Mathematisch-Naturwissenschaftliche Fakultät

Human-Computer Interaction

Bachelorarbeit

Erstellung und Bewertung einer Webplattformbasierten Datenhaltungs- und Bewertungssoftware für Studien

Eberhard Karls Universität Tübingen
Mathematisch-Naturwissenschaftliche Fakultät
Wilhelm-Schickard-Institut für Informatik
Human-Computer Interaction
Roufayda Salaheddine, roufayda.salaheddine@student.uni-tuebingen.de, 2020

Bearbeitungszeitraum: 15.08.2020-14.12.2020

Betreuer/Gutachter: Prof. Dr. Enkelejda Kasneci, Universität Tübingen
Zweitgutachter: Dr. Wolfgang Fuhl

Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe und dass alle Stellen, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen sind, durch Angaben von Quellen als Entlehnung kenntlich gemacht worden sind. Diese Bachelorarbeit wurde in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt.

Roufayda Salaheddine (Matrikelnummer 4116819), 9. Dezember 2020

Abstract

Aufgrund der zunehmenden Nutzung und Felxibilität des Internets, hat sich der Einsatz von Webanwendungen in vielen Anwendungsgebieten etabliert. Vor allem in der empirischen Forschung, z.B. in Bereichen des maschinellen Lernens oder der Mensch-Computer-Interaktion, findet eine steigende Integration von Webanwendungen in den Studienbetrieb statt, um schnell und flexibel große Datensätze verwalten und auswerten zu können. Hierzu ist es wichtig, eine Datenhaltungssoftware so zu gestalten, dass eine qualitativ hochwertige und effiziente Speicherung der erhobenen Studiendaten gewährleistet wird. Das Ziel dieser Arbeit ist die Entwicklung und Bewertung einer Webplattformbasierten Datenhaltungs- und Bewertungssoftware für Eye-Tracking- und Emotion-Detection-Studien. Dazu wurde eine auf dem Client-Server-Modell basierende Webplattform erstellt, die Probandendaten sammelt und anschließend in einer Datenbank verwaltet. Darauf aufbauend wird im Rahmen dieser Arbeit diese Software unter Betrachtung verschiedener Anforderungen evaluiert und bewertet.

Inhaltsverzeichnis

1 Einführung

In den letzten Jahren hat der Einsatz von Webanwendungen immer mehr an Bedeutung gewonnen [Web20b, Web20a, Web07]. Aufgrund der zunehmenden Nutzung des Internets, setzen Millionen von Unternehmen dies als kosteneffizientes Kommunikationsmittel ein, um Informationen zwischen dem Unternehmen und seinen Kunden auszutauschen [Gib16]. Solche Computerprogramme ermöglichen eine webbrowserübergreifende Interaktion zwischen mehreren Komponenten ohne zusätzliche Installation von Software auf der Festplatte [Tho19]. Webanwendungen erhöhen die Flexibilität der Anwendung, da sie auf mehreren Plattformen ausführbar sind, unabhängig vom Betriebssystem oder dem technischen Gerät. Somit reduziert der Einsatz einer Webanwendung die Kosten, sowohl für den Endnutzer als auch für den Dienstleister [Web20b, Gib16].
Diese Vorteile haben sich nicht nur Unternehmer in Bereichen des E-Commerce (z.B. Amazon) oder User Generated Content (z.B. Facebook, Wikipedia) zu Nutze gemacht [Fie20], sondern auch die empirische Forschung beispielsweise zur Durchführung von wissenschaftlichen Studien [Rei13, FKB+18, FKS+15]. Die Durchführung von Studien ist ein empirisches Entdeckungs- und Nachweisverfahren, um eine oder mehrere ungeklärte Fragen zu beantworten und neue Erkenntnisse zu gewinnen [Zah18]. Um dies zu ermöglichen ist eine systematische Sammlung von Daten notwendig, die daraufhin ausgewertet werden kann [Kli09]. Durch die Integration von Webanwendungen in den Studienbetrieb können Forscher einerseits sehr schnell eine große Anzahl von Teilnehmern rekrutieren, und andererseits ist es möglich, große Datensätze zu sammeln und flexibel auf diese zuzugreifen [Rei13].
Im Rahmen vieler Webplattformbasierten Studien in Bereichen der Verhaltensanalyse [Rei13], des maschinellen Lernens [FBK20, FKRK20, FK20a, FK20c, FK20b, FK19, FRM+20] und der Mensch-Computer-Interaktion [FGK20a] werden heutzutage vor allem quantitative Analysen von Studiendaten durchgeführt. Bei der Forschung geht es darum, aus den erhobenen Daten einer Studie wichtige Schlussfolgerungen zu ziehen und neue Erkenntnisse zu gewinnen. Dies wird durch eine qualitativ hochwertige Speicherung der Studiendaten erreicht. Eine hohe Qualität erreichen wir in diesem Zusammenhang, wenn die Daten bei der Erhebung durch das Softwareframeworks richtig ermittelt werden und zusätzlich in einem Format gespeichert sind, das die Stabilität und Qualität der Daten sicherstellt [Ren13]. Allerdings beinhalten die Anforderungen einer Datenhaltungssoftware neben dem zuverlässigen Erheben und Speichern von Daten, dass im Nachhinein effizient mit ihnen umgangen werden kann. Bei der Entwicklung einer effizienten Datenbank sollte man deswegen sicherstellen, dass sie Anforderungen bezüglich kurzer Zugriffszeiten und hoher

Verfügbarkeit von Daten erfüllt [Len97].
Das Ziel dieser Arbeit ist die Integration einer Datenbank in den webplattformbasierten Studienbetrieb. Dazu sollen die im Laufe der Studie entstandenen Daten der Probanden in einer Datenbank organsiert und auf strukturierte Weise gespeichert werden. Im Anschluss sollen neben der Erstellung der Datenbank, eine Evaluierung und Bewertung der Software stattfinden. Es ist wichtig zu untersuchen, ob die Datenhaltungssoftware verschiedene Anforderungen erfüllt, um eine qualitative und effiziente Datenverwaltung und -organisation zu gewährleisten. Um diese Ziele umzusetzen, haben wir eine Datenhaltungssoftware im Rahmen zweier Anwendungsgebiete entwickelt und untersucht, in denen die elektronische Datenverwaltung für Studien immer mehr an Bedeutung findet: Eye-Tracking und Emotion-Detection [Fuh19, FSG+16, FSG+17, FSK17a, ESF+17, EFK17, EHF+17, BFG+16, Fuh20].
Beim Eye-Tracking (Blickbewegungsmessung) handelt es sich um eine apparative und rezeptionsbegleitende Sensor-Technologie, mit dem Ziel die Blickrichtung [FSR+16, FKH+17, FGS+18] und den Blickverlauf von Personen zu ermitteln [FGK20b, FRE20, FSK+18, FCK18a, FBH+19, FCK+19]. Durch eine effiziente Datenverwaltung, können (z.B. im Rahmen von Studien) genaue Aussagen darüber gemacht werden, in welcher Reihenfolge und wie lange Nutzer Bereiche einer Benutzeroberfläche betrachten, wodurch die Evaluierung interaktiver Anwendungen durchführbar ist [Bla13, FCK18b, FK18]. Die Emotion-Detection (Emotionserkennung) hingegen ist ein Zweig der Stimmungsanalyse, der sich mit der Extraktion und Analyse von Emotionen befasst [AWNM20]. In diesem Gebiet können mithilfe von Computern und Programmen die Emotionen in Gesichtern gelesen und klassifiziert werden [Dwi19, FGRK19, FRK19, FCZ+18].
Vor diesem Hintergrund wird in dieser Bachelorarbeit eine webplattformbasierte Datenhaltungs- und Bewertungssoftware für Eye-Tracking- sowie Emotion-Detection-Studien erstellt und untersucht. Dabei soll der Proband die Möglichkeit haben Bilder aufzunehmen, welche anschließend zusammen mit den personenbezogenen Daten des Probanden in einer Datenbank verwaltet werden. Zusätzlich hat der Proband die Möglichkeit diese Bilder im Nachhinein einzusehen und bei Bedarf zu löschen. Um dies zu erzielen, wurde einerseits mit Hilfe von JavaScript die Akquise der Bilddaten bzw. der Probandendaten umgesetzt. Auf der anderen Seite wird eine PHP Schnittstelle aufgebaut, sodass eine serverseitige Interaktion mit JavaScript durchführbar ist. Dies ermöglicht anschließend, dass die über JavaScript erworbenen Daten in einer Datenbank gespeichert und verwaltet werden können.
Die Bachelorarbeit ist demzufolge in die folgenden Kapitel gegliedert:

- Kapitel 2: In diesem Kapitel werden alle erforderlichen Hintergrundinformationen geliefert, um ein Verständnis für die Entwicklung der Software zu ermöglichen. Dazu werden insbesondere auf die serverseitigen Funktionen der Anwendung eingegangen, wie PHP, XAMPP und MySQL.
- Kapitel 3: Die Spezifikation, wie die Bewertungs- und Datenhaltungssoftware funktioniert und wie sie serverseitig umgesetzt wurde. Das Kapitel beinhaltet

den Entwurf und die Struktur der Datenbank sowie die Datenbankschnittstelle für eine effiziente Interaktion mit dem Client. Außerdem erklären wir die Funktionsweise und die GUI der gesamten Software inklusive wichtige Implementierungen der Entwicklung.

- Kapitel 4: Evaluation und Bewertung der entwickelten Software bezüglich verschiedener Anforderungen an die Datenbank. Anschließend wird noch die Genauigkeit der Emotions- und Blickpunkterkennung der Software bewertet.
- Kapitel 5: Diskussion über die Ergebnisse der Evaluation bzw. über die Funktionsweise der Software. Folgende Leitfrage wird dabei berücksichtigt: Welche Aspekte wurden gut bzw. schlecht umgesetzt und was kann folglich verbessert werden?
- Kapitel 6: Eine Zusammenfassung der Bachelorarbeit mit den wichtigsten Ergebnissen, die man anhand dieser Arbeit schließen konnte.

2 Grundlagen

Ziel dieses Kapitels ist es, eine grundlegende Darstellung der Ansätze zu Datenmodellierung, Datenbankentwurf und Datenbankentwicklung zu geben. Insbesondere beziehen sich diese Ansätze auf die in dieser Arbeit entwickelte Anwendung einer webbasierten Datenhaltungssoftware mit Hilfe der relationalen Datenbanktheorie. Aus der untenstehenden Abbildung (Abbildung 2.1) entsteht ein Überblick über die wesentlichen Komponenten „Datenbank“, „Entwicklungsumgebung“ und „Sprachen“.

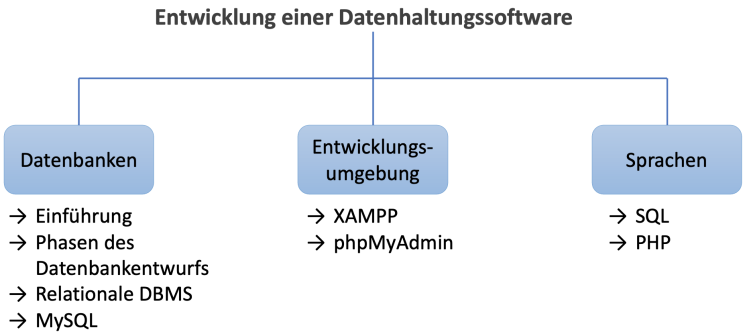


Abbildung 2.1: Überblick über den Aufbau des Kapitels "Grundlagen"

2.1 Datenbanken

2.1.1 Einführung in Datenbanken

Eine Datenbank ist eine organisierte Sammlung von strukturierten und persistenten Informationen oder Daten, welche untereinander in einer logischen Beziehung stehen. Dabei werden die Daten oft in einer bestimmten Struktur abgelegt, welche den optimierten Zugriff sowie das performante Wiederauffinden von Daten ermöglichen. Dementsprechend werden Daten, die nicht kohärieren, in separaten Datenbanken getrennt verwaltet. Das Ziel einer Datenbank ist es also, Daten effizient, dauerhaft und widerspruchsfrei zu speichern. Neben der Datenspeicherung übernimmt eine Datenbank Verwaltungsaufgaben, die von einem eigenen Datenbankverwaltungssystem (Database Management System, DBMS) organisiert werden. Die Kommunikation zwischen Client mit dem DBMS-Server erfolgt anschließend über eine logische

Schnittstelle, über die jeder Benutzer bzw. jedes Anwendungsprogramm auf die Daten in der Datenbank zugreifen kann (Abbildung 2.2.). Durch das DBMS werden diese logischen Zugriffe in physische umgewandelt und ermöglichen dem Nutzer Lese- und Schreibzugriffe der Daten auf das Speichermedium.

