Abschlussprüfung Sommer 2023

Fachinformatiker für Anwendungsentwicklung

Dokumentation zur betrieblichen Projektarbeit

MFA-Plugin

Das Multi-Faktor-Authentifizierung Plugin

Abgabedatum: [Ort], den 30.04.2023

Prüfungsbewerber:

Alexander Colga

[Adresse]

[PLZ Ort]

Ausbildungsbetrieb:

[Firma]

[Adresse]

[PLZ Ort]

Inhaltsverzeichnis

Inhaltsverzeichnis III
Abbildungsverzeichnis V
Tabellenverzeichnis VI
Verzeichnis der Listings VII
Abkürzungsverzeichnis VIII
1 Einleitung 1
1.1 Projektumfeld 1
1.2 Projektziel 1
1.3 Projektbegründung 1
1.4 Projektschnittstellen 2
1.5 Projektabgrenzung 2
2 Projektplanung 2
2.1 Projektphasen 2
2.2 Ressourcenplanung 3
2.3 Entwicklungsprozess 3
3 Analysephase 3
3.1 Ist-Analyse 3
3.2 Wirtschaftlichkeitsanalyse 4
3.2.1 Make or Buy-Entscheidung 4
3.2.2 Projektkosten 4
3.2.3 Amortisationsdauer 5
3.3 Anwendungsfälle 5
3.4 Lastenheft und Pflichtenheft 6
4 Entwurfsphase 6
4.1 Zielplattform 6
4.2 Entwurf der Benutzeroberfläche 7
4.3 Geschäftslogik 7
4.4 Maßnahmen zur Qualitätssicherung 7
5 Implementierungsphase 8
5.1 Implementierung der Datenstrukturen 8
5.2 Implementierung der Benutzeroberfläche 9
5.3 Implementierung der Geschäftslogik 10
6 Abnahmephase 12
7 Dokumentation 13
8 Fazit 14
8.1 Soll-/Ist-Vergleich 14

Inhaltsverzeichnis

8.2 Lessons Learned..... 14
8.3 Ausblick..... 14
Literaturverzeichnis..... 15
Eidesstattliche Erklärung..... 16
Anhang..... i
A1 Detaillierte Zeitplanung..... i
A2 Ressourcenplanung..... ii
A3 Anwendungsfalldiagramm..... iii
A4 Lastenheft (Auszug)..... iv
A5 Pflichtenheft für das MFA-Plugin..... v
A6 User-Interface Entwurf..... vi
A7 Endgeräte Infografik..... vi
A8 Modal Backup Codes Fertig..... vii
A9 Entity-Relationship-Modell..... vii
A10 Klassendiagramm (Auszug)..... viii
A11 Programmablaufplan Login..... ix
A12 Struktogramm Login..... ix
A13 MFA Modal..... x
A14 Class user_mfa..... x
A15 Screenshot User-Interface Fertig..... xii
A16 Screenshot Admin-Interface..... xv
A17 Screenshots Login..... xv
A18 Class MFA..... xviii
A19 VueJS Code..... xx
A20 Kommentare/Dokumentation des Quellcodes..... xxiv
A21 Mail Template..... xxiv
A22 Benutzerdokumentation..... xxv

Abbildungsverzeichnis

Abbildung 1: Anwendungsfalldiagramm.....iii
Abbildung 2: Erster Entwurf User-Interface.....vi
Abbildung 3: Endgeräte Infografik.....vi
Abbildung 4: Backup Code Modal.....vii
Abbildung 5: Entity-Relationship-Model.....vii
Abbildung 6: Klassendiagramm (Auszug).....viii
Abbildung 7: Programmablaufplan Login.....ix
Abbildung 8: Login Struktogramm.....ix
Abbildung 9: MFA-Sicherheitsrichtlinie.....x
Abbildung 10: User Interface Leer.....xii
Abbildung 11: User Interface Voll eingerichtet.....xiii
Abbildung 12: User Interface Bearbeitung Modus.....xiii
Abbildung 13: User Interface Einrichtung.....xiv
Abbildung 14: User Interface Einrichtung Fertig.....xiv
Abbildung 15: Screenshot Admin-Interface.....xv
Abbildung 16: Login Step1.....xv
Abbildung 17: Login TOTP.....xvi
Abbildung 18: Login E-Mail Step1.....xvi
Abbildung 19: Login E-Mail Stap2.....xvii
Abbildung 20: Login Backup Code.....xvii
Abbildung 21: Screenshot Benutzerdokumentation.....xxv

Tabellenverzeichnis

Tabelle 1: Grobe Zeitplanung.....3
Tabelle 2: Kostenaufstellung.....5
Tabelle 3: Detaillierte Zeitplanung.....ii

Verzeichnis der Listings

Listing 1: class user_mfa..... xii
Listing 2: Class MFA (Ausschnitt LoadUserMFA, CreateTOTP, validateTOTP).....xix
Listing 3: VueJS Ausschnitt.....xxiv
Listing 4: Beispiel Dokumentaion im Code.....xxiv
Listing 5: Mail Plain Template.....xxv
Listing 6: Mail HTML Template.....xxv

Abkürzungsverzeichnis

CRM.....*Customer-Relationship-Management*

CSS.....*Cascading Style Sheets*

ERM.....*Entity-Relationship-Model*

HOTP.....*HMAC-based One-time Password Algorithmus*

HTML.....*Hypertext Markup Language*

IDE.....*Integrierte Entwicklungsumgebung*

JS.....*Javascript*

MFA.....Multi-Faktor-Authentifizierung

OTP.....*One Time Password / Einmalpassword*

PHP.....*PHP: Hypertext Preprocessor*

PM.....*Plugin Manger*

TOTP.....*Time-based One-time Password Algorithmus*

1 Einleitung

1.1 Projektumfeld

Der Auftraggeber des Projekts ist die [Firma].
Mit vier Mitarbeitern entwickelt und vertreibt [Firma] erfolgreich Cloud Lösungen.
[Firma] wird in Ihrem Unternehmen aufgestellt und bietet Ihnen sicher die bequemliche Funktionalität moderner Cloud-Dienste.
Den Kern des Geschäfts stellt die [Produkt] dar, eine *CRM*-Lösung zur professionellen Kontaktverwaltung. Mit dem Ziel das klassische Adressbuch abzulösen. Dabei sollten die wichtigsten Funktionen eines *CRM*-Systems im Vordergrund stehen, ohne dass der tägliche Ablauf durch zu viele unnötige Features erschwert wird. [Produkt] ist also speziell für kleine und mittlere Unternehmen entworfen worden, die sich im Wachstum befinden. Höchst individualisierbar ist die Lösung in der Lage, sich an die Prozesse des Kunden anzupassen und den Kunden bei der Kundenbindung zu unterstützen.
Im Rahmen dieses Projekts wird der Prüfling vom 20.03.2023 bis zum 30.04.2023 an der Entwicklung einer webbasierten Anwendung für den internen Gebrauch teilnehmen. Das Ziel besteht darin, die Sicherheit des webbasierten CRM-Systems von [Firma] auf ein höheres Niveau im Bereich Cybersecurity zu bringen.

1.2 Projektziel

Das Ziel dieses Projekts ist es, eine Multi-Faktor-Authentifizierung (MFA) in das vorhandene CRM-System der [Firma] zu integrieren und für Kunden verfügbar zu machen. Durch die Einführung eines zusätzlichen Sicherheitselements zum Login-Prozess soll die Sicherheit der Kundendaten erhöht werden. Die MFA soll als Lösung und Verstärkung zur veralteten Variante des Login-Prozesses durch Benutzername und Passwort dienen. Sie wird im Frontend des CRM-Systems integriert.

Es werden zwei Varianten der MFA entwickelt: Variante 1 ist der Einmalcode (Google Authenticator), Variante 2 ist das E-Mail-Verfahren.

1.3 Projektbegründung

Es gibt einen wichtigen Faktor, die für die Integration dieses Projekts eine Rolle spielt. Es geht um die Erhöhung der Sicherheit des *CRM-Systems*. In einen *CRM-System* stehen oft viele Datenschutz Geschütze Daten, wie z.B. Kunden Daten, etc. Dies kann zu großem Interesse an Dritte anlocken. Eine 2-Faktor-Authentifizierung kann verhindern, dass Dritte Zugriff auf das

System erhalten, selbst wenn ein Passwort bekannt ist. Ein erfolgreicher Angriff auf ein CRM-System kann schwerwiegende Folgen haben, wie den Diebstahl von Daten, Betriebsunterbrechungen, den Verlust von Kunden und Reputation sowie finanzielle Verluste.

1.4 Projektschnittstellen

Das Projekt wird als ein Plugin entwickelt. Es arbeitet in Verbindung mit dem davor Entwickelten Plugin Manger (PM) des *CRM-Systems*. Das Projekt wird im Unternehmen [Firma], in [PLZ Ort], durchgeführt. Für die gesamte Dauer wird ein bereits vorhandener Arbeitsplatz zur Verfügung gestellt. Die Benutzer der Anwendung sind die Kunden der Firma [Firma]. Die Abnahme und Richtigkeit des Projektes werden von der Geschäftsführung und den Entwickler geprüft.

1.5 Projektabgrenzung

Die [Produkt] ist voll-modular aufgebaut. Im Rahmen dieser Modularität existieren bereit Features. Somit existiert für jedes Modul, das angeboten wird, ein eigenes Feature. Diese Features sind in der Datenbank gespeichert. Diese Logik ist bereits vorhanden außerdem ist das Plugin System ist schon vorhanden und wird hierfür lediglich genutzt

2 Projektplanung

In diesem Kapitel werden die zur Verfügung stehende Zeit sowie die Ressourcen des Projektes geplant.

2.1 Projektphasen

Bevor mit der Umsetzung des Projektes begonnen wird, muss ein Projektplan erstellt werden (siehe *Tabelle 1: Grobe Zeitplanung*). Es werden die Hauptphasen des Projektes und die Dauer in Stunden aufgelistet. Eine genauere Zeitplanung inkl. Unteraufgaben ist im Anhang *A1*: *Detaillierte Zeitplanung* Seite: *1* dargestellt

Projektphase	Geplante Zeit
Analyse	8 h
Entwurf	13 h
Implementierung	40 h
Abnahmephase	1 h
Dokumentation	10 h
Projektbewertung	1 h
Gesamt	**73 h**

Tabelle 1: Grobe Zeitplanung

2.2 Ressourcenplanung

Im Folgenden werden alle für die Umsetzung des Projektes benötigten Ressourcen, d.h. Hardware- und Softwarekomponenten sowie benötigtes Personal betrachtet. Es wurde drauf geachtet, dass bei der Umsetzung des Projektes nur Software verwendet wird, die keine Anschaffung von Lizenzen nach sich zieht. Dadurch konnten die Projektkosten reduziert werden.
Die Ressourcenplanung ist im Anhang *A2* Seite *2* dargestellt.
Die komplette Entwicklung wurde an einem Arbeitsplatzrechner mit dem Betriebssystem OSX Mojave durchgeführt. Dieses Projekt kann jedoch unter nahezu allen gängigen Computersystemen realisiert werden, da für die Entwicklungs- und Testumgebung nur ein gängiger Apache, PHP und MYSQL benötigt werden, das von allen gängigen Betriebssystemen unterstützt wird. Zur Erstellung des Quellcodes reicht ein standardmäßiger Editor. In diesem Fall wird jedoch die Eclipse IDE verwendet.
Die Projektdokumentation erfolgte unter Verwendung des Office 365 Pakets, welches durch das [Bildungsträger] gestellt wurde.

2.3 Entwicklungsprozess

Vor Beginn des Projekts wurde ein Entwicklungsprozess festgelegt, der als Leitfaden für die Umsetzung diente. Für die Entwicklung des MFA-Plugins wurde das Wasserfallmodell als geeignetste Methode aufgrund des Umfangs des Projekts gewählt. Da das Projekt von einer Einzelperson, dem Prüfling, umgesetzt wurde, wurde zusätzlich ein iteratives Softwareentwicklungsmodell eingesetzt. Dadurch konnte der Entwicklungsprozess flexibler gestaltet werden, um auf Veränderungen und Anpassungen schnell reagieren zu können.
Eine grobe Zeitplanung ist in *Tabelle 1: Grobe Zeitplanung* aufgeführt, detaillierte Informationen finden sich im Anhang *A1*: *Detaillierte Zeitplanung* Seite *1* zu finden.

3 Analysephase

Nachdem die Planung des Projektes abgeschlossen ist, kann mit der Analysephase zur Ermittlung des Ist-Zustandes begonnen werden.

3.1 Ist-Analyse

Um das genaue Anforderungsprofil für das Projekt zu ermitteln, wurden verschiedene Interviews und Gespräche mit den Stakeholdern durchgeführt, darunter auch die Mitarbeiter der Firma [Firma].
Dabei wurde festgestellt, dass es bei der [Produkt] einen Bedarf an einem sichereren Authentifizierungssystem gibt, um die Kundendaten besser zu schützen.

Die Firma [Firma] stellt derzeit unter anderem folgende PHP-Anwendungen bereit: (1) [Produkt], (2) Plugin Manger.
(1) Die [Produkt] beinhaltet ein einfaches Login System. Sie können sich mit einem einfachen Passwort und den Usernamen anmelden. Eine weitere Authentifizierung erfolgt nicht.
(2) Des Weiteren beinhaltet Sie auch ein Plugin Manger. Der dient dazu einfachere Implementierung neuer Erweiterungen. So kann jeder Kunde entscheiden was seine [Produkt] kann und ist somit voll individualisierbar.
Eine weitere Anforderung war, dass das Plugin nahtlos in den vorhandenen Plugin Manager integriert werden muss, um eine reibungslose Erweiterung der [Produkt] zu gewährleisten. Außerdem wurden auch technische Anforderungen wie die Kompatibilität mit verschiedenen Betriebssystemen und die Unterstützung von MYSQL ermittelt.
Die Analysephase endete mit einer Zusammenfassung der Anforderungen und einer detaillierten Spezifikation der Funktionen und Schnittstellen, die im Rahmen des Projekts implementiert werden sollen. Siehe Anhang *A4: Lastenheft (Auszug)* Seite *4*.

3.2 Wirtschaftlichkeitsanalyse

Durch eine Betrachtung der Projektkosten soll in diesem Kapitel untersucht werden, ob das Projekt auch wirtschaftlich gerechtfertigt ist.

3.2.1 Make or Buy-Entscheidung

Da es sich bei dem Projekt um eine sehr unternehmensspezifische Anforderung der [Firma] handelt und verschiedenste Komponenten der [Produkt] mit der Anwendung agieren müssen, lässt sich auf dem Markt keine richtige Lösung finden, die diese Wünsche erfüllen könnte. Daher soll das Projekt in Eigenentwicklung durchgeführt werden.

3.2.2 Projektkosten

Die Projektkosten, die während der Entwicklung des Projektes anfallen, sollen im Folgenden kalkuliert werden. Dafür müssen neben den Personalkosten, die durch die Realisierung des Projektes verursacht werden, auch noch die Aufwendungen für Ressourcen (Hard- und Software, Büroarbeitsplatz etc.) berücksichtig werden. Da die genauen Personalkosten nicht herausgegeben werden dürfen, wird die Kalkulation anhand von Stundensätzen durchgeführt, die von Geschäftsführer festgelegt wurden.

Der Stundensatz eines Auszubildenden beträgt demzufolge 7,20 €, der eines Mitarbeiter 22,15 €. Für die Ressourcennutzung wurde ein pauschaler Stundensatz von 15 € angenommen.

Die kosten, die für die einzelnen Vorgänge des Projektes anfallen, sowie die gesamten Projektkosten lassen sich der *Tabelle 2: Kostenaufstellung* entnehmen.

