

DOKUMENTATION PROGRAMMIERPROJEKT

Valerie Morviller, Tim Nöth, Franziska Goll, Jakob Priesner, Alejandro Stolz Mateos, Levin Hepp

Unsicherer Webshop für die Lehre

Inhaltsverzeichnis

1. Über dieses Projekt	3
2. Arbeitssprache, Arbeitsweise, Organisation	3
3. Start des Systems	3
FRONTEND - WEBSHOP	4
BASIS - FRONTEND	4
4. Struktur.....	4
5. Performance.....	5
6. Datenverwaltung	6
7. Verwaltung / Erkennung der Clients	6
KOMPONENTEN	8
8. Frontpage.....	8
9. Navbar	9
10. Navigation.....	10
11. Footer	10
12. Login Prozess	11
13. Newsletter.....	11
14. Kontaktformular	12
15. Informationsseiten der Schwachstellen	13
16. Artikelübersicht	14
17. Nutzereinstellungen	15
18. Artikelansicht	16
19. Bestellprozess	17
20. Bestellübersicht	18
ADMIN PAGE	18

21. Admin Page mit Ranking.....	18
BACKEND - WEBSHOP	20
22. Logik – Api.....	20
23. Aufbau mit dem Builder Pattern	25
24. Datenschicht.....	26
SHOP DATENBANK	27
25. ER-Modell:.....	27
26. Relationenmodell:	27
27. Beschreibung.....	28
ADMIN DATENBANK	29
28. ER-Modell:.....	29
29. Relationenmodell:	29
30. Beschreibung.....	29
SCHWACHSTELLEN - Webshop	31
31. Erkennung der Schwachstellen.....	31
32. Überprüfung der Schwachstellen	31
BEKANNTE FEHLER	34
33. Probleme mit Uuid nach Serverneustart.....	34
34. XSS wird zwei Mal in den Nutzereinstellungen angezeigt	34
35. Datenbanken werden beschädigt wenn der Server während des Resets heruntergefahren wird	34

1. Über dieses Projekt

Die Vertiefung „Information Security“ beschäftigt sich mit den Abläufen von Angriffsszenarien und wie man sie verhindern kann. Für diese Vertiefung entstand das Projekt „Unsicherer Webshop für die Lehre“. Das Ziel des Projektes war einen Webshop zu entwerfen, welcher Schwachstellen enthält.

Die Studierenden der Vertiefung sollen die enthaltenen Schwachstellen finden und werden dafür mit Punkten honoriert. Je nach Schwierigkeitsgrad der Schwachstelle, gibt es unterschiedlich viele Punkte. Der Admin kann auf einer separaten Seite eine Rankingliste der Studierenden sehen, in der die Studierenden nach ihren Punkten geordnet sind.

So findet der pädagogische Ansatz der Gamification in der Lehre Einzug in die Vertiefung.

2. Arbeitssprache, Arbeitsweise, Organisation

Die Anwendung besteht aus einem Backend und einem Frontend. Das Backend wurde in Java (Version X) geschrieben und nutzt Eclipse Jersey (ein REST-Framework, welches JAX-RS implementiert).

Das Frontend wurde mit dem Web Framework Angularjs realisiert. Durch die Verwendung von Bootstrap, können verschiedene Bildschirmgrößen unterstützt werden.

Die Anwendung unterstützt nur die deutsche Sprache, weshalb die Dokumentation auch in deutscher Sprache verfasst ist.

Zur Projekt internen Kommunikation wurde Discord verwendet. Über Discord fanden auch unsere Weeklys statt. Wir gingen unsere Ergebnisse aus dem letzten Sprint durch und organisierten unseren nächsten Sprint.

Unsere wöchentlichen Meetings, die Treffen mit Herrn Biedermann über Zoom und viele weitere Ergebnisse wurden in Confluence dokumentiert.

Das Konfigurationsmanagement Tool Bitbucket wurde zur Codeverfolgung verwendet.

Ebenfalls benutzen wir Trello als Taskboard zur Übersicht unserer ToDo's.

3. Start des Systems

Das System kann per Maven als war File exportiert und anschließend über das Terminal gestartet werden. Für das Frontend muss nodejs installiert sein. Anschließend muss in den Unterordner `/src/frontend/frontend` navigiert und anschließend mit `npm install` alle dependencies installiert werden. Nun muss die IP-Adresse des Systems, auf welchem der Webserver laufen soll in die `statics.ts` in dem Unterordner `/src/frontend/frontend/src/lib/data-access/service/` angepasst werden. Die IP des Backends muss in der `proxy.conf.json` in dem package `/src/frontend/frontend/src/lib` abgeändert werden. Danach kann man mit `ng serve – host <ip-Adresse>` das Frontend starten.

Eine Version über DockerCompose ist in Arbeit, konnte jedoch aus unerwarteten Problemen nicht fertig gestellt werden. Diese wird jedoch noch fertig gestellt.

FRONTEND - WEBSHOP

Das Frontend des Shops ist die Kundensicht. Innerhalb dieser Kundensicht, die im Verlauf der Dokumentation immer Frontend genannt wird, kann der Kunde Produkte suchen, zu seiner Wunschliste hinzufügen, in den Warenkorb legen und schlussendlich auch bestellen.

Um diese Funktionen auch nutzen zu können, muss der Kunde sich erst einmal registrieren. Nachdem sich er sich als Kunde registriert hat, kann er Benutzereinstellungen vornehmen und bei wiederholtem Besuch der Seite, kann er sich einfach einloggen.

BASIS - FRONTEND

4. Struktur

Im Frontend werden alle möglichen Bereiche der Website mit einer Komponente dargestellt. Die Komponenten sind wiederum in die Packages "Pages", "Shared" und "UI" untergliedert.

- **"UI"** beinhaltet alle Komponenten, welche überwiegend visuelle Zwecke haben und selbst keine Seite darstellen. Dazu zählen zum Beispiel die Navbar und der Footer, welche beim initialen Aufruf der Website geladen und anschließend unverändert angezeigt werden. Auch die grünen Pop-ups, die dem Endnutzer eine gefundene Schwachstelle signalisieren, fallen in diesen Bereich, da sie keiner Seite zugeordnet werden und keine tieferen Funktionen darstellen.
- **"Shared"** beinhaltet alle Komponenten, welche von den "Pages"-Komponenten benötigt werden. Sie helfen die Redundanz in den Komponenten so gering wie möglich zu halten. Sie können Funktionen bereitstellen und visuelle Inhalte anzeigen, müssen jedoch immer von einer Komponente aus dem "Pages"- oder dem "Shared"-Bereich eingebunden, bzw. aufgerufen werden. Hierzu zählen Elemente, wie der Lösch-Button oder eine Aufzählung von Artikeln in einer tabellenartigen Form. Letztere wird unter anderem in dem Bestellprozess, der Übersicht einer getätigten Bestellung, in der Übersicht von gefilterten Artikeln, im Warenkorb, und in weiteren Komponenten benutzt. Da all diese "Pages"-Komponenten dieselben "Shared"-Komponenten verwenden, werden Fehler leichter wartbar und die visuelle Konsistenz des Shops wird sichergestellt.
- **"Pages"** stellt alle Seiten unserer Single Page Application dar. Somit sind alle durch eine festgelegte URL erreichbar.

Alle Komponenten, die einen Ursprung einer Route darstellen, werden durch ein Modul repräsentiert, welches sich im selben Package befindet. Das Modul beinhaltet alle Dependencies, Provider und die Hauptkomponente der Route. Dadurch sind kompaktere Imports und Lazy-Loading möglich. Auf Lazy-Loading wird in ["Performance"](#) näher eingegangen.

Aufbau einer Komponente (gilt auch für Module in Shared oder UI):

└─ Pagename

```

├── pagename.component.ts
├── pagename.component.html
├── pagename.component.scss
└── pagename.module.ts

```

Die gerade näher beschriebenen Routes werden in dem app-routing-Module definiert. Neben dem Pfad, unter welchem das Modul aufrufbar sein soll, wird auch das Modul, welches durch Lazy-Loading geladen werden soll, festgelegt.

Alle Objekte werden in dem Models-Package deklariert.

Struktur:

Aus Gründen der Übersicht wurden für diesen Punkt irrelevante Dateien und Packages nicht mit aufgeführt.

```

├── app
│   └── app-routing.module.ts
├── lib
│   ├── data-access
│   │   ├── models
│   │   └── service
│   │       ├── store
│   │       └── backend.service.ts
│   ├── pages
│   ├── shared
│   └── ui

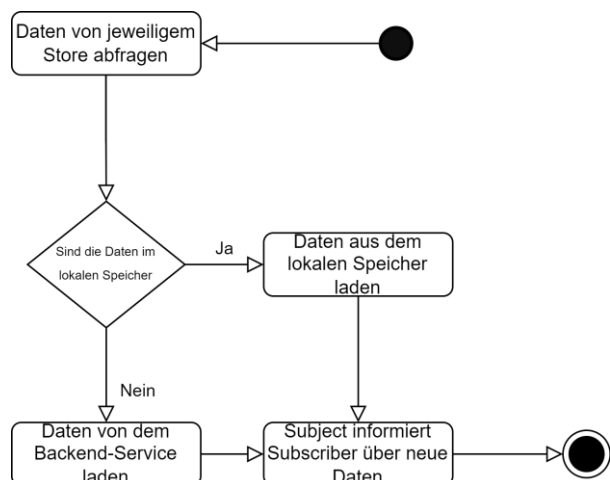
```

5. Performance

Wie bereits erwähnt, ist das Frontend in Angular implementiert. Mit Angular wurde sich für eine Single Page entschieden, wodurch lediglich eine Seite geladen werden muss und anhand dieser, die Anfragen des Nutzers, dynamisch aufgebaut werden können.

Wenn ein Nutzer durch einen Link oder Button eine andere Seite aufruft, wird diese durch Lazy-Loading nachgeladen. Durch die Verwendung, der in "Struktur" beschriebenen Module, werden nur die Dependencies geladen, welche zu dem Zeitpunkt auch notwendig sind.

