkosten 41,18 €  <b>Gesamt</b> </td><td> <div>Experten kontaktieren</div> </td></tr> </table> </div>						Voraussichtliche Lieferung	Lieferkosten	Mengenrabatt	Technische Daten	SUMME NETTO	331,27 €	Ihre Bestellung wird voraussichtlich zwischen <b>11.05.2020 - 15.05.2020</b> geliefert.	Ab einem Kaufpreis von 1.000 € liefern wir <b>versandkostenfrei</b> Unter 1.000 € berechnen wir: <b>49,00 €</b> für Fenster, Türen und Vordächer <b>16,90 €</b> für Rollläden und	Preisvorteil ab netto Bestellwert: 2,0% 5.000 € 4,0% 10.000 € 5,0% 15.000 € 6,0% 20.000 € 7,0% 25.000 € 8,0% 30.000 € 10,0% 40.000 € 12,0% 50.000 €	Wichtige Eckdaten zu Ihrer Bestellung:  Fläche gesamt: 1,38 m² Laufmeter: 4,88 m Stückzahl Fenster gesamt: 1 Stückzahl Türen gesamt: 0 Rabatt: 0%	<b>MENGENRABATT (0%)</b> 0,00 € <b>Summe netto neu</b> 331,27 € <b>RABATTAKTION (-10%)</b> -33,13 € <b>Summe netto neu</b> 298,14 € Versandkosten 41,18 € <b>Gesamt</b>	<div>Experten kontaktieren</div>
Voraussichtliche Lieferung	Lieferkosten	Mengenrabatt	Technische Daten	SUMME NETTO	331,27 €												
Ihre Bestellung wird voraussichtlich zwischen <b>11.05.2020 - 15.05.2020</b> geliefert.	Ab einem Kaufpreis von 1.000 € liefern wir <b>versandkostenfrei</b> Unter 1.000 € berechnen wir: <b>49,00 €</b> für Fenster, Türen und Vordächer <b>16,90 €</b> für Rollläden und	Preisvorteil ab netto Bestellwert: 2,0% 5.000 € 4,0% 10.000 € 5,0% 15.000 € 6,0% 20.000 € 7,0% 25.000 € 8,0% 30.000 € 10,0% 40.000 € 12,0% 50.000 €	Wichtige Eckdaten zu Ihrer Bestellung:  Fläche gesamt: 1,38 m² Laufmeter: 4,88 m Stückzahl Fenster gesamt: 1 Stückzahl Türen gesamt: 0 Rabatt: 0%	<b>MENGENRABATT (0%)</b> 0,00 € <b>Summe netto neu</b> 331,27 € <b>RABATTAKTION (-10%)</b> -33,13 € <b>Summe netto neu</b> 298,14 € Versandkosten 41,18 € <b>Gesamt</b>	<div>Experten kontaktieren</div>												

Darstellung 10: Fensterversand.com - Warenkorb

#### 4.4.2.4. Nutzer-Login

Der Nutzer soll die Möglichkeit haben sich auf der Website zu registrieren und einzuloggen.

#### 4.4.2.5. Kaufprozess

Beim Kaufprozess soll der Kunde durch den Bezahlungsprozess begleitet werden, sowie zur Angabe nötiger **Daten** aufgefordert werden. Zu diesen Daten gehören die personenbezogenen Daten des Kunden (Name, Nachname etc.), die Versandadresse und die Rechnungsadresse. Sollte der Nutzer sich vorher registriert oder eingeloggt haben, dann sollen alle bereits angegebenen Daten im Voraus ausgefüllt werden. Zur **Bezahlung** sollen möglichst viele Methoden angeboten werden, zumindest aber PayPal, Sepa-Lastschriftverfahren und Kreditkarte.

#### 4.4.2.6. Kaufbestätigung

Nach dem Kauf sollen sowohl der Käufer als auch der Verkäufer per E-Mail über den Status der Bestellung benachrichtigt werden.

#### 4.4.2.7. Bestellübersicht

Eine Übersicht der gekauften Produkte inkl. ihrer speziellen Konfiguration soll für den Verkäufer gegeben sein. Insbesondere soll auch ersichtlich sein, wer die Bestellung aufgegeben hat, wie der Status der Bestellung ist und wohin geliefert werden soll. Auch weitere, notwendige Daten sollen vorhanden sein.

#### 4.4.2.8. Rechtliche Bausteine

Der Online-Shop soll über die rechtlichen Bausteine **DSGVO**, **AGB**, **Impressum** und **Widerrufsrecht** verfügen und der Nutzer soll die Möglichkeit haben, diese einzusehen.

## 5. Entwurf

### 5.1. Plattformanalyse

Es stehen theoretisch drei Realisierungsmöglichkeiten für einen Online-Shop offen: Wix mit dem Online-Shop-Plugin, Wix ohne des Plugins (Selbstimplementierung) und WordPress. Im folgendem wird überprüft, ob sich ein Online-Shop unter den gegebenen Anforderungen mithilfe einer der zwei Wix-Methoden effektiv umsetzen lässt. Insbesondere sind die technischen Anforderungen zu beachten bzgl. des Produktangebots. Es müssen sich alle Produkte aufnehmen lassen. D. h., das konfigurierte Produkt muss sich entweder komplett dynamisch zur Laufzeit erstellen lassen, oder anhand der Konfigurationsdaten muss sich das korrekte Produkt aus der Datenbank aller Produkte auswählen lassen, inkl. der korrekten Varianten. Die beste Wix-Methode wird dann gegenüber der WordPress-Methode gestellt.