Vorgang	Zeit	Kosten / Stunde	Kosten
Analyse	8 h	*7,20 € + 15 € = 22,20 €*	*177,60 €*
Entwicklung	55 h	*7,20 € + 15 € = 22,20 €*	*1221,00 €*
Fachgespräch	2 h	*22,15 € + 7,20 € = 29,35 €*	*58,70 €*
Abnahme	1 h	*22,15 € + 7,20 € = 29,35 €*	*29,35 €*
		Gesamt	***1.486,65 €***

Tabelle 2: Kostenaufstellung

3.2.3 Amortisationsdauer

Die Wirtschaftlichkeit dieses Projektes lässt sich nicht errechnen, da kein sofortiger Nutzen vorhanden ist. Es handelt sich hierbei um ein Projekt, das die Minimierung des Sicherheitsrisikos durchführt, aus diesem Grund werden die im vorherigen Abschnitt errechneten Projektkosten nicht durch einen Kundenauftrag gedeckt. Bekannt ist jedoch, dass der Invest des Projektes sich indirekt über die generierten Projektumsätze amortisiert, die sich mit den neu gewonnenen Kunden ergeben. Der Nutzen kann zu einem späteren Zeitpunkt, nach evtl. drei bis sechs Monaten, statisch ermittelt werden.

3.3 Anwendungsfälle

Es wird im Zuge der Analyse des Projektes ein Anwendungsfalldiagramm erstellt. Dies stellt Interaktionen von Benutzern mit dem System dar und zeugt somit das erwartete Verhalten der Anwendung. Das Anwendungsfalldiagramm ist im Anhang *Abbildung 1: Anwendungsfalldiagramm* Seite *3* dargestellt.

Es gibt die Akutere Administrator und User. Die Administratoren sind spezialisierte Akteure des Users und somit an den gleichen Use-Case-Abläufen beteiligt wie der User. Der User wird sich für die [Produkt] an- und abmelden können. Bevor er sich erfolgreich anmelden kann, muss er sich zunächst Authentifizieren durch einen einmaligen Code. Außerdem kann der User seine Authentifizierung Methode aussuchen. Entweder über eine App oder über die E-Mail. Der Administrator kann zudem eine Methode als benötigt markieren. Der Administrator kann auch von anderen Benutzern die Authentifizierung Methode zurücksetzten, falls dies nötig ist.

3.4 Lastenheft und Pflichtenheft

Die Analysephase wird mit dem Lastenheft abgeschlossen. Es hält alle Anforderungen der [Firma] an das Projekt fest. Ein Auszug aus dem Lastenheft ist im Anhang *A4 Lastenheft (Auszug)* Seite *4* dargestellt.

Die einzelnen Anforderungen des Lastenheftes sind User-Stories (Anwendererzählungen), d.h. sie sind in alltäglicher Sprache formuliert und prägnant. User-Stories sind immer nach dem folgenden Schema aufgebaut: „Als Rolle möchte ich Anforderung, um Begründung. . . “ Außerdem sind sie mit der MoSCoW-Methode (Must, Should, Could, Won’t have) formuliert worden. Mit Hilfe dieser Methode können die User-Stories priorisiert werden.

Must: „Als Administrator muss ich. . . “ –> Anforderung muss umgesetzt werden.

Should: „Als Administrator sollte ich. . . “ –> Anforderung sollte umgesetzt werden, wenn die Must-Anforderungen nicht beeinträchtigt werden.

Could: „Als Administrator möchte ich. . . “ –> Anforderungen können umgesetzt werden, wenn die Must- und Should-Anforderungen nicht beeinträchtigt werden und Ressourcen frei sind.

Won’t: Anforderungen werden in diesem Projekt nicht umgesetzt oder werden nur für die Zukunft vorgemerkt

In der Planungsphase wurde auf Basis des Lastenhefts das Pflichtenheft erstellt, welches die genauen technischen Spezifikationen und Anforderungen an das Projekt festhält. Siehe Anhang *A5: Pflichtenheft für das MFA-Plugin* Seite *5*.

4 Entwurfsphase

Anschließend kann mit der Entwurfsphase begonnen werden.

4.1 Zielplattform

Wie bereits erwähnt, handelt es sich um eine Webanwendung/Erweiterung, auf die alle Benutzer über das Internet zugreifen können. Zur Erstellung der verschiedenen Faktoren wird die Spomky-Labs TOTP/HOTP Library in PHP verwendet und die QRCODE.JS Library, um QR-Codes/Bilder zu generieren. Das Webframework VueJS und der PHP-Mailer werden ebenfalls verwendet, wobei der Webserver der [Firma] die Anwendungsumgebung darstellt. Die Entwicklung wird in den Sprachen HTML, PHP, CSS, JS und JQuery durchgeführt und soll auf allen gängigen Endgeräten wie Desktop-PCs, Tablets und Smartphones funktionieren. Es ist auch geplant, die Anwendung auf mobilen Geräten zu testen, um sicherzustellen, dass sie einwandfrei funktioniert. Des Weiteren soll

auf diesem Wege der Kunden von der [Firma] als professionelles Dienstleistungsunternehmen überzeugt werden.

4.2 Entwurf der Benutzeroberfläche

Der erste Meilenstein war die Erstellung des User-Interfaces, siehe *Abbildung 2: Erster Entwurf User-Interface* Seite *6*. Es war dabei sehr wichtig, die Seite für verscheiden Endgeräte anzupassen, das so genannte Responsive Design. Im Fokus standen die Desktop-Ansicht und die Smartphone-Ansicht. Die meisten Menschen surfen nämlich mobil, *Abbildung 3: Endgeräte Infografik* Seite *6*. Basierend auf das User-Interfaces wurde noch ein Modal erstellt das für die Methode Backup Codes die Codes anzeigt, siehe *Abbildung 4: Backup Code Modal* Seite *7*. Das Admin Interface sowie die Mail werden im Laufe der Implementierungsphase entwickelt und benötigen vorher kein Entwurf.

Um einen groben Überblick über alle relevanten Daten zu erhalten für das MFA, wurde zunächst analysiert was benötigt wird. Dabei wurde hinterfragt, welche Daten gespeichert werden müssen und in welcher Beziehung sie zueinanderstehen.

Auf Grundlage der herausgearbeiteten Entitätstypen wurde ein Entity-Relationship-Model (ERM) erstellt, welches die Typen und ihre Beziehungen graphisch veranschaulichen soll. Dieses befindet sich im *Abbildung 5: Entity-Relationship-Model* auf Seite *7*. Folgende Entitätstypen wurden herausgearbeitet und abgebildet: **user_uid, backup, secret, email**

4.3 Geschäftslogik

Aufgrund der testgetriebenen Entwicklung ist es nicht möglich, vor Beginn der Implementierung der Anwendung ein Klassendiagramm mit den Klassen, mit denen die Geschäftslogik umgesetzt werden soll, zu entwerfen. Für eine nachträgliche Dokumentation der Geschäftslogik wurde ein Klassendiagramm aus dem Programmcode heraus generiert, dass sich als Ausschnitt im *Abbildung 6: Klassendiagramm (Auszug)* auf Seite *8* befindet. Des Weiteren wurde ein Programmablaufplan erstellt, um eine grobe Vorgabe zu haben, wie das Projekt abläuft, siehe Beispiel Login: *Abbildung 7: Programmablaufplan Login* auf Seite *9*.

4.4 Maßnahmen zur Qualitätssicherung

Im Rahmen des Projekts werden Maßnahmen zur Qualitätssicherung umgesetzt, um sicherzustellen, dass die entwickelte Software den Anforderungen entspricht und fehlerfrei funktioniert. Ein wichtiger Aspekt ist das parallele Entwickeln und Testen. Dies bedeutet, dass

während der Entwicklungsphase jeder neue Code kontinuierlich auf seine Funktionalität und Kompatibilität getestet wird. So können potenzielle Fehler frühzeitig erkannt und behoben werden.

Zur Qualitätssicherung werden auch Funktionstests durchgeführt. Hierbei wird jede einzelne Funktion der Software auf ihre Funktionalität geprüft, um sicherzustellen, dass sie wie erwartet funktioniert. Auch das De-buggen von Code ist ein wichtiger Teil der Qualitätssicherung. Hierbei werden einzelne Variablen getestet, um sicherzustellen, dass sie wie erwartet funktionieren.

Um die Bedienbarkeit der Software zu testen, werden Usability-Tests durchgeführt. Hierbei werden verschiedene Anwendungsszenarien simuliert, um sicherzustellen, dass die Benutzeroberfläche intuitiv und einfach zu bedienen ist. Dies ist wichtig, um sicherzustellen, dass die Benutzer die Software ohne Probleme nutzen können.

Zusätzlich werden nach Abschluss des Projekts verschiedene Nutzertests durchgeführt, um sicherzustellen, dass die entwickelte Software den Anforderungen entspricht und die gewünschten Funktionen enthält. Diese Tests werden nach dem Go-Live der Software durchgeführt, um sicherzustellen, dass die Software unter realen Bedingungen fehlerfrei funktioniert und die Benutzer zufrieden sind.

Insgesamt stellen die Maßnahmen zur Qualitätssicherung sicher, dass die entwickelte Software den Anforderungen entspricht und fehlerfrei funktioniert. Dadurch wird sichergestellt, dass die Kunden der [Firma] eine hochwertige und zuverlässige Software erhalten, die ihnen dabei hilft, ihre Daten und Geschäftsprozesse zu schützen.

5 Implementierungsphase

Als nächstes folgt nun die Implementierung des Plugins.

5.1 Implementierung der Datenstrukturen

Bei der Implementierung der Datenbank, die die Informationen für das Plugin halten, wurde sich an dem ERM (siehe *Abbildung 5: Entity-Relationship-Model* Seite *7*) orientiert. Jedoch mussten kleine Anpassungen gemacht werden, da die Anwendung an eine bereits bestehende Datenbank angeschlossen werden musste.
Daraus hat sich ergeben, dass bestimmte Felder Pflicht sind: **uid, tstamp, crdate, cruse**r. Das Ganze wurde dann in ein Datenbank Schema geschrieben. Daraus hat sich die Klasse user_mfa

ergeben die beinhaltet die Funktion für die Datenbank. Ein Ausblick ist im Anhang auf *A14 Class user_mfa* Seite *10* zu sehen.

5.2 Implementierung der Benutzeroberfläche

Die Benutzeroberfläche wurde schon in der Entwurfsphase implementiert. Das User-Interface wurde an das bestehende Design angepasst und mit kleinen Cascading Style Sheets (CSS)-Dateien fehlende Elemente hinzugefügt. Genauer erklärt im Abschnitt *5.3.* Weiter wird der Entwurf mit VueJS erweitert, um eine Datenverarbeitung zu ermöglichen. Daraus resultiert Elemente des Interface geändert werden können, ohne dass die Website neu geladen werden muss.
Der Admin Bereich wurde dementsprechend auch an das vorhandene Design angepasst und mit VueJS erweitert. Das VueJS ermöglich es ein Template ohne Daten zu Laden und nachträglich diese mit Daten zu füllen. Einfach gesagt: Erst wird die Struktur geladen dann die Informationen. (*Abbildung 15: Screenshot Admin-Interface* Seite *15*).

5.3 Implementierung der Geschäftslogik

Da die Implementierung der Geschäftslogik den Kernbestandteil des gesamten Projektes darstellt, soll diese im folgenden Abschnitt genauer erläutert werden. Um die Implementierung möglichst komfortabel durchführen zu können, ist eine geeignete Entwicklungsumgebung von großer Bedeutung. Der Prüfling hat dafür Eclipse verwendet. Eclipse bietet eine Vielzahl von Funktionen und Tools, die die Entwicklung von Softwareprojekten effizienter und einfacher machen. Durch die Verwendung von Eclipse konnte der Prüfling effektiver arbeiten, die Code-Qualität verbessern und die Entwicklungszeit verkürzen. Somit hat Eclipse dazu beigetragen, dass das Projekt erfolgreich umgesetzt werden konnte.

Die Spomky-Labs TOTP / HOTP Library wurde ausgewählt, da ein Programmieren dieser Funktionen in Eigenarbeit den zeitlichen Rahmen des Projektes sprengen würde. Die Spomky-Labs TOTP / HOTP Library ist eine PHP-Bibliothek zur Generierung von Einmalpasswörtern nach RFC 4226 (HOTP-Algorithmen) und RFC 6238 (TOTP-Algorithmen). Diese Bibliothek ist mit Google Authenticator-Apps kompatibel, die für Android und iPhone verfügbar sind. Sie ist darüber hinaus auch mit anderen Anwendungen wie z.B. FreeOTP kompatibel.
Der Time-based One-time Password Algorithmus (TOTP) liefert ein zeitbasiertes einmaliges Passwort. Es ist nur wenige Sekunden (Periode) gültig. Daher muss sichergestellt sein, dass die Uhr des Servers und die Uhr des Gerätes synchronisiert sind. Andernfalls kann es zu Problemen führen, wie z.B., dass das zeitbasierte einmalige Passwort (OTP) bereits auf dem Server abgelaufen und damit ungültig ist.
Die QRCODE.JS Library dient, wie der Name vermuten lässt, dazu, ein Bild in Form eines QR-Codes zu generieren. Da das CRM-System mit VueJS arbeitet, wird dieses Framework ebenfalls verwendet. Es dient dazu, Daten bzw. Informationen serverseitig nachzuladen oder nur solche Inhalte anzuzeigen bzw. zu laden, die auch wirklich verwendet werden. VueJS ist so konzipiert, dass beim Build nicht verwendete Komponenten ausgeschlossen werden. Das bedeutet, dass das ausgelieferte Endprodukt nur die Teile des Frameworks enthält, die wirklich verwendet werden. Die Anwendung ist nicht unnötig aufgeplustert.
Zu guter Letzt wird der bereits vorhanden PHP-Mailer verwendet. Er liefert eine Mail an die genannte E-Mail aus.

Zu Beginn der eigentlichen Implementierung wurde das Grundgerüst des Plugins erstellt. Das Grundgerüst eines Plugins besteht aus einem Ordner in dem Folgende Dateien vorhanden sein müssen: Plugin.php – Dort wird das Plugin initialisiert, Plugin.json – Informationen über das Plugin. Als nächstes wurden alle benötigten Bibliotheken hinzugefügt ggf. überprüft, ob sie schon

vorhanden sind. Das eben erwähnte Spomky-Labs TOTP / HOTP und die QRCodeJS wurden somit hinzugefügt.

Es folgt nun die Erstellung des Ordners PHP und die Datei mfa.php – Class Datei. Darauf folgt die Erstellung der Hauptklasse für das Plugin: MFA, siehe Anhang *A18: Class MFA* Seite *18*. Diese Klasse beinhaltet das Laden und Speichern der benötigten User Informationen, das Erstellen einen TOTP hash (Geheimer Code), das Überprüfen des einmaligen Passwords, das Erstellen und Senden der Mail und die Funktion, um Backup Codes zu erstellen.

Das Ganze wird dann mit Hilfe des VueJS in das Template verarbeitet. Dies wurde in einen Ordner Namens JS in die Datei firma_mfa.js gespeichert (Anhang *A19: VueJS Code* Seite *20*). Im Anhang des VueJS ist nicht der Komplette Code drin da das diese Dokumentation sprengen würde, deshalb habe ich einen Ausschnitt der VueJS Codes hinzugefügt Im VueJS gibt es dann noch die Funktionen zum Erstellen eines QR-Codes sowie die Funktionen die mit Hilfe von AJAX die PHP-Funktionen Aufrufen.

Dadurch kann man im User-Interface sich MFA-Methoden aussuchen und schon einrichten. Zeitgleich werden die Ganzen Funktionen auf Fehler überprüft und mögliche falsch eingaben werden in den Funktionen überprüft und verhindert. Um alle Informationen zu speichern die der User vorgenommen hat, muss der User sein Passwort bestätigen.
Währenddessen würde auch das Template der Mail erstellt. Diese wurde in einen weiteren Unterordner Templates erstellt. Es wurde einmal eine HTML und eine Plain Mail Template erstellt (Siehe *A21: Mail Template* Seite *24*).