Wenn bei dieser Anfrage Datensätze vom Backend benötigt werden, fragt ein Store diese an. Die Daten werden, wie in "Funktionsweise" beschrieben, gecached, wodurch beispielsweise Bilder nur ein Mal pro Session vom Backend geladen werden müssen. Sucht man beispielsweise nach dem Besuchen der Frontpage alle Handys der Marke Apple, werden die Bilder der Artikel auf der "neuen" Seite (bzw. des "neuen" Moduls) lediglich aus dem Store



aufgerufen und nicht neu vom Backend abgefragt. Dies ermöglicht kürzere Ladezeiten und dadurch ein sehr flüssiges Nutzererlebnis.

6. Datenverwaltung

Wenn eine Komponente Daten vom Backend abfragt, sollte ein Store verwendet werden. Ein Store ist ein Zwischenspeicher eines Datensatzes, welcher das Handling der Anfragen übernimmt und die Antworten cached. Dieser beinhaltet eine Variable, welche einen lokalen Stand des Datensatzes speichert und ein Subject, welches die Daten an alle Subscriber sendet. Initial ist diese Variable wahlweise ein leeres Array oder undefined. Wird der Store nach einem Datum oder Datensatz gefragt, sucht er dieses/ diesen zunächst in der eben genannten Variable. Wenn das Datum/ der Datensatz dort vorhanden ist, wird er lediglich durch das Subject an den Aufrufer gesendet. Andernfalls werden die gesuchten Daten durch den Backend-Service vom Backend abgefragt, in der Variable gespeichert und anschließend durch das Subject an den Aufrufer gesendet. Der dadurch gewonnene Performance-Boost wird in "[Performance](#)" näher beschrieben.

Der Backend-Service beinhaltet alle Anfragen ans Backend und setzt automatisch die bekannten Header.

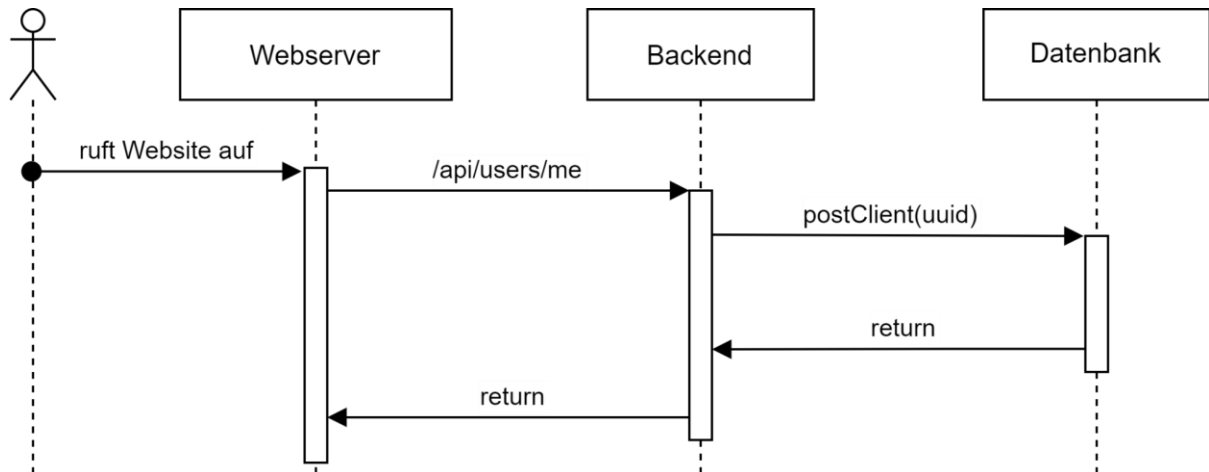
7. Verwaltung / Erkennung der Clients

Während der Planung der [Schwachstellen](#) wurden sich Gedanken über die Zuordnung der Schwachstellen zu den jeweiligen Clients gemacht. Da der Nutzer auch über externe Programme wie Postman oder eigens geschriebene Algorithmen wie Brute-Force-Algorithmen Schwachstellen finden soll, wurde ein Ansatz mit Hilfe der IP-Adresse verfolgt. Jedes Finden einer Schwachstelle sollte durch die IP-Adresse des anfragenden Clients im System hinterlegt werden. Dies ist jedoch nicht möglich, da Angular in unserer Konfiguration einen Proxy-Server nutzt, welcher die Anfragen bidirektional anpasst. Somit kommt jede Anfrage ans Backend von der IP-Adresse des Webserver (entspricht bei Angular dem Proxy). Dieser Proxy ist notwendig, um die Überprüfung durch CORS (Cross-Origin-Resource-Sharing) zu umgehen, welche auf Webseiten essentiell ist, um das Nachladen externer Ressourcen zu verbieten.

Daher wurde eine Alternative über UUIDs (Universally Unique Identifier) gewählt, welche in den Cookies abgelegt werden. Wird die Website zum ersten Mal auf dem Client aufgerufen, wird eine neue UUID in dem Uuid-Service vom Backend abgefragt. Diese wird anschließend in den Cookies abgelegt und zukünftig bei jeder Anfrage in dem Header mitgesendet. Aus Datenschutz- und Sicherheitsgründen ist diese genau einen Tag gültig. Wird die Website geschlossen und erneut aufgerufen, überprüft der Uuid-Service, ob es eine gültige UUID in den Cookies hinterlegt ist. Sofern dies der Fall ist, wird diese weiterhin verwendet, ansonsten wird das Backend erneut angefragt. Während der Erzeugung der UUID im Backend wird für das Admin-Panel ein neuer Nutzer unter dieser in der Datenbank angelegt.

Durch einen Neustart des Backends wird die Datenbank inklusive dieser Daten gelöscht. Der Client besitzt jedoch in den Cookies eine für ihn noch gültige UUID und würde diese bei jeder Anfrage mitschicken. Damit das Backend mit diesem Fall umgehen kann, wird bei jeder Anfrage an dieses überprüft, ob es die UUID in der Datenbank bereits gibt. Sollte dies nicht der Fall sein, wird ein neuer Client mit dieser in der Datenbank angelegt. Somit würde ein

Client mit einer bereits vorhandenen UUID bei dem Finden der ersten Schwachstelle in der Datenbank neu angelegt werden und ab diesem Zeitpunkt im Admin-Panel angezeigt werden.

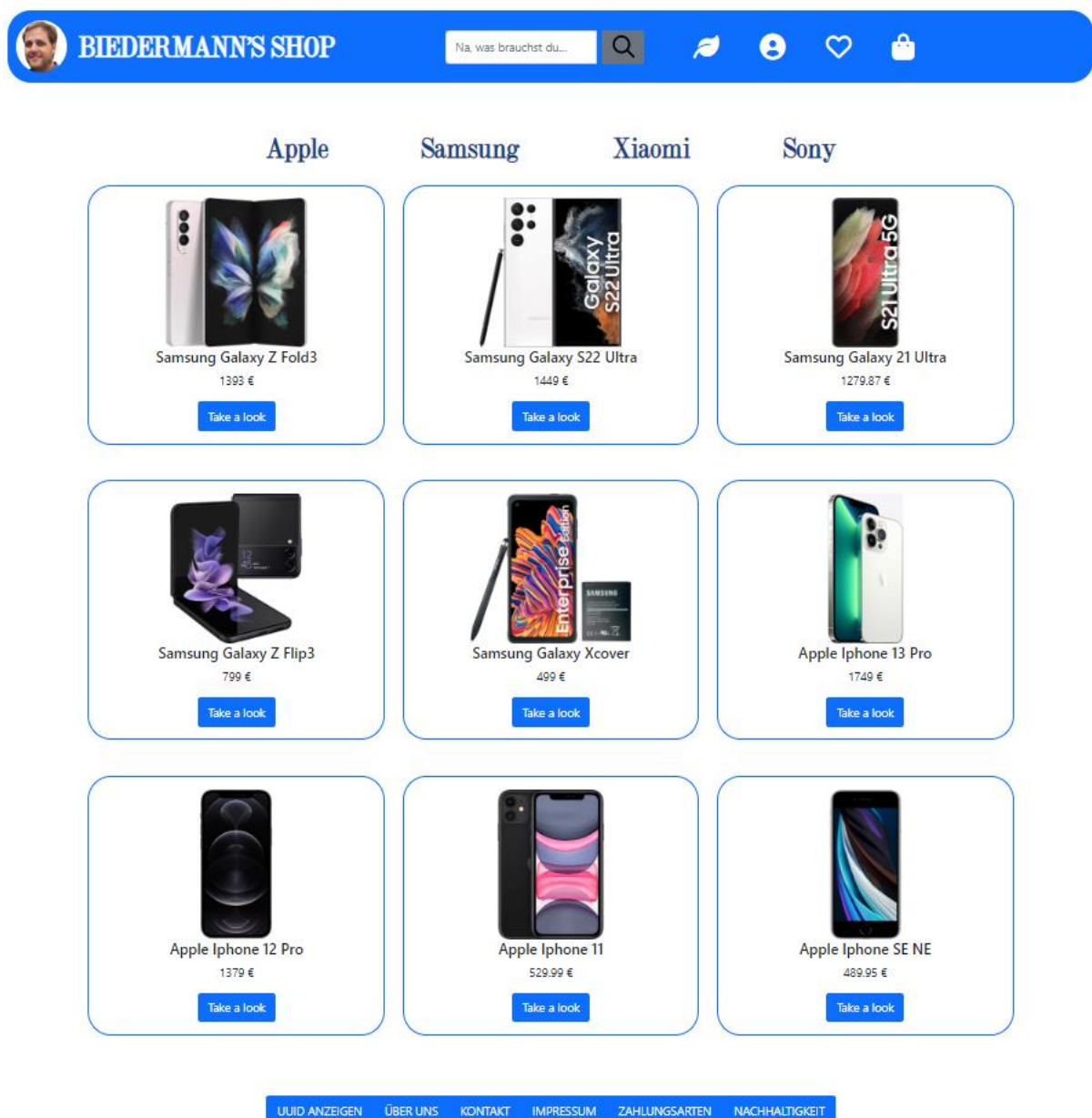


Neben der UUID wird für jeden angemeldeten Nutzer ein Session-Key gespeichert, welcher den Nutzer authentifiziert. Dieses wird ebenfalls im Header unter dem Key "sessionid" bei jeder Anfrage mit geschickt.