#### 5.1.1. Wix mit dem Online-Shop-Plugin

##### 5.1.1.1. Herausforderungen

Um ein Produkt dem Warenkorb hinzufügen zu können, muss sich dieses in der Produkt-Kollektion zum gegebenen Zeitpunkt befinden (Vgl. Kapitel 2.4.1.1). Diese Tatsache bedeutet folgendes:

Damit sich die korrekten Produkte dem Warenkorb hinzufügen lassen können, müssen sich entweder alle Produkte in der Produkt-Kollektion korrekt abbilden lassen, oder die konkreten Produkte müssen dynamisch zur Laufzeit über die Funktion „createProduct“ erstellt werden. Beim letzteren ist vor allem die Performance zu beachten.

Auch muss beim letzteren die Frage geklärt werden, wie und wann die erstellten Produkte aus der Datenbank gelöscht werden, da die Datenbank auf Dauer sonst zu voll wird.

Außerdem müssen die gekauften Produkte und deren spezielle Konfiguration gespeichert werden, sodass der Verkäufer erkennen kann, welches Produkt anzufertigen ist.

Des Weiteren muss eine Methode gefunden werden, mithilfe der sich die konfigurierten Produkte übersichtlich und verständlich im Warenkorb anzeigen lassen. Im Warenkorb werden nämlich nur die ausgewählten Optionen und ein optionales Textfeld angezeigt (Vgl. Kapitel 2.3.1.1.2).

Schließlich muss das Problem betrachtet werden, dass der Nutzer in dem im Plugin enthaltenem Warenkorb die Möglichkeit hat, ein Produkt anzuklicken und dieses so zu betrachten und zu bearbeiten, als wäre es ein „korrektes Standardprodukt“.

##### 5.1.1.2. Umsetzungsmöglichkeiten

**Die erste** im vorherigen Kapitel genannte **Methode**, bei der alle Produkte im Voraus in die Datenbank geladen werden und statisch zur Verfügung stehen, ist unter Verwendung des Wix-Online-Shop-Plugins nicht möglich. Grund dafür ist die zu große Anzahl an Variationen für ein Produkt. Die Definition eines Produkts, bei der es möglichst viele Produkte und möglichst wenige Varianten gibt, sieht wie folgt aus: Jeder Eintrag aus einer Fensterpreistabelle – sprich eine Fensterausführung mit vorgegebener Breite und Höhe



– gilt als ein Produkt. Bei dieser Definition gäbe es ca. 20 Produkte \* 20 Breitevarianten \* 20 Höhevarianten = 8000 Produkte. Zudem könnte jedes dieser Produkte mindestens 612 und sogar viele mehr Varianten besitzen. Die Anzahl der Produktvarianten bei Wix ist jedoch auf 300 beschränkt. Zudem würde auch höchstwahrscheinlich das Limit für die max. Anzahl an Produkten überschritten werden, wenn ein weiteres Profil dazukommen würde. Damit entfällt diese Methode.

**Die zweite Methode** ist die zur Laufzeit dynamische Erstellung des konfigurierten Produktes. Diese Methode besitzt die Hürde, dass die Products-Kollektion nur 50.000 Produkte halten kann. Da sich die Produkte aus der Kollektion aber über den Aufruf der Funktion `deleteProduct()` (Vgl. Kapitel 2.3.1.1.1) löschen lassen und diese von beliebigen Besuchern aufgerufen werden kann, wird es hierbei mindestens eine akzeptable Lösung geben, die das Problem beseitigt. Des Weiteren müssen die gekauften Produkte inkl. ihrer Details gespeichert werden. Auch hierbei wird es keine Probleme geben, da sich die Konfiguration des Produktes ggf. in einer anderen Kollektion speichern lässt als der Kollektion Products. Andere Kollektionen besitzen keine Einschränkungen bzgl. des Umfangs der Daten. Das größte Problem stellt die Tatsache dar, dass die sich im Warenkorb befindlichen Produkte als „korrekte Standardprodukte“ betrachtet werden und sich entsprechend anklicken und bearbeiten lassen. Der Nutzer müsste an dieser Stelle aber zum üblichen Konfigurator zurückkehren. Da es dafür keine denkbare Lösung gibt, muss der Warenkorb selbstständig implementiert werden.

#### 5.1.1.3. Fazit

Eine Erstellung eines Online-Shops mithilfe des Online-Shop-Plugins ist nur über sehr große Workarounds möglich. Insbesondere die Tatsache, dass der Warenkorb selbstständig implementiert werden muss führt dazu, dass von der Nutzung des Plugins kaum Gebrauch gemacht werden kann. Der Vorteil bei der Verwendung des Plugins ist, dass eine erfolgreiche Bestellung automatisch in einer dafür vorgesehenen Kollektion bei Wix erscheint. Diese Funktionalität lässt sich jedoch möglicherweise ohne einen großen Aufwand selbstständig implementieren. Der Aufwand, der für die Workarounds aufgebracht werden muss, gleicht die Limitierungen des Plugins nicht aus und bringt insgesamt nur mehr Einschränkungen mit sich. Eine Umsetzung des Online-Shops mithilfe des Plugins ist insofern möglich, jedoch nicht effizient, nicht performant und womöglich auch sehr Fehleranfällig.

