

FitBridge

The Bridge between your Gym Database and your Clients

Bearbeitet von Gruppe 09:

Sabti, Ahmed

Azad Saleem, Bland

Determann, Martin

Kribia, Othmane



Inhaltsverzeichnis

1 Einleitung	1
1.1 Rollenverteilung	1
1.2 SWOT-Analyse	1
2 Anwendungsfälle	2
3 FitBridge Datenmodell	3
3.1 ER-Diagramme	3
3.2 Users Tabelle	6
3.3 Messages Tabelle	6
3.4 Appointments Tabelle	7
4 Use Cases	8
4.1 Registrierung	8
4.2 Einloggen und in das Portal gelangen	9
4.3 Portalseite Kunde	9
4.4 Portalseite Trainer	10
4.5 Portalseite Admin	11
4.6 Terminbuchung und Einsehbarkeit	12
4.7 Nachrichten senden und löschen	13
5 Softwareanforderungen	14
5.1 Ablauf Einloggen	14
5.2 Ablauf Nachricht Senden	15
5.3 Ablauf Einsehen Termine und Nachrichten auf Kunden oder Trainerseite	16
5.5 Ablauf löschen der Nachrichten	17
5.6 Evaluierung der Diagramme	17
6 Sitemap	18
7 Der Code	19
7.1 Interne Funktion-Bibliothek „phptosql_inc“	20
7.2 Beispiel eines Softwareablaufes: Kunde bucht ein Termin bei Trainer	22
7.3 Beispiel eines Softwareablaufes: Nachricht senden	26
8.1 Verbindung zur Datenbank	28
8.2 SELECT Befehle und Ausführung in PHP	28
8.3 DELETE Befehle	29
8. UPDATE Befehle	30
8.5 INSERT Befehle	30
9 Testfälle	31
9.1 Automatisierte Testverfahren	31
9.2 Testverfahren aus Benutzersicht/Einrichtung	32

Abbildungsverzeichnis

Abbildung 1.1 SWOT-Analyse.....	1
Abbildung 3.1 ER-Diagramm Kunden.....	3
Abbildung 3.2 ER-Diagramm Trainer.....	4
Abbildung 3.3 ER-Diagramm Admin.....	5
Abbildung 3.4 Users Tabelle.....	6
Abbildung 3.5 Messages Tabelle.....	6
Abbildung 3.6 Appointments Tabelle.....	7
Abbildung 4.1 Registrierungsprozess der jeweiligen Benutzer.....	8
Abbildung 4.2 Use Case Diagramm für das Einloggen.....	9
Abbildung 4.3 Use Case Diagramm für die Kundenseite.....	9
Abbildung 4.4 Use Case Diagramm für die Trainerseite.....	10
Abbildung 4.5 Use Case Diagramm für die Adminseite.....	11
Abbildung 4.6 Use Case Diagramm für das Buchen und Einsehen von Terminen.....	12
Abbildung 4.7 Use Cases Diagramm des Senden Empfangen und Löschen einer Nachricht.....	13
Abbildung 5.1 Sequenzdiagramm für den Login-Vorgang.....	14
Abbildung 5.2 Ablauf für das Senden einer Nachricht.....	15
Abbildung 5.3 Termine auf User Page einsehen, als Sequenz-ähnliches Diagramm.....	16
Abbildung 5.4 Einsehen von Nachrichten als Sequenz-ähnliches Diagramm dar.....	16
Abbildung 5.5 Ablauf für das Löschen einer Nachricht.....	17
Abbildung 6.1 Ablauf für das Löschen einer Nachricht.....	18
Abbildung 7.1 Ersetzung von Eingeben von Nutzer und Passwort der Datenbank durch kleine Funktion um Ändern des Passwortes zu vereinfachen.....	20
Abbildung 7.2 „fitBridge_markAsDeleted()“ Funktion.....	21
Abbildung 7.3 Inhalt der „fitBridge_markAsDeleted()“ Funktion.....	21
Abbildung 7.4 Terminbuchungsseite.....	22
Abbildung 7.5 Eingabe für eine Terminbuchung und dazugehörige Umsetzung.....	23
Abbildung 7.6 Überprüfung der Eingaben, in PHP umgesetzt.....	24
Abbildung 7.7 verschiedene Fehlermeldungen.....	25
Abbildung 7.8 Umsetzung der Fehlermeldungsanzeige.....	25
Abbildung 7.9 Formular für Nachrichten.....	26
Abbildung 7.10 Umsetzung der Überprüfung auf leere Felder im Nachrichten Formular.....	26
Abbildung 7.11 „fitBridge_sendMessage()“ Funktion aus internen-Bibliothek.....	27
Abbildung 9.1 Anfang des automatisierten Testverfahrens.....	31

1 Einleitung

In diesem Projekt haben wir eine Plattform für kleine bis mittelgroße Fitnessstudios entwickelt. Diese Plattform soll eine Elektronische Datenbankverwaltung ermöglichen, sodass Benutzer, unter anderem, Termine Buchen können sowie Nachrichten senden können. Die Datenbankverwaltung wurde mithilfe der XAMPP Plattform realisiert. Unser Hauptziel war es eine möglichst benutzerfreundliche, stabile Benutzeroberfläche zur Anwendung unserer Datenbank zu erstellen.

1.1 Rollenverteilung

Die Rollenverteilung hat sich wie folgt ergeben:

Bland Azad Saleem - Projektleiter/ Tester

Ahmed Sabti – Softwareentwickler

Martin Determann – Softwareentwickler

Othmane Kribia – Tester

1.2 SWOT-Analyse

Im Kick-Off Meeting hat unsere Gruppe eine SWOT-Analyse erstellt, um unsere Situation zu analysieren.

SWOT-ANALYSE	
Stärken	Schwächen
<ul style="list-style-type: none">• Programmiererfahrung• Kenntnisse über Datenbanken• Motivation• Teamarbeit	<ul style="list-style-type: none">• Wissenslücken• Prokrastination• Neue Herausforderung
Chancen	Risiken
<ul style="list-style-type: none">• Internetrecherchen• Regelmäßig treffen• Hilfe von Dozenten• Kreativ sein	<ul style="list-style-type: none">• Technische Schwierigkeiten• Missverständnisse• Kein Zeitmanagement• Zeit ist begrenzt

Abbildung 1.1 SWOT-Analyse



2 Anwendungsfälle

Nach dem wir eine grobe Idee hatten, worüber wir unser Projekt gestalten wollten, und die Rollenverteilung erfolgte, haben wir angefangen konkrete Anwendungsfälle zu formulieren, nach denen wir uns im Laufe des Projektes richteten.

Erst haben wir die verschiedenen Anwender der Datenbank formuliert, um dann für die jeweiligen Anwender verschiedene Anwendungsfälle zu formulieren.

Bei unserem Fitnessstudio-Modell gibt es drei Hauptdarsteller: die **Kunden**, die **Trainer** und die Besitzer, die die Webseite als **Administratoren** verwalten sollen.

Anwendungsfall 1: Kunden, Trainer und Administrator sollen sich Registrieren können

Der Erste Anwendungsfall umfasst die spezifische Definition eines Nutzers. Ein Nutzer in unserem System soll sich über eine Benutzeroberfläche in dem FitBridge Portal ein Konto erstellen können, in dem er ein Benutzername, Passwort, Name usw. eingibt. Damit soll er sich einloggen können, um auf eine Portalseite zu gelangen, von wo er, je nach Benutzertyp, folgendes machen kann:

Anwendungsfall 2: Kunden sollen bei Trainern Termine buchen können

Im ersten Anwendungsfall soll ein Kunde bei einem Trainer einen Termin Buchen. Hierzu soll also ein Kunde einen Termin buchen können. Dazu soll der Kunde den Benutzernamen des Trainers, das Datum, die Uhrzeit, sowie zusätzliche Informationen in den neuen Eintrag in der Tabelle eingeben können.

Anwendungsfall 3: Alle Benutzer des Systems sollen miteinander über ein internes Nachrichtensystem kommunizieren können

Im zweiten Anwendungsfall wollten wir den Kunden und Trainern die Möglichkeit geben Termine Absagen zu können, und generell eine Interne Kommunikation zwischen allen Benutzern ermöglichen. Diese Kommunikation kann viele verschiedenen Anwendungen mit sich bringen, da es ein sehr einfaches und anpassungsfähiges System ist. Das System soll zudem alle grundlegenden Funktionen eines E-Mail-Systems bieten, wie Senden, einsehen und löschen einer Nachricht.

Anwendungsfall 4: Administrator soll auf einfacher Art und Weise die Datenbank verwalten können

In dem dritten Anwendungsfall ging es darum dem Administrator die Möglichkeit zu geben das gesamte System direkt von einer Benutzeroberfläche (ohne das phpMyAdmin immer benutzen zu müssen) zu verwalten. Letztendlich haben wir uns dabei für folgende zusätzliche (zur Nachrichten Funktion) Anwendungen entschieden:

Anwendungsfall 4.1: Admin soll Kunden und Trainer aus der Datenbank löschen können

Anwendungsfall 4.2: Admin soll alle gebuchten Termine im System einsehen können



3 FitBridge Datenmodell

Für diese Anwendungsfälle musste jedoch vorerst eine Datenbank aufgebaut werden, in der sämtliche Informationen gespeichert werden können.

3.1 ER-Diagramme

Die „Entity Relationship Diagrams“ verdeutlichen den Zusammenhang zwischen den jeweiligen Usern und den Nachrichten sowie zwischen den Usern und Appointments. Auch sind deren Attribute, welche als Werte in den Tabellen abgespeichert werden, dargestellt. Dabei ist das Verhältnis durch den „Type“-Wert in der User-Tabelle definiert. Ein Kunde kann z.B. einen Termin vereinbaren, während ein Admin einen Termin nicht vereinbaren kann (er kann jedoch Termine löschen). Welche Verhältnisse die User genau haben wird in den folgenden Diagrammen dargestellt:

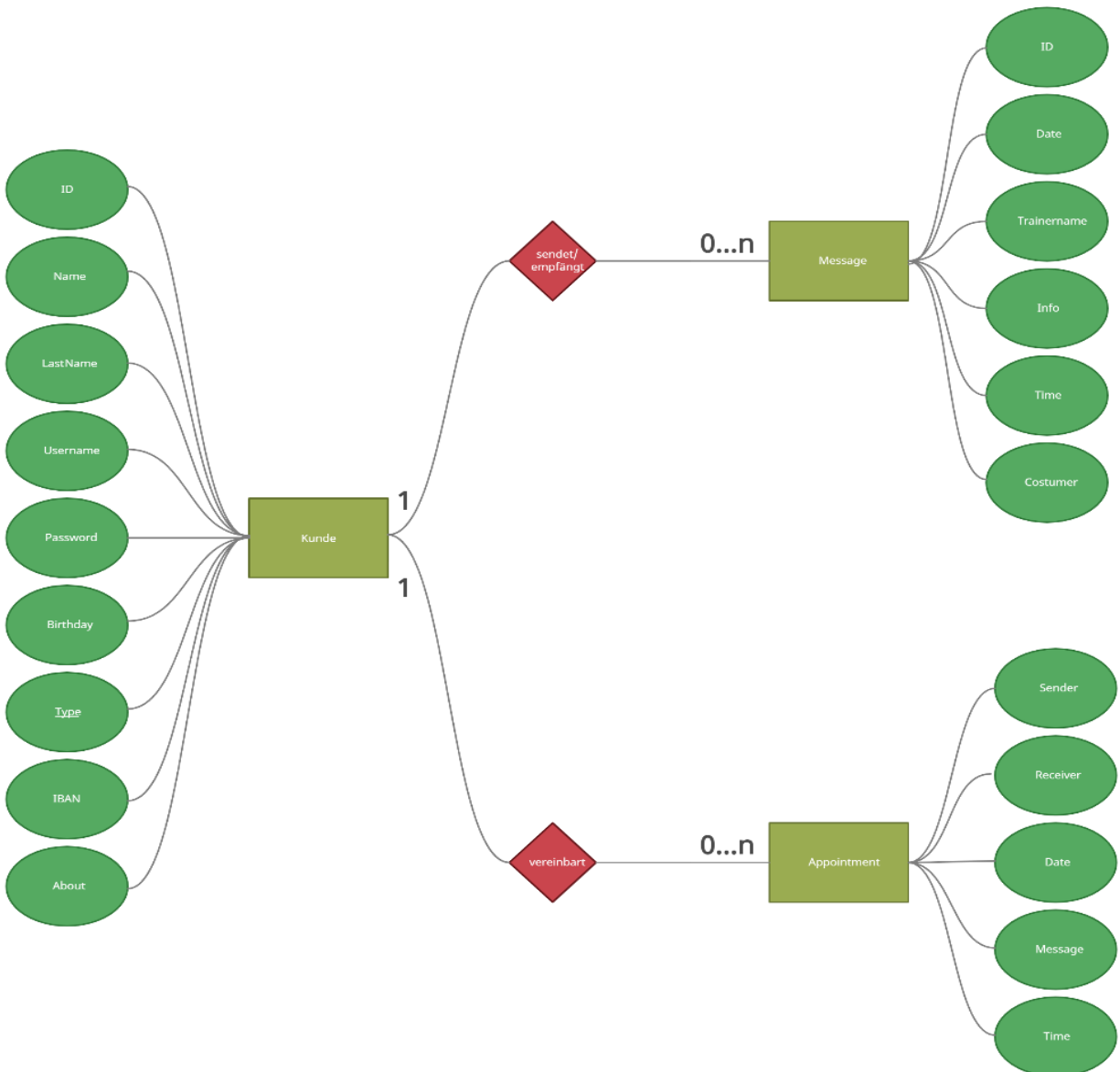


Abbildung 3.1 ER-Diagramm Kunden

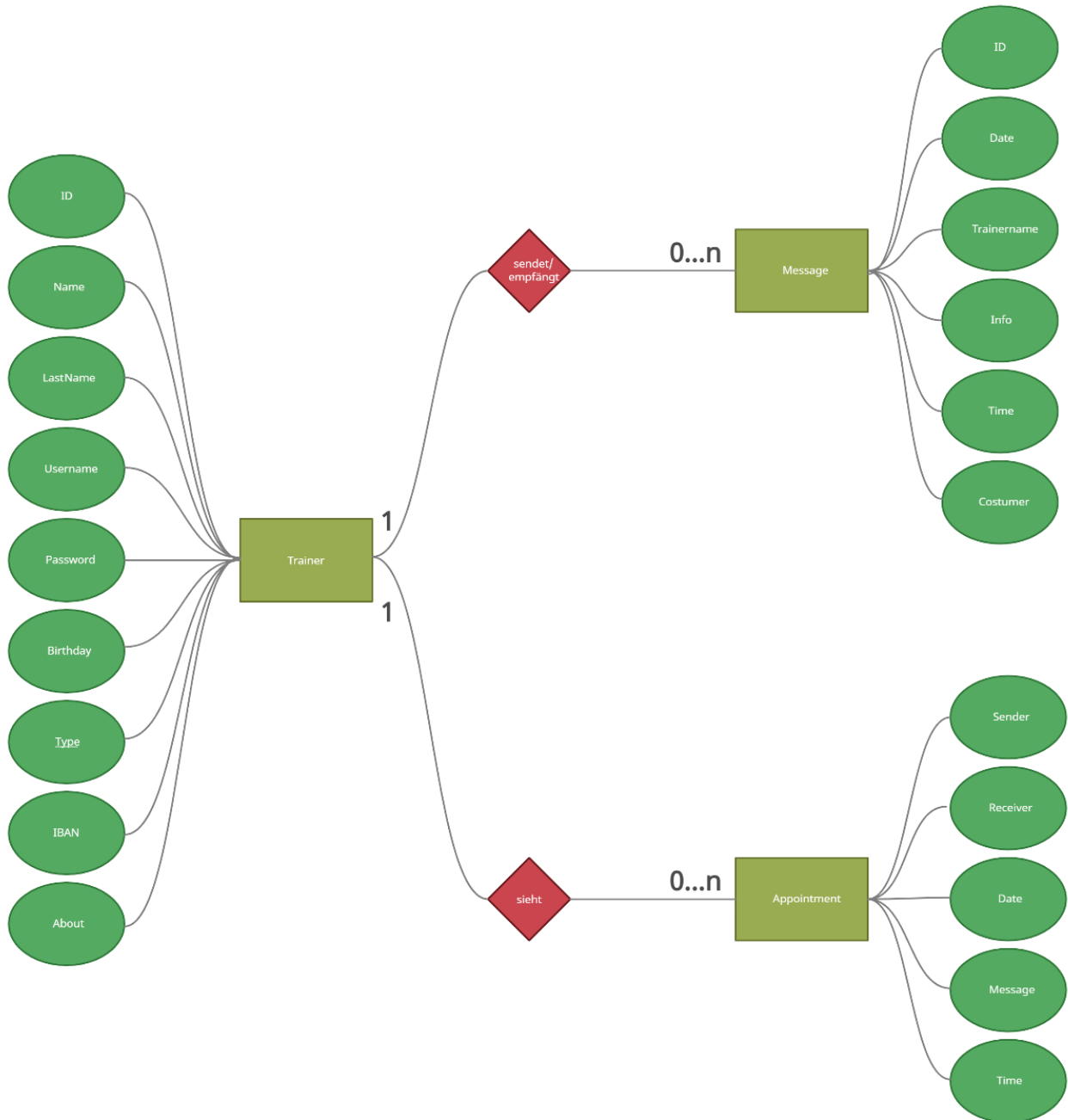


Abbildung 3.2 ER-Diagramm Trainer

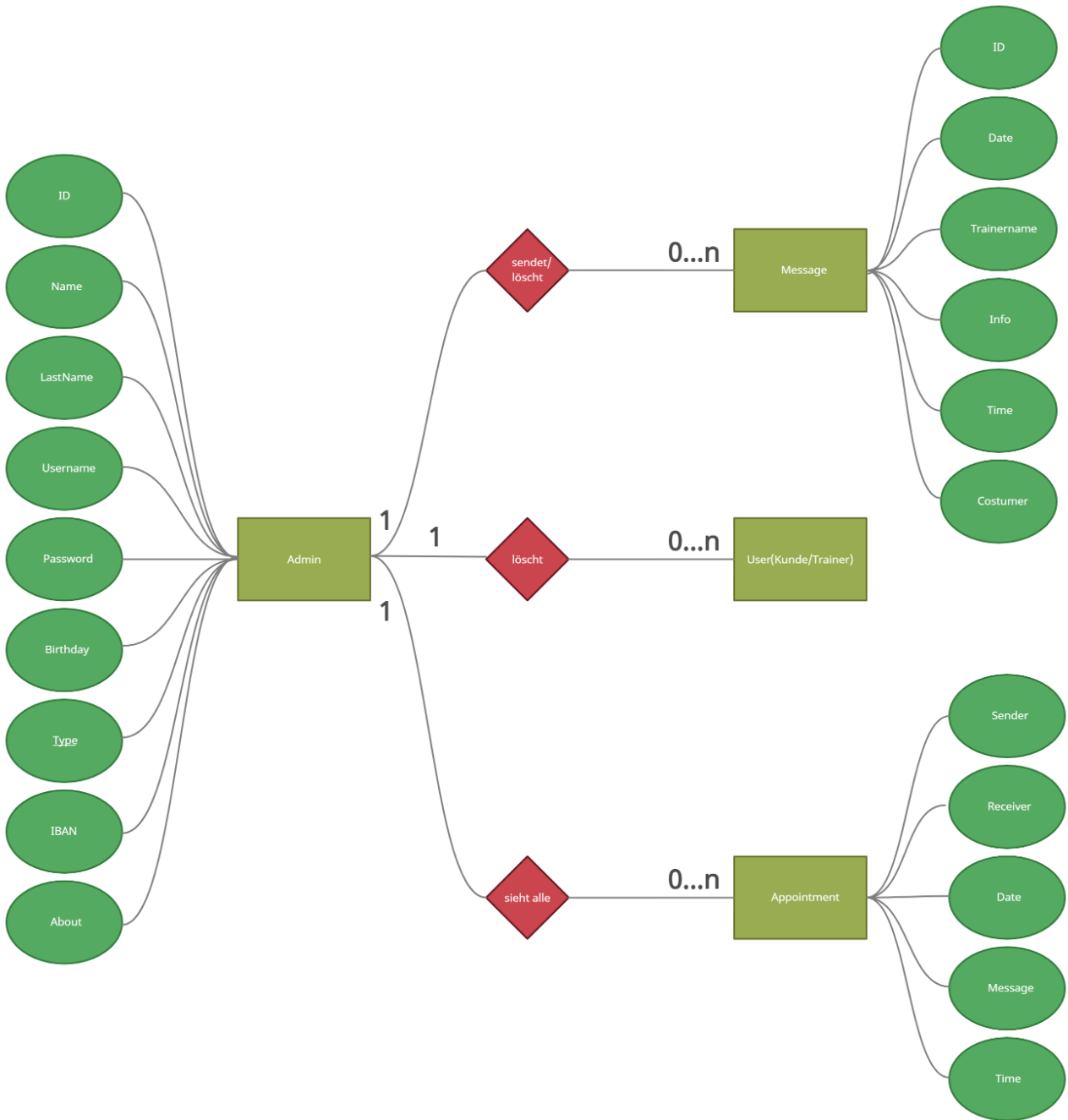


Abbildung 3.3 ER-Diagramm Admin

3.2 Users Tabelle

Beim Registrieren wird ein User in der Datenbank angelegt, dieser besteht aus den in Abbildung 3.4 dargestellten Attributen. Dort werden die üblichen Attribute wie „Username“ und „Password“ benutzt, auch benutzen wir ein „Type“-Attribut, welches beim Registrieren feststellt, ob es sich um einen Trainer/Kunden/Admin handelt, dies ist wichtig, um die Relationen aus 3.1 zu erhalten.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/> 1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
<input type="checkbox"/> 2	name	varchar(128)	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/> 3	lastname	varchar(128)	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/> 4	username	varchar(256)	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/> 5	password	varchar(256)	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/> 6	birthday	date			No	None			Change Drop More
<input type="checkbox"/> 7	type	tinyint(1)			Yes	0			Change Drop More
<input type="checkbox"/> 8	IBAN	text	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/> 9	about	text	utf8mb4_general_ci		No	None			Change Drop More

Abbildung 3.4 Users Tabelle

3.3 Messages Tabelle

Für eine Nachricht wird der Absendzeitpunkt, die Sender-ID sowie Empfänger-ID gespeichert. Ebenfalls wird die Nachricht selbst in Form eines „longtext“ abgespeichert. Das Feld „deleted“, stellt dabei dar, von wem die Nachricht gelöscht wurde.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/> 1	sender	tinytext	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/> 2	receiver	tinytext	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/> 3	date	date			No	None			Change Drop More
<input type="checkbox"/> 4	message	longtext	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/> 5	time	time			No	None			Change Drop More
<input type="checkbox"/> 6	deleted	tinyint(4)			No	None			Change Drop More

Abbildung 3.5 Messages Tabelle

3.4 Appointments Tabelle

Einen Termin kann nur ein Kunde vereinbaren, seine ID wird auch direkt als Attribut des Termins festgehalten. Außerdem wird der Trainername festgehalten (da man genau einen Trainer für einen Termin braucht). Natürlich wird auch die Terminzeit sowie Termininfo abgespeichert

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/> 1	appointment_customerid	int(11)			No	None			Change Drop More
<input type="checkbox"/> 2	appointment_date	date			No	None			Change Drop More
<input type="checkbox"/> 3	appointment_trainername	varchar(30)	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/> 4	appointment_info	text	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/> 5	appointment_time	time			No	None			Change Drop More
<input type="checkbox"/> 6	appointment_customername	text	utf8mb4_general_ci		No	None			Change Drop More

Abbildung 3.6 Appointments Tabelle

4 Use Cases

Aus diesen Anwendungsfällen haben wir Diagramme erstellt, die eine etwas vereinfachte Form eines Use Case Diagramms darstellen. Die Diagramme sollen die Anwendungsfälle unserer FitBridge Webseite visuell darstellen, um diese klarer und visueller zu formulieren.

4.1 Registrierung

Der erste Schritt zur Anwendung unserer Webseite ist die Registrierung der erstmaligen Benutzer. Von Anfang an werden hier die Benutzertypen Administrator, Trainer und Kunde unterschieden. Der Registrierungsprozess ist für alle drei Benutzertypen jedoch weitgehend ähnlich, nur dass die Trainer und Administrator sich über einen geheimen Link registrieren, während sich Kunden direkt über der Index Seite registrieren können.

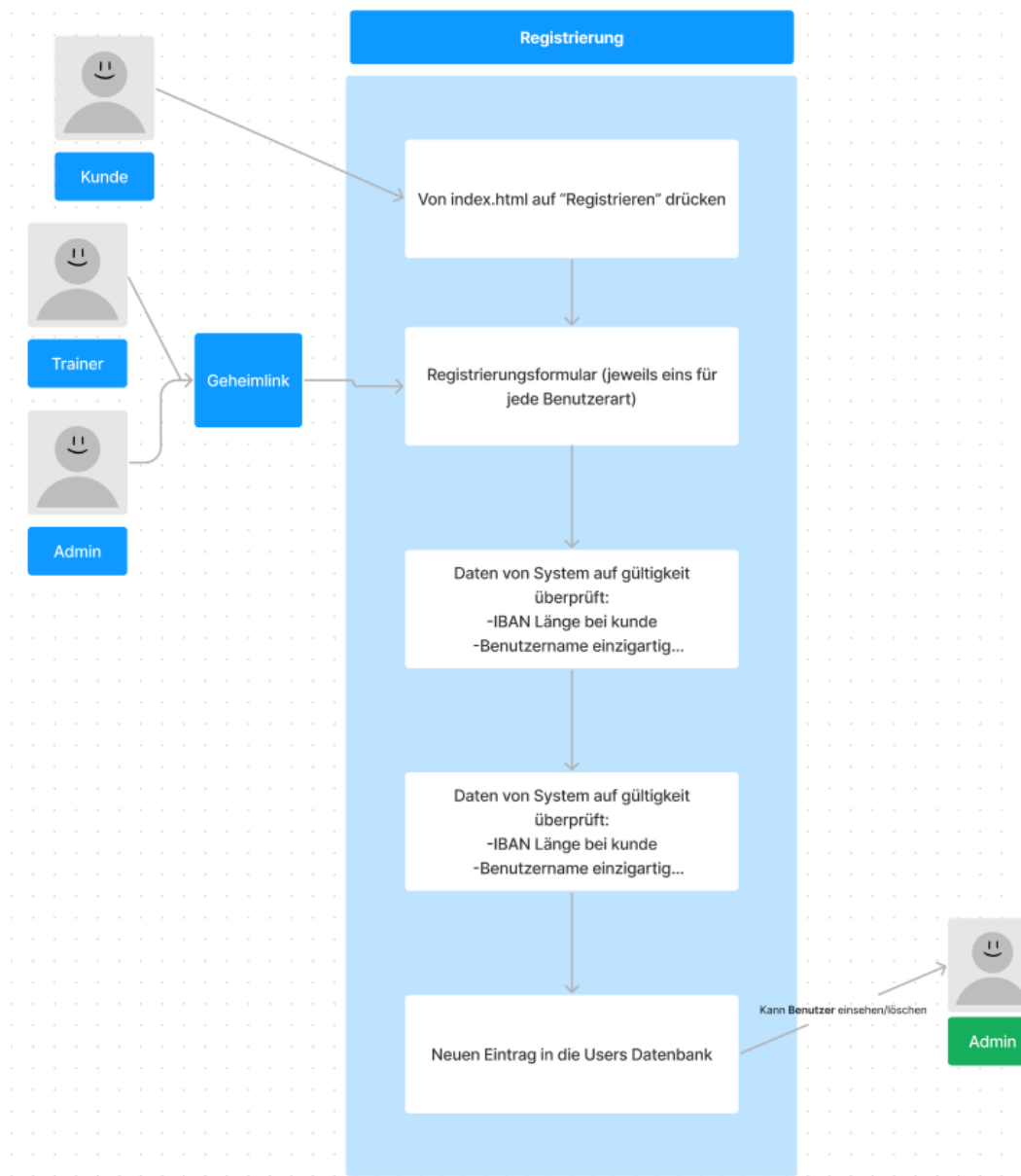


Abbildung 4.1 Registrierungsprozess der jeweiligen Benutzer

4.2 Einloggen und in das Portal gelangen

Es war uns wichtig, dass das Einloggen möglichst einheitlich und einfach für alle Nutzer ist. Daher können sich, anders als bei der Registrierung, alle Benutzertypen über dieselbe Seite einloggen, und die Software übernimmt dann das Weiterleiten zu den jeweiligen Seiten.