KOMPONENTEN

Im Voraus, während der Planungsphase, wurden Mockups erstellt, wie der Webshop später aussehen soll. Natürlich weichen die Mockups vom tatsächlichen Webshop ab, da einige Design-Entscheidungen während des Projektverlaufs überdacht und geändert wurden.

8. Frontpage



9. Navbar

Die Icons in der Navbar wurden sorgfältig ausgesucht, um Unstimmigkeiten zu vermeiden und die Bedienung so leicht wie möglich zu gestalten.

Shop Name und Bild

Der Shop Name "BIEDERMANN'S SHOP" in der Navbar bringt den User immer wieder auf die Startseite zurück. Das Shop-Bild hingegen, bringt den User auf die Website der Forschenden an der FHWS. In unserem Fall zu einem ganz bestimmten Forschenden, zu unserem Auftraggeber Herr Prof. Dr.-Ing. Sebastian Biedermann.

Ebenfalls wurde eine Suche in der Mitte der Navbar implementiert, weiters dazu im Bereich [Navigation](#).

Nachhaltigkeits Symbol

Das Blatt ist das Nachhaltigkeits Symbol, um auf die Umweltbewusstheit des Shops aufmerksam zu machen. Mehr dazu findet sich auf der Nachhaltigkeits Seite.

Account Symbol

Das Account Icon bringt den User auf die [Loginseite](#), wo er sich einloggen oder registrieren kann. Ebenfalls kommt man von dort aus auf die [Admin Login Seite](#), auf der sich die vorhandenen Admins des Webshops einloggen können. Des Weiteren führt das Account Icon den eingeloggten User auf die [Nutzereinstellungen](#), wo er seine Daten überarbeiten und/oder ergänzen kann. Es wurden mehrere Schwachstellen auf dieser Seite eingebaut. Welche Schwachstellen und wo sie zu finden sind erfährt man im Abschnitt [Schwachstellen](#).

Wunschliste und Warenkorb

Wenn man noch nicht eingeloggt ist, führen die Icons der Wunschliste und des Warenkorbs ebenfalls auf die [Loginseite](#).

Solange keine Artikel auf der Wunschliste oder im Warenkorb sind, wird eine Meldung angezeigt, dass noch keine Artikel enthalten sind.

Auf den jeweiligen Seiten wird, sobald man über den Artikel hovered, ein Delete Button angezeigt und mit betätigen des Buttons, wird der Artikel aus der jeweiligen Seite entfernt. Auf der Wunschliste ist zusätzlich oben rechts ein Button, um alle Artikel von der Wunschliste zu löschen.

Des Weiteren gibt es auf der Wunschliste bei jedem Artikel einen Button, um diesen aus der Wunschliste zu löschen und im Warenkorb hinzuzufügen.

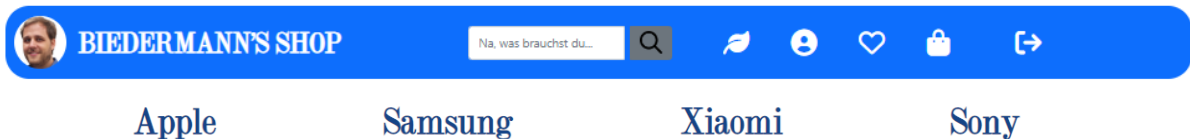
Im Warenkorb wird der Preis aller enthaltenen Artikel zu einem Gesamtpreis addiert und unten als "Gesamtsumme" angezeigt.

Vom Warenkorb aus kommt man mit dem Klick auf den "Zahlen" Button weiter zum Bezahlvorgang. Mehr Details zu diesem Vorgang unter [Bestellprozess](#).

Logout Symbol

Es erscheint ein weiteres Icon zum ausloggen, welches logischerweise nur angezeigt wird, wenn man eingeloggt ist.

10. Navigation



Suche

In das Input Feld kann man ein Teilwort, eine Marke, oder ein Modell (o. ä) eingeben und wird dann, durch einen Klick auf die Lupe, auf die entsprechende Artikelübersicht weitergeleitet.

In das Input Feld kann man ein Teilwort, eine Marke, oder ein Modell (o. ä) eingeben. Der Input wird anschließend mit dem Klick auf die Lupe, oder dem Drücken der Enter Taste, an das Backend gesendet. Dort wird mithilfe des SQL Operators "LIKE" überprüft, ob der Input ein Teil des Artikelnamens oder des Artikelmodells ist. Anschließend wird man auf die entsprechende Artikelübersicht mit den Ergebnissen weitergeleitet.

Menü

Die Menüpunkte unter der Navbar leiten einen ebenfalls auf die Artikelübersicht weiter. Mehr dazu unter dem Punkt [Artikelübersicht](#).

11. Footer

UUID anzeigen (Universally Unique Identifier == Standard für Identifikationsnummern)

Im Footer (= im unteren Ende der Website) findet man ganz links die persönliche UUID.

Diese wird im Ranking auf der Admin Seite aufgelistet. Mehr Infos dazu im Bereich [Verwaltung / Erkennung der Clients](#).

Über uns

Eine Beschreibung über das Team befindet sich in unserem Webshop unter dem Punkt "Über uns". Wir sind eine verrückte Truppe aus dem 4. Semester an der (noch) FHWS.

Kontakt

Das Kontaktformular, beschrieben unter [Kontaktformular](#), findet man unter dem Punkt Kontakt im Footer.

Impressum

Die rechtlich vorgeschriebenen Grundlagen des oder der presserechtlich Verantwortlichen für einen im eigenen Namen veröffentlichten Text-, Wort- oder Bildbeitrag, Anschrift und Ansprechpartner, findet man im Impressum.

UUID ANZEIGEN

ÜBER UNS

KONTAKT

IMPRESSUM

ZAHLUNGSARTEN

NACHHALTIGKEIT

Zahlungsarten

Die verfügbaren Zahlungsarten, die der Shop unterstützt, beschränkt sich momentan auf "Kauf auf Rechnung".

Nachhaltigkeit

Da Umweltbewusstsein in der heutigen Zeit enorm wichtig ist, findet sich auch im Footer eine Verlinkung auf die Nachhaltigkeitsseite des Shops, wo man mehr dazu erfahren kann.

Newsletter

Wenn man eingeloggt ist, erscheint ein weiterer Button im Footer. Mehr Informationen über den Newsletter findet man unter dem Punkt [Newsletter](#).

12. Login Prozess

Login

Im Login Bereich können sich registrierte Nutzer einloggen. Hier ist eine Schwachstelle im HTML Code versteckt. Weiteres dazu unter [Schwachstellen](#) unter HTML Kommentar User.

Registrieren

Vom Login Bereich aus, kommt man weiter zum Registrierungsformular. Die Schwachstellen in diesem Bereich sind Level abhängig. Es wird angezeigt, ob eine E-Mail und ein Passwort bereits vorhanden ist. Weiteres dazu unter [Schwachstellen](#).

Login Admin

Vom Login Bereich aus, kommt man weiter zum Login Bereich für die Admins. Diese sind bereits in der Datenbank hinterlegt und man kann sich nicht als Admin registrieren. Nach dem Login wird man auf die Admin Seite weitergeleitet, weiteres dazu unter dem Punkt [Admin Seite](#).

13. Newsletter

Der Newsletter befindet sich im shared-Ordner, da diese Komponente hauptsächlich visuell ist und von mehreren Seiten aus über den Footer erreichbar ist. (Details dazu s. [Footer](#)). Man kann das Newsletter-Modal nur erreichen, wenn man als User angemeldet ist. Das hat den einfachen Grund, dass der Newsletter auf den User-Store zugreift.

Wenn ein User einen Newsletter abonniert, wird das im Backend beim jeweiligen User in Form eines Flags gespeichert. Dieses Flag wird auf true gesetzt, wenn der Newsletter abonniert ist und steht auf false, wenn noch kein Newsletter abonniert wurde, beziehungsweise der Newsletter wieder deabonniert wurde.

Um den Newsletter abonnieren zu können, muss man eine Email-Adresse angeben. Ohne etwas in das Input-Feld geschrieben zu haben, ist eine Anmeldung nicht möglich, was durch die rote Markierung des Feldes und die Kennzeichnung durch ein Ausrufezeichen visuell dargestellt wird.

Des Weiteren ist das Eingabefeld des Newsletters ein weiterer Angriffspunkt, denn hier kann man einen [Blind-SQL-Injection-Angriff](#) ausführen.

NEWSLETTER ABONNIEREN ×

Mit der Anmeldung akzeptieren Sie unsere AGBs.

ANMELDEN



14. Kontaktformular

Das Kontaktformular stellt eine Page dar und ist im Footer über den Button „Kontakt“ zu erreichen (Details s. [Footer](#)). Das Formular besteht aus vier Eingabefeldern und einer Checkbox, um die Datenschutzerklärung zu akzeptieren.

KONTAKTFORMULAR

NAME*

Vorname

Nachname

EMAIL*

Deine Email

NACHRICHT*

DATENSCHUTZ*

☐ Ich willige in die Datenschutzerklärung ein.

ABSENDEN

Alle Felder müssen ausgefüllt werden. Wenn man das Kontaktformular absenden will, ohne alle Felder ausgefüllt zu haben, bekommt man eine Meldung angezeigt, in der man darauf hingewiesen wird, alle Felder auszufüllen. Das Absenden ist erst möglich, wenn das geschehen ist.