Da das ganze User-Interface ein paar Design Anpassungen benötigt und sie noch nicht von der [Produkt] vorgesehen sind musste noch eine CSS Datei erstellt werden. Dort wurden die Checkboxen als ein Switch definiert, sowie die OTP-Eingabe Boxen.
Nachdem alles so weit funktioniert wird das Plugin an die Übersetzung Funktion des CRM angepasst. Das Plugin ist dann in 2 Sprachen verfügbar: English und Deutsch. Dafür wurde ein Ordner erstellt l10n dort sind die Übersetzungen hinterlegt (de.json und en.json).

Zu guter Letzt wurde der Login angepasst. Als erstes wurden für das Login Programmablaufplan (*Abbildung 7: Programmablaufplan Login* Seite *9*) und ein Struktogramm (*Abbildung 8: Login Struktogramm* Seite *9*) erstellt, um es einfacher zu verstehen. Als nächster Schritt wurde das Login implementiert. Es wurde mit Hilfe einer Dynamischen Vue Komponente per Hook Script platziert.

Das Plugin erweitert das Login durch Setzen von verschiedenen Parametern. Diese bilden dynamisch das Login Interface. Dort werden die Einzelnen Methoden, wenn aktiv in ein Array geschrieben. Das VUE schickt das abgesendet Form immer an das Login zurück. Dort wird geprüft, ob die Methode ausgefüllt ist und wenn ja, ob sie richtig ist. Bei Erfolg alle Aktiven Methoden wird der User eingeloggt.

Da jetzt alle Methoden Funktionieren wird zuletzt noch die Backup Code variante erstellt. Die Prüft, ob der User den Backup Code hat. Wenn ja dann löscht er ihn aus der Datenbank. So kann gewährleistet werden das Jeder Code nur einmal genutzt werden kann.
Als aller Letzte Stelle wird die Letzte Funktion geschrieben, wenn das MFA eine Muss Funktion ist. Das soll den User beim Einloggen sagen, das eine MFA-Methode gesetzt werden muss. Es wird beim Einloggen geprüft ob der Admin des Systems die Variable „aktivierung durch Benutzer erzwingen“ gesetzt hat. Wenn ja wird dem User beim Einloggen eine Meldung gezeigt, siehe *Abbildung 9: MFA-Sicherheitsrichtlinie* Seite *10*. Und wird zur Konfiguration weitergeleitet. Es wird auch verhindert das der User sich ohne eine Methode auf andere Seiten der [Produkt] aufhalten kann und wird immer auf die Konfiguration weitergeleitet. Wenn eine Methode aktiv ist funktioniert alles wie gewohnt für den User.

Zu guter Letzt wird alles nochmals auf Funktionalität und Rechtschreibfehler geprüft. Wir gehen einmal Step by Step durch was jetzt alles gemacht worden ist:

1. Implementierung in Plugin Manager *Abbildung 15: Screenshot Admin-Interface* Seite *15*
2. Implementierung in Benutzerkonfiguration
    a. Nicht konfiguriert *Abbildung 10: User Interface Leer* Seite *12*
    b. Voll konfiguriert *Abbildung 11: User Interface Voll eingerichtet* Seite *13*
    c. Einrichten Step1 *Abbildung 13: User Interface Einrichtung* Seite *14*
    d. Einrichten Fertig *Abbildung 14: User Interface Einrichtung Fertig* Seite *14*
    e. Backup Codes *Abbildung 4: Backup Code Modal* Seite *7*
3. Implementierung ins Login
    a. Login Normal *Abbildung 16: Login Step1* Seite *15*
    b. Login TOTP *Abbildung 17: Login TOTP* Seite *16*
    c. Login E-Mail Step 1 *Abbildung 18: Login E-Mail Step1* Seite *16*
    d. Login E-Mail Step 2 *Abbildung 19: Login E-Mail Stap2* Seite *17*
    e. Login via Backupcode Abbildung 20: Login Backup Code Seite *17*

6 Abnahmephase

Wie bereits unter Punkt *4.4* (*Maßnahmen zur Qualitätssicherung*) erwähnt, wurde die Anwendung testgetrieben entwickelt. Während der Implementierungsphase wurden kontinuierlich Tests geschrieben, die sich in verschiedene Kategorien einteilen lassen. Während der Entwicklung wurden die Funktionen des Codes mit verschiedenen Website-URLs getestet, was als White-Box-

Tests bezeichnet wird. Dadurch konnten viele Fehlerquellen bereits während der Programmierung erkannt und behoben werden, was Zeit in der Testphase gespart hat.

Bei den Oberflächen-Tests wurde schließlich die Interaktion des Benutzers mit der Anwendung simuliert und auf mögliche Fehler getestet.

Die Projektabnahme erfolgte am 30.04.2023, dem letzten Tag der Projektarbeit des Prüflings durch den Geschäftsführer [Name] vorgenommen.
Das Go-Live war nicht Bestandteil dieses Projekts. Aus diesem Grund wurde das Testen in einer Firmeninternen Umgebung, zusammen mit dem Geschäftsführer [Name], durchgeführt. Dem Geschäftsführer wurden alle Funktionalitäten an einer echten Website demonstriert. Die Funktionalität des Plugins wurde überprüft. Es fand eine kurze Einsicht in den Quellcode statt. Alle weiteren Fehlerbehebungen und Ergänzungen werden durch die [Firma] vorgenommen. Genauere Prüfungen des Quellcodes waren zu diesem Zeitpunkt nicht notwendig, da Fehleranfällige Codeabschnitte bereits im Prototyping verbessert wurden.

Schlussendlich wurde das IHK-Projekt laut dem Geschäftsführer erfolgreich abgeschlossen und das Plugin wird nun durch die [Firma] moderiert.

7 Dokumentation

Für dieses Projekt wurde, wie in den Anforderungen *Technische Ziele* auf S*. 4*, eine Entwicklerdokumentation im Rahmen der Projektarbeit erstellt. Es handelt sich hier um eine Entwicklerdokumentation im Rahmen von Quellcode Kommentaren. (Anhang *A20: Kommentare/Dokumentation des Quellcodes* Seite *24*)

Ferner wurde eine Projektdokumentation, die von Ihnen gerade gelesen wird, angefertigt, die Kernpunkte wie Projektplanung, Analyse, Entwürfe und weitere wichtige Inhalte enthält.
Eine Benutzerdokumentation wurde innerhalb dieses Projektes erstellt (Anhang *A22: Benutzerdokumentation* Seite: *25*). Das Projekt kann auch ohne Benutzerdokumentation verstanden werden, da vom Benutzer lediglich einfache und selbsterklärende Interaktionen gefordert sind, um eine MFA-Methode einzurichten.

8 Fazit

8.1 Soll-/Ist-Vergleich

Im Rückblick betrachtet sind die Anforderungen des Projektes im vollen Umfang erfüllt worden. Das Plugin bietet den Benutzern der [Produkt] die Möglichkeit, sich zu authentifizieren. Außerdem können Administratoren das Plugin Konfigurieren und Andere Benutzer verwalten. Der zu Beginn erstellte Projektplan (siehe Kapitel *2*) konnte zeitlich eingehalten werden.

Quantitative Ziele: Stellen Sie sicher, dass alle quantitativen Ziele aus dem Projektplan bewertet und mit den tatsächlich erreichten Ergebnissen verglichen wurden. Dazu können Sie beispielsweise die geplanten und tatsächlichen Fertigstellungsdaten für wichtige Meilensteine oder die geplante und tatsächliche Anzahl der Funktionen, die implementiert wurden, vergleichen.

Qualitative Ziele: Überprüfen Sie auch, ob alle qualitativen Ziele, die im Projektplan definiert wurden, erreicht wurden. Hierzu können Sie beispielsweise Feedback von Benutzern oder anderen Interessengruppen einholen, um festzustellen, ob das Plugin ihre Erwartungen erfüllt hat.

8.2 Lessons Learned

Im Rahmen des Projekts konnte der Prüfling zahlreiche und wertvolle Erfahrungen in Bezug auf die Durchführung eines vollständigen Projekts sammeln. Die wichtigste Erkenntnis war, dass Kommunikation eine sehr große Rolle dabei spielt, um ein Projekt erfolgreich durchzuführen z.B. bei einer plötzlichen Veränderung der Vorstellungen. Außerdem wurde dem Prüfling klar, dass bei äußerst guter Planung und Analyse die anschließende Realisierung einfacher fällt, als wenn direkt mit der richtigen Programmierung begonnen wird. Ferner ist positiv zu werten, dass umfangreiche fachliche Kompetenzen erworben wurden, wie z.B. der Umgang mit verschiedenen Frameworks VueJS, Bootstrap und die Findung eigenständiger Problemlösungen.

8.3 Ausblick

Für die Erweiterung des entwickelten Plugins bestehen unbegrenzte Möglichkeiten, denn das Internet schreitet immer weiter voran. Es wird immer wieder neue Authentifizierung Möglichkeiten geben. Folgende Erweiterungen können im weiteren Verlauf eingebaut werden: Weiter Methoden wie z.B. YubiKey, digitales Zertifikat, Biometrie…. Die Modularität des Plugins ermöglicht somit eine gute Wartbarkeit und Erweiterbarkeit.
Vorrangig und an erster Stelle steht jedoch das Go-Live der Website. Das Projekt soll zum nächstmöglichen Zeitpunkt online gehen.

Literaturverzeichnis

Anon., 2023. *PHP: Hypertext Preprocessor.* [Online]
Available at: http://www,php.net/

Anon, 2023. *VueJS.* [Online]
Available at: https://vuejs.org/

[Firma], 2023. [Online]
Available at: [URL entfernt]
[Zugriff am 2023].

JGraph Ltd, A. H., 2023. *Draw.io.* [Online]
Available at: https://app.diagrams.net/

Macke, S., 2023. *. Perfekte Projektdokumentation.* [Online]
Available at: http://dieperfekteprojektdokumentation.de/

Shimjs, D., 2023. *Cross-browser QRCode generator for javascript.* [Online]
Available at: https://davidshimjs.github.io/qrcodejs/

Spomky-Labs, 2023. *TOTP / HOTP library in PHP.* [Online]
Available at: https://github.com/Spomky-Labs/otphp

Eidesstattliche Erklärung

Ich, Alexander Colga, versichere hiermit, dass ich meine Dokumentation zur betrieblichen Projektarbeit mit dem Thema

MFA-Plugin – Das Multi-Faktor-Authentifizierung Plugin

selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, wobei ich alle wörtlichen und sinngemäßen Zitate als solche gekennzeichnet habe. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

[Ort], den 28.04.2023

_____

ALEXANDER COLGA

Anhang

A1 Detaillierte Zeitplanung

Analysephase			**8 h**
1. Analyse des Ist-Zustands		2 h	
1.1. Fachgespräch Geschäftsführung	1 h		
1.2 Analyse des Programmcodes	1 h		
2. „Make or buy“-Entscheidung und Wirtschaftlichkeitsanalyse		1 h	
3. Erstellen eines Use-Case-Diagramms		2 h	
4. Erstellen des Lastenhefts mit dem Entwickler		3 h	
Entwurfsphase			**13 h**
1. Programmablaufplan		1 h	
2. Datenbankentwurf		2 h	
2.1. ER-Modell erstellen	1 h		
2.2. Konkretes Tabellenmodell erstellen	1 h		
3. Erstellen eines Klassendiagrammes der Anwendung		1 h	
4. Erstellen eines Aktivitätsdiagramm der Anwendung		2 h	
5. Erstellen eines Struktogramms für Login		1 h	
6. Benutzeroberflächen entwerfen und abstimmen		2 h	
7. Erstellen des Pflichtenhefts		4 h	
Implementierungsphase			**40 h**
1. Anlegen der Datenbank		1 h	
2. Umsetzung der HTML-Oberflächen und Stylesheets		5 h	
3. Programmierung der PHP-Module für die Funktionen		29 h	
3.1. Import der TOTP / HOTP Library	2 h		
3.2. Import der QRCodeJS	1 h		
3.3. Erstellen der Klassen	1 h		
3.4. Erstellen der Speicher und Laden Funktion	2 h		
3.5. Erstellen der TOTP Methode	8 h		
3.6. Erstellen der Mail Methode	6 h		
3.7. Anzeigen einer Liste mit Backup Codes	3 h		
3.8. Erstellen einer Übersichtsseite für der Methoden	1 h		
3.9. Anpassen des Logins	5 h		
4. Testen der Methoden/Funktionen		5 h	
Abnahmetest der Fachabteilung			**1 h**

1. Abnahmetest der Fachabteilung	1 h	
Erstellen der Dokumentation		**10 h**
1. Erstellen der Projektdokumentation	8 h	
2. Programmdokumentation	2 h	
Projektbewertung		**1 h**
1. Fazit	0,5 h	
2. Ausblick	0,5 h	
Gesamt		**73 h**

Tabelle 3: Detaillierte Zeitplanung

A2 Ressourcenplanung

Hardware

- Büroarbeitsplatz mit Computer

Software

- OSX Mojave
- Eclipse IDE für Java und Webentwicklung – Entwicklungsumgebung
- Google Chrome
- MySQL – Datenbanksystem
- PHPMyAdmin – Verwaltungswerkzeug für Datenbanksysteme
- Draw.io – Programm zum erstellen von UML und andere
- Git – verteilte Versionsverwaltung
- Office 365 – Für die Dokumentation

Personal

- Geschäftsführung – Festlegung der Anforderungen und Abnahme des Projektes
- Prüfling – Umsetzung des Projektes
- Anwendungsentwickler – Review des Codes

A3 Anwendungsfalldiagramm

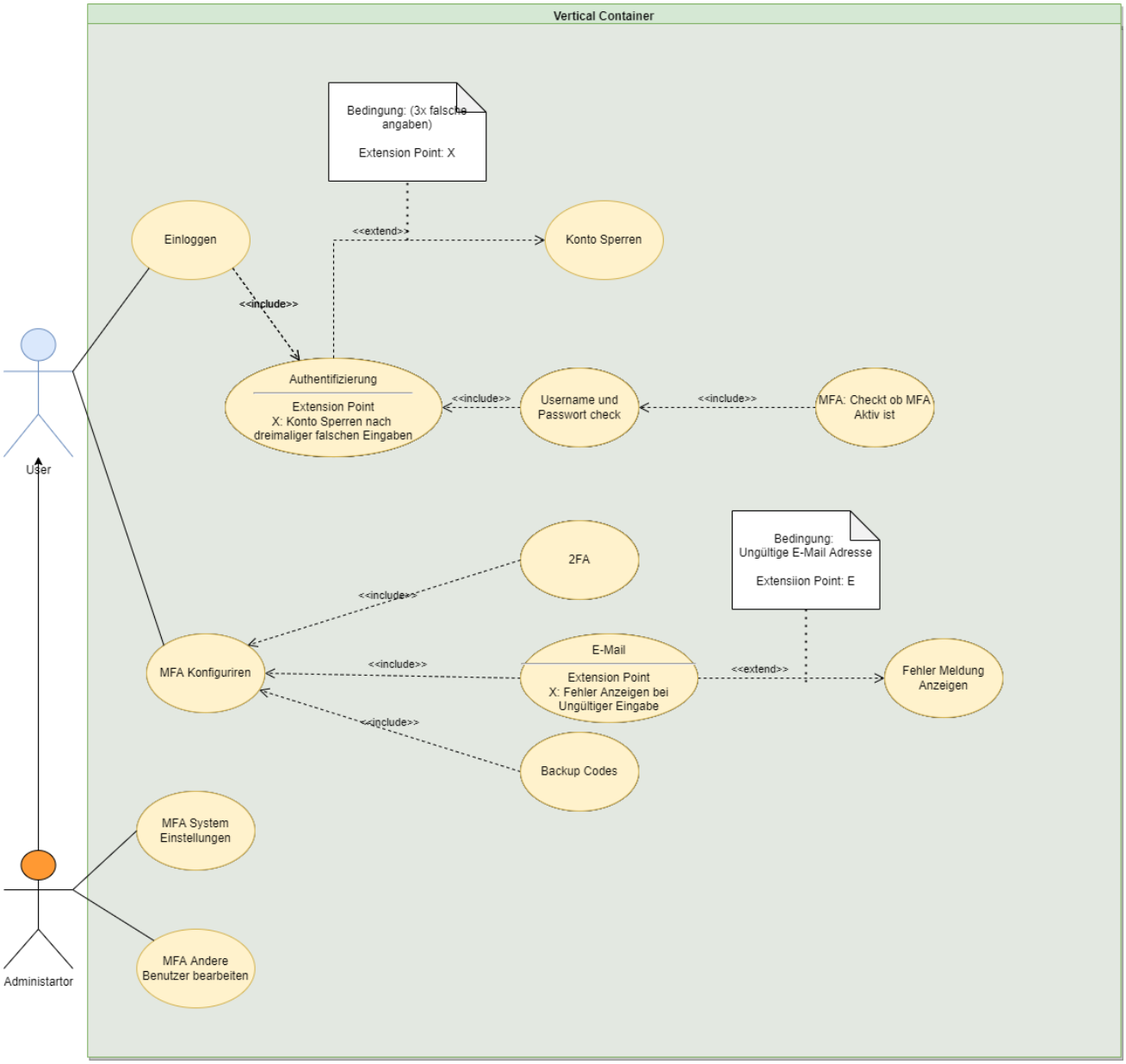


Abbildung 1: Anwendungsfalldiagramm

A4 Lastenheft (Auszug)

Es folgen die Anforderungen, die nach Abschluss des Projektes erfüllt sein müssen.