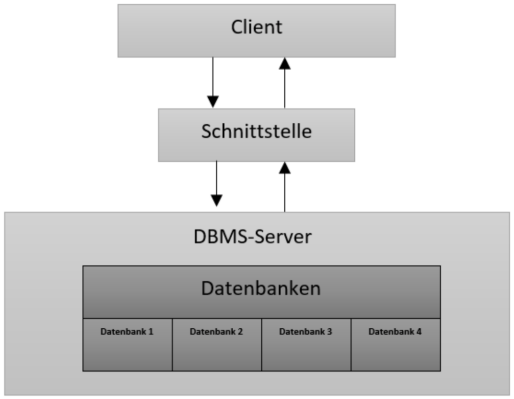


Abbildung 2.2: Prinzip der Zusammenarbeit zwischen Client und Datenbank [BDK]

Für eine effiziente und übersichtliche Gestaltung der Datenbank, sind die Daten in Zeilen und Spalten mehrere Tabellen organisiert. Dies ermöglicht dem Nutzer eine unkomplizierte und flexible Weiterverarbeitung der Daten. [BDK, Sch17, Dat20]

Wesentliche Anforderungen an eine Datenbank: [Kle06]

- **Persistenz**
  Nach Beendigung eines Prozesses bzw. Programms sollen die Daten weiterhin existieren. Durch die persistente Speicherung von Daten in einer Datenbank, sind die Daten stets präsent und können von weiteren Softwares wiederverwendet und bearbeitet werden.
- **Anlegen von Datenschemata**
  Ein Datenschemata ist eine Abstraktion, um die Speicherung von Daten in einer Datenbank darzustellen. Daruch kann eine Organisation der Daten erfolgen (z.B. anhand von Tabellen), die das Verständnis über die gespeicherten Informationen erhöht.
- **Gestaltung/Bearbeitung von Daten**
  Die Daten müssen für den Nutzer in der Datenbank so zugänglich sein, sodass das Einfügen, Löschen und Ändern von Daten möglich ist.
- **Lesen von Daten**
  Eine wichtige Anforderung an die Datenbank ist es, Daten aus der Datenbank auszulesen. Ein Nutzer muss durch gezieltes Abfragen von Daten überprüfen können, ob diese bereits in der Datenbank vorhanden sind und welche Informationen zusätzlich existieren.

2.1.2 Phasen des Datenbankentwurfs

Die Entwicklung einer Datenbank erfordert einen Planungsprozess (Entwurfsprozess), der durch mehrere Phasen durchläuft (Abbildung 2.3). Die sogenannte *Drei-Ebenen-Architektur* für Datenbanksysteme sieht vor, dass die Daten einer Datenbank unter drei Aspekten definiert werden. Zuvor findet eine Anforderungsanalyse statt.

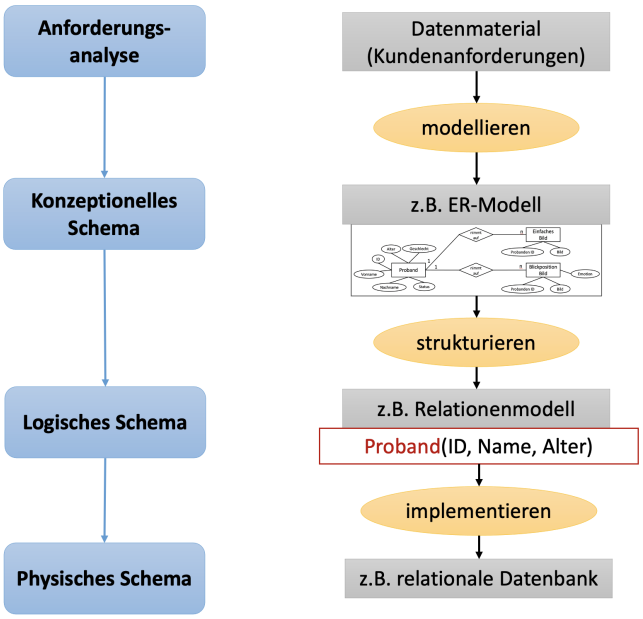


Abbildung 2.3: Graphische Darstellung über die verschiedenen Phasen des Datenbankentwurfs

1. **Anforderungsanalyse:** Informationsbedarf [Kle06]
   Die Anforderungsanalyse ist ein Prozess in der Softwareentwicklung, bei dem die Benutzererwartungen an ein neues oder modifiziertes Produkt (wie z.B. einer Datenbank) bestimmt werden. Für ein erfolgreiches Datenbank-Projekt ist es wichtig die Anforderungen präzise zu formulieren. Dazu muss für die Entwicklung und Verwendung einer Datenbank verdeutlicht werden, welche Daten in der Datenbank verwaltet werden sollen und welche Relationen zwischen diesen Daten bestehen.

2. **Konzeptioneller Entwurf:** Semantisches Modell [Mar20, Con16]
   Nach der Anforderungsanalyse, beginnt in der zweiten Phase die Anfertigung eines konzeptionellen Datenbankentwurfs. Der Zweck der konzeptionellen Entwurfsphase besteht darin, ein konzeptionelles Modell zu erstellen, das auf den zuvor identifizierten Anforderungen basiert, aber unabhängig von der Datenbanksoftware und den physischen Details ist. Das Ergebnis dieses Prozesses ist ein konzeptionelles Datenmodell, das die wichtigsten Datenentitäten, Attribute und Beziehungen eines Problems beschreibt. Ein häufig verwendetes

Modell ist hier das *Entity-Relationship-Model (ER-Modell)*, welches modelliert, wie sich die Objekte (Entitäten) zueinander verhalten.

3. **Logischer Entwurf:** Logisches Modell [Kle06, Log16]
   In der dritten Phase des Datenbankentwurfsprozesses, dem logischen Entwurf, wird das ER-Modell schrittweise in Tabellen überführt. Das Ziel in dieser Phase ist der Entwurf einer Datenbank, der auf einem bestimmten Datenmodell basiert, aber unabhängig von Details auf physischer Ebene ist. Der logische Entwurf erfordert, dass alle Objekte im konzeptionellen Modell auf Konstrukte abgebildet werden, die vom ausgewählten Datenbankmodell verwendet werden. Relationale Datenbanken bauen beispielweise auf Tabellen auf. Für den logischen Entwurf eines relationalen DBMS findet in dieser Phase die Spezifikationen für die Relationen (Tabellen) und Beziehungen statt.
4. **Physischer Entwurf:** Internes Datenmodell [FST88, Phy20]
   In der letzten Phase übersetzen wir den logischen Entwurf durch die Verwendung von Speichertechnologie und Zugriffspfade in eine physische Datenbank, also in ein aktuelles System (wie MySQL). Diese Phase umfasst zum Beispiel Tätigkeiten bezüglich Tabellendefinition, Normalisierung, Indizierung oder Leistungsverbesserung. Das Ergebnis ist schließlich eine physische Datenbank, die für eine spezielle Anwendung zugeschnitten ist.

2.1.3 Relationales Datenbankmanagementsystem (RDBMS)

Ein Datenbankmanagementsystem (DBMS) ist eine Software zur Verwaltung mehrerer, miteinander verknüpfter Daten in einer Datenbank. Dieses findet auf der logischen Ebene des Datenbankentwurfs Anwendung. Inzwischen werden verschiedene Typen von Datenbankmodellen und dazugehörigen DBMS unterschieden: relationale Datenbanken, objektorientierte Datenbanken, hierarchische Datenbanken, netzwerkartige Datenbanken oder hybrid-Datenbanken. Im Wesentlichen unterscheiden sich diese Modelle im logischen Aufbau der Datenbank. So findet das hierarchische Datenmodell heute aufgrund seiner beschränkten Anwendbarkeit kaum noch an Verwendung und das Netzwerkmodell wird hauptsächlich für große Datenmengen in Großraumrechnern eingesetzt. In dieser Arbeit werden wir das weit verbreitete relationale Datenbankmodell näher betrachten, da es die Grundlage für den Programmierteil dieser Arbeit bildet.
Bei einer relationalen Datenbank handelt es sich um eine Ansammlung von zeitlich variierenden, normalisierten Tabellen, die zueinander in Beziehung (Relationen) stehen, und die von einem Verwaltungssystem (RDBMS) organisiert werden. [Sch17]

Aufbau/Struktur einer relationalen Datenbank

Um jedoch den genauen Aufbau sowie das Funktionsprinzip der relationalen Datenbank besser zu verstehen, betrachten wir zuerst die theoretischen Grundlagen dieses

Tabelle 2.1: Formale und informelle Begriffe einer relationalen Datenbank [Sch17]

Formale relationale Bezeichner	**Informelle Bezeichnung**
Relation	Tabelle
Tupel	Eine Zeile (Reihe) einer Tabelle
Kardinalität	Anzahl der Zeilen einer Tabelle
Attribut	Eine Spalte (Feld) einer Tabelle
Grad	Anzahl der Spalten einer Tabelle
Primärschlüssel	Eindeutiger Bezeichner
Gebiet	Menge aller möglichen Werte

Datenbankmodells. Die Tabelle 2.1 definiert wichtige Begriffe, indem formale relationale Ausdrücke den mehr informellen Ausdrücken gegenübergestellt sind [Sch17].

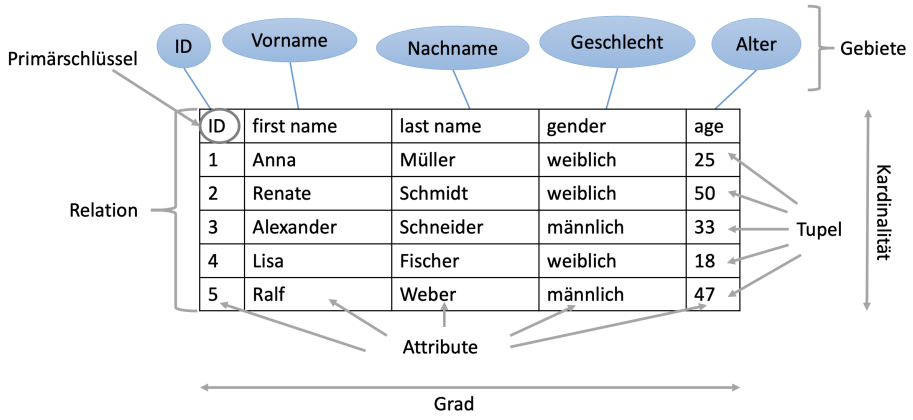


Abbildung 2.4: Aufbau und Begriffe einer Tabelle bei relationalen Datenbanken

Die Daten einer relationalen Datenbank werden in Tabellen (*Relation*) gespeichert. Eine Relation ist hierbei eine Ansammlung von *Gebieten*, bestehend aus einem Kopf und einem Rumpf. Der *Kopf* besteht aus dem Namen des Attributs inklusive seines Datentyps und hat eine feste Größe. Anhand des Kopfs kann vor allem die Anzahl an Attributen einer Tabelle festgelegt werden. Der *Rumpf* hingegen legt die Anzahl an Tupeln fest und hat eine variable Größe. Jedes Tupel entspricht dabei immer einem Datensatz in der Tabelle.
Für eine Relation kann abhängig von der Anzahl an Zeilen und Spalten die Kardinalität und der Grad bestimmt werden. Die *Kardinalität* einer Relation entspricht der Anzahl an Tupeln und ist beim Erzeugen der Relation meistens null. Diese kann anschließend durch das Einfügen oder Löschen (sobald Kardinalität größer null ist) von neuen Datensätzen laufend geändert werden. Im Gegensatz dazu wird der *Grad* einer Relation, also die Anzahl der Spalten einer Relation, beim Erzeugen definiert und ist fest vorgegeben. Der Grad ändert sich in der Regel auch nicht, da man sonst

durch das Hinzufügen oder Löschen von Spalten die Relation in eine neue Relation überführen würde.
Bei dem *Primärschlüssel* handelt es sich oft um eine ID und ist in einer Relation das erste Attribut. Dieser ermöglicht in einer relationalen Datenbank die Eindeutigkeit der einzelnen Tupel und darf sich in einer konsistenten Datenbank nicht ändern.
Der Aufbau und die Begriffe einer Tabelle bei relationalen Datenbanken sind in Abbildung 2.4 beispielhaft dargestellt. [Sch17]