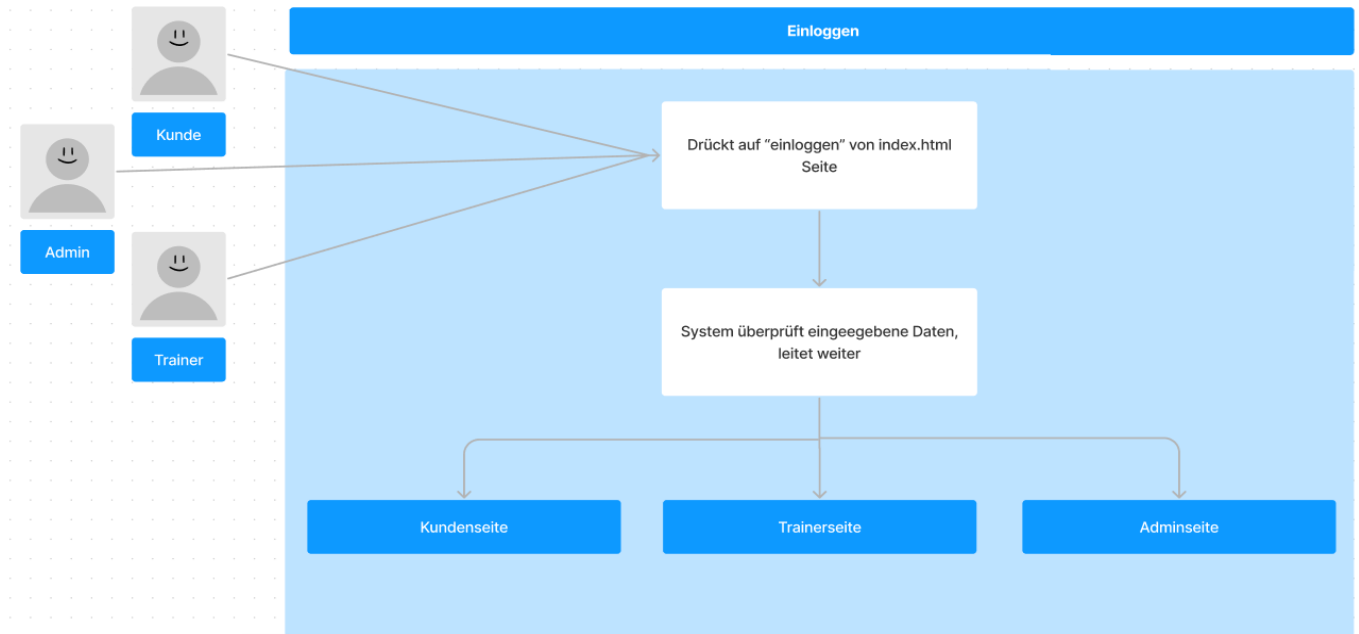


Abbildung 4.2 Use Case Diagramm für das Einloggen

Nach dem erfolgreichen Einloggen werden die Benutzerarten zur zugehörigen Portalseite weitergeleitet.

4.3 Portalseite Kunde

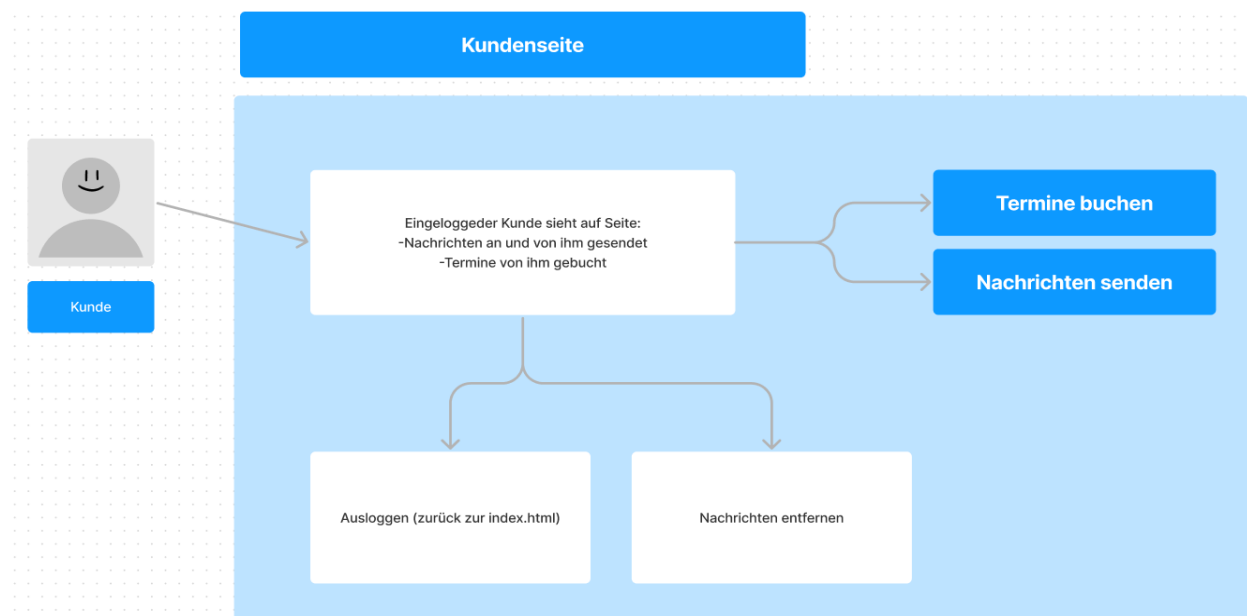


Abbildung 4.3 Use Case Diagramm für die Kundenseite



4.4 Portalseite Trainer

Sichtbar ist, dass die Kunden- und Trainerseiten weitgehend ähnlich sind. Der Unterschied zwischen den beiden ist, dass Kunden Termine Buchen können, was Trainer jedoch nicht können. Theoretisch wäre es auch möglich Trainer Termine buchen zu lassen, jedoch hielten wir es in diesem Kontext für unnötig kompliziert. Beide Seiten können jedoch ihre dazugehörigen Termine einsehen.

Viele möglichen Probleme werden zudem mit der Nachrichten Funktion gebessert, da dadurch eine sehr freie Kommunikation ermöglicht werden kann. Die Nachrichten Funktion erlaubt eine Nutzertyp-unabhängige Kommunikation, die sehr anpassbar aufgebaut ist (weil jeder jedem beliebig Nachrichten senden kann).

Die jeweiligen Portalseiten ermöglichen also den Benutzern, je nach Benutzerart, Zugriff auf verschiedene Funktionen des Systems. Manches soll direkt auf der Seite ausgeführt werden, während manches auf Anderen Seiten ausgeführt werden soll (mithilfe von Redirects). Dies ermöglicht eine bessere Übersicht, ohne dass die Webseiten überfüllt werden.

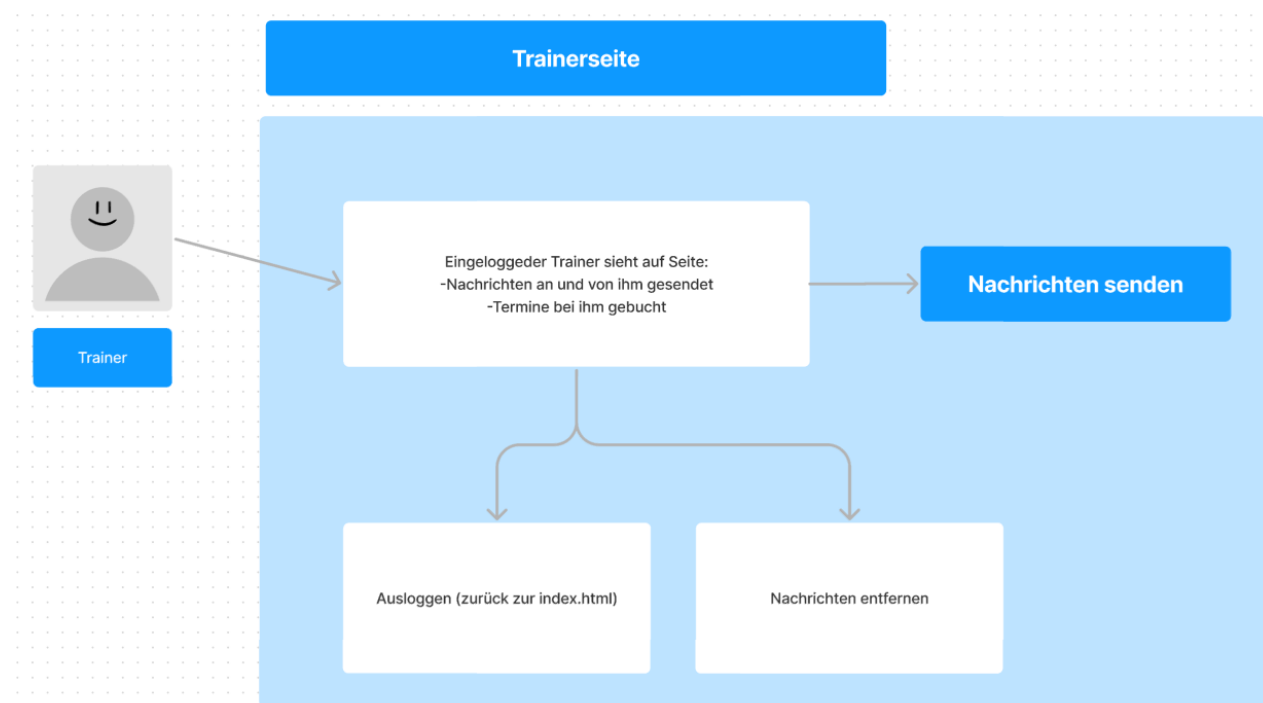


Abbildung 4.4 Use Case Diagramm für die Trainerseite

4.5 Portalseite Admin

Anders sieht es jedoch bei der Administratorseite aus. Administrator können zusätzlich zu den o.g. Funktionen auch alle Termine im System einsehen, sowie Kunden und Trainer aus dem System entfernen (diese werden dann permanent gelöscht). Diese Seite ist gut befüllt da vieles angezeigt werden muss. Dies fordert einiges von dem HTML Skript, welches die vielen Sachen in ordentlicher Art und Weise darstellen muss.

Die Admin Seite (sowie Kunde und Trainer auch) sollen nur von berechtigten Benutzern eingesehen werden können. Bei den Kunden und Trainer Seiten wäre es weniger schlimm, wenn unbefugte Nutzer Zugriff bekämen, jedoch wäre es bei der Admin Seite besonders gefährlich, da hier deutlich mehr einsehbar ist. Zudem kann der Administrator auch andere Benutzer löschen, was auch einige Gefahren mit sich bringt.