Anforderungen

- Als User muss ich mich einloggen und verifizieren können.
- Als User muss die MFA-Methoden Konfigurieren können.
- Als User sollte ich die MFA-Methoden Deaktivieren können.
- Als Administrator sollte ich MFA-Methoden erzwingen können.
- Als Administrator muss ich andere User verwalten können.

Allgemeine Ziele

- Das Plugin dient zu Überprüfung des Users und soll Dritte aussperren.
- Das Plugin soll die Sicherheit der Website erhöhen.
- [Firma] möchte, die Reichweite der [Produkt] erweitern.
- Modularer Aufbau, um Veränderungen im Nachhinein vornehmen zu können.
- Zufriedene Kunden, d.h. positive Rückmeldungen bzgl. der einfachen Bedienbarkeit bzw. den Punkten, die dem Kunden aufgefallen sind.

Technische Ziele

- Das Plugin soll mit HTML, CSS, PHP, JS erstellt werden.
- Es soll die Datenbank verwenden.
- Das Framework VueJS soll verwendet werden.
- Die Bibliothek von Spomky Labs soll verwendet werden.
- Integration des QRCODE.js
- Erstellung des User-Interface und System Konfiguration.
- Das Erweitern des Login-Bereichs des Webbasierten-CRMs
- Berücksichtigung des Responsive Design Konzeptes, so dass eine optimale Darstellung der Präsentation auch auf mobilen Endgeräten gegeben ist.
- Das Plugin soll in Deutsch und English verfügbar sein.
- Das Plugin soll 2 Methoden mitliefern (TOTP, Mail)
- Das Plugin soll Backup Funktion bieten
- Entwicklerdokumentation im Rahmen von Quellcode Kommentaren bzw. API-Dokumentation.

Qualitätsbezogene Ziele

Das Plugin soll

- Übersichtlich
- Einfach
- Verständlich und von Benutzer nachvollziehbar sein.
- Hohe Sicherheit

Außerdem soll das Plugin ein angepasstes Layout erhalten.
Die oben aufgeführten Ziele widersprechen sich nicht und sind somit als indifferent zu beurteilen.

A5 Pflichtenheft für das MFA-Plugin

Einführung

- Das Plugin soll die Sicherheit der Website erhöhen, indem es die Authentifizierung des Users über eine Zwei-Faktor-Authentifizierung (MFA) durchführt.
- Das Plugin wird von Prüfling entwickelt und dient dazu, die Reichweite der [Produkt] zu erweitern.
- Das Plugin soll in Deutsch und Englisch verfügbar sein.

Anforderungen

- Als User soll es möglich sein, sich einzuloggen und zu verifizieren.
- Als User soll es möglich sein, MFA-Methoden zu konfigurieren und zu deaktivieren.
- Als Administrator soll es möglich sein, MFA-Methoden zu erzwingen.
- Als Administrator soll es möglich sein, andere User zu verwalten.

Ziele

- Das Plugin soll Dritte aussperren und die Sicherheit der Website erhöhen.
- Das Plugin soll ein angepasstes Layout erhalten.
- Das Plugin soll modular aufgebaut werden, um Veränderungen im Nachhinein vornehmen zu können.
- Das Plugin soll hohe Qualität, Übersichtlichkeit und Benutzerfreundlichkeit aufweisen, um Kunden zufrieden zu stellen.
- Das Plugin soll in HTML, CSS, PHP, JS erstellt werden und das Framework VueJS, die Bibliothek von Spomky Labs und die Integration des QRCODE.js verwenden.
- Das Plugin soll eine Backup-Funktion bieten.
- Das Plugin soll Entwicklerdokumentation im Rahmen von Quellcode-Kommentaren bzw. API-Dokumentation beinhalten.

Technische Anforderungen

- Das Plugin soll die Datenbank verwenden.
- Das Plugin soll eine Erweiterung des Login-Bereichs des webbasierten CRMs ermöglichen.
- Das Plugin soll das Responsive Design Konzept berücksichtigen, um eine optimale Darstellung auf mobilen Endgeräten zu gewährleisten.
- Das Plugin soll 2 Methoden mitliefern: TOTP und Mail.

Qualitätsbezogene Ziele

- Das Plugin soll übersichtlich, einfach, verständlich und von Benutzern nachvollziehbar sein.
- Das Plugin soll eine hohe Sicherheit aufweisen.