Nach Absenden des Formulars wird man auf die Startseite weitergeleitet. Das Kontaktformular nutzt den Store „contact“ um die eingegebenen Daten an das Backend zu übermitteln. Hier wird, wie beim Newsletter mit der Methode „emailWithoutAt“ überprüft, ob in der angegebenen Email-Adresse ein „@“-Zeichen enthalten ist und wenn nicht, dann hat man eine [Schwachstelle](#) gefunden. Außerdem birgt das Eingabefeld für die Nachricht die Schwachstelle eine [Blind-SQL-Injection](#) ausführen zu können.

15. Informationsseiten der Schwachstellen

Findet man eine Schwachstelle, wird ein Pop-up angezeigt. Dieses gibt an, um welche Schwachstelle es sich bei der eben gefundenen handelt.

✓ **Schwachstelle gefunden!**
blind_sql_injection

Man kann das Pop-up anklicken. Dadurch gelangt man auf eine Informationsseite, die die entsprechende Schwachstelle im Detail beschreibt. Insgesamt gibt es 14 Informationsseiten. Durch diese, sollen die Nutzer der Website, sich nach dem Finden einer Schwachstelle durchlesen können, warum die eben gefundene Schwachstelle Sicherheitsrisiken mit sich bringt und wie man sie eventuell beheben beziehungsweise verhindern kann. Darüber hinaus finden sich Links zu weiteren

Beschreibungen auf den Seiten. Dadurch soll der Lehr-Aspekt der Website ausgebaut und neben dem reinen Finden auch das Vorbeugen nahegelegt werden.

Der Admin muss die Schwachstelle nicht erst finden, um auf die Informationsseiten zugreifen zu können. Er kann über seine Admin-Seite, auf der er das Ranking sieht, auf das Fragezeichen neben der jeweiligen Schwachstelle klicken und gelangt dann auf die entsprechende Informationsseite zur Schwachstelle.


16. Artikelübersicht

Auf der Übersichtsseite finden sich alle Artikel, die dem benutzten Filter oder dem benutzten Suchbegriff entsprechen.

The screenshot displays a web interface for mobile phones. On the left is a blue sidebar with the heading 'Hersteller' and a list of brands: Apple, Samsung, Xiaomi, and Sony. The main content area is titled 'Handys & Smartphones' with the subtitle 'Top Angebote zu den besten Preisen'. A dropdown menu in the top right corner is set to 'Bestseller'. The first product listed is the 'Galaxy Z Fold3' priced at '1393€'. It includes a color selection row with five circles (red, blue, green, black, white) and a storage selection row with three buttons: '64 GB', '128 GB', and '256 GB'. The second product, 'Galaxy S22 Ultra', is partially visible below with a price of '1449€'.

Links findet man alle Hersteller gelistet. Hierbei handelt es sich um Filter Buttons und wenn man alle Smartphones eines bestimmten Herstellers angezeigt bekommen möchte, kann man das mit einem Klick auf den entsprechenden Namen erreichen. Wenn man sich für ein Smartphone interessiert, kann man dieses anklicken und gelangt dann auf die Artikelübersicht-Page.

17. Nutzereinstellungen



[Profilbild ändern](#)

Max Mustermann
max.mustermann@student.fhws.de
[About](#)
Herzlichen Glückwunsch, du hast den Dummy User gefunden!
☐ Newsletter angemeldet

Anrede
Herr

Vorname
Max

E-Mail
max.mustermann@student.fhws.de

Beschreibung
Ein temporärer Nutzer zur Demonstration der Website. `<script>alert("tolle Website")</script>`

Titel
Dr.

Nachname
Mustermann

Adressen

Max Mustermann
Musterstraße 7
123456 Musterstadt
Musterland

[Adresse hinzufügen](#)

Altes Passwort

Neues Passwort

Neues Passwort bestätigen

Passwort Anforderungen

Um ein neues Passwort zu erstellen, musst Du alle folgenden Voraussetzungen erfüllen Anforderungen erfüllen:

- Mindestens 8 Zeichen
- Mindestens ein Sonderzeichen
- Mindestens eine Nummer

abbrechen

speichern

Die Page „Nutzer Einstellungen“ dient dazu, alle nutzerbezogenen Daten zu verwalten und gegebenenfalls zu verändern. Die Page besteht aus neun Input-Feldern, wobei die Felder „Vorname“, „Nachname“ und „E-Mail“ nach dem Registrieren mit den entsprechenden Daten ausgefüllt sind. Die Felder „Anrede“ und „Titel“ sind als Dropdown-Menu realisiert, sodass man die entsprechende Anrede und den entsprechenden Titel einfach auswählen kann.

Das Feld „Beschreibung“ realisiert eine Sicherheitslücke. Man kann hier eine [Cross Site Scripting \(XSS\)-Attacke](#) ausführen. Eine Eingabe im Beschreibungsfeld wird unter dem Punkt About in der linken Hälfte angezeigt und sollte es sich beispielsweise Java-Script Code handeln, interpretiert und ausgeführt.

Unter „About“ gibt es die Möglichkeit sich für den Newsletter an- und abzumelden. Außerdem kann man ein Profilbild hochladen, was eine weitere [Schwachstelle](#) darstellt. Anstatt einer Bilddatei können hier auch andere Dateien hochgeladen werden.

Unter dem Punkt „Adresse“ kann man eine oder mehrere Adressen anlegen. Wenn man noch keine Adresse angelegt hat, kann man mit einem Klick auf „Klick hier“ eine neue Adresse anlegen. Man gelangt zu einem Eingabefeld und mit einem weiteren Klick auf den Stift erscheint eine Art Pop-Up, in das man die Adressdaten eingeben kann. Wenn man bereits eine existierende Adresse hat, kann man mit einem Klick auf „Adresse hinzufügen“ noch eine weitere Adresse anlegen.



Darunter gibt es die Möglichkeit das Passwort zu ändern. Die Eingabe wird auf

- altes Passwort ist falsch
- neues Passwort erfüllt nicht die Voraussetzungen
- neue Kennwörter stimmen nicht überein

überprüft und im Falle einer Unstimmigkeit angezeigt.

Fehler! Das neue Kennwort erfüllt nicht alle Voraussetzungen!

18. Artikelansicht

Galaxy Z Flip3

128 GB
256 GB
512 GB

Modellname	Galaxy Z Flip3
Marke	Samsung
Betriebssystem	Android 11.0, ONE UI
Modelljahr	3.1.1, KNOX 3.7
Display	17.04.2020
Auflösung	Super AMOLED 2.640x1.080

799€

Menge: 1

In den Einkaufswagen

★★★★☆

- ✓ pro Verkauf ein Baum
- ✓ Gratis Rückversand
- ✓ 30 Tage Rückgaberecht

Auf die Wunschliste

Kundenrezensionen

Walter Schmitt

flippst sich gut <3

Kommentar hinzufügen

Die Artikelübersichtsseite besteht aus dem Bilderslide, den gerätespezifischen Details, den Aktionsmöglichkeiten und der Kommentarsektion. Die "Image-Section" holt sich mit Hilfe des „imageStores“ die Bilder aus der Datenbank und zeigt diese im Wechsel an. Man kann aber auch selbst zwischen den Bildern wechseln, indem man auf die gräulichen Striche klickt. Rechts neben den Bildern findet man die „Specifications“ mit allen Details des Artikels. Daneben findet sich die „Overview“. Hier kann man den Preis sehen, die Menge, in Form eines Drop-Down-Menüs, auswählen und die ausgewählte Menge des Artikels entweder „In den Einkaufswagen“ legen oder „Auf die Wunschliste“ setzen. Außerdem werden hier noch Bewertungen in Form von Sternen angezeigt. Darunter kann man die Nutzerkommentare zu dem Artikel lesen und selbst einen Kommentar hinzufügen, aber nur wenn man bereits eingeloggt ist. Hier versteckt sich eine Sicherheitslücke, denn die Kommentarfunktion ermöglicht [persistentes Cross Site Scripting](#) (XSS).


19. Bestellprozess


Wie bei jedem guten Webshop muss man, bevor man einen Artikel bestellt hat, den Bestellprozess durchlaufen. Um den Überblick zu behalten, wie weit man mit diesem bereits ist, gibt es eine „Fortschrittsanzeige“. Der Schritt bei dem man sich aktuell befindet, sowie die Schritte, die man bereits abgeschlossen hat, werden in blau angezeigt, alle anderen in grau.

Diese Übersicht dient auch dazu während des Bestellprozesses nochmal einen oder mehrere Schritte zurück zu gehen. Das geht einfach, indem man den Schritt anklickt, zudem man zurückkehren möchte.

Der Bestellprozess beginnt damit, das man eine Übersicht der Artikel angezeigt bekommt, die zuvor im Warenkorb waren. Hier hat man nochmal die Möglichkeit die Menge der Artikel zu verändern oder Artikel, die man doch nicht bestellen will zu löschen.

Übersicht



Galaxy Z Flip3
Artikelnr: 4 799€ Menge:  128 GB

Gutschein

Zwischensumme:

799 €

TOTAL

799 €

Außerdem kann man einen Gutscheincode einlösen, worin eine [Schwachstelle](#) verborgen ist. Denn die Gutscheincode sind so einfach gewählt, dass man leicht einen erraten kann. Mit einem klick auf den „Weiter“-Button gelangt man zum Schritt „Adresse“. Hier kann man mit einem Klick auf die Adresse die gewünschte Lieferadresse auswählen. Falls man keine auswählt, wird man durch eine Meldung darauf hingewiesen, dass man eine Adresse auswählen muss, um den nächsten Schritt zu machen. Wenn man noch keine Adresse

angegeben hat, muss man hier noch eine hinzufügen. Dies läuft genauso ab, wie bei den [Nutzereinstellungen](#).

Mit einem Klick auf „Weiter“ gelangt man zum Schritt „Zahlung“. Hier muss man seine Kontodaten wie Kontoinhaber, IBAN und BIC angeben. Sollte man nicht alles ausgefüllt haben, wird man durch eine Meldung darauf hingewiesen dies erst zu tun, bevor man fortfahren kann.

Mit einem Klick auf „zahlungspflichtig bestellen“ ist die Bestellung dann abgeschlossen und man gelangt auf eine Seite mit der Übersicht über die Bestellung und den angegebenen Daten.