#### 5.1.2. Wix ohne des Online-Shop-Plugins

Um Zahlungen bei Wix annehmen zu können, ist ein Business-Paket notwendig. In jedem Business-Paket ist das Online-Shop-Plugin standardmäßig vorhanden. Insofern wird kein Geld gespart, wenn auf das Plugin verzichtet wird. Nach der Analyse der Umsetzungsmöglichkeiten eines Online-Shops mit dem Online-Shop-Plugin von Wix (Vgl. Kapitel 5.1.1) hat es sich jedoch ergeben, dass ein sehr großer Teil der Funktionalitäten ohnehin selbstständig implementiert werden muss aufgrund hoher Limitierungen.

##### 5.1.2.1. Herausforderungen

Die Herausforderungen ergeben sich unmittelbar aus den Anforderungen (Vgl. Kapitel 4.4.2). Diese sind die Aufnahme und Verwaltung der Produkte, die Erstellung eines

Konfigurator, die Erstellung eines Warenkorbs, die Ermöglichung eines Kaufvorganges, das Senden einer Bestellbestätigung und die Speicherung einer Bestellung.

#### 5.1.2.2. Umsetzungsmöglichkeiten

Die Produkte lassen sich grundsätzlich bei einer korrekten Strukturierung der Datenbanken inkl. aller beliebiger Optionen und Varianten aufnehmen, da die Kollektionen von Wix keine Einschränkungen bzgl. der Datenmenge besitzen. Ihre Aktualisierung kann ebenfalls einfach sein, wenn die Struktur der Datenbanken vernünftig ausgelegt wird.

Ein Konfigurator ist umsetzbar, da bei Wix reines JS verwendet werden kann und weil Wix viele grafische Elemente inkl. vorgefertigter Listener-Funktionen zur Verfügung stellt.

Der Warenkorb ist bereits vor langer Zeit realisiert worden. Damit ist die Möglichkeit der Erstellung eines Warenkorbs bewiesen worden. In der Zukunft müssten die Sicherheit sowie die Performance des Warenkorbs durchdacht werden.

Der Kaufvorgang wird über die Bibliothek Wix-Pay realisiert. Es lassen sich beliebige Zahlungsaufträge definieren.

Eine Bestellbestätigung wird automatisch nach jeder Zahlung an die Käufer und Verkäufer gesendet. Sollte diese Bestätigung nicht die nötigen Details zu den gekauften Produkten beinhalten, so kann über die Wix-Bibliotheken eine angepasste E-Mail gesendet werden, die diese Informationen beinhaltet. Diese kann beispielsweise anhand des Zahlungsstatus, der von `startPayment()` zurückgegeben wird, getriggert werden.

Die Rechnungsdaten und die bestellten Produkte sind unter Transaktionen im Dashboard nachvollziehbar. Da nur die Namen der verkauften Produkte ersichtlich sind, müssen diese in einer anderen Kollektion nachträglich zu finden sein. Dies lässt sich jedoch problemlos realisieren.

#### 5.1.2.3. Fazit

Die Umsetzung eines Online-Shops ohne des Online-Shop-Plugins ist möglich. Der dafür benötigte Aufwand ist nicht klein, jedoch ein großer Teil der Arbeit ist bereits getan worden.

#### 5.1.3. WordPress

Eine Umsetzung des Online-Shops mit WordPress ist die langsamste Lösung, da jegliches Vorwissen im Umgang mit PHP oder WordPress fehlt.

##### 5.1.3.1. Herausforderungen

Der Konfigurator lässt sich aufgrund des Mangels an Vorwissen über Session-Verwaltung und PHP nicht manuell programmieren. Das Risiko, das beim Versuch einer solchen Umsetzung auftreten würde, wäre zu groß. Deshalb müssen sich die Produkte inkl. aller Optionen über das WooCommerce-Plugin darstellen lassen können. Beispielsweise wäre ein Fenster ein einziges Produkt, das alle denkbaren Varianten beinhaltet. Das WooCommerce-Plugin besitzt keine Einschränkungen bzgl. der Menge der Daten. Die Produkte müssen sich jedoch weiterhin übersichtlich und vernünftig gestalten lassen. Des



Weiteren müssen User-Inputangaben, wie beispielsweise die Fenstermaßen, sich in gültige Optionen umwandeln lassen. Eine Alternative würde ein WooCommerce-Plugin darstellen, bei der ein Konfigurator professionell aufgesetzt werden kann. Sind diese Dinge gegeben, dann stehen der Umsetzung theoretisch keine weiteren Hürden bevor.

#### 5.1.3.2. Umsetzungsmöglichkeiten

WooCommerce verfügt über ein sehr flexibles System zur Erstellung mehrerer Produkte mit vielen Varianten. Die Produkte ließen sich somit theoretisch leicht verwalten, aktualisieren und importieren über einen CSV-Import (Vgl. Kapitel 2.3.3). Ein Fensterprodukt müsste jedoch, wie im vorherigen Kapitel beschrieben (Vgl. Kapitel 5.1.3.1), ein einziges Produkt sein, das alle möglichen Varianten beinhaltet. Eine Berechnung der Anzahl aller Varianten für ein Kunststofffenster nach dem aktuellen Stand (25.03.2020) würde wie folgt aussehen:

#Varianten = #Profile \* #Farben \* #Fenstertypen \* #Fensterlicht \* #Fensteröffnungsarten \* #Breiten \* #Höhen \* #Verglasungen \* #Eigenschaft1 \* #Eigenschaft2 \* ...

Eine Schätzung der Anzahl dieser Varianten würde in folgenden Zahlen resultieren:

#Varianten = 1 \* 40 \* 4 \* 3 \* 10 \* 20 \* 20 \* 30 \* 20 \* 20... = 23.040.000.000

Dies sind schlicht und einfach zu viele Varianten.