A6 User-Interface Entwurf

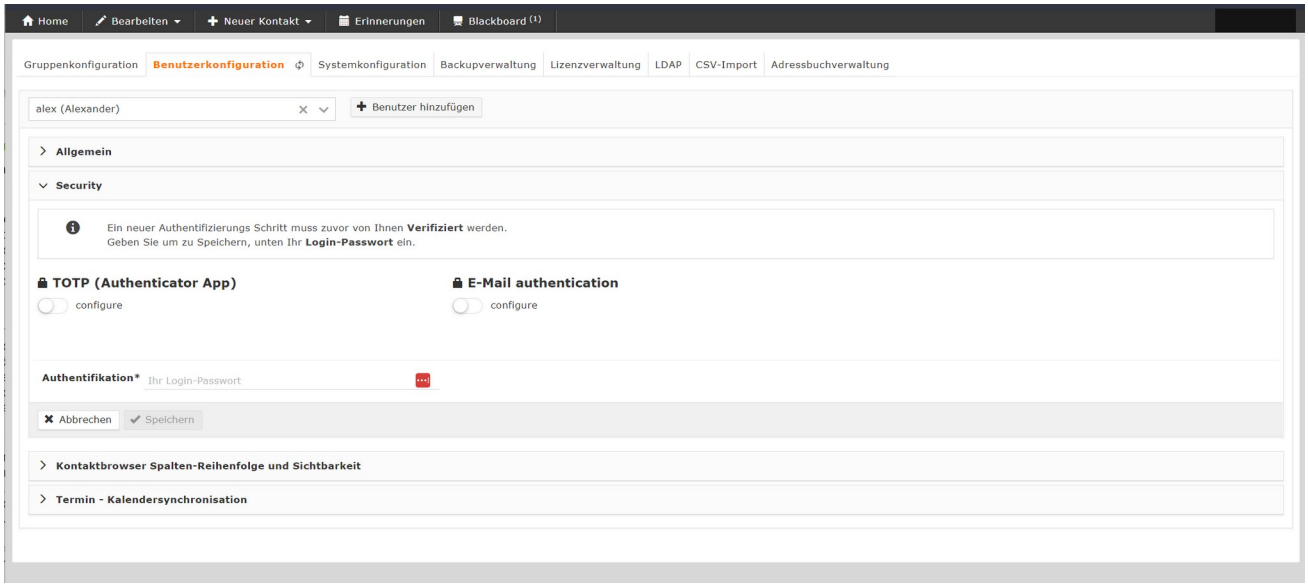


Abbildung 2: Erster Entwurf User-Interface

A7 Endgeräte Infografik

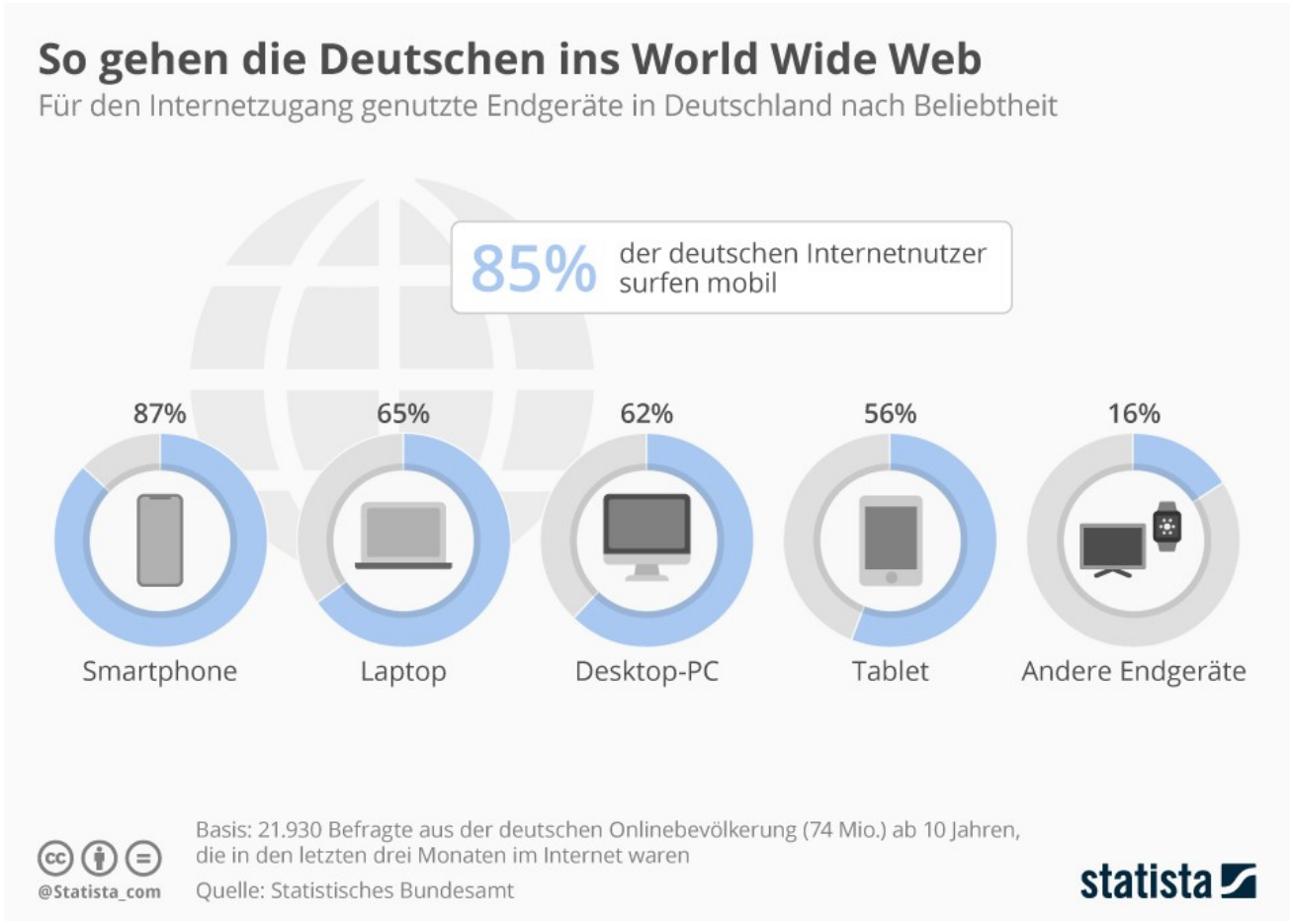


Abbildung 3: Endgeräte Infografik

A8 Modal Backup Codes Fertig

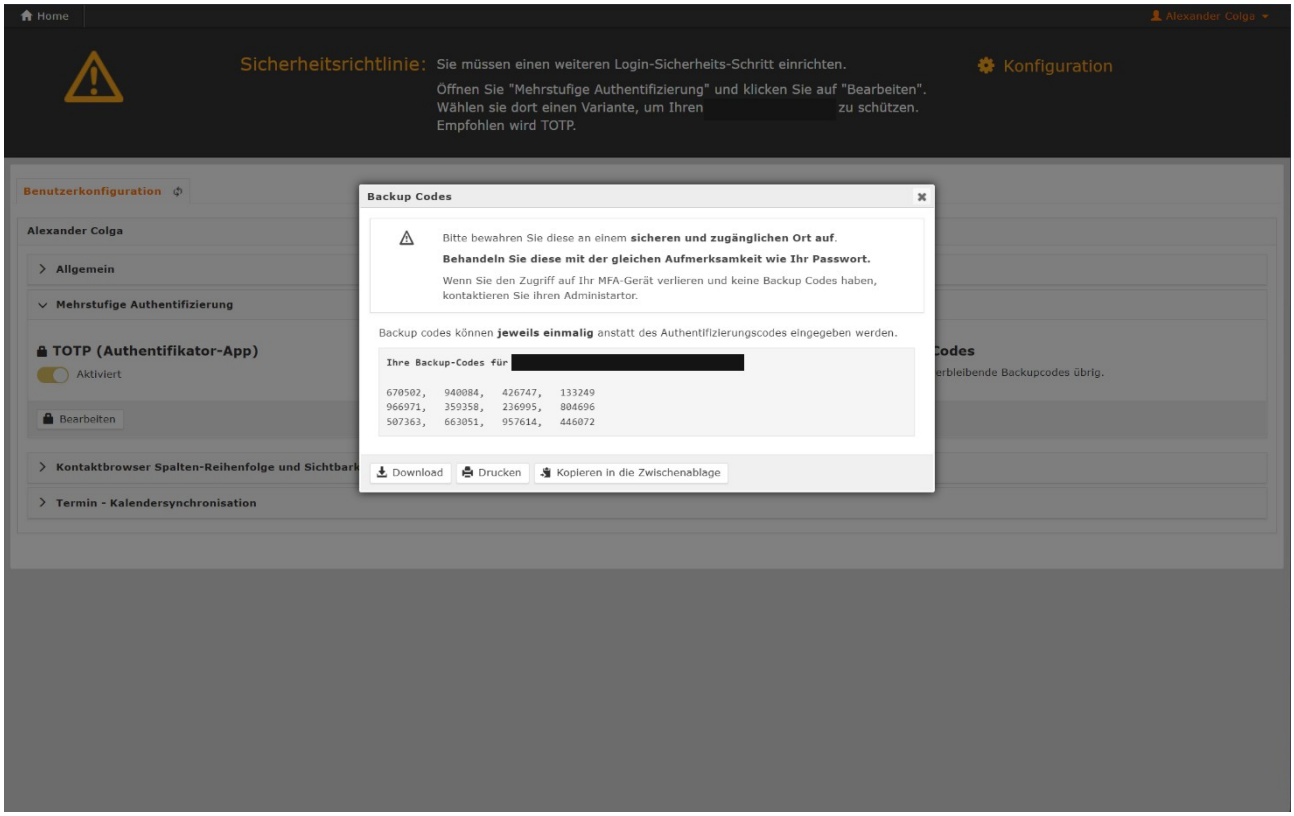


Abbildung 4: Backup Code Modal

A9 Entity-Relationship-Modell

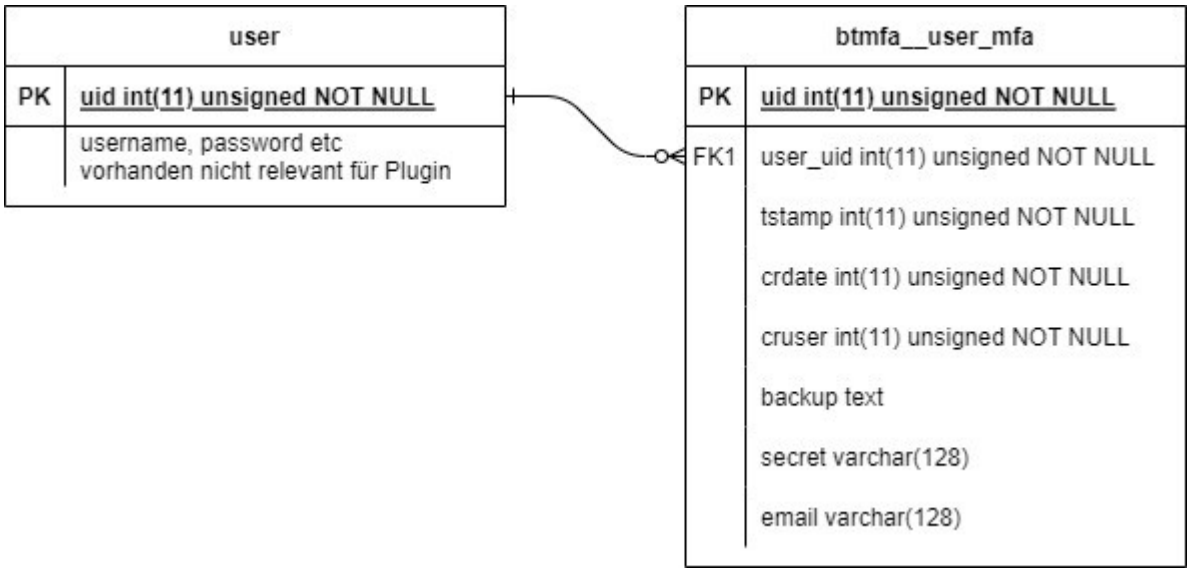


Abbildung 5: Entity-Relationship-Model

A10 Klassendiagramm (Auszug)

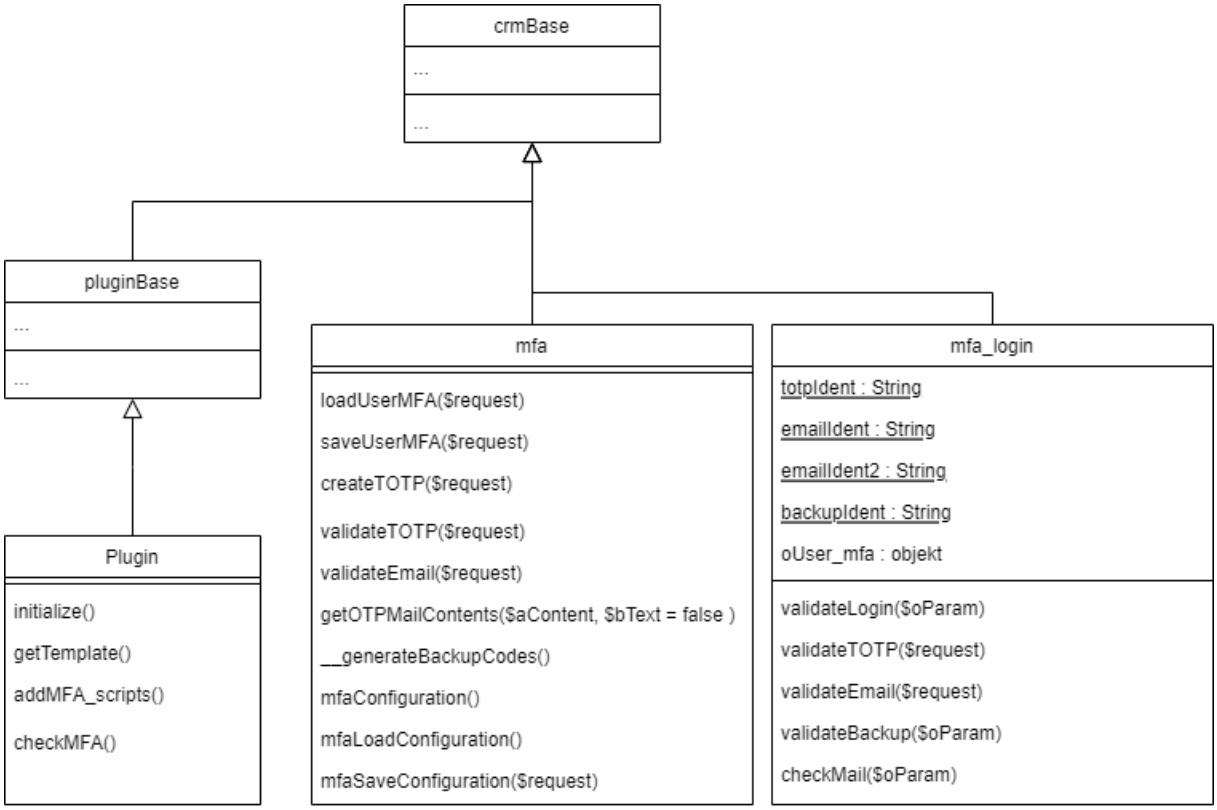


Abbildung 6: Klassendiagramm (Auszug)

A11 Programmablaufplan Login

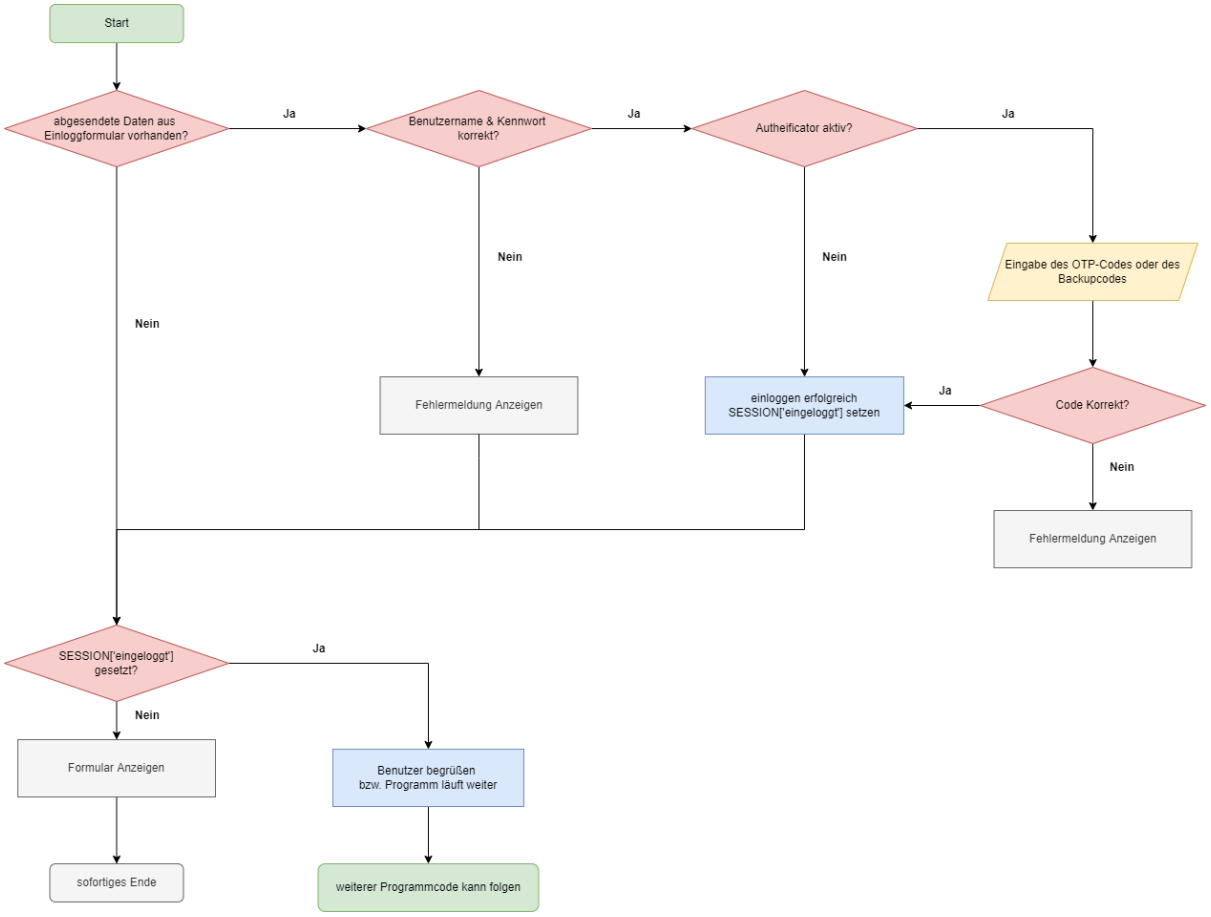


Abbildung 7: Programmablaufplan Login

A12 Struktogramm Login

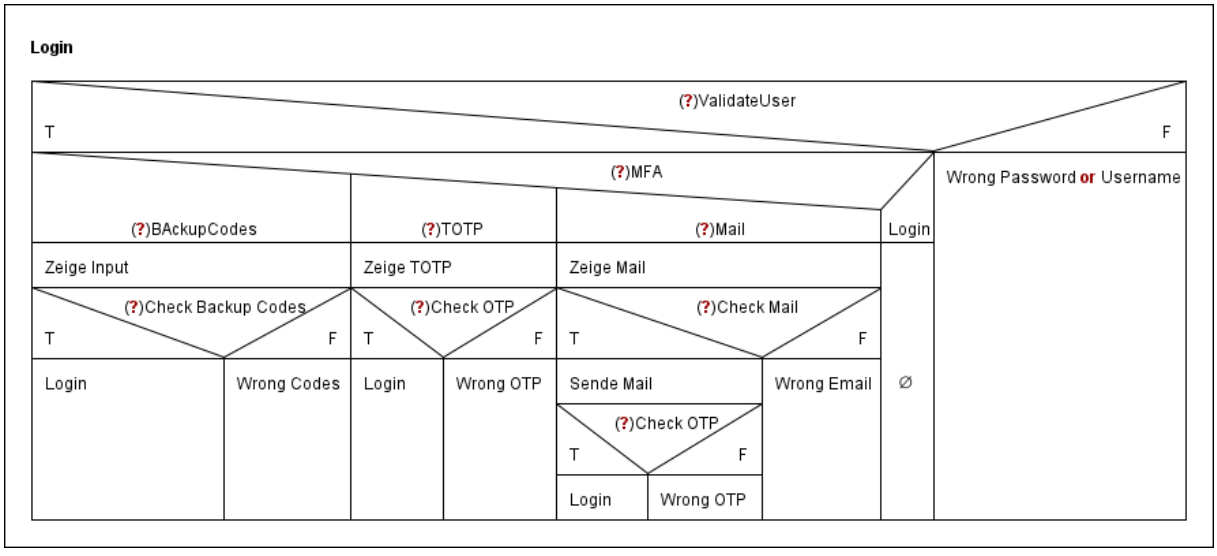


Abbildung 8: Login Struktogramm

A13 MFA Modal

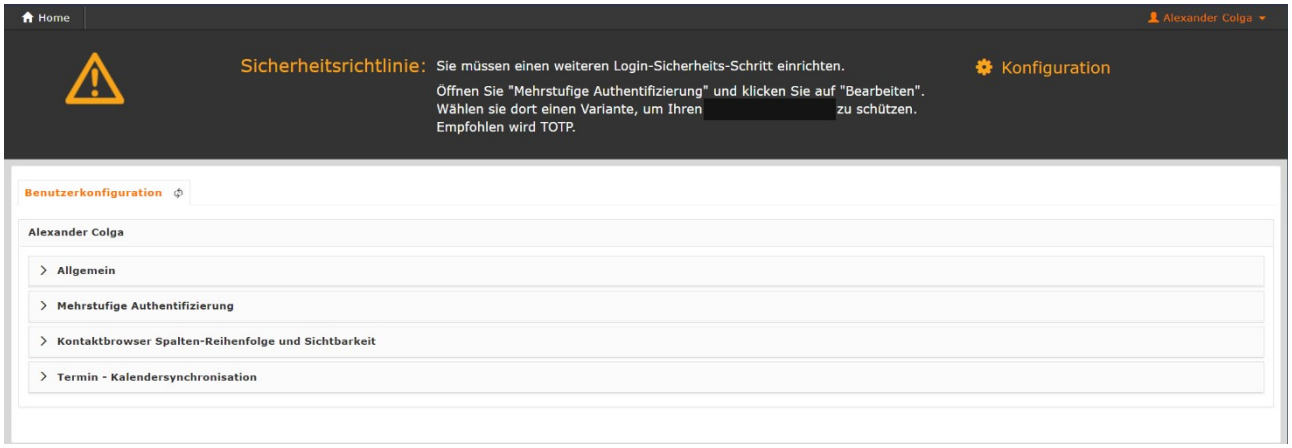


Abbildung 9: MFA-Sicherheitsrichtlinie

A14 Class user_mfa

```php
namespace [Produkt]\Plugins\firma_mfa\db
/**
 * This class represents a single row of the
 * database table "user_security".
 *
 * @author Alexander Colga
 *
 */
class user_mfa extends \dbrepresentation
{
    var $sTablename = 'firmamfa__user_mfa';
    /**
     * Stores the user_uid
     *
     * @var integer
     */
    var $user_uid;
    /**
     * Stores the backup codes as csv
     *
     * @var integer
     */
    var $backup;
    /**
     * Stores the secret for otp
     *
     * @var integer
     */
    var $secret;
    /**
     * Stores the mail
     *
     * @var integer
```

```
     */
    var $email;
    /**
     * Sets the user_uid
     * @param integer $i
     */
    function setUserUid($i) {
        $this->user_uid = $i;
        return $this;
    }
    /**
     * Get the user_uid
     */
    function getUserUid() {
        return $this->user_uid;
    }
    /**
     * Sets the Backup Codes
     * @param string $a
     */
    function setBackup($s) {
        $this->backup = $s;
        return $this;
    }
    /**
     * Get the backup codes
     */
    function getBackup() {
        return $this->backup;
    }
    /**
     * Sets the email
     * @param String $s
     */
    function setEmail($s){
        $this->email = $s;
        return $this;
    }
    /**
     * Get the email
     */
    function getEmail() {
        return $this->email;
    }
    /**
     * Sets the otp secret
     * @param string $s
     */
    function setSecret($s) {
        $this->secret = $s;
        return $this;
    }
```

```
    /**
     * Get the otp secret
     */
    function getSecret() {
        return $this->secret;
    }
}
```