2.1.4 MySQL

MySQL ist ein relationales Datenbankmanagementsystem, d.h. es speichert Informationen und Daten in mehreren, miteinander verknüpften Tabellen. Jede Tabelle besteht aus separaten Feldern, die jedes Informationsbit repräsentieren. Die Speicherung von Daten in mehreren Tabellen, statt in einem großen gemeinsamen Speicherraum, bietet Vorteile bezüglich Flexibilität und Geschwindigkeit der Datenbank.
MySQL basiert auf der Client-Server-Architektur und ermöglicht beliebig vielen Client-Programmen und Back-ends durch Programmierschnittstellen den Zugriff auf das Datenbanksystem. Für den Zugriff und die Steuerung der Datenbank wird die Datenbanksprache SQL (Structured Query Language, strukturierte Abfragesprache) eingesetzt, welche im nächsten Kapitel ausführlicher behandelt wird.
MySQL hat sich in den letzten Jahren als sehr solides und stabiles System etabliert und ist heute der beliebteste quelloffene Open-Source-Datenbankserver der Welt. MySQL eignet sich sowohl für kleine als auch große Anwendungen. [Rei10, WG14]

2.2 Sprachen

2.2.1 Datenbanksprache SQL

Als Standard-Computersprache für relationale Datenbanken, wie MySQL-Datenbanken, wird die Datenbanksprache SQL (*Structured Query Language*) zum Abfragen, Einfügen, Aktualisieren und Modifizieren von Daten eingesetzt. Sie stellt eine Schnittstelle zwischen dem Anwenderprogramm und der relationalen Datenbank zur Verfügung. Aus diesem Grund hat sich SQL als Sprache für den Datenbankprogrammierer umso mehr etabliert und spielt heute in fast allen Datenbank-Anwendungen eine fundamentale Rolle. [UM12]
SQL hat eine wichtige Eigenschaft, welche die Zugriffssprache erheblich vereinfacht und somit ein Grund für die weite Verbreitung von relationalen Datenbanken und SQL ist. Der Zugriff auf Daten über SQL findet direkt statt, unabhängig von der Art und Weise, wie sie abgespeichert (z.B. squentiell oder über einen Index) sind. [Sch17]
Die meisten relationalen Datenbanken unterstützen SQL, was ein zusätzlicher Vorteil für Datenbankadministratoren ist, da sie häufig Datenbanken über verschiedene

Plattformen hinweg unterstützen müssen. [WG14]

Der SQL-Code kann in vier Hauptkategorien unterteilt werden: [WG14, Sql19]

- **Data Query Language (DQL):** Abfrage und Suche von Daten in der Datenbank unter Verwendung der SELECT-Anweisung. DQL ist in Klauseln unterteilt wie FROM, SELECT, WHERE, JOIN, ORDER BY, usw.
- **Data Manipulation Language (DML):** Hinzufügen, Aktualisieren oder Löschen von Daten und ist genau genommen eine Untermenge von SELECT-Anweisungen. Sie setzt sich aus den Anweisungen INSERT, UPDATE und DELETE, sowie den Steueranweisungen wie SAVEPOINT, COMMIT und ROLLBACK zusammen.
- **Data Definition Language (DDL):** Verwaltung von Tabellen und Indexstrukturen. Beispielhafte DDL-Anweisungen sind CREATE, ALTER, TRUNCATE und DROP.
- **Data Control Language (DCL):** Wird verwendet, um Datenbankrechte und Datenbankberechtigungen zuzuweisen und zu widerrufen. Die wichtigsten Anweisungen sind GRANT und REVOKE.

2.2.2 PHP

Um Webanwendungen in Zusammenarbeit mit Datenbanken zu entwickeln, ist der Einsatz einer Skriptsprache wie PHP notwendig.
1995 entwicklete Rasmus Lerdorf PHP als eine Sammlung von Skripten basierend auf der Sprache Perl unter dem Namen „Personal Homepage Tools“, wurde jedoch später unter der Abkürzung „PHP Hypertext Preprocessor“ bekannt. PHP 2 wurde anschließend als umfangreiche Implementierung in der Programmiersprache C entwickelt und von Andi Gutmann und Zeev Suraski als PHP 3 komplett neu implementiert. Diese Entwickler haben dann die Firma ZEND Technologies-Ltd. gegründet, welche als Kern für das PHP-Framework ZEND die vierte Version von PHP (PHP 4) entwickelte. 2004 etablierte sich anschließend PHP 5 aufgrund von Fähigkeiten bezüglich optimierter Objektorientierung und Datenbankintegration.
Bei PHP handelt es sich also um eine weit verbreitete und für den allgemeinen Gebrauch bestimmte Open Source-Skriptsprache. PHP wurde entwickelt, um dynamische Webanwendungen, wie z.B. E-Commerce-Systeme, Chats oder Foren zu erzeugen. Im Gegensatz zu statischen Webseiten, basiert der Inhalt der Webseite auf der Interaktion mit dem Benutzer. Neue Inhalte oder Basisinformationen werden dafür beispielsweise aus Datenbanken ausgelesen und angezeigt.
PHP wird für den Einsatz in der Webprogrammierung in die textbasierte Auszeichnungssprache HTML (Hypertext Markup Language) eingebettet. Dies erfolgt durch die Einbettung des PHP-Codes in die Anfangs- und Abschluss-Verarbeitungsinstruktionen <?php und ?>. Charakteristisch an PHP, im Gegensatz zu anderen Skript-

sprachen ,wie Python oder Perl, ist, dass PHP Variablen nicht deklariert werden müssen und stets mit einem Dollarzeichen beginnen.
PHP ist eine serverseitige Skriptsprache, d.h. die Ausführung des Codes findet auf dem Server statt und sendet die HTML-Ausgabe zurück an den Client. Die einzige Information, die der Client anschließend erhält, ist die Ausgabe des PHP-Skript und nicht den ausgeführten PHP-Code. [Php20, Php18, The06]

Kommunikation mit einer Datenbank über PHP

PHP ist ein System, das besonders für die serverseitige Webentwicklung verwendet wird und somit auf dem Webserver läuft. Mit Hilfe von PHP können dynamische Webseiten mit Datenbankzugriff ausgeführt werden. Die Funktionsweise basiert auf dem Client-Server-Modell und wird anhand des folgenden Schaubildes (Abbildung 2.5) erklärt:
Ein Nutzer erstellt im Browser eines Clients eine HTML-Anfrage, sodass ein HTTP-Request an den Webserver stattfindet. Der Webserver fordert das zugehörige PHP-Skript an und leitet dieses an das PHP-Modul des Apaches weiter. Dieser liest das PHP-Skript zeilenweise ein und übermittelt enthaltene Datenbankbefehle an das Datenbankmanagementsystem. Die angeforderten Daten aus der Datenbank werden auf dem Webserver in den HTML-Code eingebettet. Zum Schluss antwortet der Web-Server dem Client mit den angefragten Informationen, die daraufhin auf dem Bildschirm des Nutzers angezeigt werden. [Kö00, Gib16, BvM01]

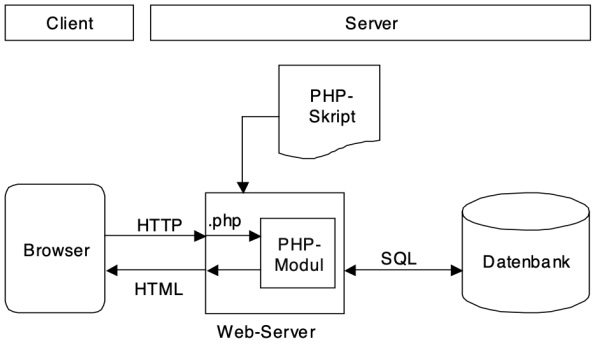


Abbildung 2.5: Ablauf und Funktionsweise einer Datenbankabfrage über PHP [BvM01]

2.3 Entwicklungsumgebung

Zur erfolgreichen Entwicklung einer Webanwendung mit Datenbankanbindung ist eine Umgebung zum Testen erforderlich, die die notwendigen Anforderungen erfüllt. Dementsprechend benötigen wir eine passende Umgebung, die die verschiedenen

Komponenten zur Erstellung einer Webanwendung mit Datenbankanbindung vereint, wie zum Beispiel XAMPP.

2.3.1 XAMPP

Unabhängig davon, ob man ein Content-Managementsystem, eine Groupware oder eine Webanwendung aufsetzen will, bietet das XAMPP-Paket eine gute Grundlage. Damit wird gewährleistet, dass die Bestandteile verknüpft werden, die für die Nutzung von Lösungen basierend auf dem Apache-MySQL-PHP-Gespann erforderlich sind. [Rei10]
XAMPP steht für Cross-Platform (X), Apache (A), MySQL (M), PHP (P) und Pearl (P). Es handelt sich um eine simple Apache-Distribution, die die gängigsten Webentwicklungstechnologien in einem einzigen Paket enthält. Sein Inhalt, seine geringe Größe und seine Portabilität machen es zu einem idealen Werkzeug für Softwareentwickler, die Anwendungen in PHP und MySQL entwickeln und testen. Alle Mittel, die benötigt werden, um einen Webserver einzurichten, – Serveranwendung (Apache), Datenbank (MySQL) und Skriptsprache (PHP oder Perl) – sind in einer einfachen, extrahierbaren Datei enthalten. XAMPP ist außerdem plattformübergreifend, d.h. es funktioniert für mehrere unterschiedliche Betriebssysteme wie Linux, MacOS und Windows gleichermaßen gut. [Dvo07, WG14]
XAMPP dient nicht als Produktionssystem, wie z.B. als öffentlicher Webserver, sondern ist in erster Linie für Entwickler geeignet, um in kurzer Zeit ein kompaktes Testsystem aufzusetzen. Deswegen gibt es für diese Apache-Distribution bewusste Einschränkungen bezüglich der Sicherheit.

Das XAMPP-Paket beinhaltet vier wichtige Komponenten: [WG14]

1. **Apache**: Der Apache HTTP Server stellt einen Open-Source-HTTP-Server für moderne Betriebssysteme einschließlich UNIX und Windows bereit. Dieser ermöglicht es mittels serverseitiger Skriptsprachen Webseiten dynamisch zu entwickeln und zu warten. Das Ziel von dem Apache Server ist es, einen sicheren, effizienten und erweiterbaren Server bereitzustellen, der HTTP-Dienste synchron zu den aktuellen HTTP-Standards anbietet.

2. **MySQL**: Jede Webanwendung erfordert, unabhängig vom Schwierigkeitsgrad, eine Datenbank zur Speicherung der gesammelten Daten. Das quelloffene MySQL ist ein SQL-Datenbank-Managementsystem und bildet somit die Grundlage für dynamische Webseiten.

3. **PHP**: PHP (*PHP Hypertext Preprocessor*) ist eine weit verbreitete und für den allgemeinen Gebrauch bestimmte Open Source-Skriptsprache, welche speziell für die Webprogrammierung geeignet ist und in HTML eingebettet werden kann. [Php20]

4. **Perl**: Perl ist eine dynamische high-level Programmiersprache, die in der

Netzwerkprogrammierung, Systemadministration usw. ausgiebig verwendet wird. Zwar wird sie für Webentwicklungszwecke weniger verwendet, hat aber dennoch eine Menge Nischenanwendungen.