Eine valide, jedoch aufgrund nicht vorhandener Möglichkeiten nicht getestete Alternative ist das „Extra Product Options“-Plugin für WooCommerce, das verspricht, alle nötigen Funktionalitäten zur Berechnung des Preises zu bieten. Unklar bleibt jedoch, wie eine vom Nutzer eingegebene Fensterbreite/-höhe in eine valide Zahl umgerechnet werden sollte. Vermutlich müssten die Preise für alle möglichen Maßen sogar händisch eingetragen werden, was bei 20 Fensterprodukten mit je 400 Maßkombinationen eine Unmenge an Arbeit bedeuten würde.

#### 5.1.3.3. Fazit

Höchstwahrscheinlich wäre die Umsetzung eines Online-Shops mit WordPress mithilfe von WooCommerce und weiterer Plugins möglich, jedoch lässt sich der Aufwand dafür nur sehr schwer einschätzen.

#### 5.1.4. Plattformvergleich

Da der Online-Shop sich mithilfe des Online-Shop-Plugins von Wix nicht umsetzen lässt, wird die Umsetzungsmöglichkeit bei Wix ohne des Plugins mit der Umsetzungsmethode über WordPress verglichen. Im Folgendem ist ein tabellarischer Vergleich zu sehen. Jedes Kriterium besitzt eine Wichtigkeit von 1 bis 10. Jeder Umsetzungsvariante werden zu dem jeweiligen Kriterium Punkte vergeben.

Kriterium	Wichtigkeit (1-10)	Wix ohne Plugin	Punkte (1-10)	WordPress mit WooCommerce Plugin	Punkte (1-10)

Preis	5	Ca. 200€ pro Jahr + Domain	5	<ul style="list-style-type: none"> <li>- Ab ca. 100€ pro Jahr + Domain</li> <li>- Pauschal ca. 200€ fürs Plugin</li> </ul>	7
Implementierungsaufwand	6	mittel, da vieles selbstprogrammiert werden muss jedoch vieles bereits implementiert.	5	Nicht einschätzbar aufgrund fehlender Kenntnisse.	5
Pflegeaufwand des Shops	7	Schätzungsweise normal	5	Schätzungsweise normal	5
Aufwand für Produktaktualisierungen	9	Schätzungsweise sehr gering bei richtiger Implementierung	10	Kann nicht eingeschätzt werden, da die Methode zur Berechnung des Preises der Fenster basierend auf ihren Maßen unbekannt ist. Möglicherweise sehr hoher Aufwand, wenn der Preis für jede Maßkombination händisch eingetragen werden muss!	3
Performance	6	Schätzungsweise langsam aufgrund vieler DB-Zugriffe und Workarounds	3	Schätzungsweise hoch, da professionelles Plugin	7
Skalierbarkeit	7	Skaliert bei richtiger Implementierung gut	9	Skaliert schätzungsweise gut	10

Übertragbarkeit	3	Eher schlecht, da an Wix und ihre API gebunden. JS-Code jedoch übertragbar	6	Eher schlecht, an WordPress gebunden	5
Projektrisiko	8	Eher gering, da sich alle Funktionalitäten voraussichtlich umsetzen lassen	8	Hoch, da keine Erfahrung mit WordPress und nur eine ungetestete Umsetzungs-idee vorhanden	2
Adminfreundlichkeit	5	Eher schwach aufgrund der Workarounds und voraussichtlich eigener Kollektionen für Bestellungen und Produkte	3	Schätzungswiese in Ordnung, da professionelles Plugin	7
Sicherheit	7	Eher durchschnittlich, da bei einer Selbstimplementierung einige Fehler denkbar sind	6	Eher gut, da durch viele User verwendet	10
Zuverlässigkeit	7	Eher schwach, da bei einer Selbstimplementierung einige Fehler denkbar sind	6	Schätzungswiese zuverlässig, da durch viele User verwendet	10

#### 5.1.5. Fazit

Im Folgendem wird die Gesamtpunktzahl der beiden Umsetzungsvarianten aus dem Vorherigen Kapitel (Vgl. Kapitel 5.1.4) angegeben. Die Gesamtpunktzahl wird errechnet, indem zunächst für jede Umsetzungsvariante zu jedem Kriterium die Wichtigkeit mit der Anzahl der vergebenen Punkte multipliziert wird und anschließend die errechneten Punkte für jedes Kriterium einer Variante zusammen addiert werden:

	Wix ohne Plugin	WordPress mit WooCommerce-Plugin
<b>Punkte</b>	442	445

Viele der im vorherigen Kapitel tabellarisch dargestellten Kriterien im Vergleich der beiden Umsetzungsmöglichkeiten für den Online-Shop (Wix und WordPress) sprechen für WordPress. Jedoch die Tatsache, dass der Aufwand bei der WordPress-Variante nicht einschätzbar ist und zudem der Erfolg des Projekts bei dieser Variante aufgrund des fehlenden Vorwissens in WordPress und PHP gefährdet wäre, bringen der Wix-Variante viele Punkte. Damit gewinnt WordPress sehr knapp mit 4 Punkten. Bei einem so geringen Unterschied fällt die Entscheidung nach den persönlichen Vorlieben. **Der Online-Shop wird in Wix realisiert werden.**

## 5.2. Konzeptioneller Entwurf

### 5.2.1. Ablauf des Einkaufs im Online-Shop

Im folgendem wird der Ablauf eines Einkaufs im Online-Shop anhand eines Aktivitätsdiagramms erläutert [8].