20. Bestellübersicht

Über die Nutzereinstellungen kommt man mithilfe des “Letzte Bestellungen” Buttons auf die Übersichtsseite der letzten Bestellungen.

Hier werden zuerst nur die Namen der Artikel und das Bestelldatum angezeigt, damit die Seite nicht mit Informationen überflutet wird. Wenn man nun auf eine Bestellung klickt, klappt die Bestellung auf und die weiteren Informationen der Bestellung werden vom Backend nachgeladen. Hier wird nun die Bestellnummer und die Gesamtsumme angezeigt. Anschließend werden die einzelnen Artikel, mit einem Trennstrich getrennt, aufgelistet. Am Ende ist die Versandadresse und die IBAN der Zahlungsart dargestellt.

Mit erneutem klicken auf den Bestellungsheader der einzelnen Bestellung, kann die Bestellung wieder eingeklappt werden. Es können mehrere Bestellungen parallel geöffnet werden.

ADMIN PAGE

21. Admin Page mit Ranking

Wenn man sich erfolgreich als Admin eingeloggt hat, ist man auf der Rankingseite. Hier werden die Nutzer mit ihren zugehörigen UUIDs angezeigt. Man sieht die Punkteübersicht, nach der die Nutzer sortiert werden (höchste Punktzahl ganz oben, niedrigste Punktzahl ganz unten) und wie viele Schwachstellen von den insgesamt 14 gefunden wurden.

Über den zugehörigen Button kann man die Datenbank oder nur den Shop und seine Schwachstellen auf ihren Anfangszustand zurücksetzen.

Durch ein Top-Down Menü kann das Level (Beginner, Tutor, Endboss) angepasst werden.

Bei den jeweiligen Nutzern gibt es ein kleines + Symbol. Damit öffnet man ein Bootstrap-Modal. Es zeigt die genauen Schwachstellen an, welche von Nutzern gefunden worden sind (grüne Schriftfarbe = gefunden, graue Schriftfarbe = noch nicht gefunden). Außerdem steht zu jeder Schwachstelle eine detaillierte Beschreibung, wo sie zu finden ist. Diese

Beschreibung ist levelabhängig. Ein kleines Fragezeichen, neben der Beschreibung, führt zu einer weiteren Seite, wo die Schwachstelle im Allgemeinen beschrieben ist und wie man sie verhindern kann (s. [Informationsseiten](#)).

Um als Admin wieder auf die Ranking Seite zu kommen, wurde ein Gamepad Symbol in die Navbar implementiert.

BACKEND - WEBSHOP

22. Logik – Api

Admin Service (/admin)

GET Login

[Open Request →](#)

http://localhost:8080/api/admin/login?username=admin&password=admin

Rückgabe der sessionID.

Query Params

username	admin
password	admin

GET Read Level

[Open Request →](#)

http://localhost:8080/api/admin/level

Rückgabe des aktuell eingestellten Levels.

Request Headers

sessionid	4dm1nufhbtn#d\$e&8#i3i349s#4492h=
-----------	-----------------------------------

GET Get Ranking

[Open Request →](#)

http://localhost:8080/api/admin/interface

Rückgabe des Rankings.

Request Headers

sessionid	
-----------	--

Query Params

sessionid	4dm1nufhbtn#d\$e&8#i3i349s#4492h=
-----------	-----------------------------------

PUT Reset Database Shop

[Open Request →](#)

http://localhost:8080/api/admin/interface

Zurücksetzen der Datenbank.

Request Headers

sessionid	4dm1nufhbtn#d\$e&8#i3i349s#4492h=
-----------	-----------------------------------

POST Logout

[Open Request →](#)

http://localhost:8080/api/admin/logout

Löscht die sessionID aus der Datenbank, sodass man sich nicht mehr mit der sessionID anmelden kann.

Request Headers

sessionid	4dm1nufhbtn#d\$e&8#i3i349s#4492h=
-----------	-----------------------------------

POST Set Level

[Open Request →](#)

http://localhost:8080/api/admin/interface?level=1

Setzen der Level.

Request Headers

sessionid	4dm1nufhbtn#d\$e&8#i3i349s#4492h=
-----------	-----------------------------------

Query Params

level	1
-------	---

DEL Reset Ranking

[Open Request →](#)

http://localhost:8080/api/user/interface

Zurücksetzen / Löschen des Rankings.

Request Headers

sessionid	4dm1nufhbtn#d\$e&8#i3i349s#4492h=
-----------	-----------------------------------

Article Service (/articles)

GET Read Articles

[Open Request →](#)

http://localhost:8080/api/articles?page=1&specifications=false&brand=apple&name=iphone 12

Alle Artikel werden aufgerufen und als JSON zurück gegeben. Man kann die Artikel mit den Parametern "page", "specification", "brand" und "name" filtern.

Query Params

page	1
specifications	false
brand	apple
name	iphone 12

GET Read Article By ID

[Open Request →](#)

http://localhost:8080/articles/1

Nur der Artikel mit entsprechender ID wird zurückgegeben.

Cart Service (/cart)

GET

Read Cart Items

Open Request →

http://localhost:8080/api/cart/items

Alle Artikel, die sich im Warenkorb des Users befinden (was durch die sessionId überprüft wird), werden zurückgegeben.

Request Headers

sessionId	snHxIebosNPqjKxigV8VF6wsuG4NW6mV278zbbkm=
-----------	---

POST

Create Cart Item

Open Request →

http://localhost:8080/api/cart/items

Ein Artikel wird im Warenkorb des Users "erstellt". Um den Warenkorb einem User zuordnen zu können, muss die sessionId mitgegeben werden. Im JSON-Format muss das Specified-Item mitgegeben werden.

Request Headers

sessionId	snHxIebosNPqjKxigV8VF6wsuG4NW6mV278zbbkm+8=
uuid	208947012974

Body raw (json)

json

```
{
  "amount": 1393,
  "articleNumber": 1,
  "color": "red",
  "gbSize": 128,
  "id": 1,
  "name": "Galaxy Z Fold3",
  "pictureId": 1,
  "quantity": 18
}
```

View more

DEL

Delete Cart Item

Open Request →

http://localhost:8080/api/cart/items/1

Der Artikel mit der entsprechenden ID wird aus dem Warenkorb des Users gelöscht. Um den Warenkorb einem User zuordnen zu können, muss die sessionId mitgegeben werden.

Request Headers

sessionId	snHxIebosNPqjKxigV8VF6wsuG4NW6mV278zbbkm+8=
-----------	---

PUT

Modify Cart Item

Open Request →

http://localhost:8080/api/cart/items/1

Der Artikel mit der entsprechenden ID wird durch den mitgegebenen Artikel ersetzt. Um den Warenkorb einem User zuordnen zu können, muss die sessionId mitgegeben werden.

Request Headers

sessionId	snHxIebosNPqjKxigV8VF6wsuG4NW6mV278zbbkm+8=
-----------	---

Body raw (json)

json

```
{
  "amount": 4,
  "articleNumber": 2846,
  "color": "Red",
  "gbSize": 64,
  "id": 24,
  "name": "Iphone",
  "picture": "picture",
  "quantity": 1
}
```

View more

Commentary Service (/comments)

POST

Create Comment

Open Request →

http://localhost:8080/api/comments

Wird verwendet um Schwachstellen zu erkennen. (xss)

Request Headers

uuid	208947012974
------	--------------

Contact Service (/contact)

POST

Create Contact Information

Open Request →

http://localhost:8080/api/contact

Wird verwendet um Schwachstellen zu erkennen. (email without @)

Body raw (json)

json

```
{
  "firstname": "Max",
  "lastname": "Mustermann",
  "mail": "maimail.com",
  "message": "message"
}
```

Coupon Service (/coupons)

GET

Read Coupon

Open Request →

http://localhost:8080/coupons/sommer2022

Informationen eines Coupons werden abgefragt.

Example

Read Coupon

Request

HTTP

GET /coupons/mycoupon15 HTTP/1.1
Host: localhost:8080

Response

Body Headers

json

```
{
  "active": true,
  "id": 2,
  "name": "mycoupon15",
  "percent": 15
}
```

Flaw Service (/flaws)

GET

Read Flaws

Open Request →

http://localhost:8080/api/flaws/

Rückgabe der vom Client gefundenen, aber noch nicht abgefragten Schwachstellen.

Request Headers

sessionId	OKmRFgO2EuPE94o2xew7LuIM+YcXG79qYDTom9nj6Q8=
-----------	--

Order Service (/orders)

GET

Read All Orders

Open Request →

http://localhost:8080/api/orders

Rückgabe der Bestellungen eines Nutzers.

Request Headers

sessionid

snHxiebosNPqjxxigV8VF6wsuG4NW6mV278zbbkm+8=

POST

Create Order

Open Request →

http://localhost:8080/api/orders?cleanUpWishlist=true

Eine entsprechende Bestellung wird bei dem entsprechenden Nutzer "erstellt". Mit dem Parameter cleanUpWishlist kann man den Warenkorb leeren.

Request Headers

sessionid

snHxiebosNPqjxxigV8VF6wsuG4NW6mV278zbbkm+8=

Query Params

cleanUpWishlist

true

Picture Service (/pictures)

GET

Read Picture

Open Request →

http://localhost:8080/api/pictures/1

Rückgabe des entsprechenden Bildes (Base64 encodiert).

User Service (/user)

GET

Read User

Open Request →

http://localhost:8080/api/users

Rückgabe der Nutzerinformationen.

Request Headers

sessionid

g08FHQPBLVUpfcXlKad7IOZtNdXxquRnJn4ql+t9YGU=

POST

Create User

Open Request →

http://localhost:8080/api/user/register

Erstellung eines neuen Users.

DEL

Delete User

Open Request →

http://localhost:8080/api/users/

Löschen eines Users.

Request Headers

sessionid

snHxiebosNPqjxxigV8VF6wsuG4NW6mV278zbbkm+8=

GET

Read User By Id

Open Request →

http://localhost:8080/api/users/1

Rückgabe der Nutzerinformationen eines bestimmten Nutzers.