2.3.2 XAMPP-Tool phpMyAdmin

Webanwendungen die auf der Zusammenarbeit zwischen Apache, MySQL und PHP basieren, benötigen den Zugang zu einem Werkzeug, mit dessen Hilfe einfache Eingriffe in den Datenbankserver vorgenommen werden können. So muss zum Beispiel für einen Online-Shop eine Datenbank aufgesetzt werden, in der die Zugangsdaten der Kunden übersichtlich und sicher gespeichert werden. Zwar bietet MySQL viele verschiedene Möglichkeiten, mit dessen Hilfe ein Datenbankserver verwaltet und gewartet werden kann, doch sind diese Tools wenig benutzerfreundlich.
Hier bietet das XAMPP-Tool *phpMyAdmin* eine Lösung, um diesem Problem entgegenzukommen. Dieses benutzerfreundliche und leistungsfähige Werkzeug ist eine freie Webanwendung zur Administration und Verwaltung von MySQL-Datenbanken und ist in PHP implementiert. Mit Hilfe von phpMyAdmin können sowohl lokal MySQL-Datenbanken aufgesetzt werden als auch für den Betrieb einer typischen Webanwendung eingesetzt werden. Durch phpMyAdmin lassen sich MySQL-Datenbanken einfach und schnell per HTTP mit einem Browser ansprechen, ohne auf komplizierte SQL-Befehle zurückgreifen zu müssen. Sind die Benutzerrechte richtig gesetzt, bietet phpMyAdmin eine Reihe vielseitiger Funktionen, um die MySQL-Server zu administrieren und ganze Datenbanken zu steuern. Der Datenbankmanager unterstützt eine Vielzahl von Aktionen zur Verwaltung, Organisation und Administration von Datenbanken und Tabellen, dem Export von Daten oder den Einstellungen der Benutzerrechte. Die gesamte Bedienung von phpMyAdmin läuft dazu problemlos über das Browserfenster ab (siehe Abbildung 2.6). [Rei10]

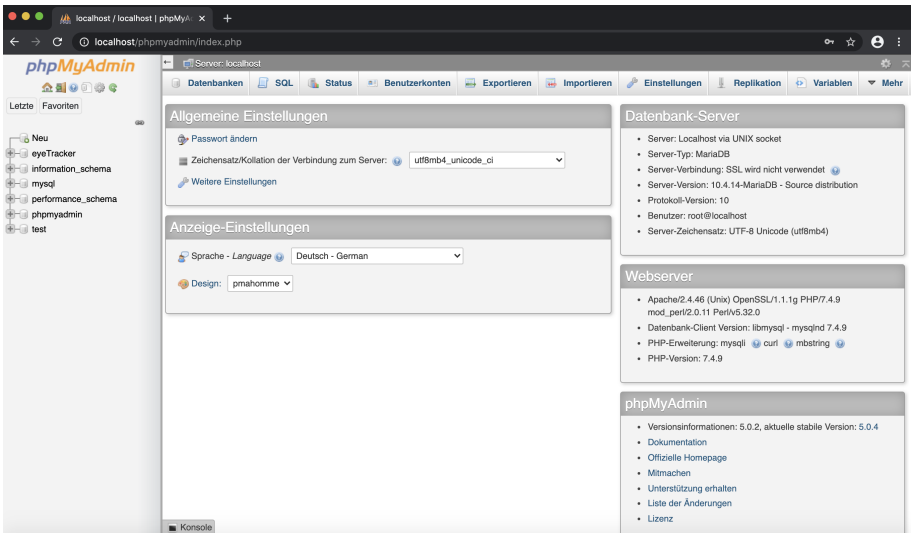


Abbildung 2.6: Grafische Benutzeroberfläche des XAMPP-Tools *phpMyAdmin*

3 Methoden

In vielen Bereichen des maschinellen Lernens und der Mensch-Computer-Interaktion werden große Datenmengen im Rahmen von Studien gewonnen. Diese Datenmengen müssen erst aus der Software erworben und anschließend konsistent und effizient verwaltet werden. Dies findet mit Hilfe eines Datenverwaltungsprogramm statt, welches sicherstellt, dass die Daten für den späteren Gebrauch und zur Bewertung für den Anwender bereitstehen. Im Rahmen dieser Arbeit haben wir auf Basis eines Anwendungsbeispiels (Eye-Tracking und Emotion-Detection) eine Software entwickelt, die die Verwaltung von Daten im Rahmen von Studien sicherstellt.
Das Ziel des Programmierteils ist es eine Webplattformbasierte Datenhaltungs- und Bewertungssoftware für eine Eye-Tracking- und Emotion-Detection-Studie zu entwickeln. Die Software soll anhand eines Eye-Trackers bzw. anhand einer Emotion-Detection entweder die Blickposition oder die Emotion des Probanden auswerten können. Diese Daten werden anschließend inklusive eines Bildes des Probanden und seinen Angaben in einer Datenbank gespeichert und können daraufhin beispielsweise für Studienzwecke verwendet werden.
Die Funktionsweise dieser Software sieht wie folgt aus: Der Proband hat die Möglichkeit zwischen verschiedenen Optionen auszuwählen. Erstens kann der Proband ein einfaches Foto von sich aufnehmen, das anschließend in der Datenbank gespeichert wird. Zweitens kann der Proband durch den Einsatz des Eye-Trackers seine Blickposition auslesen lassen. Zu diesem Zweck betrachtet er einen Punkt in der Mitte des Bildschirms und der Computer erfasst die x- und y-Koordinaten des Blickpunktes. Anschließend wird ebenfalls ein Bild des Probanden erzeugt und gemeinsam mit den passenden Koordinaten in der Datenbank abgelegt. In der dritten Option kann der Proband seine Emotionen durch die Verwendung einer Emotion-Detection erfassen lassen. Hierzu wird dem Nutzer eine Emotion gezeigt, die er nachahmen soll. Die Software erkennt anschließend die Emotion und nimmt ein Foto auf. Genauso wie beim Eye-Tracker wird das Bild mit der dazugehörigen Emotion in der Datenbank gesichert.
Um dies umzusetzen, lässt sich die Implementierung in zwei Teile gliedern: Einerseits findet eine Akquise der Bild- und Probandendaten anhand von JavaScript statt. In diesem Teil der Implementierung wurde hierzu ein Eye-Tracker bzw. ein Emotion-Detector aufgesetzt sowie das Frontend der Webanwendung entwickelt. Auf diesen Teil wird in dieser Arbeit jedoch nicht genauer eingegangen. Der zweite Teil des Programmierteils beinhaltet das Backend, also den serverseitigen Teil der Software. Hier wurde unter anderem eine PHP Schnittstelle für die Kommunikation mit JavaScript angelegt. Zusätzlich beinhaltet dieser Teil noch das Aufsetzen eines

Webservers und einer Datenbank, um die über JavaScript empfangenen Daten zu speichern und zu verwalten.

3.1 Datenbankentwurf und Datenbankstruktur

Bei der Entwicklung einer Datenhaltungssoftware ist eine genaue, schrittweise Planung entscheidend. In diesem Kapitel erklären wir in Anlehnung an die verschiedenen Phasen des Datenbankentwurfs, wie die Datenbank für die oben genannte Anwendung nach und nach realisiert wurde.

3.1.1 Anforderungen an die Datenbank

Die Datenbank soll alle erforderlichen Studiendaten verwalten. Zu den erforderlichen Daten zählen das Erfassen des Probanden, die Resultate der Studie in Form von Bildern und Zahlen sowie Inhalte, die auf der Webplattform bereitgestellt werden. In der Datenbank sollen alle Komponenten, die zur Durchführung einer Eye-Tracking- und Emotion-Detection-Studie benötigt wurden, gespeichert werden. Dies betrifft folgende Daten:

- Probanden-Daten inklusive Namen, Geschlecht, Alter und Status des Probanden. Diese zusätzlichen Angaben sind vom Probanden nur freiwillig anzugeben.
- Bilder der Smileys für die Emotion-Detection
- Textinhalte bezüglich Anleitung und Datenschutzhinweis
- Bildresultate, bei der Nutzung der Variante „Einfaches Bild/Nichts“
- Bildresultate und die dazugehörigen Emotionen, bei der Nutzung der Variante „Emotion/Smiley“
- Bildresultate und die dazugehörigen Blickpositionen, bei der Nutzung der Variante „Blickposition“

Bei der Speicherung in der Datenbank müssen genauso die Relationen der einzelnen Komponenten untereinander berücksichtigt werden. Dazu muss vermerkt werden, wie genau der Proband aufgebaut ist und wie die aufgenommenen Bilder jeweils abgespeichert werden sollen.
Ein Proband setzt sich aus Daten bezüglich seines Namens, Geschlecht, Alter und Status zusammen. Falls diese Daten nicht angegeben werden wollen, sind diese mit Platzhaltern zu füllen. Diese Informationen müssen gemeinsam mit einer ID (für die Eindeutigkeit der Teilnehmer) für jeden Probanden gespeichert werden. Hierbei ist weiterhin zu berücksichtigen, dass man Probanden nicht doppelt anlegen kann.
Außerdem hat jeder Proband die Möglichkeit Bilder in den Varianten „Einfaches

Bild/Nichts“, „Emotion/Smiley“ und „Blickposition“ aufzunehmen. Für die Varianten müssen sowohl die Bilddaten als auch die dazugehörigen Emotionen bzw. Blickpositionen aufbewahrt werden. Der Proband hat die Möglichkeit sowohl an allen drei Varianten teilzunehmen als auch nur an einer oder zwei. Dazu ist es sinnvoll, diese Bildresultate extern auszulagern, anstatt sie gemeinsam mit den Probanden-Daten zu sichern. Auf diese Weise reduziert man das Anlegen von redundanten Daten und erhöht die Effizienz der Datensicherung.
Neben dem Anlegen von Daten, soll jeder Teilnehmer an der Studie die Möglichkeit haben, seine Daten bei Bedarf einzusehen und zu löschen. Dazu muss jeder Proband Zugriff auf seine Studienergebnisse bekommen.
Eine weitere wichtige Anforderung an die Datenbank ist, dass sie eine Schnittstelle nach außen bereitstellt. Diese ist erforderlich, damit andere Programme und technische Geräte Zugriff auf die Datenbank haben, um Daten zu schreiben oder auslesen zu können. In dieser Arbeit erstellen wir hierzu eine PHP-Schnittstelle für die Kommunikation mit JavaScript. So kann anschließend die Webplattform mit der Datenbank interagieren.

3.1.2 Konzeptioneller Entwurf

Sobald alle Anforderungen an die Datenbank erfasst und analysiert wurden, fand der konzeptionelle Entwurf der Datenbank statt. Die erste Aufgabe war es, ein Modell zu erstellen, welches alle relevanten Daten und Anforderungen für die zu entwickelnde Software umfasst. Dazu haben wir ein *Entity-Relationship-Modell* ausgearbeitet, um eine sinnvolle Darstellung von Zusammenhängen zwischen den Daten zu erzielen.
Das ER-Modell beschreibt in Form eines Diagramms, alle existierenden Entitäten (Objekte) für die Datenbank gemeinsam mit ihren Attributen zur Modellierung der Eigenschaften. Dazu werden die Entitäten im ER-Modell als Rechtecke dargestellt, während die zugehörigen Attribute in Ovalen stehen, die mit dem Rechteck verbunden sind. Eine wichtige Eigenschaft im ER-Modell ist die Relation, die zwischen Objekten auftreten kann. Um zu definieren, in welcher Beziehung zwei Entitäten stehen, wird sie in Form einer Raute benannt und zwischen den betroffenen Entitäten verbunden. [Kle06] Im Folgenden wird das Datenbankmodell für unsere Webplattform anhand eines ER-Diagramms veranschaulicht (Abbildung 3.1).

Das zentrale Objekt im konzeptionellen Modell der Datenbank ist das Objekt „Proband“. Jeder „Proband“ wird dabei durch eine ID eindeutig. Neben diesem Attribut setzt sich das Objekt aus den weiteren Attributen „Vorname“, „Nachname“, „Geschlecht“, „Alter“ und „Status“ zusammen. Das Objekt „Proband“ verweist auf weitere Objekte, die die Funktionsweise und den Aufbau der Studiensoftware widerspiegeln. Jeder Proband hat die Möglichkeit an unterschiedlichen Varianten der Studie teilzunehmen. In jeder Variante nimmt der Proband Bilder auf, die extern vom Probanden abgespeichert werden müssen. Demzufolge resultiert eine Beziehung zwischen dem Probanden und jeder dieser Varianten. Somit hat ein Proband die

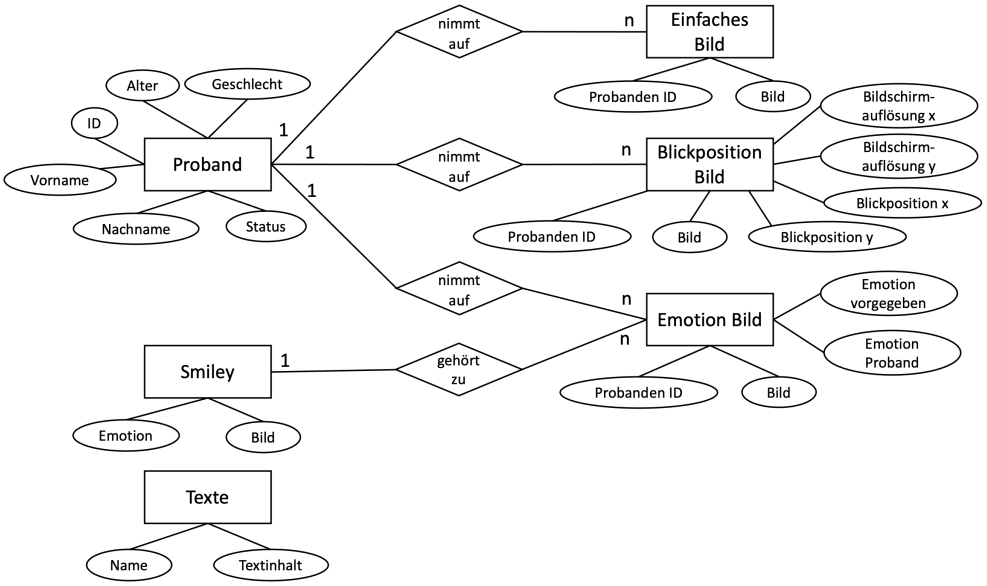


Abbildung 3.1: ER-Modell über den grundlegenden Aufbau der Datenbank und der Tabellen

Möglichkeit entweder ein einfaches Bild, ein Bild mit einer Emotion oder ein Bild zusammen mit der Blickposition aufzunehmen. Für jede dieser Varianten ergibt sich ein Objekt „Einfaches Bild“, „Blickposition Bild“ und „Emotion Bild“.
Im Modell von oben nach unten betrachtet, ist das Objekt „Einfaches Bild“ das erste Objekt, auf das „Proband“ verweist. In dem Objekt sollen alle Bildresultate verwaltet werden, die die Probanden unter der Option „Einfaches Bild/Nichts“ aufnehmen. Um zuordnen zu können, welches Bild in dem Objekt, welchem Probanden zugeordnet ist, muss die ID des Probanden zusätzlich zum Bild gespeichert werden. Daraus ergeben sich für das Objekt „Einfaches Bild“ die Attribute „Probanden ID“ und „Bild“. Jeder Proband hat die Möglichkeit ein oder mehrere Bilder unter der Variante „Einfaches Bild“ aufzunehmen. Somit ergibt sich die erste Relation in unserem ER-Modell zwischen dem Objekt „Proband“ und dem Objekt „Einfaches Bild“. Da jeder Proband mehrere Bilder machen kann, aber zu jedem Bild nur ein Proband gehört, erhalten wir Kardinalitäten, die in das ER-Modell mitaufgenommen werden können. Kardinalitäten zwischen zwei Objekten beschreiben, in welcher Beziehung sie zueinanderstehen. Dazu werden die Ziffer 1 und die Buchstaben m und n benutzt. Anhand der Kardinalitäten kann man verschiedene Formen von Beziehungen unterscheiden: [Kle06]

- **„Eins-zu-Eins-Beziehung“ (1:1)**
  Jede Entität des einen Entitätstyps steht in Beziehung mit höchstens einer Entität aus dem zweiten Entitätstypen.