Das Diagramm zeigt vier Aktivitätsframes auf: **product section, shopping cart, payment, user account.**

**Product section** kapselt den Prozess der Produktauswahl ab. Der Nutzer hat grundlegend die Wahl zwischen einem simplen und einem komplexen Produkt. Ein simples Produkt wird ausgewählt und ggf. konfiguriert, ein komplexes Produkt wird dynamisch erstellt. In beiden Fällen wird das gegebene Produkt schließlich dem Warenkorb hinzugefügt (cart). Nach dem Hinzufügen eines Produkts zum Warenkorb wird der Nutzer zum Warenkorb weitergeleitet.

**Shopping cart** kapselt alle Aktivitäten ab, die im Warenkorb (cart) getätigt werden können. Der Nutzer kann sich den Warenkorb jederzeit anschauen. Im Warenkorb hat er die Möglichkeit ein vorher hinzugefügtes Produkt, falls es konfiguriert werden kann, zu bearbeiten oder zu duplizieren und das erstellte Duplikat anzupassen. In beiden Fällen wird das gegebene Produkt zur Bearbeitung vorgemerkt, woraufhin der Konfigurationsbereich geöffnet wird. Der Nutzer hat außerdem die Möglichkeit ein beliebiges Produkt zu löschen oder seine Anzahl zu ändern. Des Weiteren kann der Nutzer seinen Einkauf fortsetzen, wodurch er zur Produktauswahl weitergeleitet wird, oder zur Bezahlung (payment) fortfahren.

**User account** kapselt die Aktivitäten ab, die mit dem Ein- und Ausloggen sowie mit dem Registrieren eines Nutzers zu tun haben. Diese werden grundsätzlich über ein Popup anhand der Wix-API umgesetzt. Ein registrierter Nutzer kann sich einloggen. Ein nicht registrierter Nutzer kann sich registrieren. Ein eingeloggter Nutzer kann sich ausloggen. Nach dem Ein- oder Ausloggen wird das Popup geschlossen und der Nutzer kann seine vorherige Tätigkeit fortsetzen.

**Payment** kapselt den Bezahlungsprozess ab. Hat sich der Nutzer vorher eingeloggt, dann werden seine Daten zur Kaufabwicklung im Voraus ausgefüllt, sonst werden sie während des Kaufabwicklungsprozesses vom Nutzer eingegeben. Der Nutzer kann vor der endgültigen Bezahlung seine Bestellung bearbeiten, wodurch er zurück zur Warenkorbansicht weitergeleitet wird, oder den Einkauf fortsetzen, wodurch er zur Produktauswahl weitergeleitet wird. Außerdem kann er den Bezahlungsprozess

fortsetzen und bezahlen. Bei einer erfolgreichen Zahlung werden die Bestelldaten in der Datenbank aufgenommen und die Thank-You-Page wird gezeigt. Restliche Prozesse werden automatisch über Wix durchgeführt, beispielsweise das Versenden einer Bestätigungsemail.

### 5.2.2. Produkte

**Im Folgendem** wird der Aufbau eines Produktes sowie die Erstellung, Konfiguration, Auswahl und Verwaltung von Produkten besprochen. Anschließend wird die Art der Speicherung von Produkten diskutiert.

#### 5.2.2.1. Produktaufbau

Die meisten Produkte besitzen **Optionen**. Optionen besitzen **Varianten**. Jede Produktvariante kann als ein separates Produkt angesehen werden, da sie eine Ausführung des Produktes darstellt und ggf. einen anderen Preis besitzt. Bestimmte Varianten lassen sich nur auswählen, wenn diverse andere Optionen oder Varianten vorher ausgewählt worden sind. Einige andere Optionen ergeben sich durch variable **Nutzereingaben**. Diese Optionen werden in Form eines variablen Textes abgebildet, dynamisch erstellt und an die Bestellung und das gegebene Produkt gebunden.

Grundsätzlich wird zwischen zwei **Produkttypen** unterschieden: Simple- und Komplexes Produkt. **Simple Produkte** besitzen eine *übersichtliche Anzahl* an Varianten und Kombinationen. Jede Variante ist textueller Form. Für jede Kombination wird der Preis manuell festgelegt. **Komplexe Produkte** sind sehr umfangreich in ihren Kombinationsmöglichkeiten. Ihre Varianten besitzen mehrere Informationen in verschiedenen Formaten. Diese Produkte werden dynamisch vom Nutzer durch einen vorher geschriebenen Algorithmus erstellt.

Nach dem aktuellen Stand (23.04.2020) wird zw. Zwei **Produktmodellen** unterschieden: Fenstergriff und Fenster. Produkte zweier verschiedener Typen unterscheiden sich in der Regel jeweils in ihren Attributen und den Methoden zur Preisberechnung.

#### 5.2.2.2. Produkterstellung

**Simple Produkte** und ihre Variationen werden im Voraus manuell erstellt, da der Umfang ihrer Kombinationen übersichtlich ist.

**Komplexe Produkte** werden dynamisch erstellt, da der Umfang ihrer Varianten und Kombinationen zu groß ist. Komplexe Produkte und ihre einzigartigen Kombinationen werden nach der dynamischen Erstellung zu allen Produktkombinationen hinzugefügt und zur Auswahl bereitgestellt.