PUT

Modify User

Open Request →

http://localhost:8080/api/users/

Modifizierung eines Users.

DEL

Delete User With ID

Open Request →

http://localhost:8080/api/users/1

Unautorisiertes löschen eines Users. Setzt eine Schwachstelle um.

GET Read Payment

Open Request →

http://localhost:8080/api/users/payment

Rückgabe der Zahlungsinformationen.

Request Headers

sessionId snHxiebosNPqjxigV8VF6wsuG4NW6mV278zbbkm+8=

POST Create Payment

Open Request →

http://localhost:8080/api/users/payment

Erstellung der Zahlungsinformationen.

Request Headers

sessionId snHxiebosNPqjxigV8VF6wsuG4NW6mV278zbbkm+8=

Body raw (json)

```
{
  "bic": "9385663890988",
  "iban": "DE19 2347 2837 8376",
  "accountHolder": "Max"
}
```

GET Read Addresses

Open Request →

http://localhost:8080/api/users/addresses

Rückgabe aller Adressen eines Users.

Request Headers

sessionId snHxiebosNPqjxigV8VF6wsuG4NW6mV278zbbkm+8=

POST Create Address

Open Request →

http://localhost:8080/api/users/addresses

Erstellung einer Adresse eines Users.

Request Headers

sessionId snHxiebosNPqjxigV8VF6wsuG4NW6mV278zbbkm+8=

Body raw (json)

```
{
  "address": "Example Street",
  "address2": "Example House",
  "city": "Example City",
  "country": "Example Land",
  "deliveryInstructions": "Example Instructions",
  "id": 1,
  "name": "Example",
  "zipcode": "12345"
}
```

View more

GET Read Address

Open Request →

http://localhost:8080/api/users/addresses/1

Rückgabe einer bestimmten Adresse eines Users.

Request Headers

sessionId snHxiebosNPqjxigV8VF6wsuG4NW6mV278zbbkm+8=

PUT Update Address

Open Request →

http://localhost:8080/api/users/addresses/1

Modifizierung einer Adresse eines Users.

Request Headers

sessionId snHxiebosNPqjxigV8VF6wsuG4NW6mV278zbbkm+8=

Body raw (json)

```
{
  "address": "Example Street",
  "address2": "Example House",
  "city": "Example City",
  "country": "Example Land",
  "deliveryInstructions": "Example Instructions",
  "id": 1,
  "name": "Example",
  "zipcode": "12345"
}
```

View more

DEL Delete Address

Open Request →

http://localhost:8080/api/users/addresses/1

Löschen einer Adresse.

Request Headers

sessionId snHxiebosNPqjxigV8VF6wsuG4NW6mV278zbbkm+8=

GET Read Mail

Open Request →

http://localhost:8080/api/users/mail

Rückgabe einer Email-Adresse eines Users.

Request Headers

sessionId snHxiebosNPqjxigV8VF6wsuG4NW6mV278zbbkm+8=

POST Create Mail

Open Request →

http://localhost:8080/api/users/mail

Erstellung einer Email-Adresse eines Users.

Request Headers

sessionId snHxiebosNPqjxigV8VF6wsuG4NW6mV278zbbkm+8=

Wishlist Service (/wishlist)

PUT	Update Wishlist Item	Open Request →	POST	Create Wishlist Item	Open Request →
<div>http://localhost:8080/api/wishlist/items/1</div>			<div>http://localhost:8080/api/wishlist/items</div>		
<div>Der Artikel mit der angegebene id im Warenkorb vom angegebener (mit sessionId) Nutzer wird durch ein mitgegebener Artikel ersetzt.</div>			<div>Erstellung/Hinzufügen eines Artikels auf der/die Wunschliste.</div>		
Request Headers			Request Headers		
<div>sessionIdsnHxiebosNPqjxigV8VF6wsuG4NW6mV278zbbkm+8=</div>			<div>sessionIdsnHxiebosNPqjxigV8VF6wsuG4NW6mV278zbbkm+8=</div>		
GET	Read Wishlist Items	Open Request →	DEL	Delete Wishlist	Open Request →
<div>http://localhost:8080/api/wishlist/items?sessionId=snHxiebosNPqjxigV8VF6wsuG4NW6mV278zbbkm+8=</div>			<div>http://localhost:8080/api/wishlist</div>		
<div>Alle Artikeln vom angegebener Nutzer (mit sessionId) werden zurückgegeben.</div>			<div>Alle Artikel im Warenkorb vom User mit der angegebene sessionId wird gelöscht.</div>		
Query Params			Request Headers		
<div>sessionIdsnHxiebosNPqjxigV8VF6wsuG4NW6mV278zbbkm+8=</div>			<div>sessionIdsnHxiebosNPqjxigV8VF6wsuG4NW6mV278zbbkm+8=</div>		

<i>AdminLogic</i>
dataAccessAdminPanel: DataAccessAdminPanel dataAccessShopDatabase: DataAccessShopDatabase level: int
login (String mail, String password): String logout (String session): Nullable getRanking (String session): List<RankingRow> resetDatabaseShop (String session): Nullable resetDatabaseRanking (String session): Nullable setLevel (int level, String session): void authorizeRequest (String session): Nullable

<i>AuthenticationLogic</i>
Database: DataAccessShopDatabase
register (User user, String uuid): String login (String mail, String password, String uuid): String logout (String session): Nullable createSessionId (): String

<i>OrderLogic</i>
createOrder (boolean cleanup, String session, Order order, UriInfo uriInfo): URI

<i>FlawHandler</i>
DataAccessAdminPanel: DataAccessAdminPanel
imageWithWrongDataType (String uuid): void emailWithoutAt (String uuid): void htmlCommentUser (String uuid): void putXSS (String uuid): void priceOrder (String uuid): void guessCoupon (String uuid): void guessUserLogin (String uuid): void commentXSS (String uuid): void sqlInjection (String uuid): void deleteOtherUser (String uuid): void blindSqlInjection (String uuid): void endpointScanning (String uuid): void deleteOtherUser (String uuid): void deleteOtherUser (String uuid): void

<i>DataHandler</i>
DataAccessShopDatabase: DataAccessShopDatabase
getArticles (int page, String search, boolean specifications): List<Articles> getArticle (int id): Article getCartItems (String session): List<ArticleVersion> createCartItem(ArticleVersion articleVersion, String session): Nullable modifyCartItem (int id, ArticleVersion articleVersion): ArticleVersion deleteCartItem (String session, int id): Nullable getCoupon (String name): Coupon getPicture (int id): String getOrder (int id): Order createOrder (Order order, String session, boolean cleanup): int getUser (String session): User createUser (User user): int deleteUserById (String session, long userId, String ipOfRequest): Nullable getUserPayment (String session): Payment createUserPayment (String session, Payment payment): int getAllUserAddresses (String session): List<Address> getUserAddress (String session, int addressID): Address createAddress (String session, Address address): Address modifyAddress (String session, Address address): Address deleteAddress (String session, Address address): Nullable getUserMail (String session): String checkNewsletter (String session): boolean turnOnNewsletter (String session): Nullable turnOffNewsletter (String session): Nullable getWishlist (String session): List<ArticleVersion> createWishlistItem (String session, ArticleVersion articleVersion): Nullable modifyWishlistItem (String session, ArticleVersion articleVersion, int id): Nullable deleteWishlistItem (int id): Nullable deleteWishlist (String session): Nullable

23. Aufbau mit dem Builder Pattern

Das Backend baut auf einer funktionalen Aufteilung auf. Zum einen auf den Services, welche die Endpunkte definiert (welche Header, welcher Pfad, usw.) und die erhaltenen Daten an die States weitergibt. Dabei werden nicht direkt die States, sondern die Builder für die States verwendet. In den Buildern werden die jeweiligen Daten festgelegt und überprüft. Ein Beispiel für einen voll ausgebauten Builder-Aufruf sieht folgendermaßen aus:

```
return new DemonstrationState.Builder()
    .withSession( session )
    .withUuid( uuid )
    .withCheckId( model.getId(), requestedId )
    .withAuthorize( isModelIdLargerThan0( model) )
    .withModel( model )
    .withUriInfo( uriInfo )
    .withNotNull( model )
    .withHeader( new Header("HeaderName", "HeaderValue") )
    .defineResponseBody( actionOnModel( model ) )
    .build( )
    .ok( );
```

1. `withSession(final String session)`

`WithSession()` überprüft, ob der übergebene String nicht leer oder null ist. Sollte dies der Fall sein, wird eine `NotAuthorizedException` geworfen und somit eine 401 an den Client gesendet.

2. `withUuid(final String uuid)`

`WithUuid()` funktioniert analog zu `withSession()`.

3. `withCheckId(final int modelId, final int requestedId)`

`WithCheckId()` kommt bei einem Put zum Einsatz. Man übergibt der Funktion zum einen die Id des Models aus dem Request und zum anderen die Id aus der URL. Stimmen beide nicht überein, wird eine `BadRequestException` geworfen und somit eine 400 an den Client gesendet.

4. `withAuthorize(final boolean authorized)`

`WithAuthorize()` nimmt das Ergebnis einer externen Funktion entgegen, welche einen boolean zurückgibt. Diese externe Funktion überprüft zum Beispiel die Zugriffsrechte auf verwendete Ressourcen. Sollte diese Funktion false liefern, wird eine `ForbiddenException` geworfen und somit eine 403 an den Client gesendet.

5. `withModel(final Object model)`

`WithModel()` gibt ein Objekt an den Builder, welches anschließend in diesem zum Beispiel in der internen `lookForFlaw()`-Funktion verwendet werden kann. Dieses wird gleichzeitig auf null überprüft. Sollte es null sein, wird eine `BadRequestException` geworfen und somit eine 400 an den Client gesendet.

6. `withUriInfo(final UriInfo uriInfo)`

`WithUriInfo()` wird für Post Anfragen genutzt. Das Attribut wird verwendet, wenn man als Rückgabecode `create()` definiert, um die Location der erstellten Resource festzulegen.