- **„Eins-zu-N-Beziehung“ (1:N)**

Jede Entität des ersten Entitätstyps steht in Beziehung mit einer oder mehreren Entitäten aus dem zweiten Entitätstypen. Umgekehrt kann jede Entität aus dem zweiten Entitätstypen höchstens einer Entität aus dem ersten Entitätstypen zugeordnet werden.

- **„M-zu-N-Beziehung“ (M:N)**
  Jede Entität aus dem ersten Entitätstypen steht in Beziehung mit einem oder mehreren Entitäten aus dem zweiten Entitätstypen. Diese Beziehung ist für die umgekehrte Richtung äquivalent.

Jeder „Proband“ kann ein oder mehrere Bilder („Einfaches Bild“) aufnehmen. Umgekehrt kann jedes Bild („Einfaches Bild“) nur einem Probanden zugeordnet werden. Infolgedessen ergibt sich für die Relation zwischen „Proband“ und „Einfaches Bild“ eine 1:N-Beziehung.
Das Objekt „Blickposition Bild“ ist das zweite Objekt, auf das „Proband“ verweist. In diesem Objekt werden die Bildresultate für die Eye-Tracking Variante „Blickpunkt“ gespeichert, gemeinsam mit der x- und y-Koordinate der Blickposition sowie der x- und y-Koordinate der Bildschirmauflösung. Die x- und y-Koordinate der Bildschirmauflösung benötigen wir, um am Ende die Genauigkeit des Eye-Trackers in Relation zu der Blickposition des Probanden zu evaluieren. Folglich umfasst das Objekt „Blickposition Bild“ die Attribute „Probanden ID“, „Bild“, „Blickposition x“, „Blickposition y“, „Bildschirmauflösung x“ und „Bildschirmauflösung y“. Genauso wie für das Objekt „Einfaches Bild“ ist eine Beziehung zwischen „Proband“ und „Blickposition Bild“ notwendig, um die aufgenommenen Bilder extern auszulagern. Es handelt sich ebenfalls um eine 1:N-Beziehung zwischen diesen beiden Entitäten.
Für die dritte Option „Emotion/Smiley“ wird ebenfalls ein Objekt „Emotion Bild“ erzeugt, auf das „Proband“ deutet. Dieses Objekt verweist auf die Attribute „Probanden ID“ für die Zuordnung des Probanden, „Bild“ für die Bildresultate und „Emotion“, um die dazugehörige Emotion abzuspeichern. Zusätzlich wird unter dem Attribut „Emotion vorgegeben“ die Emotion des Smileys gespeichert, um am Ende die Emotionserkennung und -genauigkeit der Software zu untersuchen. Die Relation zwischen „Proband“ und „Emotion Bild“ ist notwendig für die Aufbewahrung der Bilder unter der Option „Emotion/Smiley“. Gleichermaßen wie für die bereits genannten Objekte, ergibt sich zwischen dem Objekt „Proband“ und „Emotion Bild“ wieder eine 1:N-Beziehung.
Neben dem Objekt „Proband“ verweist noch ein weiteres Objekt „Smiley“ auf „Emotion Bild“. In diesem Objekt sollen alle Smiley-Bilder aufbewahrt werden, die der Proband unter der Variante „Emotion/Smiley“ nachahmen kann. Das Objekt „Smiley“ lässt sich aus diesem Grund in folgende zwei Attribute gliedern. Einerseits gibt es das Attribut „Emotion“, in dem die Emotion des Smileys festgehalten wird, und das zweite Attribut „Bild“, welches den passenden Smiley für die Emotion in Form eines Bildes speichert. Da das Objekt „Smiley“ für die Ausführung der „Emotion/Smiley“-Variante notwendig ist, liegt eine Relation auf „Emotion Bild“ vor. Jede Emotion des Objektes „Smiley“ kann zu mehreren Bildern in „Emotion Bild“ gehören. Andererseits gehört aber nur ein „Emotion Bild“ zu einem „Smiley“.

Demzufolge ergibt sich für diese Relation eine 1:N-Beziehung.
Das letzte Objekt in dem ER-Modell ist das Objekt „Texte“. Dies ist notwendig, um alle größeren Texte in der Datenbank zu sichern, die auf der Webplattform angezeigt werden sollen, wie zum Beispiel Anleitungen zur Anwendung der Webplattform oder der Datenschutzhinweis. Das Objekt „Texte“ verweist auf das Attribut „Name“ zur Definition der Art des Textes und auf das Attribut „Textinhalt“, in dem sich der passende Textinhalt befindet. „Texte“ ist unabhängig zu den restlichen Entitäten im ER-Modell und weist deshalb keine Beziehungen zu den anderen Objekten auf.

3.1.3 Logischer Entwurf

Die Datenbank basiert auf der Grundlage des relationalen Datenbankmodells. Dementsprechend gibt es für die Software eine Datenbank mit mehreren Tabellen, die miteinander in Relation stehen. Im logischen Entwurf des Datenbankdesign müssen wir das Entity-Relationship-Modell aus dem vorherigen Kapitel systematisch in Tabellen übersetzen.
Bei der Übersetzung des ER-Modells in Tabellen betrachten wir dazu drei zentrale Ziele: [Kle06]

1. Die Tabellen sollen mit Daten so gefüllt werden, dass sich redundante Daten vermeiden lassen. Möchten wir also Informationen, wie zum Beispiel die aufgenommenen Bilder, mit einem Probanden verknüpfen, wird nur seine ID als Referenz genutzt.

2. Die Verwendung von NULL-Werten zum Füllen der Tabellen soll nur eingesetzt werden, wenn dies notwendig ist. Dementsprechend sollte mit Bedacht entschieden werden, ob und welche optionalen Werte von einem Probanden benötigt werden. Da für unsere Anwendung die Teilnahme an der Studie in erster Linie anonym ist, können die Felder in der Datenbank, in denen die freiwilligen Angaben des Probanden stehen, mit optionalen Werten gefüllt sein.

3. Unter Berücksichtigung des 1. und 2. Ziels sollte eine möglichst geringe Anzahl an Tabellen modelliert werden.

Zur Übersetzung unseres ER-Modells in Tabellen müssen, zusätzlich unter Berücksichtigung der oben genannten Ziele, folgende Vorgehensweisen beachtet werden. Jeder Entitätstyp im ER-Modell lässt sich in eine Tabelle übersetzen, wobei die Attribute der Entitäten den Attributen der Tabelle entsprechen. Somit ergeben sich für unser Modell sechs Tabellen, die mit den sechs Entitäten übereinstimmen. Außerdem lässt sich jede Beziehung im ER-Modell individuell übersetzen. Da wir im ER-Modell nur 1:N-Beziehungen zwischen den Entitäten haben, wird auf die Übersetzung von 1:1-Beziehungen und M:N-Beziehungen nicht weiter eingegangen. Für die Übersetzung der 1:N-Beziehungen im ER-Modell betrachten wir die Beziehungen „nimmt auf“ und „gehört zu“. Zur Übersetzung der Beziehung „nimmt

auf“ wird das identifizierende Attribut des Probanden („ID“) in die Tabellen aller drei Objekte „Einfaches Bild“, „Blickposition Bild“ und „Emotion Bild“ mit eingetragen. Somit ergibt sich für „Einfaches Bild“, „Blickposition Bild“ und „Emotion Bild“ die Spalte „Probanden-ID“ zur Identifizierung des Probanden zum jeweiligen Bild. Die Übersetzung der Beziehung „gehört zu“ funktioniert analog. Hier wird das identifizierende Attribut des Smileys zusätzlich in die Tabelle „Emotion Bild“ miteingenommen. Dementsprechend speichern wir für das Objekt „Emotion Bild“, neben dem aufgenommenen Bild, die ID der Emotion „Smiley“.
Aus der Übersetzung des Entity-Relationship-Modells und unter Berücksichtigung der Anforderungen resultiert folglich eine Datenbank „eyeTracker“ mit insgesamt sechs Tabellen (Abbildung 3.2):

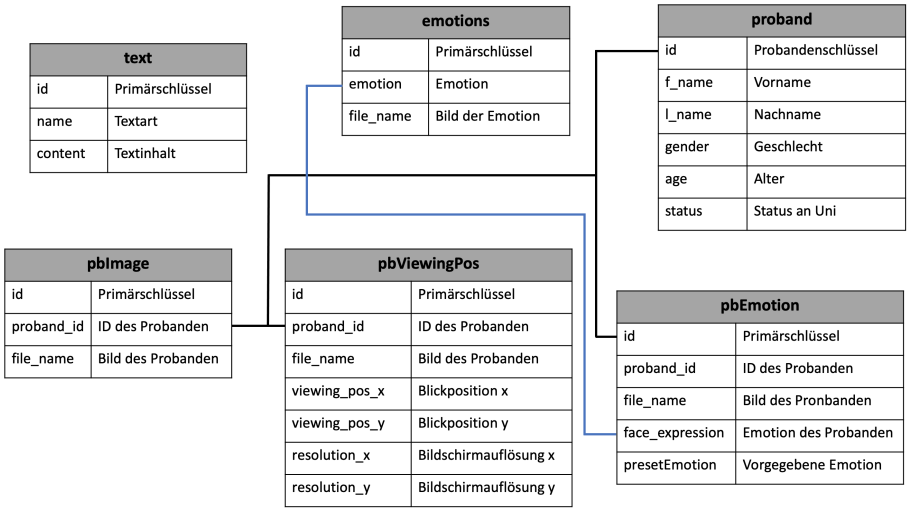


Abbildung 3.2: Datenmodell des logischen Entwurfs der Datenbank „eyeTracker“

- Tabelle „emotions“:
  Speichert die Bilder der Smileys, die für den Emotion-Tracker eingesetzt werden. Hierzu werden fünf Smileys bereitgestellt, die die Emotionen glücklich, neutral, traurig, überrascht und ängstlich repräsentieren.

- Tabelle „text“:
  Speichert den Text für den Datenschutzhinweis, den Text für die verschiedenen Anleitungen auf der Startseite sowie die Kurzanleitungen in den Pop-Ups beim Öffnen des jeweiligen Trackers.

- Tabelle „proband“:
  Speichert die Informationen des Probanden, welche freiwillig angegeben werden können. Hier werden Informationen wie Vor- und Nachname, Geschlecht, Alter und Status des Probanden zusammengetragen. Falls vom Probanden keine Daten angegeben werden wollen oder die Angaben unvollständig sind, bleiben die Felder dieser Spalten anonymisiert.

- Tabelle „pbImage“:
  Speichert die Bilder, die der Proband unter der Option „Nichts/Einfaches Bild“ aufnimmt. Hierzu wird jedem aufgenommenem Bild, die ID des Probanden zugeordnet.