Die **Strategieunterscheidung** bei der Erstellung von Produkten zwischen diesen zwei Produktmodellen ist sinnvoll, da dadurch vermieden wird, dass entweder sehr viele Varianten eines komplexen Produktes per Hand erstellt werden müssen, oder dass die Applikation zur Erstellung von Produkten komplex und umfangreich wird, damit sich beliebige Produkterstellungsregeln erstellen lassen. Stattdessen kann für komplexe

Produkte ein angepasster, einziger Algorithmus geschrieben werden, der die Produkte entsprechend erstellt.

#### 5.2.2.3. Bearbeitung/Konfiguration von Produkten

**Alle simplen Produkte** werden auf dieselbe Art bearbeitet, da sie vieles gemeinsam haben. Eine Bearbeitung simpler Produkte bedeutet lediglich die Auswahl einer anderen Produktausführung. Beim Bearbeiten wird also die passende Ausführung durch ein kontinuierliches Einschränken der Anzahl der möglichen Ausführungen, basierend auf den ausgewählten Optionen, ausgewählt. Die alte Ausführung wird rückgängig ausgewählt und wird durch die neue Ausführung ersetzt. Die **Konfigurationsmöglichkeiten** ergeben sich hier durch einfaches Filtern. Bei simplen Produkten steckt keine höhere Konfigurationslogik hinter.

**Alle komplexen Produkte** werden individuell bearbeitet. Komplexe Produkte werden neu erstellt, basierend auf den ausgewählten Produktoptionsvarianten der zu bearbeitenden Produktausführung. Das konfigurierte Produkt wird als eine neue Produktausführung gespeichert und ersetzt die alte Produktausführung in der Liste der durch den Nutzer zum Warenkorb hinzugefügten Items. Die **Konfigurationsmöglichkeiten** werden hier individuell nach einem Tag-System ermittelt.

Auf einer **grafischen Oberfläche** werden dem Nutzer alle Konfigurationsmöglichkeiten zu dem aktuell konfigurierenden Produkt angezeigt. Durch das Anklicken verschiedener Eigenschaften kann der Nutzer diese für das aktuelle Konfigurationsprodukt auswählen. Ggf. führt dies zur Änderung der zur Verfügung stehenden Eigenschaften.

Damit eine fehlerfreie Konfiguration möglich ist, müssen auf der Konfigurationsseite folgende **Funktionalitäten und Daten** zugriffsbereit sein:

- **Darstellungsdaten** wie Bilder, Elementbezeichnungen etc. – Die Darstellungsdaten lassen sich über zwei Wege an die grafischen Elemente anbinden: Über Datasets oder über direkte Datenübergabe an die Repeater.
- **Kombinationsmöglichkeiten** einzelner Eigenschaften – Die Kombinationsmöglichkeiten müssen manchmal über Tabellenelemente aus höhergradigen Referenzen ermittelt werden.
- **Ids der ausgewählten Konfigurationselemente** – Zwecks **Identifikation** der Konfigurationselemente muss sich die zu dem angeklickten Element zugehörige Id aus der Kollektion ermitteln lassen.
- Das **resultierende Fensterprodukt** und sein **Preis** – Zur präsentativen Darstellung des konfigurierten Produktes werden grundsätzlich Daten aus den Selektionselementen direkt bezogen. Die Preisberechnung geschieht anhand eines Fensterproduktobjektes, das ständig konfiguriert wird.
- Eine **eindeutige Id des Produktes** – jedes Produkt, auch ein dynamisch erstelltes Produkt, muss sich identifizieren lassen, damit es von dieser und anderen Domänen vernünftig **weiterverarbeitet** werden kann.

#### 5.2.2.4. Datenspeicherung

Alle zu einem **Produkt** hinzugehörigen **Daten** werden in der Datenbank dauerhaft gespeichert, da sie für alle Nutzer zugänglich sein müssen und sich vom Admin ggf. verwalten lassen müssen.

**Fertige Produkte** werden ebenfalls in der Datenbank gespeichert, da sie für alle Nutzer sichtbar sein müssen. Dies bezieht sich nicht zwangsläufig auf komplexe Produkte. Um die Vorgänge simpel zu halten, wird hierbei jedoch zunächst nicht unterschieden und simple und komplexe Produkte werden auf dieselbe Art und Weise gespeichert. Beim Design wird darauf geachtet, dass eine mögliche Änderung dieser Gegebenheiten in naher Zukunft möglich ist.

#### 5.2.3. Warenkorb

Die dynamisch erstellten Produkte müssen sich unter anderem, laut 4.4.3.2, dem Warenkorb **hinzufügen**, dort **darstellen** und aus dem Warenkorb heraus **bearbeiten** und **duplizieren** lassen. Schließlich müssen sie sich **kaufen** lassen. Dies bedeutet für die Produkte insb. Folgendes:

- **Die genaue Konfiguration des Produktes** muss nach dem Hinzufügen eines Produktes zum Warenkorb gespeichert werden, damit ein nachträgliches Bearbeiten der Produkte möglich ist.
- Ein sich im Warenkorb befindendes Produkt kann bearbeitet werden, wenn es über **Optionen** verfügt.

Eine **dauerhafte Speicherung des Warenkorbs** ist nicht notwendig. Aus diesem Grund wird der Warenkorb in der Session-/Local-Storage gespeichert, damit die mit der Speicherung der Daten in einer Datenbank verbundenen Probleme umgangen werden (vgl. Kapitel 2.4).

#### 5.2.4. Nutzerverwaltung

Die Nutzer werden fast vollständig über die **Wix-Users-API** verwaltet, da diese die nötigen Funktionalitäten bereits bietet.