Listing 1: class user_mfa

A15 Screenshot User-Interface Fertig

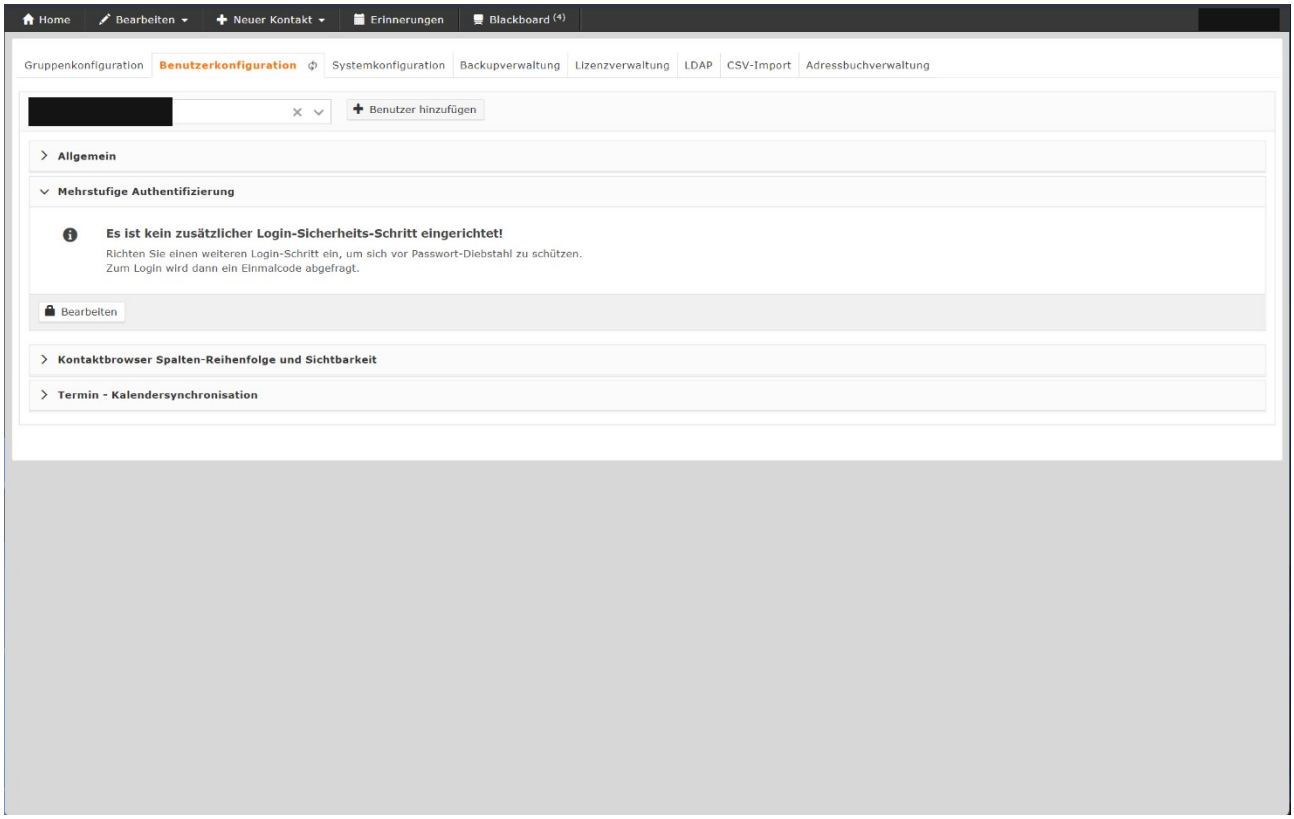


Abbildung 10: User Interface Leer

Anhang

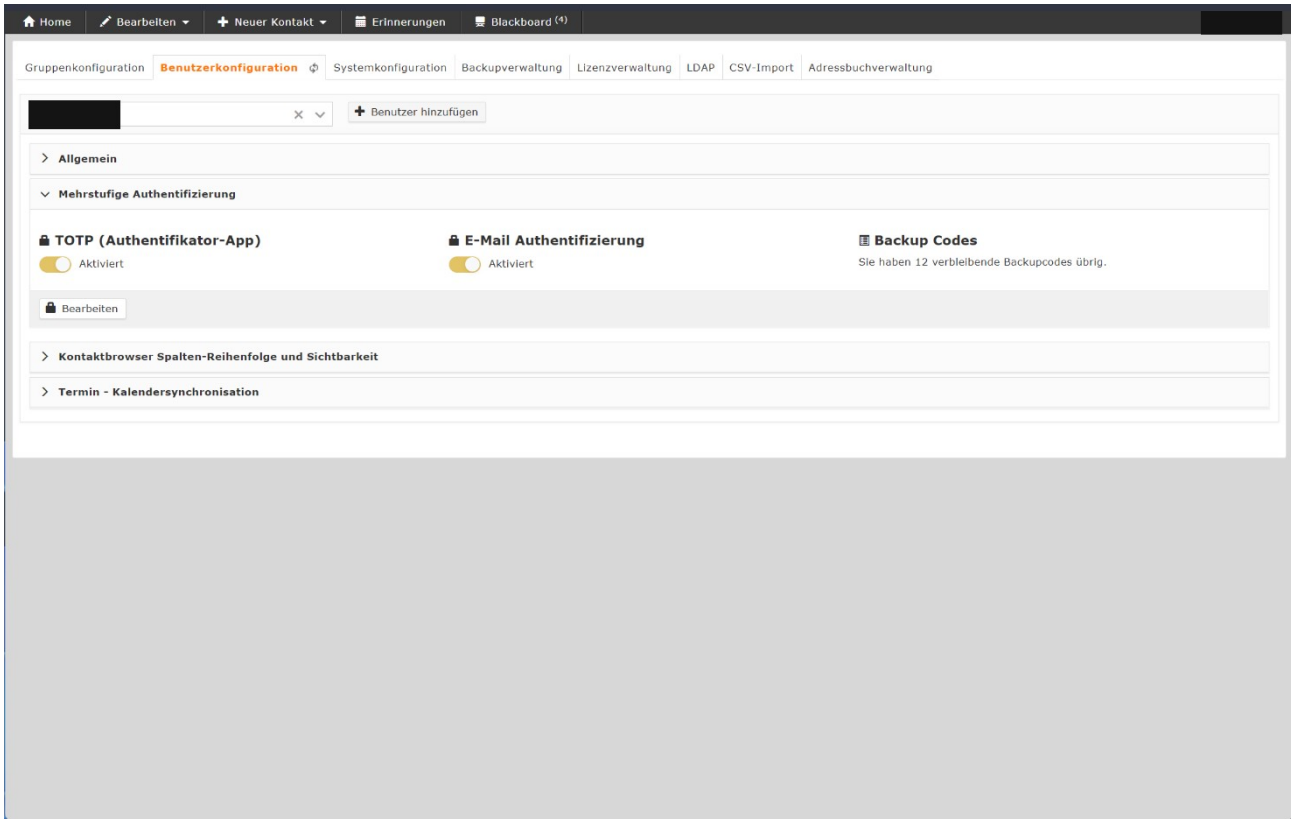


Abbildung 11: User Interface Voll eingerichtet

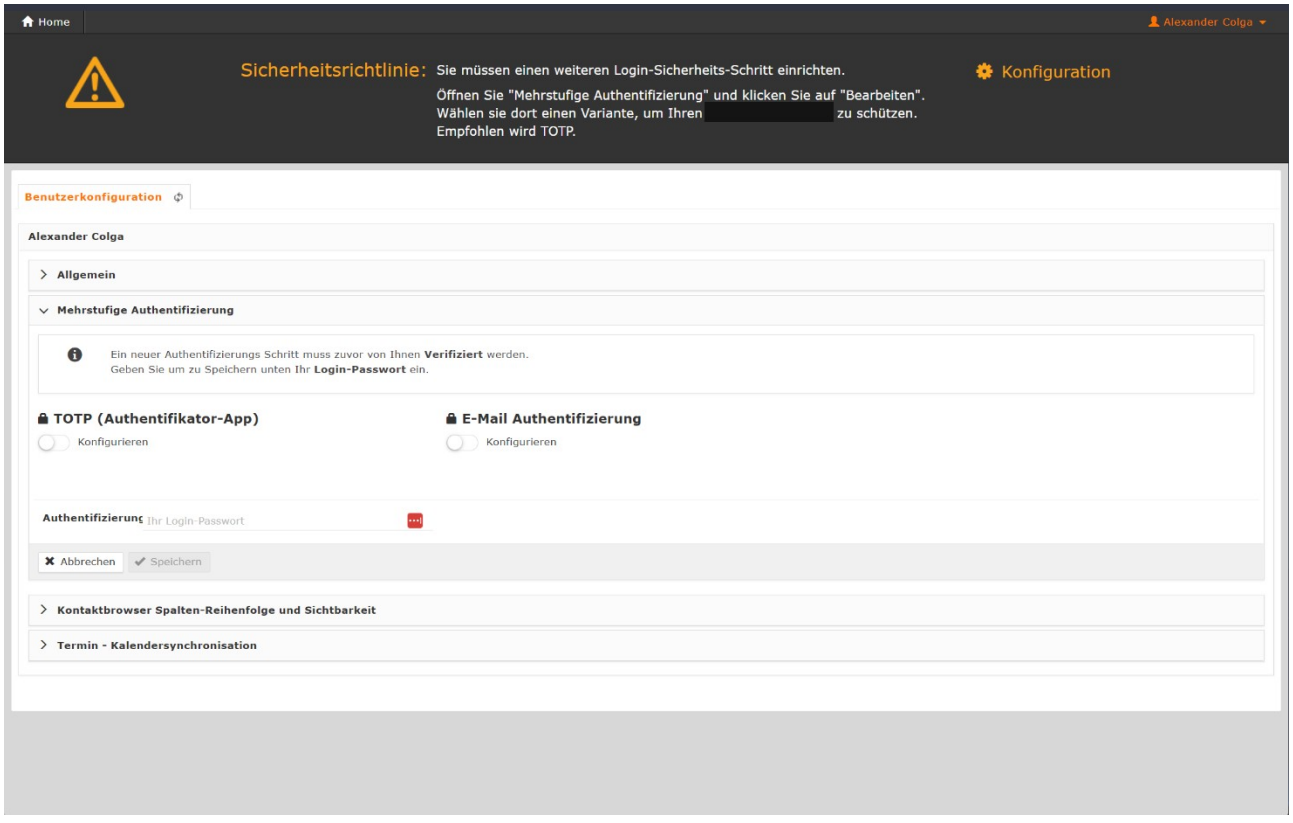


Abbildung 12: User Interface Bearbeitung Modus

Anhang

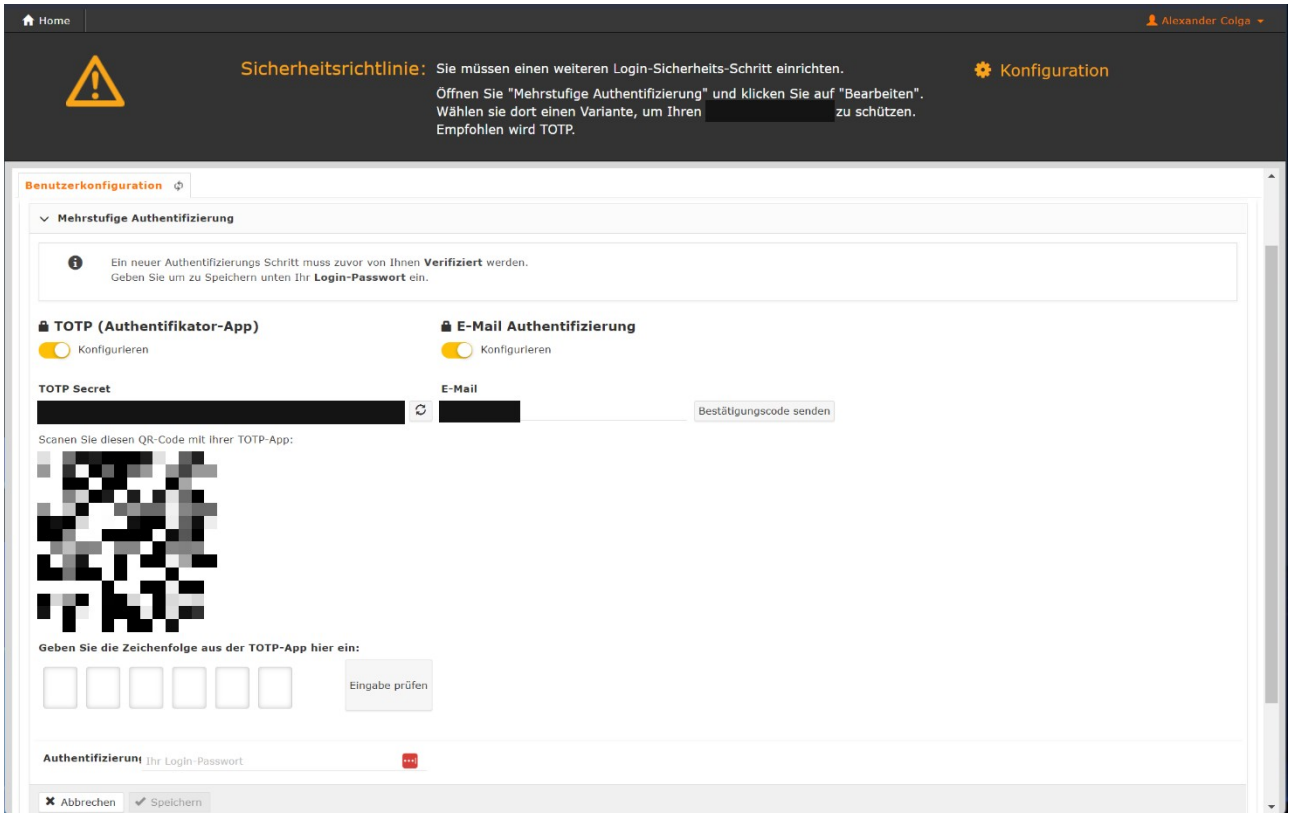


Abbildung 13: User Interface Einrichtung

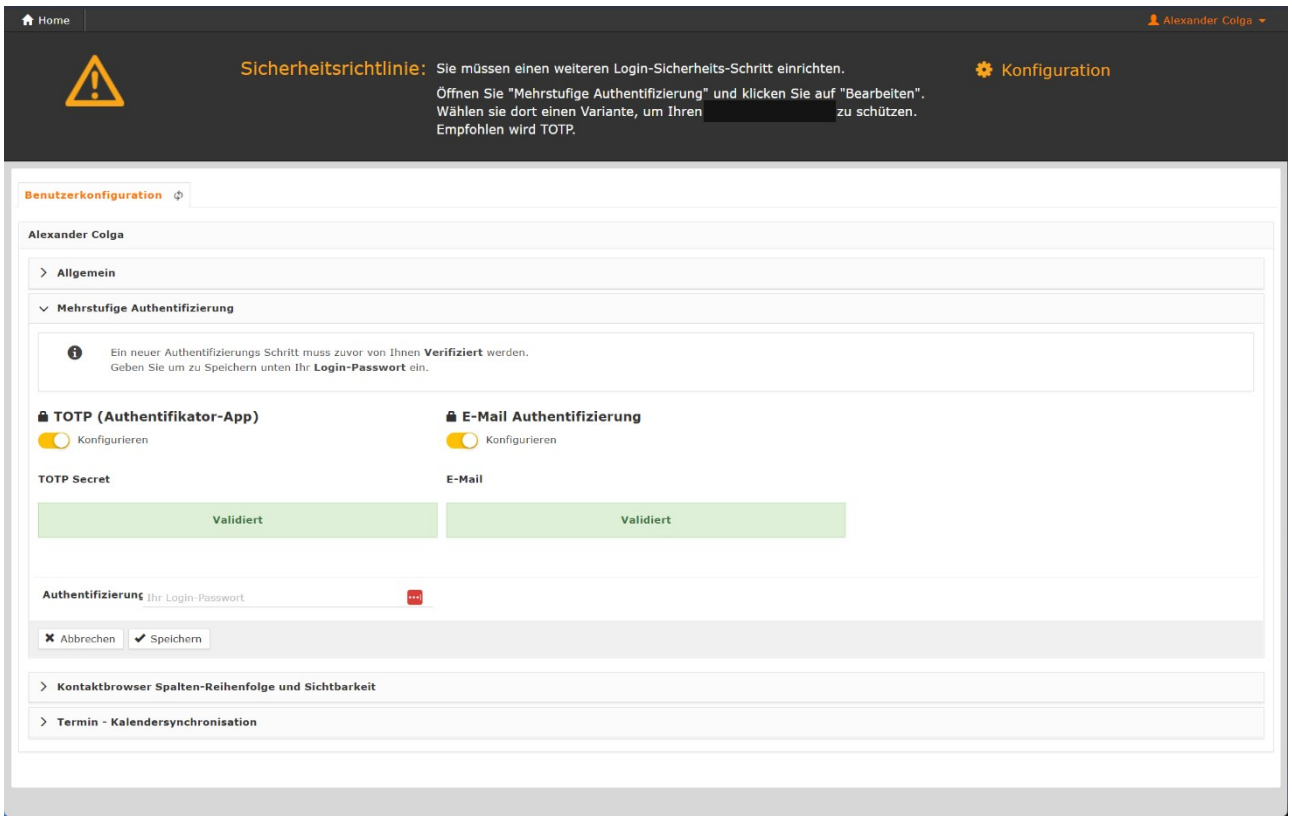


Abbildung 14: User Interface Einrichtung Fertig

A16 Screenshot Admin-Interface

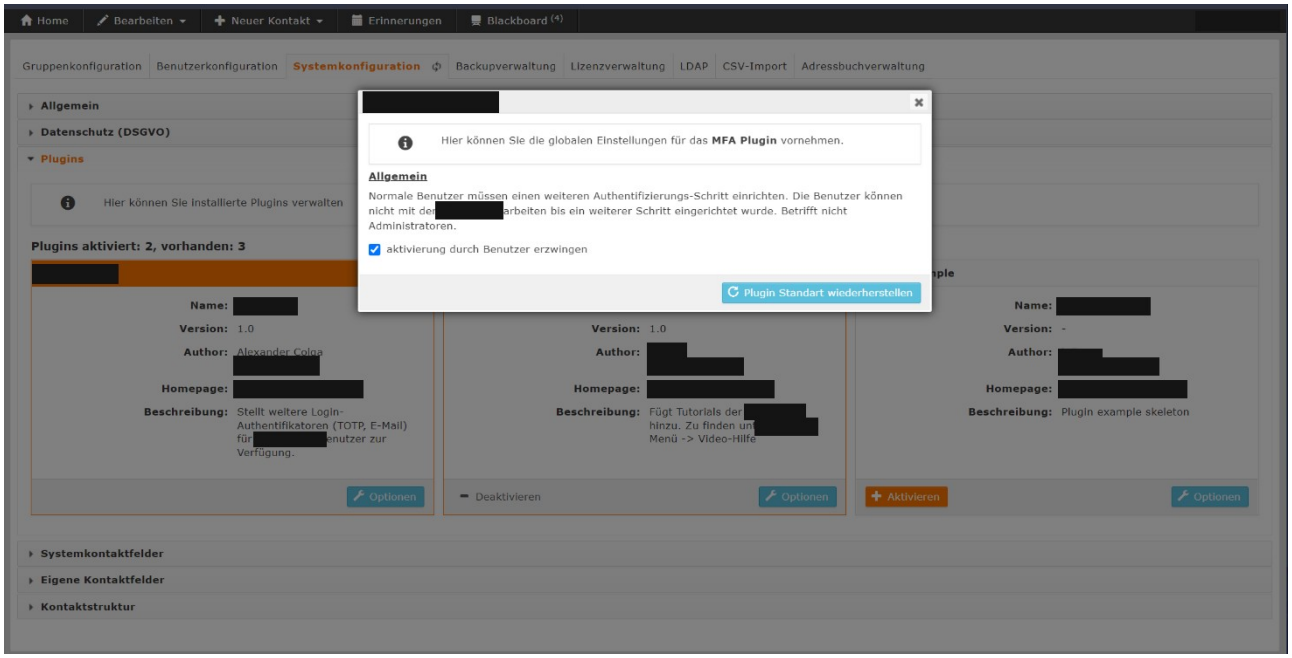


Abbildung 15: Screenshot Admin-Interface

A17 Screenshots Login

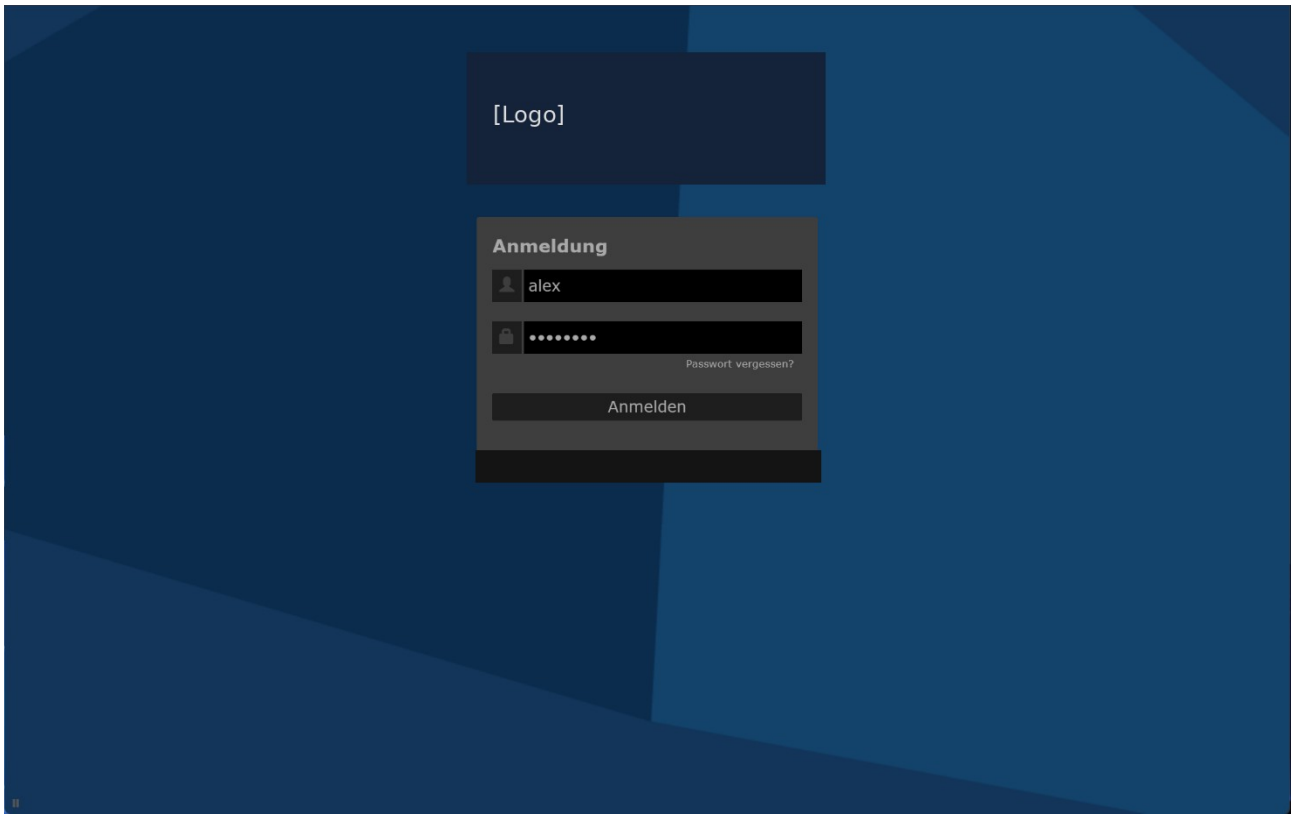


Abbildung 16: Login Step1

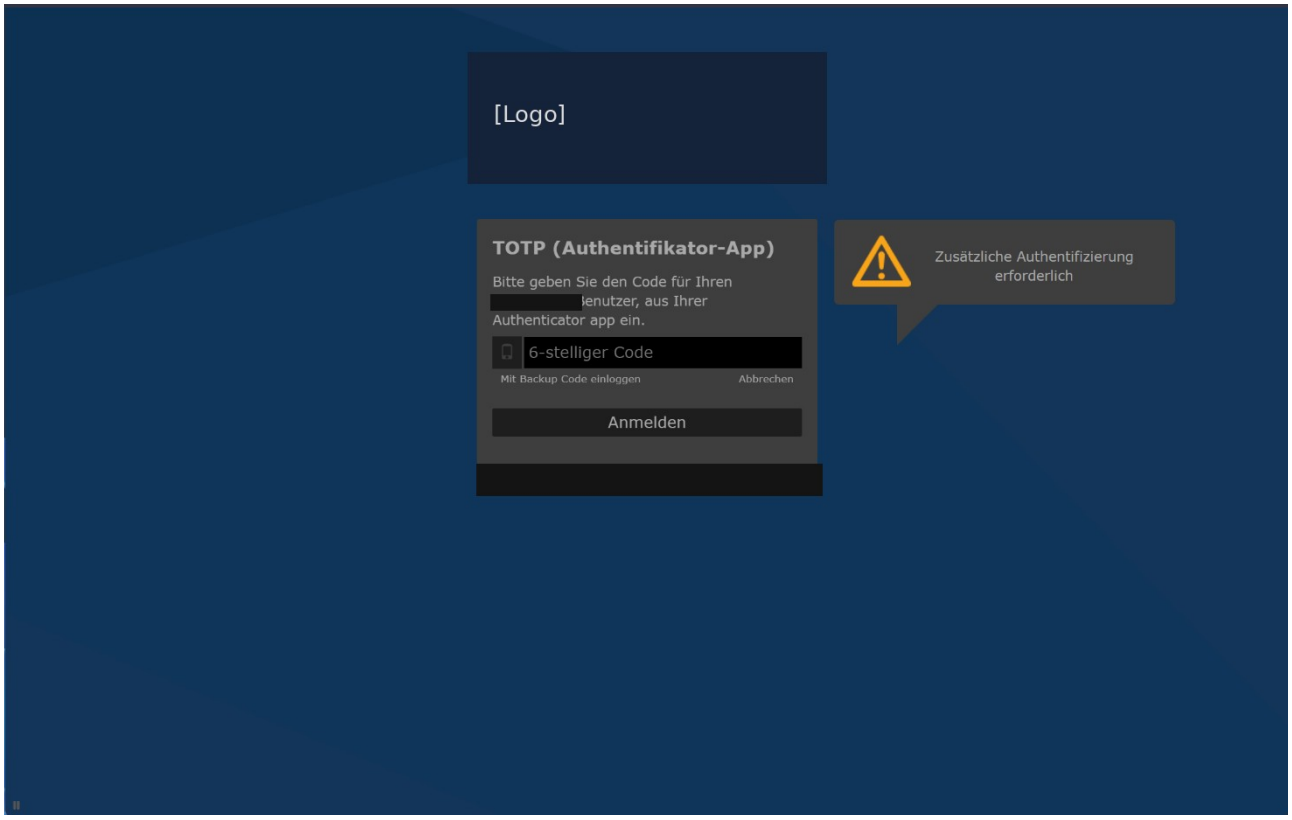


Abbildung 17: Login TOTP

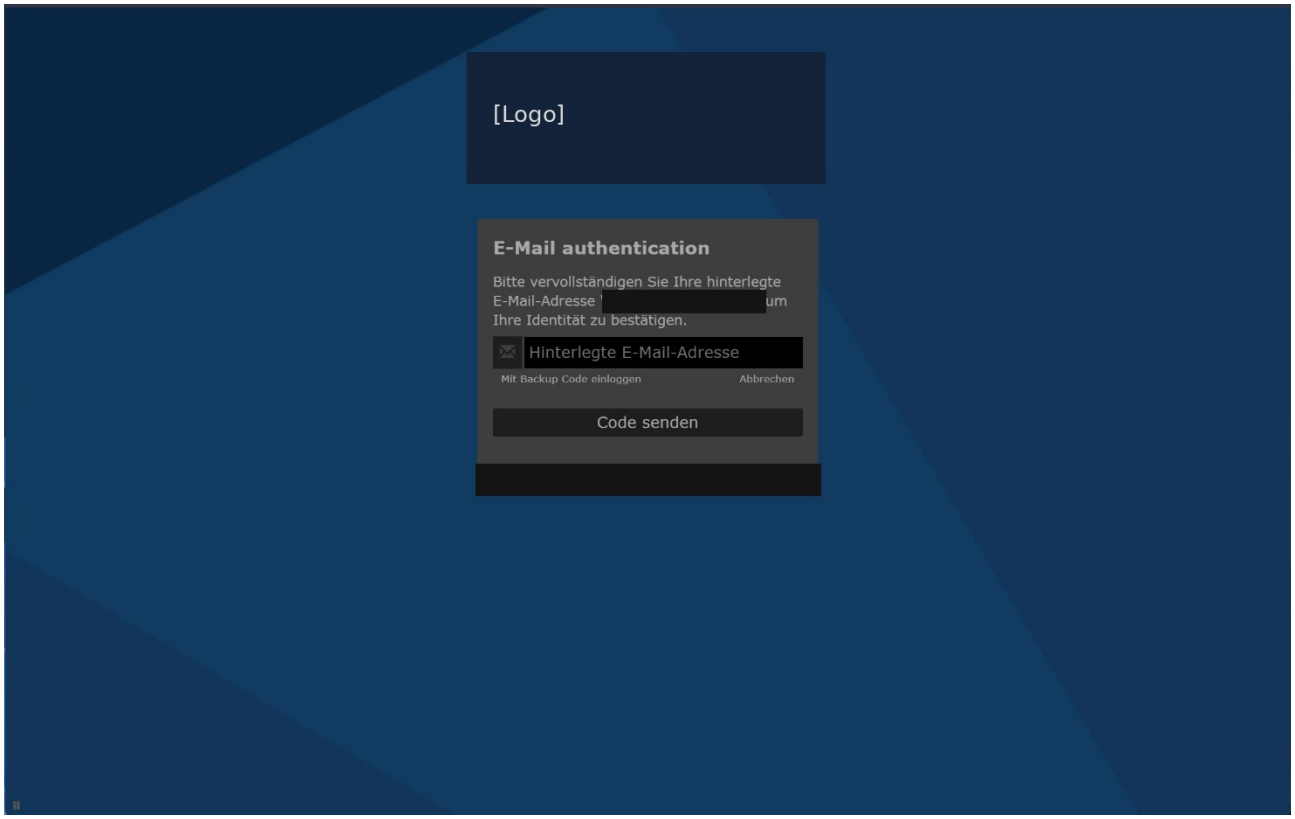


Abbildung 18: Login E-Mail Step1

Anhang

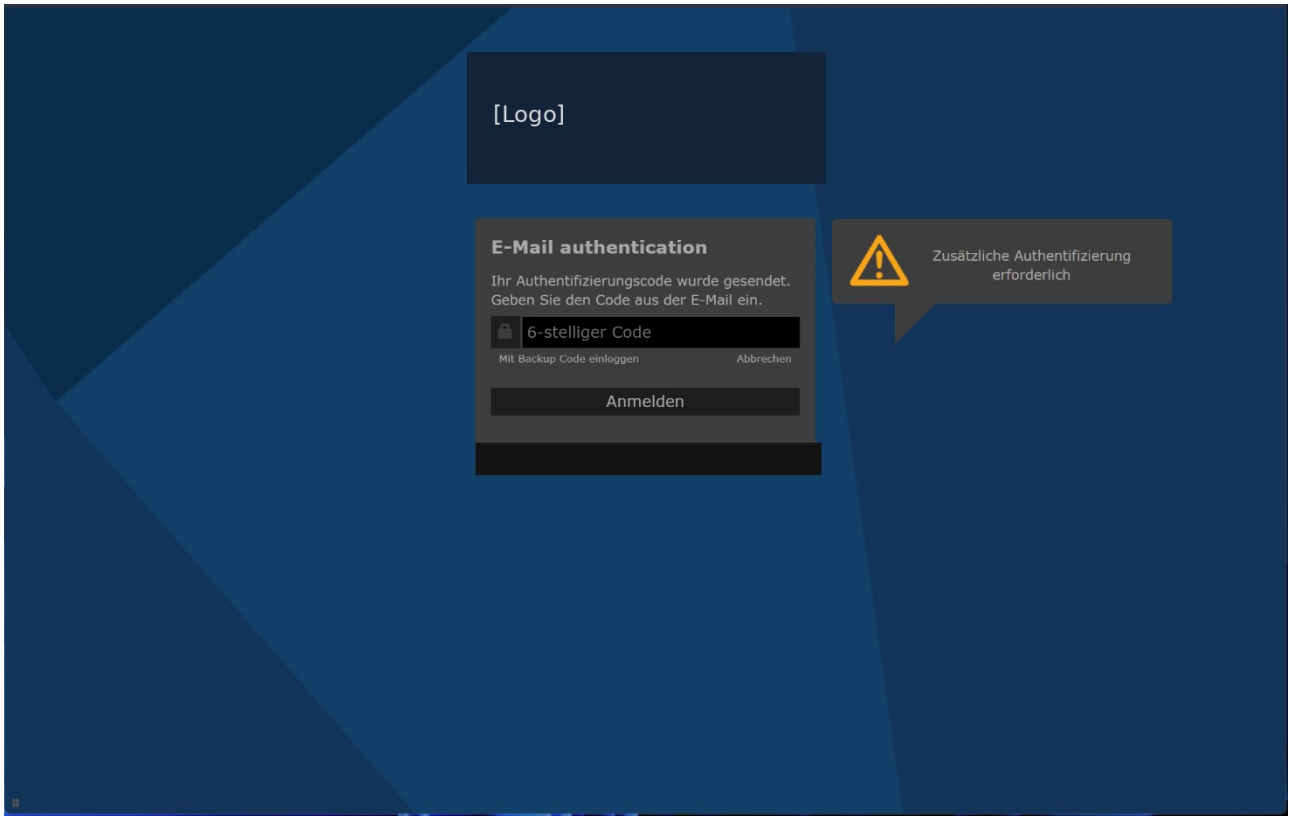


Abbildung 19: Login E-Mail Stap2

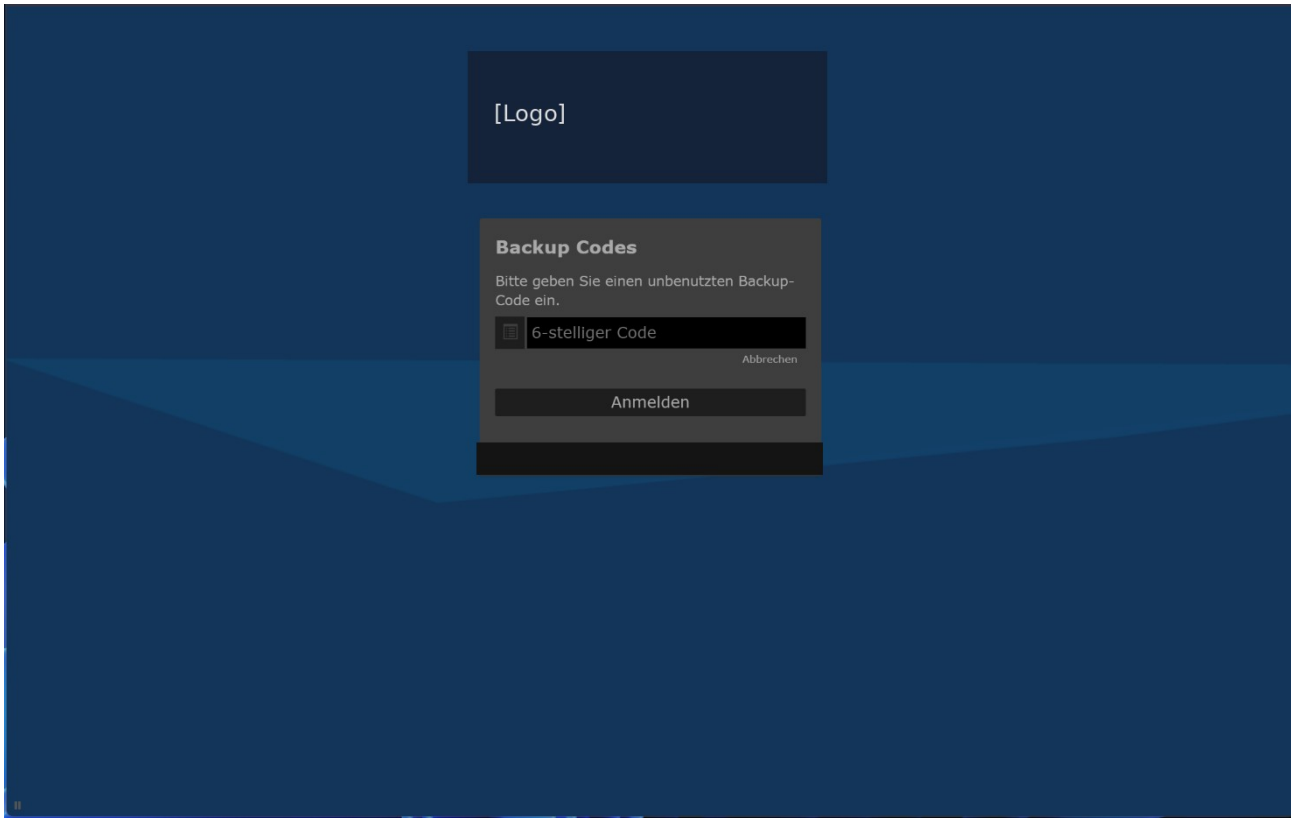


Abbildung 20: Login Backup Code

A18 Class MFA

```
Class MFA extends \crmBase
{
    /**
     * Load all data that require for MFA
     * @param unknown $user_uid
     * @return unknown[]|boolean[][]|string[][][][]|boolean[][][][]|
    NULL[][]
     */
    function loadUserMFA($request)
    {
        if ($this->_crm()->isLoginUserAdmin()) {
            $user_uid = $request["user_uid"];
        } else {
            $user_uid = (int) $this->_crm()->getLoginUserUid();
        }

        // Kein Eintrag vorhanden
        $userMfa = [
            "user_uid" => $user_uid,
            "totpActive" => false,
            "mailActive"=> false,
            "bkpCodesLeft"=> 0,
        ];

        $user_mfa = new db\user_mfa;
        if ($user_mfa->loadByFields(["user_uid" => $user_uid])) {
            $userMfa["totpActive"] = !! $user_mfa->getSecret();
            $userMfa["mailActive"] = !! $user_mfa->getEmail();

            $iBkpCount = count(explode(',', $user_mfa->getBackup()));
            $userMfa["bkpCodesLeft"] = $iBkpCount;
        }

        return \main::respondSuccess($userMfa);
    }

    /**
    * Methoden Ausgeschnitten
    * Sonnst zulanger Anhang
    */
    /**
     * Create a Secret for 2FA and
     * Create the Link for the QR Code
     * @return array
     */
    function createTOTP($request)
    {
        $totp = TOTP::create();
        $secret = $totp->getSecret();

        $user = isset($request["user"]) ? trim($request["user"]) : '';
```

```
        $domain = isset($request["domain"]) ? trim($request["domain"]) :
    '';

        $totp->setLabel(htmlspecialchars($user)); // The label (string)
        $totp->setIssuer(htmlspecialchars($domain)); // cbx domain
        $uri = $totp->getProvisioningUri();

        $response = [
            "secret" => $secret,
            "uri" => $uri,
            "validated" => 0
        ];

        return \main::respondSuccess($response);
    }

    /**
     * Checkt if the OTP correct
     * @param {OTP} $input - The Code from the Authenication App
     * @return array
     */
    function validateTOTP($request)
    {
        $secret = isset($request['secret']) ? $request['secret'] : false;
        $token = isset($request['token']) ? $request['token'] : false;

        if ($secret === false || $token === false) {
            // return validation failed
            return \main::respondSuccess(-1);
        }

        $totp = TOTP::create($secret);
        // 1 = ok; -1 = fail; 0 = ui-default
        $v = $totp->verify($token, null, 1) ? 1 : -1;
        return \main::respondSuccess($v );
    }
}
```

Listing 2: Class MFA (Ausschnitt LoadUserMFA, CreateTOTP, validateTOTP)

A19 VueJS Code

```
// vue component defer loading
// and get its temlate dynamic from server
Vue.component('firma_mfa_userconfiguration', function(resolve) {
    // get template
    $.ajaxq('getMfaAdminTemplate', {url:
    firmaMfaPlugin.templates.userconfiguration})
    .then(function(response) {
        resolve({
    props: ['selectedUserData']
template: response,
    computed: {
        // this computed value will trigger loading new data on user
    change
        // and returns selected users mfa status
        mfaStatus: function() {