- Tabelle „pbEmotion“:
  Speichert die Bilder, die der Proband unter der Option „Emotion/Smiley“ aufnimmt. Das Verfahren basiert auf der gleichen Methode wie in der Tabelle „pbImage“. Zusätzlich zu dieser Information wird die Emotion, die vom Emotion-Detector beim Probanden abgelesen wurde, eingetragen sowie die Emotion, die der Proband nachahmen sollte.

- Tabelle „pbViewingPos“:
  Speichert die Bilder, die der Proband unter der Option „Blickpunkt“ aufnimmt. Hier werden genauso wie in den Tabellen „pbImage“ und „pbEmotion“ die Bilder mit der passenden Probanden-ID gesichert. Zusätzlich zu dieser Information gibt es noch zwei weitere Spalten, um die x- und y-Koordinate der Blickposition abzuspeichern, und zwei Spalten für die x- und y-Koordinate der Bildschirmauflösung.

3.1.4 Physischer Entwurf

Nachdem in den vorangegangenen Kapiteln die theoretischen Grundlagen und das konzeptuelle Datenbankschema aufgebaut wurde, können wir die Tabellen und Abhängigkeiten mithilfe von SQL in Datenbanken definieren, sodass eine Bearbeitung mit der Datenbank-Software möglich ist.
Für diese Phase des Datenbankentwurfs wurde das relationale Datenbanksystem MySQL zusammen mit der standardisierten Skriptsprache SQL eingesetzt. Es ist geplant eine mehrbenutzerfähige Webplattform mittels eines Client-Server-Modells aufzubauen. Dieses Modell wird von MySQL unterstützt und bietet mehreren Nutzern die Möglichkeit, auf gleiche Datenbestände zuzugreifen. Es kann auf verschiedenen Arten von Betriebssystemen eingesetzt werden, wie zum Beispiel Windows, Linux und MacOS und ermöglicht so eine flexible Anwendung. Für die hohen Datenbestände bei Studien, ist dies ein wichtiges Kriterium bei der Entwicklung der Software. Außerdem stellt das MySQL-Datenbanksystem für jede unterstützte Programmiersprache Schnittstellen an, sodass eine Kommunikation mit der Datenbank möglich ist. Somit kann PHP mittels der in dieser Arbeit entwickelten Algorithmen eine Verbindung zur Datenbank aufbauen.
Nachdem das Datenbanksystem für die praktische Entwicklung der Tabellen gewählt wurde, konnten die Tabellen aus dem logischen Schema des Datenbankentwurfs anhand von DDL-Statements (Data-Definition-Language) für das MySQL System aufgestellt werden. Dazu haben wir alle Tabellen analysiert und entsprechende DDL-Statements erstellt.
Als Beispiel wird das DDL für das Relationsschema „Proband“ beschrieben und

erklärt.

CREATE TABLE proband(id INT NOT NULL PRIMARY KEY AUTO_INCREMENT, f_name VARCHAR(30) NOT NULL, l_name VARCHAR(30) NOT NULL, gender VARCHAR(30) NOT NULL, age INT(20) NOT NULL, status VARCHAR(30) NOT NULL)

Jedes Mal wenn ein neuer Proband in die Tabelle aufgenommen wird, bekommt dieser einen eindeutigen Schlüssel durch MySQL zugeordnet. Dieser Primary Key wird automatisch inkrementiert und kann zudem als ID des Probanden verwendet werden. Der Vor- und Nachname, das Geschlecht und der Status des Probanden werden in der Tabelle als ein String variabler Länge (*varchar*) gespeichert, für die jeweils 30 Zeichen zur Verfügung stehen. Ebenfalls ist vorausgesetzt, dass ein Proband all diese Attribute zugewiesen bekommt, d.h. diese Attribute dürfen nicht leer sein (NOT NULL). Das Alter des Probanden wird jeweils als integer abgespeichert. Aus diesem DDL ergibt sich schließlich unsere erste Tabelle in der MySQL Datenbank und sieht durch den Einsatz des XAMPP-Tools phpmyadmin wie folgt aus (Abbildung 3.3):

+ Optionen

	id	f_name	l_name	gender	age	status
Bearbeiten Kopieren Löschen	313	Anna	Müller	w	25	student
Bearbeiten Kopieren Löschen	314	Renate	Schmidt	w	50	mitarbeiter
Bearbeiten Kopieren Löschen	315	unknown	unknown	unknown	0	unknown
Bearbeiten Kopieren Löschen	317	Alexander	unknown	m	0	student
Bearbeiten Kopieren Löschen	318	Oliwia	Oles	w	23	student
Bearbeiten Kopieren Löschen	319	Ralf	Weber	m	43	sonstige
Bearbeiten Kopieren Löschen	320	Roufayda	Salaheddine	w	21	student
Bearbeiten Kopieren Löschen	321	unknown	unknown	unknown	0	unknown
Bearbeiten Kopieren Löschen	322	unknown	unknown	unknown	0	unknown

Abbildung 3.3: Screenshot der Tabelle „proband“ unter dem Tool *phpMyAdmin*

Die Relation „proband“ speichert die freiwillig angegebenen personenbezogenen Daten der Probanden. Beim Anlegen eines Probanden wird erstmal geprüft, ob dieser schon in der Datenbank vorhanden ist. Hierzu müssen alle Angaben des Probanden genau mit den Daten einer bereits vorhandenen Reihe in der Tabelle übereinstimmen. Damit soll das Anlegen von doppelten Probanden vermieden werden. Dies haben wir mit Hilfe einer if-Abfrage umgesetzt, die die Angaben jedes neuen Probanden mit den schon in der Tabelle vorhandenen Datensätzen vergleicht. Jeder Proband hat ebenso die Möglichkeit keine Informationen anzugeben und unbekannt an der Studie teilzunehmen. Wenn dies der Fall ist oder die Angaben unvollständig ausgefüllt sind, werden die Felder in dieser Reihe mit NULL-Werten gefüllt. So wird der Proband anonymisiert in der Datenbank angelegt und bekommt

lediglich eine ID zugeordnet. Mit der ID kann außerdem jeder Proband seine Daten zu einem späteren Zeitpunkt einsehen oder gegebenfalls löschen.

Tabelle „emotions“

Die Relation „emotions“ speichert die PNG-Dateien der Smileys, die für die Emotion-Detection verwendet werden. Die Tabelle setzt sich aus drei Attributen zusammen: Erstens die „id“ (*int*) als Primärschlüssel, um die Eindeutigkeit der Reihen in der Relation zu gewährleisten. Das zweite Attribut „emotion“ (*varchar*) kennzeichnet die verschiedenen fünf möglichen Emotionen glücklich, neutral, traurig, überrascht und ängstlich. Das dritte Attribut „file_name“ (*varchar*) sichert die Dateinamen der Bilder, um diese für den Emotion-Detector aufrufen zu können. Die Bilder der Smileys werden dazu in einem Ordner auf dem XAMPP-Server aufbewahrt und können anschließend über den Dateinamen in der Datenbank gesichert werden. Jede Emotion entspricht einer Zeile, sodass die Tabelle eine Kardinalität von 5 besitzt. (Abbildung 3.4)

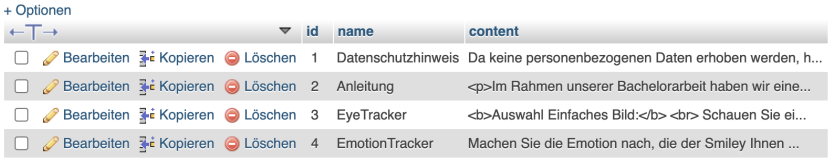
Abbildung 3.4: Screenshot der Tabelle „emotions“ unter dem Tool *phpMyAdmin*

Tabelle „text“

Die Relation „text“ speichert die verschiedenen Textinhalte, die auf der Webseite angezeigt werden. Die Tabelle besitzt die Attribute „id“ (*int*) für den Primärschlüssel, „name“ (*varchar*) für die Art des Textes und „content“ (*varchar*) für den Inhalt des Textes. Es werden insgesamt vier Texte in der Tabelle gesichert: der Datenschutzhinweis, die Anleitung auf der Startseite und die Kurzanleitungen für die Pop-Ups beim Öffnen des jeweiligen Trackers. Dementsprechend hat die Relation eine Kardinalität von 4. (Abbildung 3.5)

Tabelle „pbImage“, „pbEmotion“ und „pbViewingPos“

In den Relationen „pbImage“, „pbEmotion“ und „pbViewingPos“ werden die Bilder gespeichert, die alle Probanden im Laufe der Studie aufnehmen. Hierzu werden

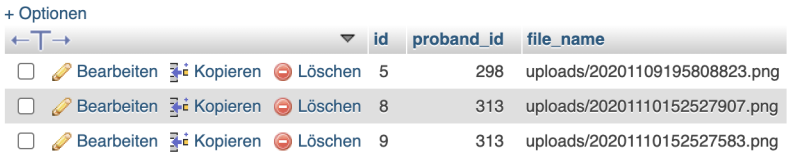


+ Optionen

	id	name	content
Bearbeiten Kopieren Löschen	1	Datenschutzhinweis	Da keine personenbezogenen Daten erhoben werden, h...
Bearbeiten Kopieren Löschen	2	Anleitung	<p>Im Rahmen unserer Bachelorarbeit haben wir eine...
Bearbeiten Kopieren Löschen	3	EyeTracker	Auswahl Einfaches Bild: Schauen Sie ei...
Bearbeiten Kopieren Löschen	4	EmotionTracker	Machen Sie die Emotion nach, die der Smiley Ihnen ...

Abbildung 3.5: Screenshot der Tabelle „text“ unter dem Tool *phpMyAdmin*

in der Tabelle „pbImage“ die Bilder unter der Option „Nichts/Einfaches Bild“, in der Tabelle „pbEmotion“ die Bilder unter der Option „Emotion/Smiley“ und in der Tabelle „pbViewingPos“ die Bilder unter der Option „Blickpunkt“ gesichert.
Alle drei Relationen arbeiten nach dem gleichen Funktionsprinzip und ähneln sich dementsprechend in ihrer Struktur. Jede neue Reihe bekommt unter dem Attribut „id“ (*int*) einen Primärschlüssel für die Eindeutigkeit der Inhalte zugeordnet. Zusätzlich zu dieser ID gibt es ein weiteres Attribut „proband_id“ (*int*). In diesem Attribut wird die ID des Probanden gesichert, welche dieser in der Tabelle „proband“ zugeordnet bekommen hat. Mit Hilfe dieser Probanden-ID kann zugeordnet werden, welches Bild welchem Probanden zugeordnet ist. Im dritten Attribut „file_name“ ist für alle drei Tabellen jeweils der Dateiname des Bildes hinterlegt, das der Proband erzeugt hat. Die Bilder sind hier ebenfalls in einem Ordner auf dem Server abgelegt und können über den Dateinamen abgerufen werden.
Die Tabelle „pbImage“ setzt sich nur aus den drei oben genannten Attributen zusammen (Abbildung 3.6).



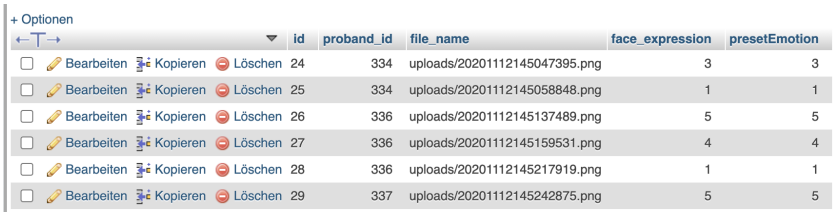
+ Optionen

	id	proband_id	file_name
Bearbeiten Kopieren Löschen	5	298	uploads/20201109195808823.png
Bearbeiten Kopieren Löschen	8	313	uploads/20201110152527907.png
Bearbeiten Kopieren Löschen	9	313	uploads/20201110152527583.png

Abbildung 3.6: Screenshot der Tabelle „pbImage“ unter dem Tool *phpMyAdmin*

Im Gegensatz dazu wird in „pbEmotion“ zusätzlich zu diesen Daten unter dem Attribut „face_expression“ (*int*) die jeweilige Emotion eingetragen, die vom Emotion-Detector beim Probanden abgelesen wurde. Dazu ist jedem der fünf möglichen Emotionen eine Zahl zwischen 1 und 5 zugeordnet, die gemeinsam mit dem Bild in dieser Tabelle bewahrt wird. Das fünfte Attribut „presetEmotion“ (*int*) speichert die vorgegebene Emotion des Smileys, welche der Proband nachahmen sollte (Abbildung 3.7). Diese zusätzliche Information wird für die Evaluierung der Emotionserkennung und -genauigkeit benötigt.