#### 5.2.5. Bezahlung

Die Bezahlung wird fast vollständig unter Verwendung des **Wix-Payment-Modules** abgewickelt, da diese die nötigen Funktionalitäten besitzt, sicher ist und vermutlich die einzige Zahlungsmöglichkeit bei Wix darstellt. Aus der Nutzerdatenbank werden die zu vorher ausfüllenden Daten bezogen.

#### 5.2.6. Bestellaufnahme

Eine erfolgreiche Bestellung muss in der Datenbank hinterlegt werden, sodass sie für den Admin ersichtlich ist. Spätestens zum Zeitpunkt des Verkaufs müssen die gekauften Produkte auf eine Art in der Datenbank abgelegt werden, sodass klar ist, welche Produkte gekauft worden sind. Eine Bestellung wird somit komplett in der Datenbank abgespeichert.

#### 5.2.7. Speicher- und Datenverwaltung

Üblicherweise würde man ein Object-Related-Mapping-Modul benutzen. Doch dafür braucht man vernünftige Dbs.

**Im Folgendem** werden die Strukturen der einzelnen Speichermechanismen für diese Software erläutert.

##### 5.2.7.1. Datenbankstruktur

Die Tabellen der Datenbanken und ihre Zusammenhänge werden anhand eines ER-Diagramms [2] dargestellt. Im folgendem werden die im Diagramm vorkommenden Entitäten erklärt. Entitäten, die echte Fenstereigenschaften repräsentieren, werden zusätzlich unterstrichen.

##### 5.2.7.2. Session-/Local-Storage-Struktur

#### 5.3. Technischer Entwurf

##### 5.3.1. Schichten-Übersicht

##### 5.3.2. Packages-Übersicht



## 6. Implementierung

## 7. Test

## 8. Fazit

## 9. Old TO REFACTOR

### 9.1. Generelle Funktionsweise

1. Der Konfigurator wird initialisiert.
  - a. Ein Defaultprodukt wird initialisiert.
  - b. Wurde der Konfigurator nicht initialisiert und es wird versucht eine fortgeschrittene Seite aufzurufen, so wird man auf die Produktauswahl zurückgeleitet.
2. Man wird auf die Auswahlseite für die erste Komponente des Produkts weitergeleitet.
3. Die zur Verfügung stehende Auswahl wird nach Kompatibilitätskriterien gefiltert.
4. Eine Komponente wird ausgewählt.

### 9.2. Vererbung

Alle eigenen Klassen erben von AbstractObject.

AbstractObject bietet die Funktionalität der Vererbung und weiteres.

Eine Klasse „inherits“ alle Elternteile und Interfaces und alle Elternteile von Interfaces.

Ein Interface kann keine Interfaces implementieren. Es kann aber ein anderes Interface erweitern.

Beispiel:

Class A extends AbstractObject

```
{  
    methodA(){}  
}
```

Class B extends class A()

```
{  
    Constructor()  
    {  
        ....  
        pushInheritance("PrintableInterface", "B");  
    }  
    methodB(){}  
}
```

Class ShitableInterface

```
{  
    Shit(){}  
}
```

Class PrintableInterface extends ShitableInterface

```
{  
    Print(){}  
}
```

b = new B();

b besitzt die folgenden Methoden: (ACHTUNG: Methoden aus Interfaces müssen über call aufgerufen werden)

b.methodA

b.methodB

b.print

b.shit

### Einfache Vererbung

Eine Klasse erbt von der Elternklasse (extends), von allen Eltern, Großeltern, GroßGroßEltern etc. der Elternklasse alle Methoden, statischen Methoden und Felder.

Methoden die auf einfache Weise vererbt werden können direkt aufgerufen werden über das Objekt und müssen nicht über call gerufen werden, können aber.

### Abstrakte Klassen

Von abstrakten Klassen sollten keine Instanzen erzeugt werden. Deshalb gibt es auch keine Abstrakte Klasse in factory.

### Interfaces

Eine Klasse erbt von Interfaces, die sie selbst besitzt oder eines der Elternteile.

Vererbt werden nur Methoden.

Diese Methoden müssen über call() gerufen werden.

### Statische Methoden

Statische Methoden werden Analog vererbt, müssen aber über call gerufen werden.

### Abstrakte Methoden

Abstrakte Methoden werden auch normal vererbt, müssen aber ausimplementiert werden von Kindern.

Eine Abstrakte Methode sollte immer `MissingImplementationError` schmeißen.

### 9.3. Komponenten

#### `typing.js`

Besitzt alle Bibliotheken nötig für die Vererbung sowie alle Klassen.

Factory: Funktion, über die alle Klassen erzeugt werden sollen. (Interfaces haben eine getrennte Factory.) Alle neuen Klassen sollten hier vermerkt werden.

Inherits: checkt die gesamte Vererbungskette ab nach dem oben beschriebenen Schema.

`classOf`: gibt den Namen der Klasse der Instanz wieder.

`Typesize`: Typisiert das gesamte Objekt (deep). Rekursiv.

#### `storageManager.js`

Verwaltet die storage.

Nur `StorageItem` kann zu der Storage hinzugefügt werden. `StorageItem` ist ein Wrapper für das eigentliche Objekt. Zurückgegeben wird das typisierte Objekt, damit kann ohne weiteres wie gewohnt weitergearbeitet werden.

Interne Struktur:

- Eine interne Indextabelle wird verwaltet:
- `ACT_MAX_INDEX`: [aktuelle Anzahl an items, length]
- `ITEM0`: [id0]
- `ITEM1`: [id1]
- ...
- [Id0]: Objekt...
- [Id1]: Objekt...