            if (! this.selectedUserData) {
                return null;
            }
            if (! this.mfa_tmp || ! this.mfa_tmp.user_uid) {
                this.loadUserMfaStatus();
            }
            return this.mfa_tmp;
        },
        // check if saving is allowed
        isModified: function() {
            return      this.status.totp  &&
    this.status.totp.validated  == 1
                    ||  this.status.email &&
    this.status.email.validated == 1
                    ||  this.status.newBackupCodes;
        },

        [Produkt]Host: function () {
            return window.location.host;
        }
    },
    data: function() {
        return {
            // has mfa status of last selected user
            mfa_tmp: {
                user_uid: null,
                totpActive: false,
                emailActive: false,
                bkpCodesLeft: 0
            },
            // gets a copy of the editing user object
            editMfa: null,

            // same as resetEditStatus
            status: {
                loading: false,
```

```
                enableTotp: false,
                enableEmail: false,
                error: ‘‘,

                // updated from server
                totp: {
                },
                // loaded from server
                email: {
                },
                // just a switch
                newBackupCodes: false
            },

            // qrcodejs instance
            qrcode: null,
            // dialog referenz
            dlgBackup: null,
            // api calls entry point
            apiPath: firmaMfaPlugin.apiPath,
            // shows some object
            debug: crm.debug
        }
    },
    created: function() {
    },
    mounted: function() {
    },
    destroy:function() {
    },
    methods: {

...... Viele Methoden .....
Beispiel Methoden:
loadUserMfaStatus: function() {
            if (this.status.loading) {
                return
            }
            this.status.loading = true;
            this.resetEditStatus();
            var vm = this;

            console.log("loading user mfa for:", this.selectedUserData);

            // call plugin api
            $.ajaxq(‘loadusermfa’, {
                url: this.apiPath + 'MFA/loadUserMFA';,
                data: {
                    user_uid: this.selectedUserData.uid
                },
                success: function(response) {
                    if (response.success) {
```

```
                        vm.mfa_tmp = response.success;
                        vm.status.loading = false;
                    }
                }
            })
        },
/**
         * create Secret_Key HASH
         */
        createTOTP: function()
        {
            var vm = this;
            $.ajaxq('createTOTP', {
                url: this.apiPath + 'MFA/createTOTP',
                data: {
                    user: vm.selectedUserData.username,
                    domain: window.location.host
                },
                success: function(response) {
                    if (response.success) {
                        vm.status.totp = response.success;

                        // wait for template become visible
                        vm.$nextTick(function() {
                         vm.createQRCode();
                        });

                    } else {
                        // disable totp on error,
                        vm.status.enableTotp = false;
                    }
                }
            });
        },
        /**
         * Create the QR Code, davidshimjs/qrcodejs
         *
         * - https://stackoverflow.com/questions/30796584/qrcode-js-
    error-code-length-overflow-17161056
         *   + fixed with padEnd(220), correctLevel Q, dont use old min
    code
         */
        createQRCode: function()
        {
            if(! $("#mfa_totp_qrcode").children().length) {
                this.qrcode = new QRCode("mfa_totp_qrcode", {
                    text: this.status.totp.uri.padEnd(220),
                    width: 196,
                    height: 196,
                    colorDark : "#000000",
                    colorLight : "#ffffff",