In der Tabelle „pbViewingPos“ hingegen gibt es neben den drei oben genannten Spalten noch die Attribute „viewing_pos_xg“ (*int*) und „viewing_pos_y“ (*int*), wo die x- und y-Koordinate der Blickposition erfasst werden. Ebenfalls werden in dieser Tabelle die x- und y-Koordinate der Bildschirmauflösung unter den Attributen „resolution_x“ (*int*) und „resolution_y“ (*int*) für die Untersuchung der Eye-Tracking

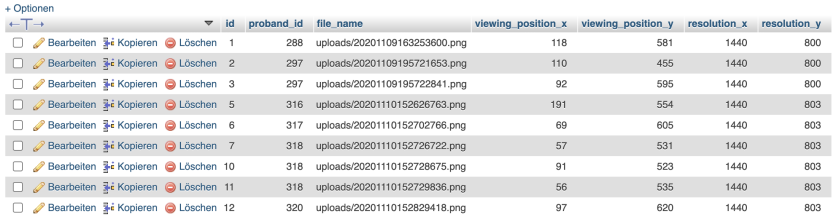


+ Optionen

	id	proband_id	file_name	face_expression	presetEmotion
Bearbeiten Kopieren Löschen	24	334	uploads/20201112145047395.png	3	3
Bearbeiten Kopieren Löschen	25	334	uploads/20201112145058848.png	1	1
Bearbeiten Kopieren Löschen	26	336	uploads/20201112145137489.png	5	5
Bearbeiten Kopieren Löschen	27	336	uploads/20201112145159531.png	4	4
Bearbeiten Kopieren Löschen	28	336	uploads/20201112145217919.png	1	1
Bearbeiten Kopieren Löschen	29	337	uploads/20201112145242875.png	5	5

Abbildung 3.7: Screenshot der Tabelle „pbEmotion“ unter dem Tool *phpMyAdmin*

Genauigkeit gesichert (Abbildung 3.8).



+ Optionen

	id	proband_id	file_name	viewing_position_x	viewing_position_y	resolution_x	resolution_y
Bearbeiten Kopieren Löschen	1	288	uploads/20201109163253600.png	118	581	1440	800
Bearbeiten Kopieren Löschen	2	297	uploads/20201109195721653.png	110	455	1440	800
Bearbeiten Kopieren Löschen	3	297	uploads/20201109195722841.png	92	595	1440	800
Bearbeiten Kopieren Löschen	5	316	uploads/20201110152626763.png	191	554	1440	803
Bearbeiten Kopieren Löschen	6	317	uploads/20201110152702766.png	69	605	1440	803
Bearbeiten Kopieren Löschen	7	318	uploads/20201110152726722.png	57	531	1440	803
Bearbeiten Kopieren Löschen	10	318	uploads/20201110152728675.png	91	523	1440	803
Bearbeiten Kopieren Löschen	11	318	uploads/20201110152729836.png	56	535	1440	803
Bearbeiten Kopieren Löschen	12	320	uploads/20201110152829418.png	97	620	1440	803

Abbildung 3.8: Screenshot der Tabelle „pbViewingPos“ unter dem Tool *phpMyAdmin*

3.2 Motivation XAMPP

Es ist geplant eine webplattformbasierte Datenhaltungs- und Bewertungssoftware für Studien zu entwickeln. Studiendaten liegen jedoch meist in großen Mengen vor und erfordern eine gut organisierte und sichere Verwaltung. Um dies zu realisieren, können die Daten in Datenbanken gesichert werden. Eine häufige und weit verbreitete Möglichkeit, um Daten zu speichern und zu verwalten, ist in Form eines relationalen Datenbankmodells. Dort werden sie logisch in Tabellen abgesichert, welche zueinander in Relation stehen. Für unsere Software haben wir uns erst überlegt, welche Anforderungen sie erfüllen muss und uns daraufhin für das relationale Datenbankmodell als Grundlage zur Datensicherung entschieden. Nachdem die theoretischen Grundlagen geklärt waren, ging es darum, sich für die effizienteste Methode zur Entwicklung und Implementierung der Software zu entscheiden. Die gesamte Software basiert auf dem Client-Server-Modell inklusive einer Datenbankanbindung. Der clientseitige Teil beinhaltet die Datenakquise unter Anwendung von JavaScript. Auf diesen Teil wird in dieser Bachelorarbeit aber nicht näher eingegangen. Der serverseitige Teil, sprich die Datenverarbeitung und die Datensicherung, beinhaltet zu aller Erst das Aufsetzen einer Datenbank. MySQL, als eines der bekanntesten Datenbanksysteme, schien uns als eine gute Wahl für die Verwaltung der Studiendaten.
Eine wichtige Komponente bei der Entwicklung einer Web-Applikation ist eine Umgebung zum Testen. Für den Einsatz des XAMPP-Pakets, der alle notwendigen Komponenten (Apache, MySQL und PHP bzw. Perl) zur Entwicklung einer

Webanwendung beinhaltet, gibt es verschiedene gute Gründe. Erstens ist es nicht notwendig jede einzelne Komponente über den Paketmanager zu installieren, da mithilfe von XAMPP eine funktionsfähige Apache-Umgebung aufgesetzt werden kann. Zudem stellt das XAMPP-Paket immer die aktuellste Version der jeweiligen Komponenten in einer aufeinander abgestimmten Form zur Verfügung, welches ein wichtige Voraussetzung für die Entwicklung von Webanwendungen ist. Ein letzter entscheidender Punkt für den Einsatz von XAMPP bietet der vereinfachte Deployment-Prozess. XAMPP bietet ein Entwicklungssystem, auf dem die Webanwendungen unter realen Bedingungen ausgiebig getestet und verbessert werden können, welches für den Entwicklungsprozess einer Software ein maßgeblicher Schritt ist. [Rei10]

3.3 Datenbankschnittstellen

Nachdem die Datenbank erstellt wurde, geht es im nächsten Schritt darum, von außen mit der Datenbank zu arbeiten und zu kommunizieren. Zur Implementierung von Anwendungen, verfügen Datenbanken im Allgemeinen über mindestens eine Programmierschnittstelle, die als eine Bibliothek von Datenstrukturen und Funktionen bereitgestellt wird. Mit Hilfe dieser Funktionen kann eine Kommunikation mit dem Datenbanksystem, die Definition und Ausführung von Anfragen, sowie die Verarbeitung von Ergebnissen erreicht werden. Für die in dieser Bachelorarbeit implementierten Programme, waren die Schnittstellen für die Programmiersprache PHP von Relevanz.
PHP bietet dazu zwei verschiedene APIs (Application Programming Interface) an, um eine Verbindung zu dem MySQL-Server aufzubauen: MySQLi (Improved MySQL) und PDO (PHP Data Objects). Bei MySQLi handelt es sich um eine verbesserte Erweiterung von PHP zum Zugriff auf die MySQL-Datenbank, die sich sowohl prozedural als auch objektorientiert einsetzen lässt. Die MySQLi-Schnittstelle unterstützt ausschließlich MySQL-Datenbanken und bietet eine einfache Möglichkeit, mit der Datenbank zu kommunizieren und Queries auf dem Server auszuführen. PDO hingegen, bieten eine Datenzugriffs-Abstraktionsschicht für die Arbeit mit unterschiedlichen SQL-basierten Datenbanken in PHP und ermöglichen so eine konsistente API für die Arbeit mit verschiedenen Datenbanksystemen. [API20]
In dieser Arbeit haben wir uns zur Kommunikation mit dem Datenbanksystem für die MySQLi -Schnittstelle (objektorientiert) entschieden, da es sich bei der Arbeit um ein MySQL-spezifisches Projekt handelt und MySQLi Vorteile bezüglich Geschwindigkeit und Funktionalität mit sich bringt.

Verbindungsaufbau zum MySQL-Server

In PHP kann die Verbindung zum MySQL-Server mit Hilfe der mysqli_connect()-Funktion aufgebaut werden. Die gesamte Kommunikation zwischen PHP und der

MySQL-Datenbank findet dabei über diese Verbindung statt. Der Verbindungsaufbau wurde in Form einer Funktion realisiert, die gleichzeitig auch die „eyeTracker"-Datenbank initialisiert. Im Folgenden wird die createDB-Funktion beschrieben und erklärt.

```
1     <?php
2     // build connection to MySQL and create the Database eyeTracker
3     function createDB($dbHost, $dbUsername, $dbPassword) {
4         $db = new mysqli(
5             $dbHost,
6             $dbUsername,
7             $dbPassword
8         );
9
10        // Check connection
11        if($db->connect_error){
12            die("ERROR: Could not connect. \n"
13                . $db->connect_error . "\n"
14                . $db->connect_errno
15               );
16        }
17
18        // Attempt create database query execution
19        $sql = "CREATE DATABASE eyeTracker";
20        if($db->query($sql) === true){
21            echo "Database created successfully <br>";
22        } else{
23           echo "ERROR: Could not able to execute $sql. " . $db->error;
24        }
25    }
26    ?>
```

Die Funktion createDB bekommt drei Parameter dbHost, dbUsername und dbPassword übergeben, die den Eingabeparametern der mysqli_connect()-Funktion entsprechen. Der Verbindungsaufbau zur Datenbank findet direkt im Anschluss statt und wird in der Variable $db gespeichert. Da wir uns für die objektorientierte Variante der MySQLi-Schnittstelle entschieden haben, benutzt man nicht direkt die mysqli_connect()-Funktion, sondern erzeugt die Verbindung mit Hilfe des Befehls:

new mysqli("hostname", üsername", "password", "database")

Der Parameter *hostname* spezifiziert den Hostnamen (z.B. localhost), oder die IP-Adresse des MySQL-Servers. Für die Parameter *username* und *password* werden die Zugangsdaten für den Zugriff auf den MySQL-Server bestimmt und der letzte Parameter *database* gibt die MySQL-Datenbank an, die beim Ausführen der Abfragen verwendet werden soll.
Im Anschluss wird durch eine if-Abfrage erstmal überprüft, ob der Verbindungsaufbau erfolgreich stattgefunden hat, ansonsten gibt er den passenden Error aus.

Wenn der Verbindungsaufbau erfolgreich war, kann anschließend die „eyeTracker“-Datenbank mittels des DDL-Statements CREATE DATABASE aufgesetzt werden. Genauso wie für den Verbindungsaufbau überprüfen wir, ob das Anlegen der Datenbank erfolgreich war und geben für den gegenteiligen Fall eine Fehlermeldung aus.
Die Funktion createDB haben wir in einer Datei db.php, gemeinsam mit den DDL-Statements der Tabellen und wichtigen Funktionen zur Verwaltung des Datenbanksystems, ausgelagert. Um die Verbindung zur Datenbank aufzubauen, muss die createDB-Funktion mit den passenden Parametern ausgeführt werden. Dazu wird die Funktion in einer externen Datei (index.php) ausgelagert und bekommt dort die notwendigen Parameter zum Verbindungsaufbau übergeben.
Die Verbindung zum MySQL-Datenbankserver wird automatisch geschlossen, sobald die Ausführung des Skripts beendet ist. Wenn sie aber früher geschlossen werden soll, kann man dies durch einen Aufruf der PHP-Funktion $mysqli->close() bewerkstelligen.

3.4 Tutorial und GUI

Zur Veranschaulichung der implementierten Software, wird im Folgenden anhand eines Tutorials und einer grafischen Benutzeroberfläche die Funktionsweise und die Anwendungsmöglichkeiten der Webanwendung erklärt.

Start und Datenschutz

Die Software startet mit dem Öffnen der Startseite (index.html). Bevor man jedoch auf die Webseite geleitet wird, öffnet sich ein Pop-Up, um den Probanden über den Datenschutz bei der Teilnahme dieser Studie zu informieren (Abbildung 3.9). Der Text des Datenschutzhinweises ist dazu in der Datenbank (Tabelle: *text*) gespeichert und wird über die PHP Schnittstelle auf der Webseite angezeigt. Nun hat der Proband die Möglichkeit, entweder den Datenschutzhinweis zu akzeptieren oder aber abzulehnen. Bei einer Ablehnung ist die Teilnahme an der Studie nicht möglich und die Software leitet den Nutzer automatisch auf die Google-Seite weiter. Stimmt der Nutzer jedoch den Datenschutzbedingungen zu, findet eine Weiterleitung auf die Startseite statt, sodass er an der Studie teilnehmen kann.