Hinzufügen von neuen Objekten:

Immer ans Ende der Kette.

Finden von Objekten:

Nach der ID oder Tags.

Löschen von Objekten:

Löschen der beiden, zugehörigen Einträge zu dem item.

Wenn es das letzte Item in der Liste war: fertig. Sonst:

Verschiebe das letzte Item an die neue, leere Stelle.

`debug.js`

Bietet nützliche Debugfunktionalitäten. Verwaltet die Printausgaben.

#### 9.4. Fehlerbehandlung

Fehler werden durch throws von Errors signalisiert.

Checked Errors werden behandelt, UncheckedErrors nach oben geschmissen, um den Programmfluss zu unterbrechen.

Checked errors gelten als Ausnahmen, die man möglicherweise beheben kann.

Unchecked errors gelten als Fehler, die sich nicht beheben lassen.

Generelle Richtlinien bei Fehlerbehandlung

Ein Fehler trifft auf, wenn ein unerwartetes Verhalten auftritt. Ein gutes Beispiel:

Die Methode `getStorageItem(id)` gibt das StorageItem mit der gegebenen ID aus der Storage wieder.

Im Normalfall wird das zugehörige StorageItem zurückgegeben.

Existiert kein Item mit der zugehörigen ID, wird null zurückgegeben (oder ein anderer Standardwert, der das nicht-vorhandensein des Items symbolisiert). Ein vermeindlicher Versuch ein nicht-existierendes Item abzufragen ist kein Fehlerfall. Es ist einer der üblichen Zustände von `getItem`.

Wenn als ID-Parameter keine ID sondern ein Objekt eines falschen Datentyps übergeben wird, dann wird ein Fehler zurückgegeben und `throw Error` wird durchgeführt.

Note: Die `getItem` Funktion könnte auch einfach den Fehler vermeiden und auch in diesem Fall null zurückgeben. Im Normalfall wird jedoch nie ein Parameter eines falschen Datentyps zurückgegeben. Dieser Zustand weist also auf einen klaren Fehler hin und schmeißt deshalb einen Error.

Halterichtlinien für `throw`

Beim `throw` wird immer ein am besten passender Error gewählt.

Halterichtlinien für `catch`

Gecatcht werden nur errors, die `CheckedError` erweitern.

Gecatcht wird an der Stelle, wo ein möglicher Fehler gut behandelt werden kann.

## Besonderheiten

### Tests

Automatisch getestet werden alle Methoden/Funktionen und Klassen außer:

- Funktionen/Methoden und Klassen, die nicht extern sind.
- Funktionen/Methoden und Klassen, die nicht automatisch getestet werden können, bzw. nur optisch getestet werden können
- Funktionen/Methoden und Klassen, die besonders schwer automatisch zu testen sind und besonders leicht manuell getestet werden können. Diese sollten als @untested markiert werden.
- Getter und setter
- Konstruktoren

## 10. Verweise

- [1] <https://support.wix.com/de/article/anzahl-der-produkte-die-in-wix-stores-hinzugef%C3%BCgt-werden-kann>, Produktlimits bei Wix mit dem Online-Shop-Plugin. Zuletzt zugegriffen am 25.03.2020
- [2] Diagramme/ER-Diagramm/, Pfad zum ER-Diagramm.
- [3] <https://www.youtube.com/watch?v=XlhqvEsrupo>, CSV-Import und Export bei WooCommerce erklärt. Zuletzt zugegriffen am 25.03.2020
- [4] <https://codecanyon.net/item/woocommerce-extra-product-options/7908619>, Extra-Product-Options-Plugin. Zuletzt zugegriffen am 25.03.2020
- [5] <https://shop.trustedshops.com/dashboard/legaltexts>, Rechtstexte für die Website. Zuletzt zugegriffen am 31.03.2020
- [6] <https://www.wix.com/corvid/reference/wix-pay.html#PaymentResult>, Corvid-API, PaymentResult. Zuletzt zugegriffen am 06.04.2020
- [7] <https://www.wix.com/corvid/reference/wix-pay-backend.Events.html#onPaymentUpdate>, Corvid-API, Event-Handler für Änderungen des Payment-Status. Zuletzt zugegriffen am 06.04.2020.
- [8] Diagramme und Modelle/Aktivitätsdiagramm/, Pfad zum Aktivitätsdiagramm
- [9] Diagramme und Modelle/Schichtenmodell/, Pfad zum Schichtenmodell
- [10] Diagramme und Modelle/Paketdiagramm/, Pfad zum Paketdiagramm
- [11] <https://www.wix.com/corvid/reference/wix-dataset.Dataset.html>, Corvid-API, Dataset. Zuletzt zugegriffen am 13.03.2020.
- [12] [https://www.wix.com/corvid/reference/\\$w.html#\\$w](https://www.wix.com/corvid/reference/$w.html#$w), Corvid-API, Selectors. Zuletzt zugegriffen am 13.04.2020.
- [13] [https://www.wix.com/corvid/reference/\\$w.Repeater.html](https://www.wix.com/corvid/reference/$w.Repeater.html), Corvid-API. Wix-Repeater. Zuletzt zugegriffen am 13.04.2020.
- [14] [https://www.wix.com/corvid/reference/\\$w.Container.html](https://www.wix.com/corvid/reference/$w.Container.html), Corvid-API, Wix-Container. Zuletzt zugegriffen am 13.03.2020.
- [15] <https://docs.microsoft.com/de-de/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/ddd-oriented-microservice>, Domain-Driven-Design. Zuletzt zugegriffen am 23.04.2020.