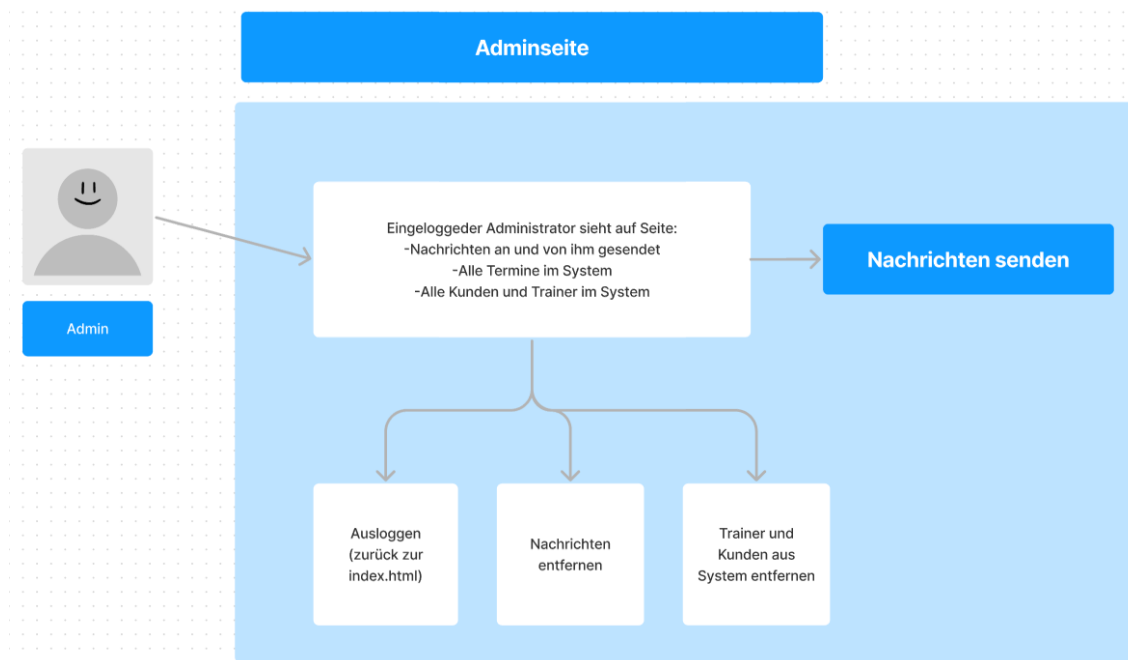


Abbildung 4.5 Use Case Diagramm für die Adminseite



4.6 Terminbuchung und Einsehbarkeit

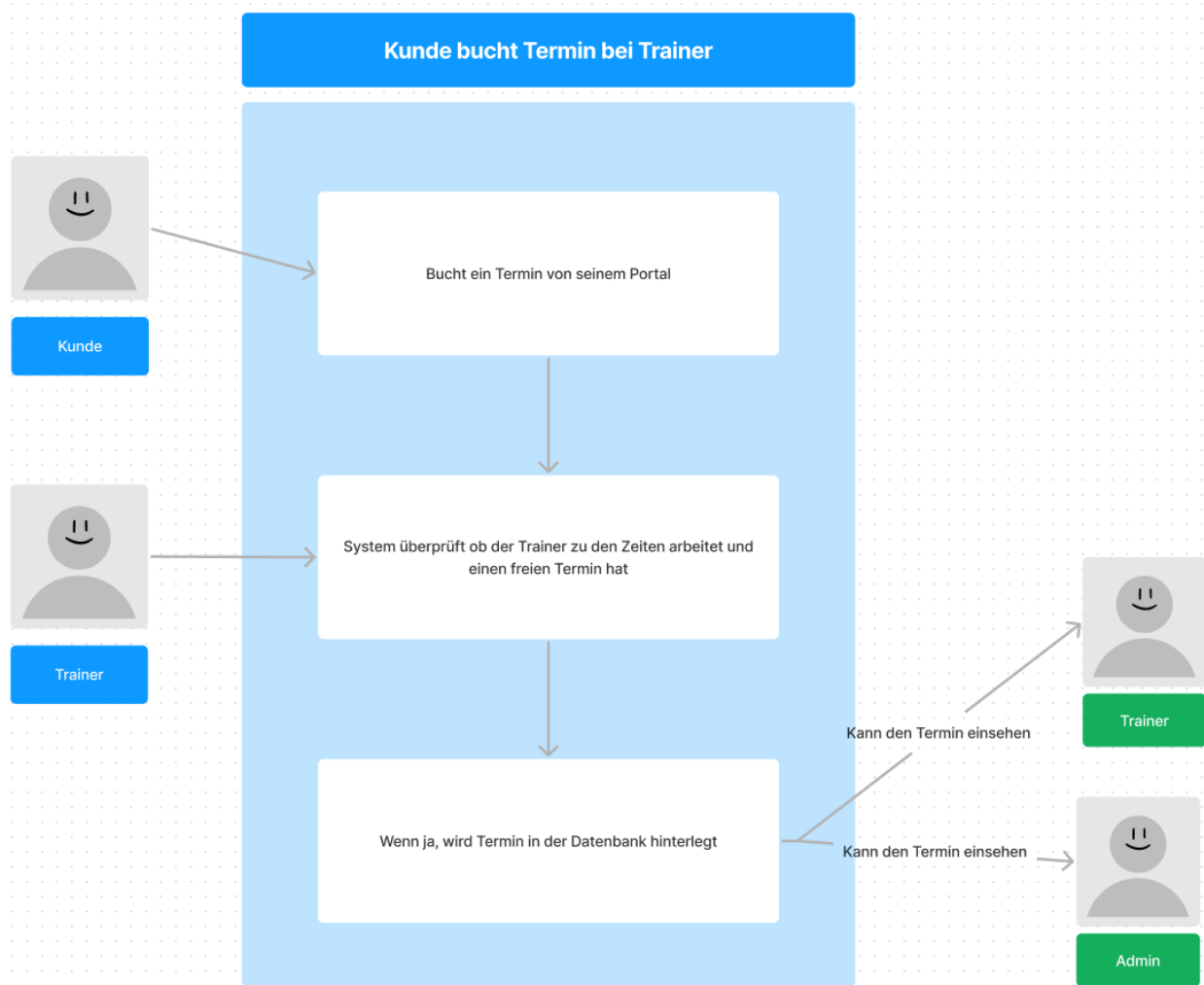


Abbildung 4.6 Use Case Diagramm für das Buchen und Einsehen von Terminen

Da die Terminbuchung nur einseitig erfolgt, ist der dazugehörige Programmablauf relativ einfach. Daher wollten wir Anfangs keine zusätzlichen Funktionen hier einbinden. Bemerkenswert ist, dass der Admin über seinem Portal die gebuchten Termine aller Kunden einsehen kann. Diese Funktion ist zwar nicht unbedingt notwendig, könnte aber anderswo implementiert werden, wo es sinnvoller wäre.

Wichtig ist jedoch, dass das System keine bereits existierenden Termine vergibt, und auch keine in der Vergangenheit gebuchten Termine erlaubt. Zudem müssen die Termine in irgendeiner Form löscher sein. Im Laufe des Projektes haben wir uns dafür entschieden, die Termine erst zu löschen, wenn sie in der Vergangenheit geraten. Somit braucht sich der Benutzer nicht um die Terminverwaltung zu kümmern. Das Absagen der Termine erfolgt dann über das Senden einer Nachricht an den Trainer.

4.7 Nachrichten senden und löschen

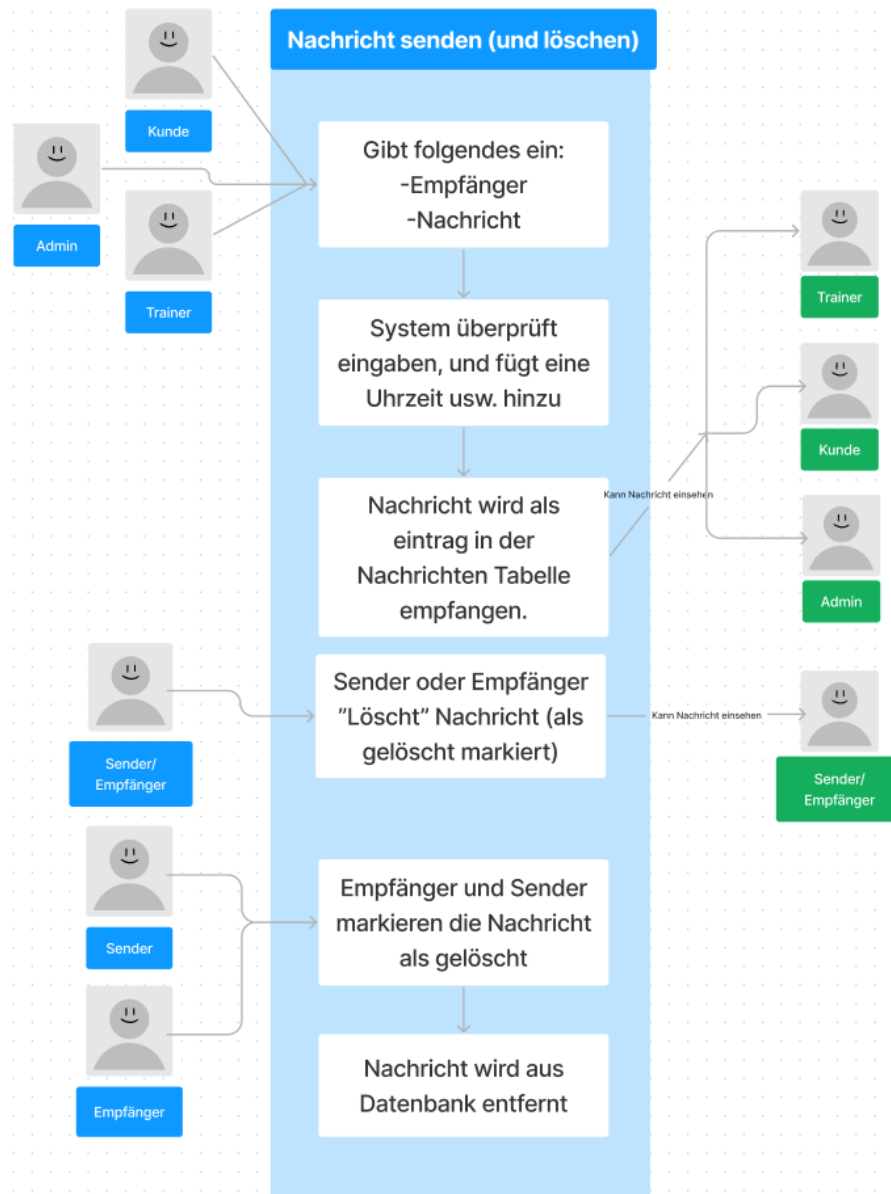


Abbildung 4.7 Use Cases Diagramm des Senden Empfangen und Löschen einer Nachricht

Sichtbar ist, dass die Nachrichten-Sendung erheblich mehr Funktionalität bietet, als die Terminbuchung (die nur einseitig ohne Löschen erfolgt).

Das Interne Nachrichtensystem erlaubt ein beidseitiges Löschen von Nachrichten. Damit sollen die Nachrichten für die jeweiligen Benutzer erstmal nur als gelöscht markiert werden. Gelöscht, bzw. permanent entfernt, sollen diese Nachrichten allerdings nur werden, wenn der Sender und der Empfänger die Nachricht gelöscht haben. Sonst würden die Nachrichten ungewollt verloren gehen. Dieses Verhalten entspricht dem eines E-Mail-Systems.

5 Softwareanforderungen

Um ein paar der wichtigsten Prozesse nochmal visuell darzustellen, haben wir Sequenz-ähnliche Diagramme erstellt. Diese entsprechen jedoch keine offizielle Designvorschrift, und dienen nur zur Veranschaulichung der Funktion unseres Codes.

5.1 Ablauf Einloggen

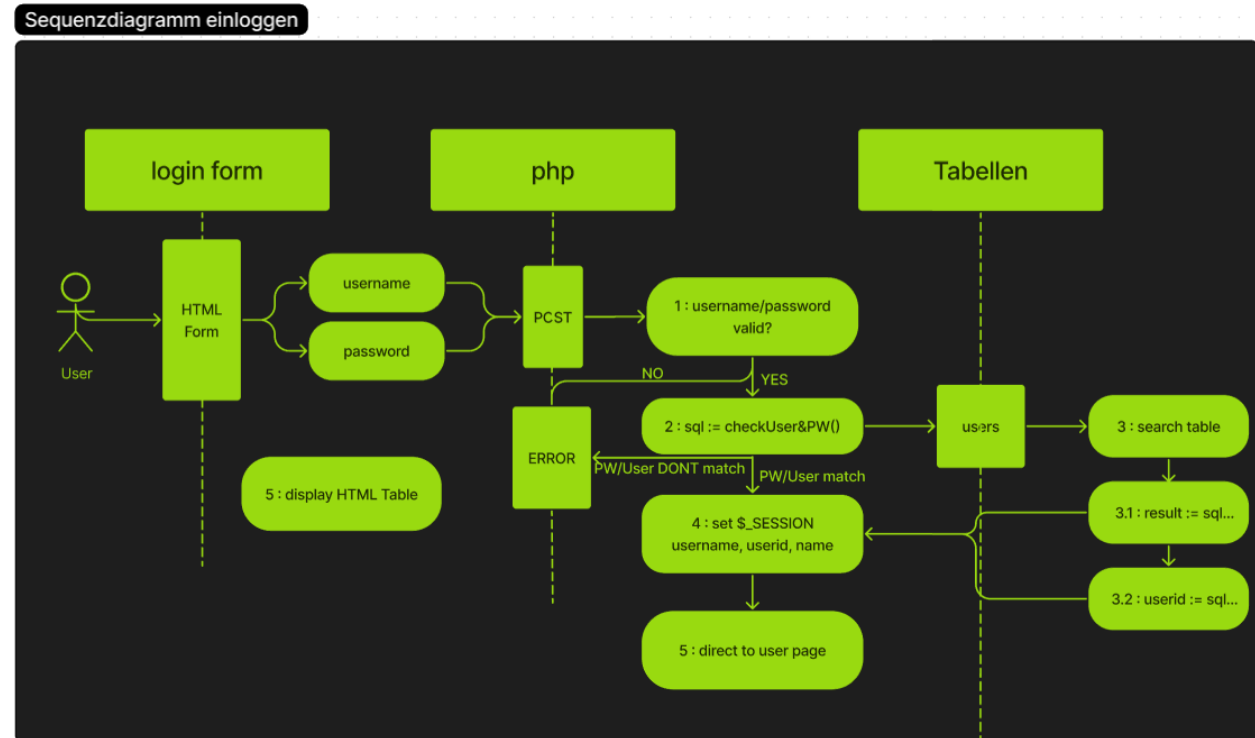


Abbildung 5.1 Sequenzdiagramm für den Login-Vorgang

Das obige Diagramm zeigt, wie das Einloggen in unserem System ermöglicht wird. Eine Überprüfung der eingegebenen Informationen geschieht mithilfe eines Vergleiches mit den in der Datenbank Vorhandenen Benutzer. Sofern die Eingaben mit den Daten in der „users“ Tabelle übereinstimmen, werden sämtliche Informationen über den Benutzer als Session Variablen gespeichert, auf die Spätere Programme Zugriff haben. Diese Session Variablen bilden im Wesentlichen den wichtigsten Aspekt des Einloggens. Die Verwaltung der Fehlermeldungen sowie das Erzeugen und Ausführen der SQL-Befehle sind hier nicht direkt von Interesse, und werden im Abschnitt 8 genauer erklärt.



5.2 Ablauf Nachricht Senden

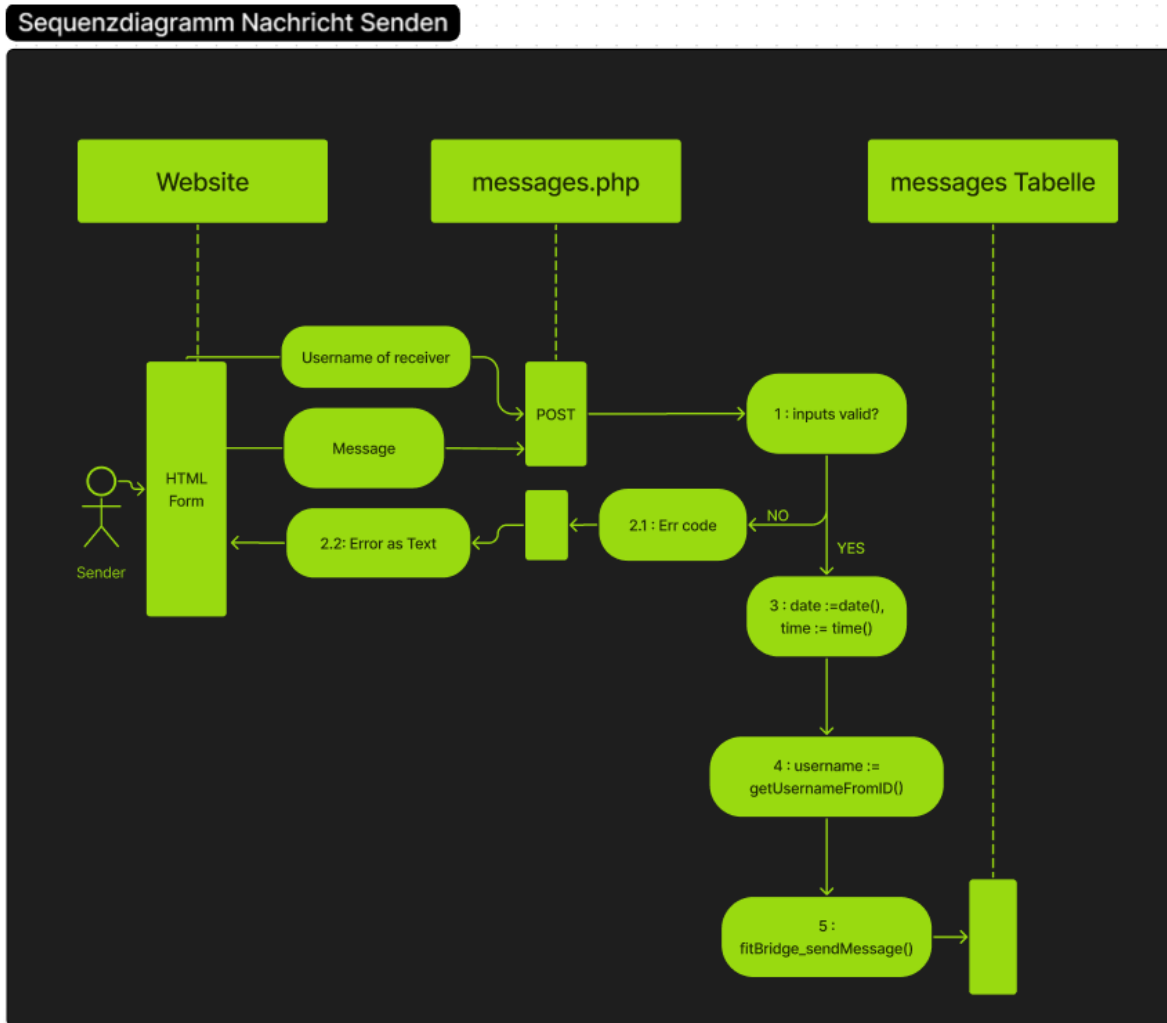


Abbildung 5.2 Ablauf für das Senden einer Nachricht

Beim Senden einer Nachricht werden von dem Benutzer nur zwei Informationen in das Formular übergeben: Benutzername des Empfängers, und die Nachricht an sich. Vorerst werden diese Eingaben auf Korrektheit überprüft, sodass keine Ungültigen Daten weitergegeben werden, und der Benutzer sofort über die Fehleingabe informiert werden kann. Sofern die Eingaben jedoch stimmen, bestimmt das Programm einige zusätzlich benötigte Daten, wie z.B. Datum, Zeit und Benutzername des Senders. Zeitliche Informationen werden mit den `date()` und `time()` Funktionen bestimmt. Der Benutzername des Senders wird aus einer Session Variable, die bereits beim Einloggen festgelegt wurde, bestimmt, um den Sender der Nachricht zu definieren.