7. `withNotNull(final Object... objects)`

WithNotNull() bekommt beliebig viele Objekte als Attribut, welche auf null überprüft werden. Ist eines davon null, wird eine BadRequestException geworfen und somit eine 400 an den Client gesendet.

8. `withHeader(final Header header)`

WithHeader() legt einen Header fest, welcher beim Erstellen der Response gesetzt wird.

9. `defineResponseBody(final Object responseBody)`

DefineResponseBody() setzt den Body der Request. Hier sollte - wie im Beispiel gezeigt - eine Funktion in den Argumenten Bereich, welche die Logik des Endpunktes darstellt. Wenn kein Inhalt in den Body soll, sollte der Rückgabotyp der Funktion als Nullable definiert werden und die Funktion sollte null zurück geben. Wird die Logik vor dem Builder ausgeführt, können Änderungen an der Datenbank durchgeführt werden, ohne dass jegliche Art der Autorisierung stattgefunden hat. Wird defineResponseBody() nicht als letztes auf dem Builder aufgerufen, können ebenfalls Änderungen auf der Datenbank ausgeführt werden, ohne dass das Modell zum Beispiel auf null geprüft wurde.

Anschließend wird mit build der Build-Prozess abgeschlossen und mit

- `ok()` → 200
- `create()` → 201
- `noContent()` → 204

der Rückgabecode der Response festgelegt werden. Wenn andere Rückgabecodes benötigt werden, kann entweder die Funktion `statusCode(final int status)` verwendet werden oder alternativ eine neue Funktion für den Code im AbstractState implementiert werden. Letzteres sollte dabei bevorzugt werden.

24. Datenschicht

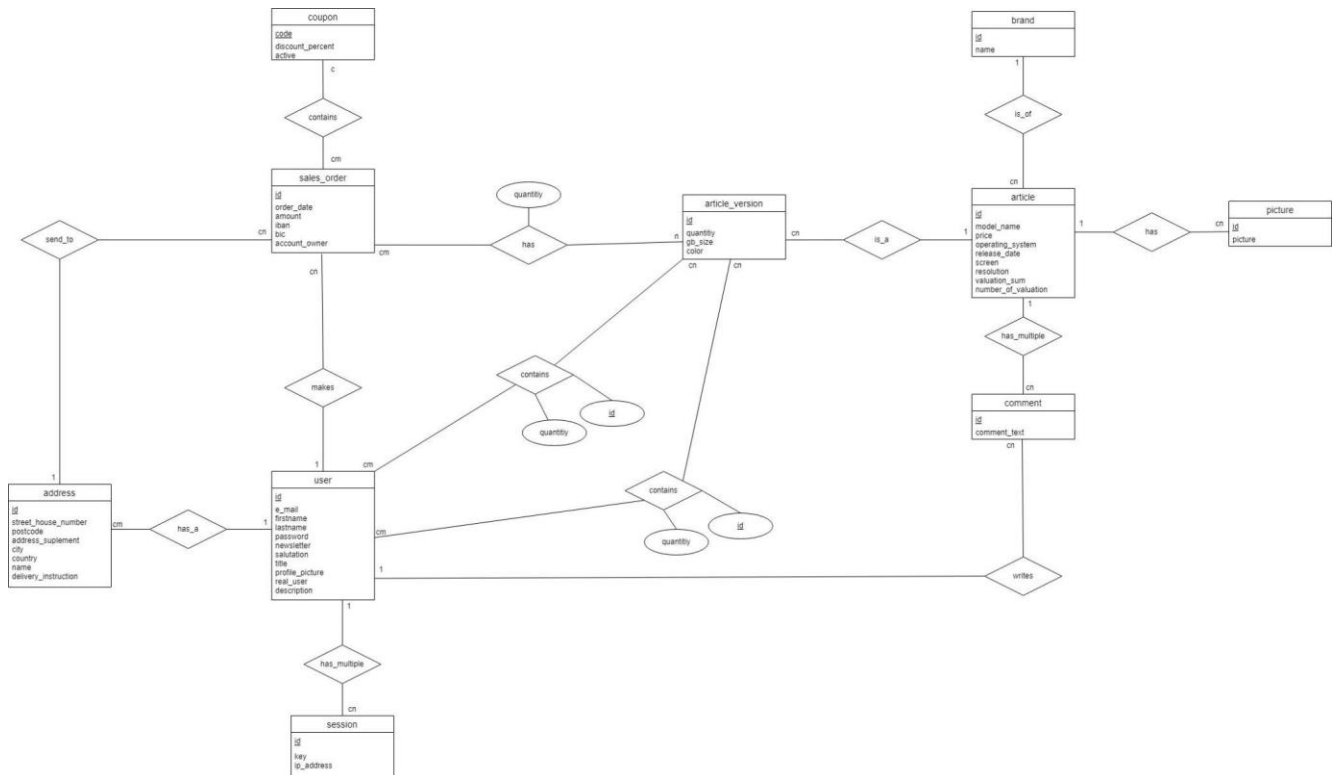
Die Datenschicht teilt sich in zwei Datenbanken auf, in die shopDatabase.db und die adminPanel.db.

Für die Datenbanken wurde sich für SQLite entschieden, da etwas Einfaches und Leichtes für die Zwecke des Shops am besten passt. Der Zugriff erfolgt über JDBC und die Datenbank ist auch nur eine Datei.

In der Klasse DatabaseQueries sind verschiedene Datenbankbefehle und Objekte, die für das Zurücksetzen und neu befüllen der Datenbanken benötigt werden. Das Zurücksetzen wird bei jedem Start des Servers durchgeführt kann aber auch von dem Admin manuell über das [Admin Panel](#) durchgeführt werden.

SHOP DATENBANK

25. ER-Modell:



26. Relationenmodell:

coupon (code, discount_percent, active)

sales_order (id, order_date, amount, iban, bic, account_owner, user_id, address_id, coupon_code)

sales_order_article_version (id, quantity, sales_order_id, article_version_id)

article_version (id, quantity, gb_size, color, article_id)

article (id, model_name, price, operating_system, release_date, screen, resolution, valuation_sum, numer_of_valuation, brand_id)

brand (id, name)

picture (id, picture, article_id)

comment (id, comment_text, article_id, user_id)

wish_list (id, quantity, user_id, article_version)

shopping_cart (id, quantity, user_id, article_version_id)

address (id, street_house_number, postcode, address_supplement, city, country, name, delivery_instruction, user_id)

user (id, e_mail, firstname, lastname, password, newsletter, salutation, title, profile_picture, real_user, description)

session (id, key, ip_address, user_id)

27. Beschreibung

Die Shop Datenbank entspricht der Datenbank, auf welcher der ganze Shop aufbaut. In dieser Datenbank werden sämtliche Daten, die im Zusammenhang mit dem Shop stehen gespeichert. Außerdem befinden sich darin Daten, die für die Sicherheitslücken relevant sind wie die User.

Die Passwörter der Nutzer werden verschlüsselt gespeichert. Dabei wurden sich für zwei Verschlüsselungsverfahren entschieden, dem SHA-1 und dem SHA-512. Die Passwörter der Dummy User werden mit dem SHA-1 verschlüsselt, damit diese dann von den Angreifern entschlüsselt werden können. Die Passwörter von realen Usern, also Accounts, die von den Angreifern erstellt werden, werden mit SHA-512 verschlüsselt damit die verschiedenen Angreifer sich nicht gegenseitig stören können.

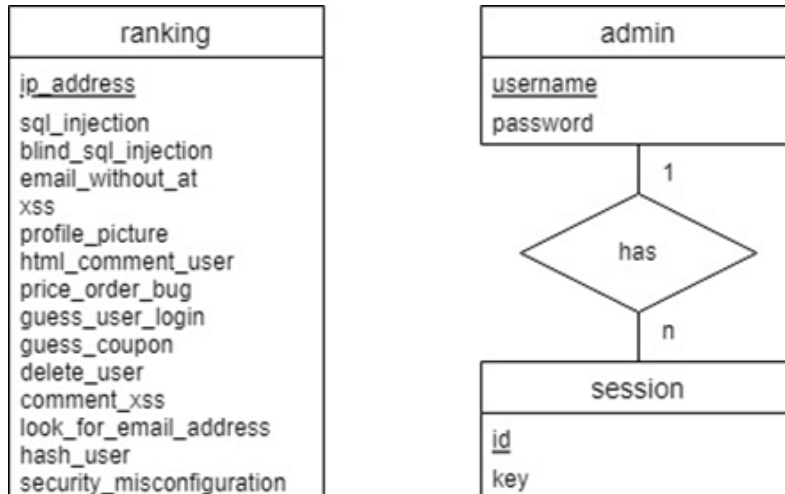
Die Bilder der Artikel wurden Base64 codiert und dann in der Datenbank gespeichert, dies soll dafür sorgen, dass die Bilder nicht extra gespeichert werden müssen und immer schnell auf diese zugegriffen werden kann.

Die Klasse, welche diese Datenbank mit der Logik verbindet und sämtliche Zugriffe auf diese bereitstellt ist die "DataAccessShopDatabase".

Bei einem Reset werden alle Tabelle wieder zurückgesetzt, mit Aufnahme von brand, article, coupon und picture.

ADMIN DATENBANK

28. ER-Modell:



29. Relationenmodell:

admin (username, password)

session (id, key, admin_username)

ranking (ip_address, sql_injection, blind_sql_injection, email_without_at, xss, profile_picture, html_comment_user, price_order_bug, guess_user_login, guess_coupon, delete_user, comment_xss, look_for_email_address, hash_user, security_misconfiguration)

30. Beschreibung

In dieser Datenbank befinden sich die Rankings, also welcher Angreifer welche Sicherheitslücken gefunden hat und die Zugangsdaten für den Admin, der das alles überwachen kann. Diese Tabellen werden extra in einer anderen Datenbank gespeichert, damit der Admin Zugang sicher ist und die die Angreifer nicht die Möglichkeit haben sich als Admin einzuloggen.