                    correctLevel : QRCode.CorrectLevel.Q
```

```
                });
            } else {
                this.qrcode.clear(); // clear the code.
                this.qrcode.makeCode(this.status.totp.uri.padEnd(220));
    // make another code.
            }
        },

        inputValueTOTP: function(otp) {
            this.status.totp.code = otp;
        },

        /**
        * Check the OTP from the TOTP
        */
        validateTOTP: function()
        {
            var vm = this;
            $.ajaxq('checkTOTP', {
                url: this.apiPath + 'MFA/validateTOTP',
                data: {
                    secret: this.status.totp.secret,
                    code: this.status.totp.code
                },
                success: function(response)
                {
                    vm.status.totp.validated = response.success * 1;
                }
            })
        },
/**
            print by open a new window
            - alternativ show codes as fullsize fixed content before
    printing
         */
        printBackupCodes: function ()
        {
            var content = $(this.$refs.formatedBkpCodesOutput).text();

            var myWindow = window.open('', 'Print-Backup-Codes',
    'height=480,width=640');
            myWindow.focus(); // necessary for IE >= 10
           myWindow.document.write('<!doctype html><head><meta
    charset="utf-8">');
           myWindow.document.write('<title>Backup-Codes</title>');
           myWindow.document.write('<style>@page { size: A5 }</style>');
           myWindow.document.write('</head><body><pre>');
           myWindow.document.write(content);
           myWindow.document.write('</pre></body></html>');
           myWindow.document.close(); // necessary for IE >= 10

           myWindow.print();
```

```
            myWindow.close();
        },
}
```

Listing 3: VueJS Ausschnitt

A20 Kommentare/Dokumentation des Quellcodes

```
Beispiel Kommentare
/*
* MFA for [Produkt]
* - call by: /?path=/firma_mfa/php/totp|[Produkt]\\Plugins\\firma_mfa\
    \mfa/loadUserMFA'
* - extends crmBase to access [Produkt] instance
*/
_____________________________

// TOTP - disable / enable
        $factorKey = "totp";
        if (isset($request[$factorKey])) {
            $factor = $request[$factorKey];
            $factor_validated = isset($factor["validated"]) ? (int)
    $factor["validated"] : 0;
            // validation was done
            if ($factor_validated > 0) {

                // - disable if no secret else enable
                $totp_secret = isset($factor["secret"]) &&
    strlen(trim($factor["secret"])) ? trim($factor["secret"]) : null;
                $user_mfa->setSecret($totp_secret);
            }
        }

_________________________
/**
     * Checkt if the OTP correct
     * @param {OTP} $input - The Code from the Authenication App
     * @return array
     */
```

Listing 4: Beispiel Dokumentaion im Code

A21 Mail Template

```
Hallo ###NAME###,

Ihr Authentification Code lautet:

###CODE###

Geben Sie Bitte diese Zeichenfolge in die anfordernde Webseite ein.


[automatisch versandte email]
```

```
Diese E-Mail wurde aufgrund eines Aufrufs eines Web- oder Mobilgeräts mit
    der IP-Adresse ###IP### generiert.
Diese E-Mail wurde im Authentifikationsprozess zur Account Sicherung
    hinterlegt.

###REASON###
```

Listing 5: Mail Plain Template

```
<p>Hallo ###NAME###,<br />
<br />Ihr Authentification Code lautet:<br />
<br /></p>
<table width="100%" border="0" cellspacing="0" cellpadding=="0">
    <tr>
        <td style="font-size: 48px; line-height: 52px; font-family:
    Arial, sans-serif;
            font-weight: bold; text-align: center;">###CODE###</td>
    </tr>
</table>
<br /><br />
<p>[automatisch versandte email]
<br />Diese E-Mail wurde aufgrund eines Aufrufs eines Web- oder
    Mobilgeräts mit der IP-Adresse ###IP### generiert.
<br />Diese E-Mail wurde im Authentifikationsprozess zur Account
    Sicherung hinterlegt.
<br />###REASON###</p>
```

Listing 6: Mail HTML Template

A22 Benutzerdokumentation

Benutzerdokumentation: Screenshots aus Datenschutzgründen entfernt

Abbildung 21: Screenshot Benutzerdokumentation