Damit sind alle Daten die für eine Nachricht benötigt werden bestimmt, sodass diese an der „`fitBridge_sendMessage()`“ Funktion übergeben werden können. Diese Funktion sorgt dafür, dass die Nachricht in die entsprechende Tabelle eingefügt wird.

5.3 Ablauf Einsehen Termine und Nachrichten auf Kunden oder Trainerseite

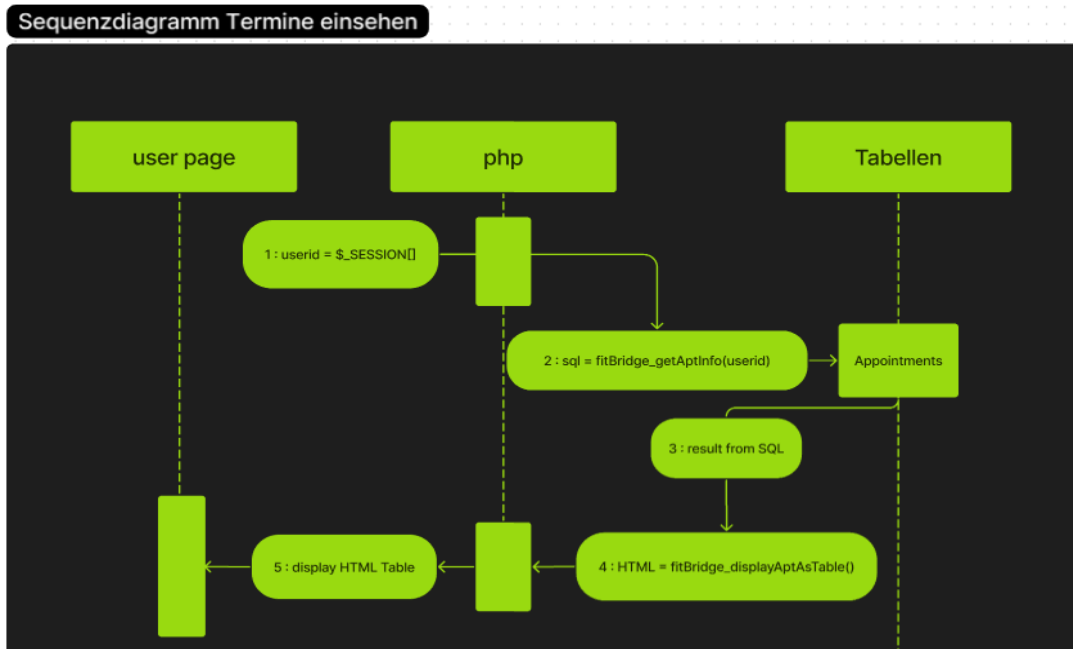


Abbildung 5.3 Termine auf User Page einsehen, als Sequenz-ähnliches Diagramm

Das Einsehen der Termine erfolgt mithilfe eines kleinen PHP Skriptes innerhalb der User Page. Dieses Diagramm stellt, auf sehr vereinfachter Art und Weise, das Darstellen der Termine dar. Das Einsehen von Nachrichten erfolgt auf ähnlicher Art und Weise. Hierzu wurden einige Interne Funktionen angewandt, um den Code zu abstrahieren:



Abbildung 5.4 Einsehen von Nachrichten als Sequenz-ähnliches Diagramm dar

5.5 Ablauf löschen der Nachrichten

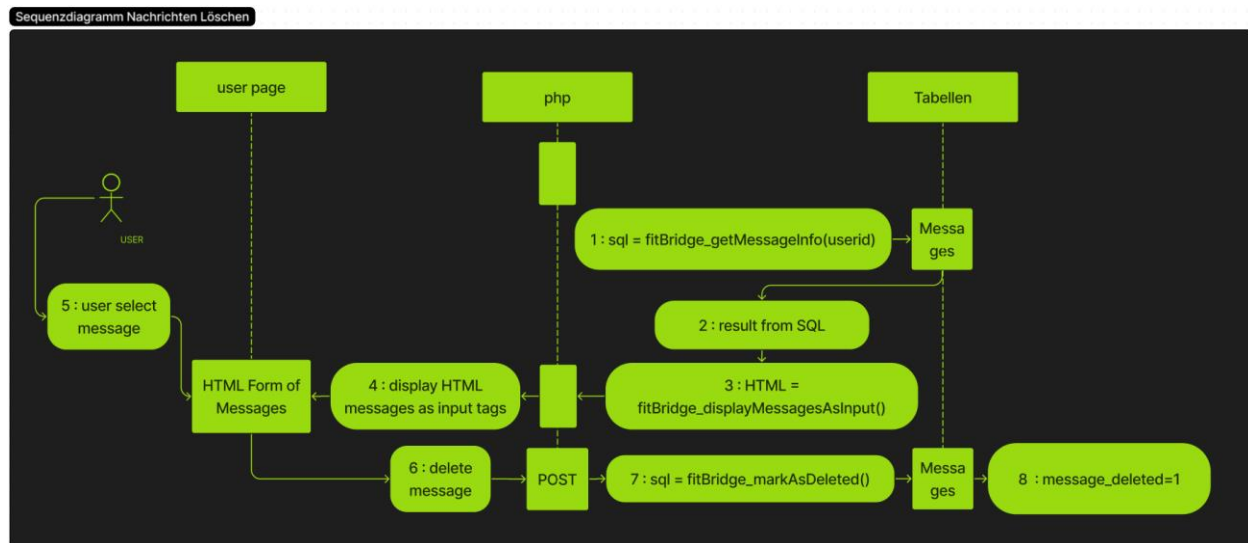


Abbildung 5.5 Ablauf für das Löschen einer Nachricht

Damit der Benutzer Nachrichten möglichst, ohne viel eingeben zu müssen löschen kann, werden die Löschraren Nachrichten als Input Tags eines HTML Formulars dargestellt. Dazu müssen die für den Eingeloggten Nutzer löschraren Nachrichten erst geladen, bzw. rausgesucht werden. Dies geschieht mithilfe einer SQL Query die von der „getMessageInfo()“ ausgeführt wird. Aus dem Query Ergebnis wird innerhalb der „displayMessagesAsInput()“ einen HTML Skript erzeugt, der diese Nachrichten als Input Tags darstellt, die in einem Formular auf der User Page gezeigt werden können. Dies ermöglicht vorerst die Auswahl einer Nachricht von der Benutzerseite. Wenn der Benutzer eine Nachricht ausgewählt hat, wird die Nachricht von der „markAsDeleted()“ Funktion für den Benutzer als gelöscht markiert (bzw. nicht mehr angezeigt). Wirklich gelöscht (also permanent aus der Datenbank entfernt) wird diese Nachricht jedoch erst wenn sie von dem Sender UND dem Empfänger als gelöscht markiert wurde.

5.6 Evaluierung der Diagramme

Wir haben die Diagramme und visuellen Methoden in unserem Projekt nicht erfolgreich anwenden können. Rückblickend hätten uns solche Diagramme jedoch bei der Klärung von den Prozessabläufen deutlich unterstützt, wenn sie besser erstellt gewesen wären. Durch die Mangelnde Planung, ist es in dem Projekt öfters zu nachträglichen Änderungen des grundlegenden Aufbaus der Software gekommen, weil den Testern Probleme aufgefallen sind, die die Programmierer nicht berücksichtigt haben. Dies führte wiederum zu große Effizienzminderung bei der Programmierung.

Wir haben die Sequenz- und Use Cases Diagramme aus Zeitlichen Gründen nicht mehr nachträglich bessern können, da wir andere Aspekte des Projektes priorisieren mussten. Die minderwertige Qualität dieser Diagramme hätte durch bessere Planung, und eine frühere Erstellung (vor dem Programmieren) vermieden werden können.

Trotzdem konnten uns diese Diagramme bei dem Projekt unterstützen. Zudem waren sie bei einem kleineren Projekt nicht unbedingt notwendig.

6 Sitemap

Um diese ganzen Funktionen und Abläufe innerhalb unserer Webseite einordnen zu können, haben wir folgende Sitemap erstellt. Diese gibt ein Überblick der oben beschriebenen Funktionen, aus Anwendersicht.

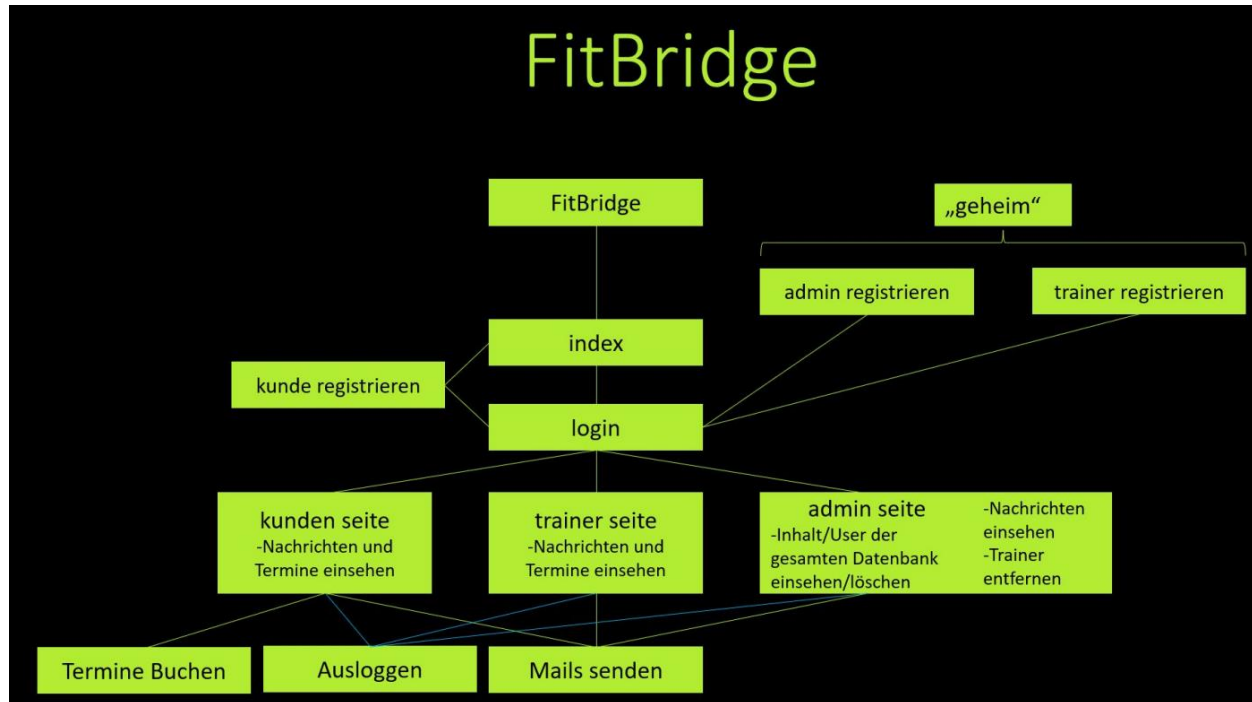


Abbildung 6.1 Ablauf für das Löschen einer Nachricht

Mithilfe der Sitemap wird auch der Registrierungsprozess für Administrator sowie Trainer noch einmal etwas klarer dargestellt. Die Verbindungslinien in dem Diagramm haben nicht nur eine symbolische Bedeutung, da sie (idealerweise) auch so in der Software umgesetzt sind, dass ein Zugriff außerhalb der Linien nicht möglich ist. Das Ganze wird über Session Variablen geregelt, sodass auf den Seiten erst nach erfolgreichem Einloggen zugegriffen werden kann.

Gibt man z.B. den Link für die „user page“ in den Browser ein, ohne eingeloggt zu sein, erkennt das Programm, dass die Session Variable nicht gesetzt ist, und leitet den Nutzer mit einer Warnmeldung zurück zur Homepage.

Anders sieht es jedoch bei den Admin- und Trainerregistrierungsseiten aus. Auf diese Seiten kann jeder mit der Eingabe des entsprechenden Links zugreifen, um sich als Trainer oder Admin zu registrieren.

Vor einem solchen Angriff kann jedoch sehr leicht geschützt werden, z.B. in dem die Seiten nur auf einem lokalen Rechner am Standort des Servers (sofern der Server sich im Fitnessstudio befindet) freigegeben werden. Da die Trainerseite keinen Zugriff auf geschützte Daten ermöglicht, muss diese allerdings nicht geschützt werden. Sollte sich aus Spaß eine fremde Person als Trainer registrieren, kann der Admin diese Person sehr leicht aus dem System entfernen.

Die Sitemap zeigt auch dass, das Senden von Mails für alle eingeloggeden Benutzer über dieselbe Seite stattfindet. Aus diesem Grund kann diese Funktion sehr einfach für verschiedene Anwendungen eingesetzt werden, da nur ein zusätzlicher link platziert werden muss.

7 Der Code

Ziel unseres Entwicklerteams, war es eine möglichst hochwertige, anwendbare Softwareinfrastruktur zu erzeugen, aus der, mit wenig Aufwand, viele Funktionen bereitgestellt werden konnten. Zwar ist unsere Software in vielen Aspekten noch nicht vollständig ausgereift, jedoch ist die Software weitgehend funktionstüchtig und zufriedenstellend.