Die Klasse, welche diese Datenbank mit der Logik verbindet und sämtliche Zugriffe auf diese bereitstellt ist die "DataAccessAdminPanel", mit Ausnahme der Verschlüsselung der Passwörter, dafür wird die Methode der "DataAccessShopDatabase" genutzt.

Die Zugangsdaten für den Admin sind folgende:

Username	Passwort
admin	Xb935uFoLaC!82?Z

Loggt sich der Admin ein, so wird dieser auch gleichzeitig als normaler Nutzer angesehen, er kann also auch alle Sicherheitslücken testen. Die Zugangsdaten für den Nutzer sind:

Username	Passwort
biedermann	Xb935uFoLaC!82?Z

SCHWACHSTELLEN - Webshop

31. Erkennung der Schwachstellen

Wenn der Nutzer eine Anfrage an das Backend schickt, überprüft dieses diese auf Schwachstellen. Ist eine Schwachstelle in der Anfrage enthalten, wird in einer Tabelle in der Datenbank (siehe [Admin Datenbank](#)) die jeweilige Schwachstelle als gefunden markiert. Dabei steht eine

- 0 für "nicht gefunden"
- 1 für "gefunden und **nicht** vom Client abgerufen"
- 2 für "gefunden und vom Client abgerufen"

Im Client läuft ein Store, welcher alle 3 Sekunden im Backend den Endpunkt Flaws abfragt, ob er eine Schwachstelle gefunden und noch nicht abgefragt hat. Also ob in seiner Zeile in der Tabelle in einem Feld eine 1 eingetragen ist. Wenn dies der Fall ist, werden die Ergebnisse an den Client zurückgegeben.

32. Überprüfung der Schwachstellen

SQL Injection

Die Überprüfung auf die Sicherheitslücke SQL Injection befindet sich in der Klasse VulnerabilityCheck. Es gibt drei unterschiedliche Schwierigkeitsgrade für die SQL Injection, das jeweilige Level, welches der [Admin](#) einstellt, wird dabei verwendet. Löst man die Schwachstelle erfolgreich aus, bekommt man eine Email und ein gehashtes Passwort zurück.

Auf Stufe 1 wird nur überprüft, ob es sich um einen SQL Injection handelt, es wird vor der Überprüfung keine Veränderung der Eingabe durchgeführt. Geprüft wird auf ' OR 1=1;# und ' AND 1=1;#

Bei Stufe 2 wird wie bei Stufe 1 wieder auf ' OR 1=1;# und ' AND 1=1;# geprüft, jedoch werden vor der Überprüfung alle ' und # am Anfang und Ende des Strings entfernt.

Bei Stufe 3 gibt es 2 verschiedenen Möglichkeiten, zusätzlich muss eine Anfrage über ein API Testtool wie Postman kommen:

1. Es wird geprüft und verändert wie bei Stufe 2, zusätzlich werden aber noch alle „1=1“ aus dem String entfernt
2. Es wird ohne Veränderung auf ' UNION SELECT email, password FROM user;# geprüft

Blind SQL Injection

Bei der Blind SQL Injection wurde sich für eine zeitbasierte SQL Injection entschieden. Die Überprüfung befindet sich in der Klasse VulnerabilityCheck. Bei dieser gibt es keine verschiedenen Stufen oder Level. Geprüft wird auf „sleep (“, „benchmark (“, „; wait for delay“ und „; wait for time“.

Wird die Sicherheitslücke ausgelöst, wird eine 15 Sekunden Pause ausgelöst, wodurch der Angreifer seinen Erfolg bemerkt.

Cross Site Scripting (XSS)

Die Überprüfung auf Cross Site Scripting befindet sich in der Klasse VulnerabilityCheck. Hier gibt es drei unterschiedliche [Schwierigkeitsgrade](#).

Geprüft wird, um die Schwachstelle zu erkennen, ob die Eingabe <script> und </script> enthält.

Auf Stufe 1 wird lediglich geprüft. Es findet vorher keine Veränderung der Eingabe statt.

Bei Stufe 2 werden vor der Prüfung alle „script“ aus dem String entfernt. Danach wird geprüft.

In Stufe 3 werden zusätzlich zum Entfernen aller „script“ auch noch das erste „<“, „>“ und „\“ aus dem String entfernt und anschließend überprüft.

GuessUserLogin

Die Schwachstelle Hash User wird in der Klasse AuthenticationLogic beim Login überprüft. Um diese Schwachstelle auszulösen, muss man vorher eine erfolgreiche SQL Injection durchführen oder es per brute force herausfinden. Der Hash ist dabei ein SHA-1 Hash.

Bei dieser Schwachstelle gibt es wieder drei Stufen, die der Admin einstellen kann.

	E-Mail	Password
Stufe 1	Test1@test.de	123456789
Stufe 2	Test2@User.de	test123456789
Stufe 3	Test3@TestUser.de	0112358132134

E-Mail ohne @

Bei dieser Sicherheitslücke muss der Angreifer eine E-Mail ohne das @ Zeichen beim [Kontaktformular](#) eingeben. Überprüft wird die Sicherheitslücke in der ContactState Klasse

Profilbild

Bei dieser Sicherheitslücke muss der User eine Datei hochladen, welche keine der folgenden Endungen hat: png, jpg, jpeg, tiff, gif und bmp hat. Überprüft wird die Sicherheitslücke in der Klasse SecurityBreachDetection.

HTML Kommentar User

Um diese Schwachstelle auszulösen, müssen sich die User als derjenige Nutzer einloggen, dessen Zugangsdaten im HTML-Code als Kommentar verstecken. Überprüft wird das in der Klasse SecurityBreachDetection.

Preis manipulieren

Die Preis manipulieren Schwachstelle ist im [Bestellprozess](#) zu finden. Hier wird auf Level 1 die URL bereits mit dem entsprechenden Queryparameter "amount" und dem entsprechenden Wert angezeigt. Bei Level 2 wird der Queryparameter in der URL nicht mehr automatisch angezeigt, kann aber trotzdem in der URL hinzugefügt werden. Der Wert des Queryparameters kann beliebig verändert werden und wird mit dem Klick auf den Button "zahlungspflichtig bestellen" mit an das Backend gesendet. Im Backend wird überprüft, ob der berechnete Wert der gekauften Artikel der Gleiche wie der übergebene Wert ist. Wenn diese nicht übereinstimmen, wurde die Schwachstelle gefunden.

Coupon erraten

Bei dieser Sicherheitslücke wird überprüft, ob der Angreifer einen gängigen Rabattgutschein bei einer Bestellung eingibt. Die Überprüfung findet in der Klasse SecurityBreachDetection statt. Geprüft wird, ob die Eingabe folgende Wörter enthält: blackfriday, summersale, summer, ausverkauf, neujahr, newyear, winter, biedermann, biedisshop.

User löschen

Um diese Schwachstelle auszulösen muss der Angreifer eine Delete Anfrage über ein API Testtool machen auf /api/user/{id}. Überprüft wird die Schwachstelle in der UserState Klasse.

Kommentar Cross Site Scripting (XSS)

Bei dieser Sicherheitslücke muss der Angreifer einen Kommentar zu einem Artikel schreiben, welcher Cross Site Scripting enthält. Überprüft wird wie bei der Sicherheitslücke Cross Site Scripting (XSS)

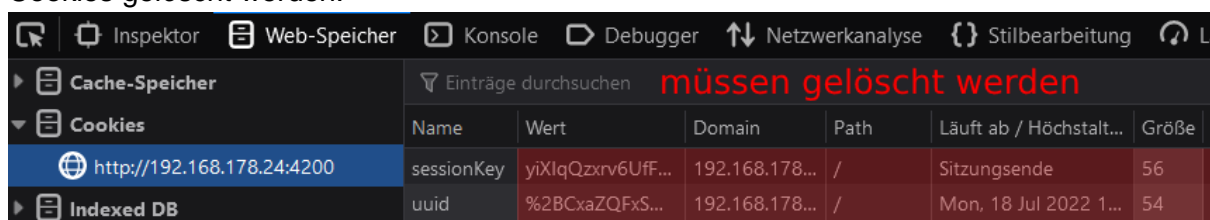
Link Finden

Bei dieser Sicherheitslücke muss der Angreifer mithilfe zusätzlicher Software einen versteckten Endpunkt finden. Ausgelöst wird die Sicherheitslücke sobald eine Anfrage an diesen gesendet wird.

BEKANNTE FEHLER

33. Probleme mit Uuid nach Serverneustart

Wenn Nutzer angemeldet sind und der Server währenddessen neu gestartet wird, ohne dass der Nutzer sich ausloggen konnte, schlagen alle nachfolgenden Anfragen an das Backend fehl. Sollte dies der Fall sein, muss im Client lediglich die Uuid und die Session Id aus den Cookies gelöscht werden.



	Name	Wert	Domain	Path	Läuft ab / Höchstalt...	Größe
http://192.168.178.24:4200	sessionKey	yiXlqQzrv6UfF...	192.168.178...	/	Sitzungsende	56
	uuid	%2BCxaZQFxS...	192.168.178...	/	Mon, 18 Jul 2022 1...	54

34. XSS wird zwei Mal in den Nutzereinstellungen angezeigt

Wenn man eine XSS-Schwachstelle gemäß VERLINKUNG in die Beschreibung in den Nutzereinstellungen eingibt, wird der darin enthaltene Javascript-Code anschließend und bei jedem Neuladen der Seite 2x statt 1x ausgeführt.

35. Datenbanken werden beschädigt wenn der Server während des Resets heruntergefahren wird

Beim Start des Servers werden am Anfang beide Datenbanken auf den initialzustand zurückgesetzt. Wird der Server während des Zurücksetzens heruntergefahren werden beide Datenbanken beschädigt, was dafür sorgen kann, dass sie nicht mehr nutzbar sind. Deswegen den Server erst herunterfahren sobald er vollständig hochgefahren ist. Dies dauert auf einem handelsüblichen Pc etwa 40 Sekunden.