Bei einer solchen Softwareinfrastruktur, wie die von FitBridge, ist es essenziell einen klaren, einfachen und effizienten Code zu schreiben, der skalierbar, integrierbar und modifizierbar ist. Hierzu müssen von Grund auf einfache und zuverlässige Funktionen geschrieben werden, um wiederholbare Vorgänge zu abstrahieren. Zudem sollen die Funktionen möglichst selbsterklärend sein. Sie sollen zukünftigen Programmierern bei dem Erzeugen von größeren, abstrakteren Funktionen unterstützen, sodass keine Zeit bei wiederholbaren Vorgängen verschwendet wird.

Ganz kontraproduktiv sind hier lange, unverständliche Skripte, von denen wir leider noch einige besitzen. Solche Skripte erschweren den Umgang mit dem Code, und sollten vermieden werden. Somit ist der Schwachpunkt unserer Software, eine Mangelnde Abstraktion. Es sind viele uneinheitliche Teilskripte innerhalb vieler Dateien verteilt, die das Debugging und Testen deutlich erschweren. Zudem ist es schwierig auf diese etwas wirre Software aufzubauen, um neue Funktionen hinzuzufügen.

Um dieses Problem zu bessern, haben wir eine recht große „Renovierung“ unserer Software betrieben. Dabei haben wir einige kleinere und wiederholbare PHP Skripte in Funktionen innerhalb der „phptosql_inc“ Datei ausgelagert, sodass die Prozesse etwas mehr vereinheitlicht und abstrahiert werden konnten. Das Ergebnis ist zwar aus Benutzersicht weitgehend unverändert, jedoch ermöglichte diese Transition eine vereinfachte Integration vieler Funktionen innerhalb anderer Systeme. Zudem wurde ein Automatisiertes Testverfahren in der „test_functions.php“ implementiert (siehe Abschnitt 9) welches ein sehr schnelles Testen der Einzelfunktionen auf verschiedene Eingaben ermöglichte, um Probleme bei der späteren Integration zu vermeiden.

Somit besteht unsere Software im Wesentlichen aus Funktionen der „phptosql_inc“ Datei, die dann in größeren PHP „_inc“ Skripten sowie kleinere, in HTML eingebettete Skripte angewendet werden. Idealerweise gäbe es mehrere Funktionsdateien für verschiedene Anwendungen, sodass es in keinem Skript zu längeren Codeabschnitten kommen würde. Zukünftig werden wir uns an eine Art Maximallänge beim Programmieren halten, sodass keine Funktionen zu lange und unübersichtlich werden können. Zudem sollten alle Einzelfunktionen gründlich getestet werden, bevor sie eingesetzt werden, um späteres debuggen zu erleichtern.

In der aktuellen Version sind noch einige längere, unübersichtliche Code-Abschnitte Vorhanden, die wir aus Zeitlichen Gründen nicht beseitigen konnten. Natürlich hätte das durch einen besseren Programmierstil am Anfang des Projektes vermieden werden können.



7.1 Interne Funktion-Bibliothek „phptosql_inc“

```
* @author Ahmed Sabti, Martin Determann, Bland Azad Saleem, Othmane Kribia
* @license Only to be used by Fitness24 as permitted by contract [LINK TO CONTRACT]
*/

////////////////////////////////////
////////////////////////////////////SET LOCALHOST USER/PASSWORD HERE ///////////////////////////////////
////////////////////////////////////

function fitBridge_getHostname()
{
    $hostname = 'localhost'; //set hostname
    return $hostname;
}

function fitBridge_getUsername()
{
    $username = 'root'; //set username
    return $username;
}

function fitBridge_getPassword()
{
    $password = ''; //set password
    return $password;
}

function fitBridge_getDatabase()
{
    $database = 'fitness_datenbank'; //set database name
    return $database;
}
```

Abbildung 7.1 Ersetzung von Eingaben von Nutzer und Passwort der Datenbank durch kleine Funktion um Ändern des Passwortes zu vereinfachen

Die Abbildung 7.1 zeigt ein positives Beispiel aus unserer Internen Bibliothek. Sichtbar ist, dass die Funktionen eine Benennungsvorschrift folgen, um Kollisionen zu vermeiden. Zudem sind alle größeren, komplizierteren Funktionen mit Doxygen Kommentaren versehen. Dies ermöglicht ein effizienteres Programmieren, da die Anwender der Funktionen mithilfe einer IDE, wie CLion, diese Doxygen Kommentare beim Verwenden der Funktionen einsehen können. In dem Kommentar sind alle Übergabeparameter sowie Rückgabewerte dokumentiert. Die Dokumentation hilft auch zukünftigen Anwendern unserer Plattform ein schnelles Verständnis der Funktionen zu bekommen.



```
/**
 * This function sets the deleted aspect of a message to 1 if deleted by sender and 2 if deleted by receiver. If a message has been deleted by
 * both, the message is removed from DB
 * @param $time /enter the message send time
 * @param $date /enter the message send date
 * @param $val /enter 1 if sender deleted, enter 2 if receiver deleted
 * @param $senderUsername /enter username of the sender
 * @return bool/mysqli_result/void returns -1 for bad connection to DB otherwise returns query result
 */
function fitBridge_markAsDeleted($time, $date, $senderUsername, $val)
{
    $conn = new mysqli(fitBridge_getHostname(), fitBridge_getUsername(), fitBridge_getPassword(), fitBridge_getDatabase());
    if (isset($conn->connection_error)) {
        die('Connection failed : ' . $conn->connect_error);
        return -1;
    } else {
        $sql = "SELECT * FROM `messages` WHERE `sender` LIKE '$senderUsername' AND `time` LIKE '$time' AND `date` LIKE '$date'";
        $result = $conn->query($sql); //Store query result
        $d = 1;
        foreach ($result as $row) {
            if ($row["deleted"] != 0) {
                fitBridge_deleteMessage($time, $date, $senderUsername); //Message permanently deleted
            }
        }
    }
}
```

Abbildung 7.2 „fitBridge_markAsDeleted()“ Funktion

Diese Funktion ist eine etwas umfangreichere Funktion in unserem Projekt, die auf einigen Unterfunktionen aufbaut. Trotzdem soll sie leicht anwendbar sein.

Wir haben unseren gesamten Code vollständig in Englisch programmiert, obwohl die Benutzeroberfläche weitgehend in Deutsch geschrieben ist. Das liegt daran, dass innerhalb unserer Gruppe Englisch als Sprache bevorzugt wird. Zudem hielten unsere Programmierer Englisch generell für eine bessere Sprache, da der Code an sich sowieso auf der englischen Sprache basiert.

```
function fitBridge_markAsDeleted($time, $date, $senderUsername, $val)
{
    $conn = new mysqli(fitBridge_getHostname(), fitBridge_getUsername(), fitBridge_getPassword(), fitBridge_getDatabase());
    if (isset($conn->connection_error)) {
        die('Connection failed : ' . $conn->connect_error);
        return -1;
    } else {
        $sql = "SELECT * FROM `messages` WHERE `sender` LIKE '$senderUsername' AND `time` LIKE '$time' AND `date` LIKE '$date'";
        $result = $conn->query($sql); //Store query result
        $d = 1;
        foreach ($result as $row) {
            if ($row["deleted"] != 0) {
                fitBridge_deleteMessage($time, $date, $senderUsername); //Message permanently deleted
                $d = 0;
            }
        }
        if ($d) {
            $sql = "UPDATE `messages` SET `deleted` = $val WHERE `sender` LIKE '$senderUsername' AND `time` LIKE '$time' AND `date` LIKE '$date'";
            $result = $conn->query($sql); //Store query result
        }
        return $result;
    }
}
```

Abbildung 7.3 Inhalt der „fitBridge_markAsDeleted()“ Funktion

Sichtbar ist, dass der Inhalt der Funktion weitgehend selbsterklärend ist. Der etwas kompliziertere Löschoprozess wird mithilfe der internen „fitBridge_deleteMessage()“ Funktion abstrahiert, sodass dieser Prozess unabhängig von dem Restlichen betrachtet werden kann.



7.2 Beispiel eines Softwareablaufes: Kunde bucht ein Termin bei Trainer

In dem ersten Anwendungsfall soll ein registrierter, eingeloggter Kunde bei einem Trainer ein Termin buchen können. Dies geschieht über der „appointment.php“ Seite:

Abbildung 7.4 Terminbuchungsseite

Um einen Termin zu buchen kann der Kunde das Datum, Uhrzeit Trainer Name sowie zusätzliche Infos in dem Formular (siehe Abbildung #) eingeben.

Die Eingaben für Datum, Uhrzeit und Trainer erfolgen mithilfe von Leisten mit dem entsprechenden HTML und PHP code umgesetzt wurden.

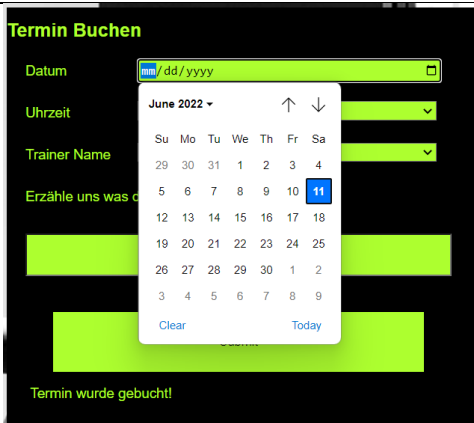


Leiste:	Software:
	<pre><div style="top:10%"> <p>Datum</p> <input type="date" id="appointment_date"> </div></pre> <p>Hier wird ein einfaches HTML „<input>“ Tag verwendet um ein Datum eingeben zu können.</p>
	<pre><div style="..."> <p>Uhrzeit</p> <select name="appointment_time" id="appointment_time"> <option value="08:00:00">8:00</option> <option value="08:30:00">8:30</option> <option value="16:00:00">16:00</option> <option value="17:00:00">17:00</option> </select> </div></pre> <p>Für die Uhrzeit wird auch ein Einfaches „<select>“ Tag verwendet, um Uhrzeiten für ein Termin auszuwählen.</p>
	<pre><div style="..."> <p>Trainer Name</p> <select name="user"> <?php //This is an integrated PHP Script that displays all trainers in system. It i \$conn = new mysqli(...); if (\$conn->connect_error) { echo "<h2>Verbindung Fehlgeschlagen! Kontaktiere unseren Helpdesk!</h2>"; die('Connection failed : ' . \$conn->connect_error); } else { \$sql = "select * from users where type like '%1%'"; //QUERY: All users ma \$result = \$conn->query(\$sql); //Stored in result echo "<option></option>"; if (\$result->num_rows > 0) { //If there were any trainers in the FitBridg foreach (\$result as \$row) { //Display these trainers on website echo "<option value=''>"; echo \$row["username"]; echo "
"; echo \$row["name"]; echo "</option>"; } } } <?> </select> </div></pre>

Abbildung 7.5 Eingabe für eine Terminbuchung und dazugehörige Umsetzung

Sichtbar ist, dass für die Leiste mit den Trainer namen ein PHP skript verwendet wird, der einen SQL Befehl an der Datenbank übergibt, um die Namen aller Trainer rauszusuchen, und dann daraus eine HTML „<select>“ Leiste mit den dazugehörigen „<options>“ erzeugt. Angezeigt wird der Vorname des Trainers, aber ausgewählt wird der Benutzername, da der ein Schlüsselattribut ist.



Bevor aus diesen Eingaben ein Termin erzeugt werden kann, müssen die Eingaben überprüft werden. In diesem Fall gibt es folgende Fehleinabemöglichkeiten:

- 1: Datum in Vergangenheit
- 2: Kein Trainer ausgewählt
- 3: Bereits gebuchten Termin ausgewählt (die möglichkeit könnte auch mit einem PHP Skript der nur die Vorhandenen Termine anzeigt ermöglicht werden, jedoch haben wir es aus zeitlichen Gründen nicht geschafft)

Das Textfeld kann auch leer bleiben, da es nicht unbedingt wichtig ist. Um diese Eingaben zu überprüfen wird folgender Code im PHP Skript eingesetzt:

```
$userId = $_SESSION["userid"]; //Current (logged in) users id from session var
$name = $_SESSION["firstname"]; //Name of currently logged in individual
$appointment_date = date( format: 'Y-m-d', strtotime($_POST['appointment_date'])); //Date entered by customer
$appointment_trainerName = $_POST['user']; //Name entered by customer
$appointment_info = $_POST['appointment_info']; //Info entered by customer
$appointment_time = $_POST['appointment_time']; //Time entered by customer
$appointment_time .= ":00"; //Appends ms to satisfy database format
$_SESSION["warning"] = 0; //In case user books appointment in past
if ($appointment_trainerName=='') {
    //Don't submit to table, notify user of empty fields
    //Goes back to the user site...
    $_SESSION["warning"] = 2;
    header( header: "Location: ../appointment.php");
    exit();
} else if ($appointment_date<date( format: 'Y-m-d')) {
    $_SESSION["warning"]=3;
    header( header: "Location: ../appointment.php");
    exit();
} else { //Submit form etc.
    //Get username from users table:
    $loggedInUsername = fitBridge_getUserInfo( searchBy: 1, $userId, getThisUsers: "username");
    //Add appointment with information entered in form:
    $cErr = fitBridge_addApt($appointment_date, $appointment_time, $appointment_trainerName, $loggedInUsername, $appointment_info);
    if ($cErr==4) {
        $_SESSION["warning"] = 4;
    }
}
```

Abbildung 7.6 Überprüfung der Eingaben, in PHP umgesetzt

Wenn die an dem „appointment_inc.php“ geposteten Felder für Datum oder Benutzername nicht valid sind, werden entsprechende Zahlen in eine Session Variable namens „warning“ gespeichert, und der Skript leitet automatisch wieder zurück zum Formular, wo je nach Wert der Session Variable, eine Fehlermeldung angezeigt wird.

Wenn die Felder jedoch keine invaliden Werte besitzen, wird die ID des eingeloggten Benutzers der „fitBridge_getUserInfo()“ Funktion übergeben, die daraus den Benutzernamen des eingeloggten Benutzers sucht.

Dieser Benutzername wird dann mit den Anderen, vom Formular geposteten Felder, der „fitBridge_addApt()“ Funktion übergeben, die dann überprüft ob es noch keinen Termin mit dieser Uhrzeit gibt, und diesen Termin dann der „Appointments“ Tabelle hinzufügt. Wenn es diesen Termin schon gibt, gibt die Funktion einen Wert zurück der dann in der Session Variable gespeichert wird.



Die Funktion leitet dann wieder zur „appointment_inc.php“ Seite zurück, wo entweder eine Erfolgsmeldung oder, die entsprechende Fehlermeldung angezeigt wird.

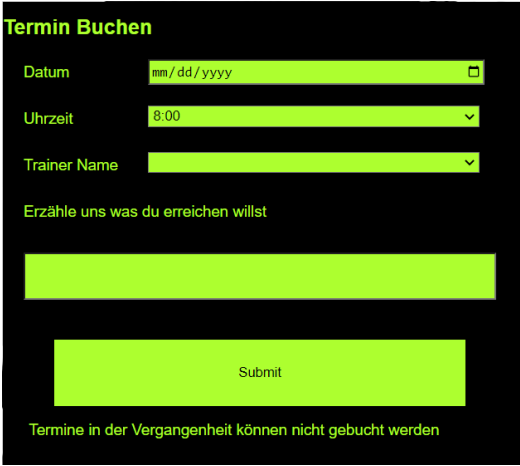
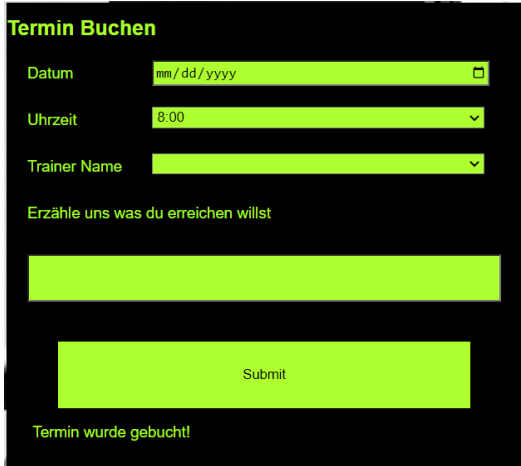
Falsches Datum:	Erfolg:
	

Abbildung 7.7 verschiedene Fehlermeldungen

Hinter den Warnmeldungen steckt folgender PHP skript, der je nach Session Variable eine entsprechende Meldung zurückgibt:

```
<div style="background-color: #f0f0f0; padding: 10px; border: 1px solid #ccc;">
  <?php
  if (isset($_SESSION["warning"])) {
    if ($_SESSION["warning"] == 3) {
      echo "<p>Termine in der Vergangenheit können nicht gebucht werden</p>"; //Notify user
      $_SESSION["warning"] = 0; //Clear session id
    } else if ($_SESSION["warning"] == 2) {
      echo "<p>Bitte keine Felder leer lassen</p>"; //Notify user
      $_SESSION["warning"] = 0; //Clear session id
    } else if ($_SESSION["warning"]==4){
      echo "<p>Dieser Termin ist bereits besetzt!</p>"; //Notify user
      $_SESSION["warning"] = 0; //Clear session id
    } else if ($_SESSION["warning"] == 21) {
      echo "<p>Termin wurde gebucht!</p>"; //Notify user
      $_SESSION["warning"] = 0; //Clear session id
    }
  }
  ?>
</div>
```

Abbildung 7.8 Umsetzung der Fehlermeldungsanzeige

Die Termine werden mit der „fitBridge_addApt()“ Funktion anschließend in die Appointments Tabelle der Datenbank eingefügt.



7.3 Beispiel eines Softwareablaufes: Nachricht senden

Die Funktionen, die für das Nachrichten senden zuständig sind, sind etwas besser aufgebaut als die für die Termine. Das liegt daran, dass diese erst später implementiert wurden, nachdem wir mehr Erfahrung besaßen. Zudem ist das Nachrichten senden für alle Benutzer gleich, das heißt also jeder Nutzer (egal ob Admin, Kunde oder Trainer) von deren Seite mit dem Drücken des Entsprechenden Knopfes zur „message.php“ Seite weitergeleitet wird. Von da aus sehen Nutzer Folgendes Formular:

Abbildung 7.9 Formular für Nachrichten

Dieses Formular ist weitgehend ähnlich zu dem, bereits erklärten, Terminbuchungsformular, nur dass hier die Benutzernamen aller Benutzer im System angezeigt werden, und die Nachrichten Eingabe mit einer „textarea“ als Input realisiert wurde. Sobald auf „Submit“ gedrückt wird, wird dem Formular namens „message_inc.php“ die eingegebenen Daten übergeben. Hier werden wieder die Eingaben vorerst auf leere Felder überprüft:

```
if (($_POST['username'] == '') | ($_POST['message_text'] == '')) {  
    //Don't submit to table, notify user of empty fields  
    //Goes back to the user site...  
    $_SESSION["pastDateWarning"] = 2;  
    header( header: "Location: ../message.php");  
    exit();  
}
```

Abbildung 7.10 Umsetzung der Überprüfung auf leere Felder im Nachrichten Formular

Fehlermeldungen werden, wie in der Terminbuchung, mithilfe von Session Variablen und Anzeigen realisiert. Diese werden auch direkt in der „message.php“ Seite angezeigt. Sofern jedoch keine leeren Felder vorhanden sind, sind keine weiteren Angaben möglich, da die Auswahl an eingaben mit einem PHP Skript und entsprechend resultierenden HTML „<select>“ Tags eingeschränkt wurde. Somit werden die, im Formular eingegebenen, Daten der „fitBridge_sendMessage()“ Funktion übergeben, die daraus einen neuen Eintrag in die Tabelle hinzufügt. Diese Funktion ähnelt der „fitBridge_addApt()“ Funktion in den wesentlichen Aspekten.



```
/**
 * This function adds messages into the messages table after it connects to fitBridge DB.
 * @sql INSERT INTO messages(sender, receiver, date, message, time) values(<params entered to function>)
 * @param $senderUsername /enter username of message sender as a string
 * @param $receiverUsername /enter username of message receiver as a string
 * @param $date /enter date at which message was sent using format: 'Y-m-d'
 * @param $time /enter time at which message was sent using format: 'H:i:s'
 * @param $message /enter the message itself as a string
 * @return bool|int -1 for connection error, sql execution value
 */
function fitBridge_sendMessage($senderUsername, $receiverUsername, $date, $time, $message)
{
    $conn = new mysqli(fitBridge_getHostname(), fitBridge_getUsername(), fitBridge_getPassword(), fitBridge_getDatabase());
    if (isset($conn->connection_error)) {
        die('Connection failed : ' . $conn->connect_error);
        return -1;
    } else {
        //Prepare the necessary fields in the "messages" table:
        $stmt = $conn->prepare( query: "INSERT INTO messages(sender, receiver, date, message, time) VALUES(?, ?, ?, ?, ?)");
        //Bind values entered into form to matching row:
        $stmt->bind_param( types: "sssss", &$var1: $senderUsername, &$var2: $receiverUsername, $date, $message, $time);
        /*Execute the SQL query generated by the prepare() and bind_param() functions:*/
        $execval = $stmt->execute();
        return $execval;
    }
}
```

Abbildung 7.11 „fitBridge_sendMessage()“ Funktion aus internen-Bibliothek

Die Abbildung 7.11 zeigt wie die „fitBridge_sendMessage()“ Funktion eine Nachricht erzeugt, indem sie einen neuen Eintrag in der Nachrichten Tabelle einfügt. Das Ergebnis des Hinzufügen wird in der „\$execval“ Variable gespeichert, die von der Funktion zurückgegeben wird.



Um mithilfe des XAMPP Servers die Daten in der Datenbank zu verwalten oder einzusehen werden sog. SQL Queries verwendet. Der SQL Standard ist die Befehlssprache, die von dem XAMPP Server genutzt wird, um Befehle auf der Datenbank auszuführen. Dieser Standard beinhaltet recht viele verschiedene Befehle, von denen wir allerdings nur ein paar angewendet haben. Das liegt daran, dass wir unseren Code möglichst einfach und verständlich gestalten wollten, und auch keine Notwendigkeit hatten komplexere Queries auszuführen.

8.1 Verbindung zur Datenbank

Um die SQL Queries auf einer bestimmten Tabelle auszuführen, muss vorerst eine Verbindung zur Datenbank auf dem Server (von dem das Skript läuft) erzeugt werden.

```
$conn = new mysqli(fitBridge_getHostname(), fitBridge_getUsername(),  
fitBridge_getPassword(), fitBridge_getDatabase());
```

 (1.1)

Dazu wird ein Objekt des Typs „mysqli“ mit dem PHP „new“ Allokier. Der Constructor des „mysqli“ Objektes benötigt einige Informationen, die für die Verbindung zur Datenbank notwendig sind. Diese wurden mithilfe von einfachen „get()“ Funktionen dem Constructor übergeben. Dieser Objektorientierte Ansatz ist in PHP eine übliche Methode, um auf einer Datenbank zuzugreifen.

8.2 SELECT Befehle und Ausführung in PHP

```
$sql = "SELECT * FROM users WHERE $searchBy LIKE '$searchFor'";  
$result = $conn->query($sql);
```

 (2.1)

Der SQL-Befehl besteht aus einigen, in PHP-Variablen enthaltenen, Strings. In diesem Fall ist die „\$searchBy“ Variable als Parameter einer Funktion definiert, aus der dieser SQL-Befehl ausgeführt wird. Die „\$searchBy“ Variable könnte z.B. sowas wie „username“ oder „userid“ enthalten. Sie beschreibt das Feld aus der „users“ Tabelle was untersucht werden soll. Die „\$searchFor“ Variable enthält den Vergleichswert, der rausgesucht werden soll, als String. Hierbei ist es wichtig zu unterscheiden, ob die Variable direkt (also ohne Anführungszeichen) oder mit Anführungszeichen in dem Befehl geschrieben wird. Zudem können auch andere Zeichen verwendet werden, um verschiedene Spezifikationen zur Genauigkeit der Suche anzugeben, wie z.B. „%1“. Damit kann unterschieden werden, ob das Feld den zu Suchenden String enthält oder entspricht und ob kleinere Abweichungen (wie Groß- und Kleinschreibung) berücksichtigt werden sollen.

Der Befehl an sich wird dann in der „\$sql“ Variable als String gespeichert, um dann mithilfe der „query()“ Funktion auf dem „\$conn“ Objekt bzw. der damit verbundenen Tabelle ausgeführt zu werden. Das Query Ergebnis beinhaltet dann entweder eine gesamte Reihe mit mehreren Informationen, die dann von dem PHP Skript unterteilt werden können, oder eine Fehlermeldung bei erfolgloser Suche. Wenn bei erfolgreicher Suche (keine Systemfehler) nichts gefunden wird, wird auch nichts in der Variablen gespeichert. Dazu wird im Folgenden mehr drauf eingegangen.



Mit der SELECT Methode konnten wir recht viel erreichen. Zum Beispiel besteht auch die Möglichkeit mehrere Vergleiche innerhalb eines einzigen SQL-Befehls auszuführen:

```
SELECT * FROM `appointments` WHERE  
((`username_trainer` LIKE '$trainerUsername')  
AND (`time` LIKE '$time')  
AND (`date` LIKE '$date'))
```

 (2.3)

Der obige Befehl zeigt, wie ein Termin eindeutig über den Benutzernamen des Trainers, die Zeit und das Datum rausgesucht werden kann. Dieser Befehl gibt eine gesamte Reihe aus der Datenbank zurück. Meistens ist hier die gesamte Reihe jedoch überflüssig, da ja der Benutzername, das Datum und die Zeit schon bekannt sind. Das schadet jedoch nicht, da innerhalb des PHP Skriptes das Ergebnis wie folgt unterteilt werden kann:

```
foreach ($result as $row) {  
    $customerUsername = $row["$customer_username"];  
}
```

 (2.)

Hier wird zum Beispiel mithilfe einer PHP „foreach“ Schleife das Ergebnis in eine Reihe mit einzelnen Spalten unterteilt, sodass aus dieser Reihe der Benutzername des Kunde rausgesucht werden kann. Alternativ könnte man aus der Tabelle nur den Benutzernamen des Kunden Raussuchen, und die „foreach“ Schleife umgehen. Der SQL-Befehl wäre dann wie folgt gegliedert:

```
SELECT `username_customer` FROM `appointments` WHERE  
((`username_trainer` LIKE '$trainerUsername')  
AND (`time` LIKE '$time')  
AND (`date` LIKE '$date'))
```

 (2.5)

Die Befehle sind weitgehend ähnlich, jedoch wird in diesem Befehl das Aussortieren des Erwünschten Suchergebnis von der Reihe bereits in dem SQL-Befehl übernommen. Möglicherweise hätte diese Methode ein besseres Laufzeitverhalten.

8.3 DELETE Befehle

Mit derselben Methode wie in 1.1 und 1.2 können auch Queries ausgeführt werden, die keine Suchergebnisse zurückgeben, wie zum Beispiel „DELETE“ Befehle.

```
DELETE FROM `messages` WHERE  
`sender` LIKE '$senderUsername'  
AND `date` LIKE '$date'  
AND `time` LIKE '$time'
```

 (3.1)

Mit diesem Befehl werden Nachrichten nach dem Datum, die Uhrzeit und dem Benutzername des Senders aus der „messages“ Tabelle in der Datenbank gelöscht.



8. UPDATE Befehle

Um einzelne Felder eines Eintrages zu bearbeiten, ohne diese zu löschen, werden sog. „UPDATE“ Befehle benutzt:

```
UPDATE `messages` SET `deleted`=' $val' WHERE  
`sender` LIKE '$senderUsername'  
AND `time` LIKE '$time'  
AND `date` LIKE '$date'
```

(.1)

In diesem Beispiel (.1) werden Nachrichten aus der „messages“ Tabelle als gelöscht markiert. Das `deleted` Feld einer bestimmten Nachricht (die von dem Sender mit Benutzername = „\$senderUsername“, zu dem in den „\$time“ und „\$date“ Variablen festgelegten Zeitpunkt) wird gleich der Variable „\$val“ gesetzt, welche beschreibt, ob diese Nachricht von Sender oder von Empfänger als gelöscht markiert wurde. Diese Variablen beschreiben alle Funktionsparameter der Funktion, die diesen SQL-Befehl ausführt. Mit dem UPDATE Befehl können allerdings keine neuen Reihen in eine Tabelle eingefügt werden.

8.5 INSERT Befehle

Das Einfügen neuer Einträge in die Datenbank geschieht daher mithilfe eines etwas anderen Prozesses als bei den bislang erwähnten Methoden. Das Verbinden zur Datenbank geschieht weiterhin mit demselben, Objektorientierten Vorgang wie in 1.1 beschrieben, woraus ein Objekt namens „\$conn“ erzeugt wird. Der Unterschied liegt jedoch darin dass der SQL Befehl mit zwei separaten Funktionen erzeugt wird, um eine bessere Übersichtlichkeit zu ermöglichen.

```
$stmt = $conn->prepare("INSERT INTO users(name, lastname, username,  
password, birthday, type, about) VALUES(?, ?, ?, ?, ?, ?, ?)");  
  
$stmt->bind_param("ssssiss", $firstname, $lastname, $username,  
$password, $birthday, $type, $about);  
  
$execval = $stmt->execute();
```

(5.1)

Der Codeabschnitt 5.1 zeigt den etwas unterschiedlichen Vorgang bei dem Einfügen eines neuen Feldes in der Tabelle. Der Unterschied liegt darin dass der SQL-Befehl erstmals mit der „prepare()“ Funktion festgelegt wird, wobei dieser direkt mit dem „\$conn“ Objekt verbunden wird. Die einzufügenden Daten werden erstmals leer gelassen, bzw. mit dem „?“ Symbol markiert. Die Felder werden dann im nächsten Schritt mit der „bind_param()“ Funktion in den Befehl integriert, sodass dieser dann mit der „execute()“ Funktion ausgeführt werden kann.

Dieser Ansatz ist Objektorientierter als die bislang erwähnten, und ermöglicht eine Aufteilung des Befehls in zwei Schritten, um einen sehr langen SQL-Befehl zu vermeiden. Grundsätzlich baut er jedoch auf ähnliche Prinzipien auf.

9 Testfälle

9.1 Automatisierte Testverfahren

In unserem Projekt wurden einige Funktionen implementiert, die wir auf z.B. falsche Eingaben oder Parameter testen wollten, bevor wir diese eingesetzt haben. Dazu haben wir ein ausführliches Testprogramm namens „test_functions.php“ geschrieben. Es kann über XAMPP in einem Browser ausgeführt werden (als Webseite geladen). Die Ergebnisse (sowie Fehlermeldungen) der Tests können direkt im Browser eingesehen werden. Ziel dieses Dokumentes war es das Implementieren von Funktionen Innerhalb des Programmes zu vereinfachen. Das Dokument ist mit vielen Kommentaren versehen, sodass zusätzliche Testfunktionen leicht hinzugefügt werden können. Wie bereits erwähnt, haben wir alle Funktionen der internen „phptosql_inc“ Bibliothek gründlich getestet, bevor diese in größere Programme implementiert wurden. Um dieses Testen möglichst einfach zu gestalten, haben wir dieses Dokument erzeugt.

```
////////////////////////////////////  
/// ABLAUF: //////////////////////////////////////  
/// 1) add valid user "Bob Jeffries" to users table using fitBridge_addUser();  
/// 2) add valid user "Trainer TT" to users table using fitBridge_addUser();  
/// 3) attempt to add identical user (same username) using fitBridge_addUser();  
/// 4) search for "Bob Jeffries" in users table using fitBridge_getUserInfo();  
/// 5) search for nonexistent user using fitBridge_getUserInfo();  
/// 6) search with invalid parameters using fitBridge_getUserInfo();  
/// 7) Create an appointment using Bob and Trainer's infos  
/// 8) Create a second valid appointment  
/// 9) Attempt to create an identical appointment  
/// 10) Search for all appointments belonging to Ttrainer (by username)  
/// 11) Display all appointments belonging to Ttrainer using displayAppointments()  
/// 12) Send a mail from Ttrainer to Bob Jeffries (by usernames)  
/// 13) Search for mail as Bob Jeffries (by Bob's Username)  
/// 14) Mark message as deleted by Bob's username  
/// 15) Search for the deleted message as Bob post deletion  
/// 16) Delete message as trainer  
/// Clean all tables  
  
//1) Create a new customer in users table  
echo "Creating Bob Jeffries as a valid user with fitBridge_addUser<br>";  
$firstname = "BOB";  
$lastname = "Jeffries";
```

Abbildung 9.1 Anfang des automatisierten Testverfahrens

Die obige Abbildung zeigt den Ablauf des automatisierten Testverfahrens als Kommentar. Das Ganze Verfahren ist im code einheitlich kommentiert, um das Finden von Fehlern zu erleichtern. Sämtliche Testergebnisse werden mit der „echo()“ Funktion in dem Browser angezeigt.

9.2 Testverfahren aus Benutzersicht/Einrichtung

Da wir ein System aufbauen was von Benutzern verwendet werden soll, war das wichtigste Element der Testverfahren, die nicht-automatisierten Verfahren, um das endgültige Verhalten der Seite zu erproben. Daher haben wir ein ausführliches Testverfahren entwickelt, um das gesamte System aus Benutzersicht auf möglichst viele Fälle zu erproben. Im Laufe der Entwicklung wurden einige Testschritte hinzugefügt und geändert. Dieses Testverfahren dient auch als eine Art Bedienungsanleitung für unsere fiktiven Kunden.

1) TESTFALL 0: Einrichten der Datenbank

- 2) XAMPP vollständig einrichten: [//www.youtube.com/watch?v=081xcYZKOZA](https://www.youtube.com/watch?v=081xcYZKOZA)
- 3) Datei entpacken:
- 4) Die „fitbridge.zip“ Datei in dem „htdocs“ Folder mit dem Verzeichnis „<...>/xampp/htdocs“ entpacken
- 5) Von der „localhost“ Startseite auf „phpMyAdmin“ drücken
- 6) (Von „phpMyAdmin“ Seite) den „+ Neu“ Knopf in der Leiste auf der Linken Seite drücken, um eine neue, leere Datenbank zu erzeugen.
- 7) Als Namen der Datenbank „fitness_datenbank“ eingeben.
- 8) Auf Erstellen drücken sodass eine neue, leere Datenbank eingerichtet wird.
- 9) Von der neuen Datenbank auf „import“ drücken
- 10) Auf „Datei Auswählen“ drücken
- 11) Die „fitness_datenbank.sql“ Datei aus dem Verzeichnis „<...>/xampp/htdocs/fitbridge/fitness_datenbank.sql“ auswählen.
- 12) Nun sollten die Drei Tabellen „users“, „messages“ und „appointments“ einsehbar sein.
- 13) Bevor die Frontend geladen werden kann, müssen zuerst die Zugriffsdaten zu Ihrem XAMPP Server festgelegt werden, sofern diese dem Standard abweichen, **sonst ist dieser Schritt unnötig.**
- 14) Öffnen Sie dazu die Datei:
„<...>/xampp/htdocs/fitbridge/inc/php_sql_inc.php“
und geben Sie die von Ihnen festgelegten Werte überall ein wo „set ...“ als Kommentar steht
- 15) Geben Sie in einem beliebigen Browser „localhost/fitbridge“ ein.
- 16) Sofern die Index Seite sichtbar ist, sollten Sie mit den Testfällen beginnen können.



18) TESTFALL 1: Neues Fitnessstudio (leere Datenbank) registriert den ersten Admin/Trainer

19) Eingeben von:

„localhost/fitbridge/register_admin.php“

um zur Admin Registrierungsseite zu kommen

20) Nun können die Felder ausgefüllt werden. Folgende Eingabefehler sollten von dem System verhindert werden:

- Leeres Passwort, Leerer Vor- oder Nachname, Leerer Benutzername
- Bereits existierender Benutzername
- Geburtsdatum in Zukunft

Eine entsprechende Fehlermeldung sollte auch direkt in der Registrierungsseite angezeigt werden.

21) Eingeben von:

„localhost/fitbridge/register_trainer.php“

um zur Trainer Registrierungsseite zu kommen.

22) Nun können die Felder ausgefüllt werden. Folgende Eingabefehler sollten von dem System verhindert werden:

- Leeres Passwort, Leerer Vor- oder Nachname, Leerer Benutzername
- Bereits existierender Benutzername
- Geburtsdatum in Zukunft

Eine entsprechende Fehlermeldung sollte auch direkt in der Registrierungsseite angezeigt werden.

23) TESTFALL 2: Datenbank ist ‚eingrichtet‘ (mind. 1 Trainer/Admin in DB vorhanden), Kunden können sich registrieren

24) Webseite kann nun vom fiktiven FitBridge Kunde veröffentlicht werden. Somit können sich die ersten Kunden des Fitnessstudios anmelden.

25) Kunden geben:

„localhost/fitbridge/index.html“

im Browser ein, um zur Homepage zu kommen.

26) Auf „Registrieren“ drücken, um zur Kunden-Registrierungsseite zu gelangen

27) Nun können die Felder ausgefüllt werden. Folgende Eingabefehler sollten von dem System verhindert werden:

- Leeres Passwort, Leerer Vor- oder Nachname, Leerer Benutzername
- Bereits existierender Benutzername
- Geburtsdatum in Zukunft
- IBAN zu kurz (mind. 22 Ziffern)

Eine entsprechende Fehlermeldung sollte auch direkt in der Registrierungsseite angezeigt werden.



29) TESTFALL 3: Neukunde logged sich das erste Mal ein

30) Von der Registrierungsseite (oder von index Seite) zur login.html Seite gelangen.

31) Die entsprechenden Felder ausfüllen. Folgende Eingabefehler sollten von dem System verhindert werden:

- Nicht existierender/Falscher Benutzername/Leere Eingabe (wird gleichbehandelt)
- Falsches Passwort/Leeres Passwort (auch gleichbehandelt)

Diese Fehlermeldungen werden, im Gegensatz zu den anderen, auf einer anderen Seite angezeigt, und der Benutzer muss sich zurück-klicken.

32) Bei richtiger Eingabe sollte der Kunde eingeloggt werden, und an die „user_page.php“ Seite weitergeleitet werden. Von da aus können Termine gebucht werden.

33) TESTFALL 4: (Testfälle 0,1,2,3 sind Voraussetzung) Kunde Bucht Termin

34) Um ein Termin zu buchen können Kunden auf dem „Termin Buchen“ Knopf drücken

35) Damit werden sie zur „appointment.php“ Seite weitergeleitet. Von da aus können die Felder entsprechend ausgefüllt werden. Folgende Eingabefehler sollten von dem System verhindert werden:

- Kein Trainer ausgewählt (falscher Trainer nicht möglich)
- Datum in der Vergangenheit
- Bereits existierenden Termin bei eingegebenem Trainer

Diese Fehlermeldungen sollten direkt auf der Seite angezeigt werden.

36) Nach der erfolgreichen Terminbuchung sollte der Benutzer zurück zur „user_page.php“ Seite zurückgeleitet werden. Dort sollten die Termine auch gleich angezeigt werden.

37) TESTFALL 5: (Testfälle 0,1,2,3 sind Voraussetzung) Kunde schickt Nachricht an Trainer

38) Dazu kann der Kunde direkt von der „user_page.php“ (nach erfolgreichem Einloggen) auf dem „Nachricht Senden“ Knopf drücken.

39) Der Kunde sollte zur „message.php“ Seite weitergeleitet werden.

40) Von da aus kann der Kunde das Formular entsprechend ausfüllen. Das System sollte gegen folgende falsche Eingaben schützen:

- Kein Empfänger gewählt (falscher Empfänger nicht möglich)
- Textfeld leer
- Eine entsprechende Fehlermeldung wird direkt auf der Seite angezeigt.

Bei erfolgreicher Eingabe wird der Benutzer zurück zur „user_page.php“ geleitet.

41) Nun sollte die Nachricht einsehbar sowie löscher sein (bitte noch nicht löschen da es für die nächsten Tests noch benötigt wird).

42) Kunde ausloggen

43) TESTFALL 6: (Testfälle 0,1,2,3 sind Voraussetzung) Trainer loggt (zum ersten Mal) ein

44) Von der „index.html“ Seite auf dem „Einloggen“ Knopf drücken

45) In der „login.html“ Seite die Daten des in TESTFALL 2 erstellten Trainers eingeben, um einzuloggen.

46) Nach erfolgreichem einloggen sollten Trainer automatisch zur „trainer_page.php“ Seite weitergeleitet werden. Von da aus können Trainer Termine einsehen und Nachrichten Schicken.

47) Der in TESTFALL 3 gebuchte Termin (oder Termine) sollten nun einsehbar sein.

48) Die in TESTFALL gesendete Nachricht sollte auch einsehbar sein.



49) TESTFALL 7: (Testfälle 0,1,2,3 sind Voraussetzung) Trainer schickt Kunde eine Nachricht

- 50) Als eingeloggter Trainer auf „Nachricht Senden“ drücken
- 51) Von der entsprechenden Seite aus an dem in TESTFALL 2 erstellten Kunde eine Nachricht schicken.
- 52) Trainer sollte wieder zurück zur „trainer_page.php“ Seite zurückgeleitet werden von wo aus dieser Nachricht, sowie die in vorherigen Testfall vom Kunde geschickte Nachricht einsehbar sein sollten.

53) TESTFALL 8: (Testfälle 0,1,2,3,6/7 sind Voraussetzung) Trainer löscht alle Nachrichten

- 54) Nun soll kann der Trainer beide Nachrichten (nach Augenschein auf gleicher Art und Weise) löschen in dem er auf dem entsprechenden Knopf unter Nachrichten löschen drückt (ganz unten Auf der Seite, ist nicht direkt sichtbar).
- 55) Als Trainer ausloggen

56) TESTFALL 9: (Testfälle 0..8 sind Voraussetzung) Kunde sieht die vorherigen Nachrichten

- 57) Der eingeloggte Kunde kann nun von der „user_page.php“ aus die von ihm gesendete Nachricht sehen, kann diese auch löschen.
- 58) Kunde ausloggen

59) TESTFALL 10: (Testfälle 0..8 sind Voraussetzung) Admin loggt sich erstmalig ein

- 60) Von der index.html Seite aus auf „einloggen“ drücken
- 61) Die Daten von dem in TESTFALL 1 registrierten Admin in das „login.html“ Formular eingeben um als Admin eingeloggt zu werden.
- 62) Nun sollte Admin direkt zur „admin_page.php“ Seite geleitet werden.
- 63) Von der „admin_page.php“ Seite können auch unter Verwendung derselben Nachrichten Funktion Nachrichten geschickt sowie gelöscht werden, alle Termine können eingesehen werden und alle Trainer sowie Kunden sollten einsehbar und löscher sein.

64) TESTFALL 11: (Testfälle 0..10 sind Voraussetzung) Admin löscht alle Kunden und Trainer

- 65) Von der „admin_page.php“ Seite können nun die in den Vorherigen Anwendungsfällen erstellten Kunden/Trainer gelöscht werden. Idealerweise sollten auch die dazugehörigen Termine mit gelöscht werden, das ist aber noch nicht vollständig implementiert.
- 66) Admin loggt sich aus.

67) TESTFALL 12: nicht-eingeloggter/unberechtigter Nutzer versucht auf einer Webseite zuzugreifen

- 68) In einem frischen Browser-Fenster (oder ausgeloggten Zustand) die URL:
„localhost/fitbridge/admin_page.php“
oder eine beliebige andere Seite, auf der kein Zugriff gewährt ist, eingeben. Es soll nun eine Fehlermeldung erscheinen und weiteres Vorgehen verhindert werden.