Startseite

Die Startseite der Webseite lässt sich in mehrere Bereiche gliedern (Abbildung 3.10). Auf der rechten Seite befindet sich eine kleine Einführung sowie eine Anleitung über die Webplattform, um dem Probanden eine kleine Übersicht über das Ziel der Studie und über die verschiedenen Varianten zu geben. Auf der linken Seite

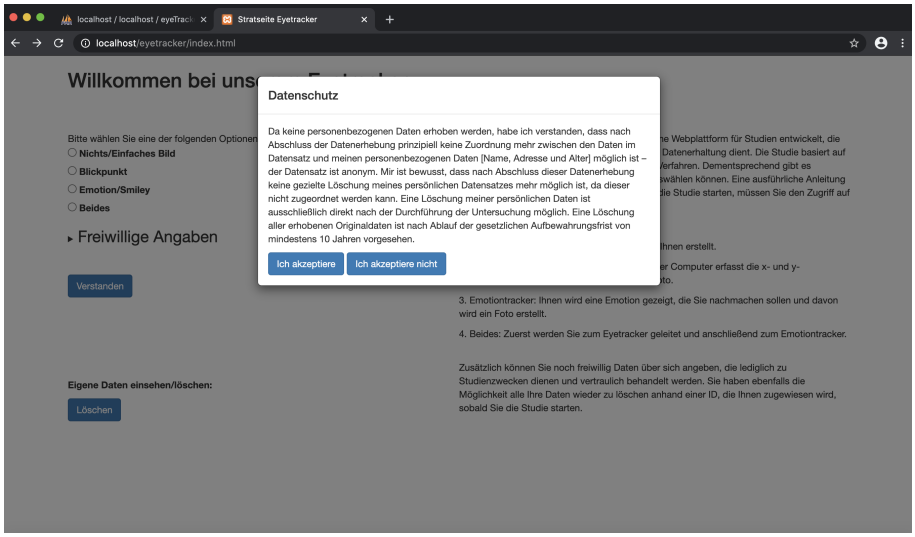


Abbildung 3.9: Grafische Benutzeroberfläche zum Anzeigen des Datenschutzhinweis

kann der Proband anhand einer Liste entscheiden, an welcher der vier Varianten er teilnehmen möchte. Die Liste enthält folgende Optionen:

- **Nichts/Einfaches Bild:** Aufnahme eines einfachen Bildes
- **Blickposition:** Betrachten eines Punktes in der Mitte des Bildschirms sowie die Erfassung der Koordinaten des Blickpunktes mit anschließender Aufnahme eines Bildes
- **Emotion/Smiley:** Nachahmen einer vorgegebenen Emotion, die vom System gedeutet wird mit anschließender Aufnahme eines Bildes
- **Beides:** Weiterleitung zum Eye-Tracker und zum Emotion-Tracker

Unterhalb der Auflistung der Varianten kann der Proband unter dem Punkt „Freiwillige Angaben“ personenbezogene Daten angeben, die ausschließlich für Studienzwecke eingesetzt werden. Unter diese Angaben fallen Informationen bezüglich Name, Geschlecht, Alter und Status des Teilnehmers. Ein wichtiger Hinweis ist, dass diese Daten nur freiwillig angegeben werden müssen. Entscheidet sich der Nutzer, anonymisiert an der Studie teilzunehmen, wird ihm lediglich eine ID zugeordnet. Anhand der ID hat jeder Teilnehmer der Studie die Möglichkeit seine Daten, die im Laufe der Studie gesammelt wurden, einzusehen und bei Bedarf zu löschen. Dies erfolgt unter dem Punkt „Eigene Daten einsehen/löschen".

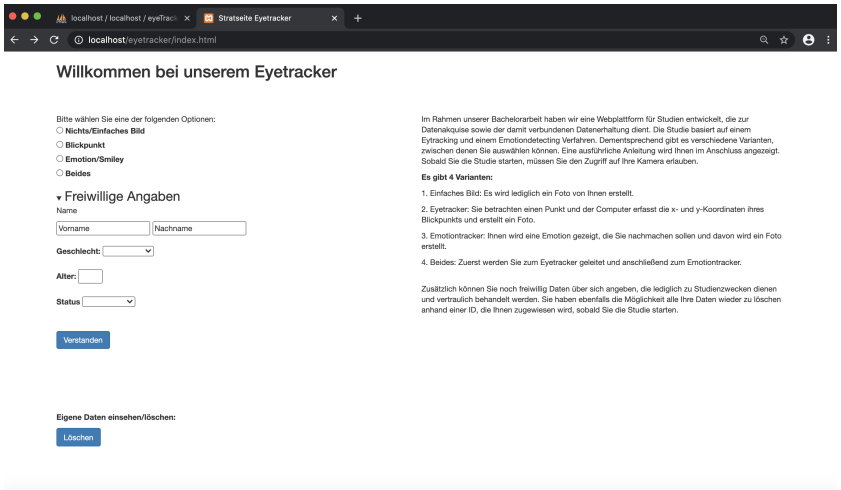


Abbildung 3.10: Grafische Benutzeroberfläche der Startseite der Software

Variante „Einfaches Bild/Nichts“

Entscheidet sich der Nutzer für die Variante „Einfaches Bild/Nichts“ wird er auf die passende Webseite weitergeleitet. Bevor der Teilnehmer jedoch starten kann, öffnet sich ein Pop-Up mit einer Kurzanleitung für diese Variante, anhand dessen er weiß, was zu tun ist. Genauso wie beim Datenschutz und bei der Anleitung auf der Startseite, ist der Text dieser Kurzanleitung in der Datenbank (Tabelle: *text*) gesichert. Anschließend öffnet sich die Seite zum Aufnehmen des Bildes (Abbildung 3.11). Hierbei ist zu beachten, dass bei der ersten Nutzung der Zugriff auf die Kamera erlaubt werden muss. Anschließend befindet sich oben links auf der Webseite das Videofeld, anhand dessen man ein Foto aufnehmen kann, Dazu muss der Button „Take photo“ betätigt werden. Ebenfalls lässt sich auf der Seite noch die ID des Nutzers anzeigen, die aus der Datenbank (Tabelle: *proband*) ausgelesen wird. Es ist wichtig, dass sich der Proband die ID merkt, da er anhand dieser seine Daten im Nachhinein einsehen und löschen kann. Oben rechts auf der Webseite gibt es zudem noch ein Fragezeichen-Button, mit dem bei Unklarheiten das Pop-Up mit der Kurzanleitung wieder geöffnet werden kann.

Variante „Blickpunkt“

Bei der Variante „Blickpunkt“ ist der Aufbau der Seite ähnlich dem der Variante „Einfaches Bild/Nichts“ (Abbildung 3.12). Zuerst öffnet sich wieder automatisch die Kurzanleitung für diese Variante, um dem Nutzer die Funktionsweise zu erklären. Die Kurzanleitung kann wieder mit Hilfe des Fragezeichen-Buttons geöffnet werden. Auch die Probanden-ID und das Videofeld befinden sich an derselben Stelle wie in der vorherigen Variante. Der Unterschied bei der Blickpunkt-Option ist, dass sich in der Mitte des Bildschirms ein Punkt befindet, der vom Probanden angesehen

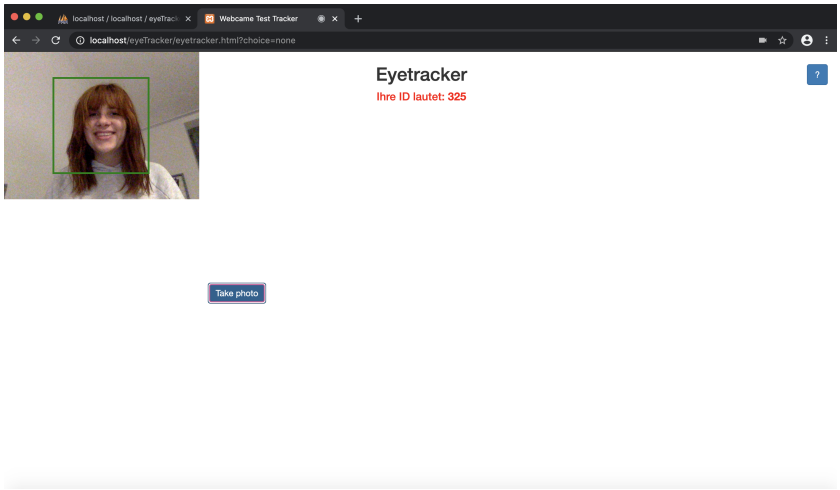


Abbildung 3.11: Grafische Benutzeroberfläche der Variante „Einfaches Bild/Nichts“

werden muss. Anschließend nimmt der Nutzer mit dem „Take photo“-Button ein Foto auf und der Computer misst die x- und y-Koordinate der Blickposition. Diese Informationen werden daraufhin gemeinsam mit dem Bild in der Tabelle „pbViewingPos“ in der Datenbank gespeichert.

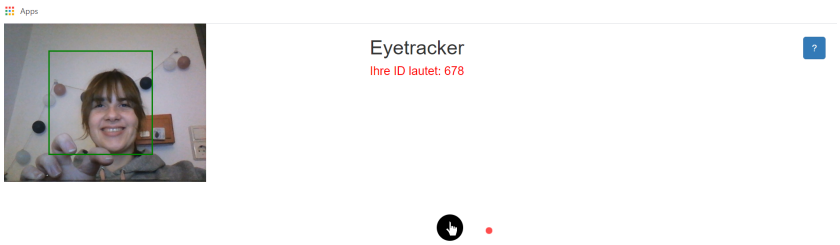


Abbildung 3.12: Grafische Benutzeroberfläche der Variante „Blickpunkt“

Variante „Emotion/Smiley“

In der Variante „Emotion/Smiley” kann der Proband seine Emotionen anhand einer Emotion-Detection von der Software auslesen lassen (Abbildung 3.13). Genauso wie in den vorherigen Varianten, erhält der Nutzer eine Kurzanleitung über die Funktionsweise und das Prinzip dieser Option. Desgleichen, wie für die anderen Optionen, wird die Probanden-ID auf der Webseite angezeigt und die Kurzanleitung kann wieder über den Fragezeichen-Button im Nachhinein geöffnet werden. Bei

der „Emotion/Smiley“-Variante geht es darum, dass der Nutzer einen Smiley mit einer Emotion (glücklich, traurig, ängstlich, neutral oder überrascht) zugewiesen bekommt. Der Smiley wird in der Mitte der Webseite angezeigt und kann über den „Neuer Smiley“-Button neu generiert werden. Die angezeigte Emotion des Smileys soll nun der Proband nachahmen. Im rechten Videofeld der Webseite wird anschließend die Emotion des Probanden von der Software gelesen und interpretiert. Dazu erscheint in dem Videofeld ein farbiges Kästchen mit der Emotion, um das Gesicht des Nutzers. Daraufhin kann der Proband mit dem „Take Photo“-Button ein Bild von sich aufnehmen, das gemeinsam mit der passenden Emotion in der Datenbank (Tabelle: *pbEmotion*) abgespeichert wird.

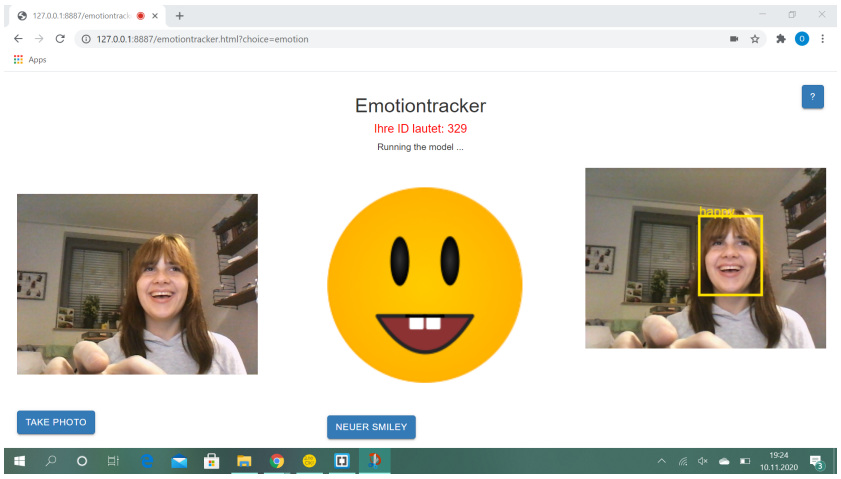


Abbildung 3.13: Grafische Benutzeroberfläche der Variante „Emotion/Smiley“

Daten einsehen/löschen

Jeder Teilnehmer an der Studie hat im Nachhinein die Möglichkeit seine Daten und Bilder einzusehen und bei Bedarf zu löschen. Dies erfolgt über den Button „Löschen“ unter dem Punkt „Eigene Daten einsehen/löschen“ auf der Startseite der Webplattform. Der Button leitet den Nutzer auf die Seite „delete.html“, wo er seine ID in einem Eingabefeld eintippen kann und anschließend über den „Weiter“-Button zu seinen Daten weitergeleitet wird (Abbildung 3.14). Der Button kann dazu nur vom Nutzer getätigt werden, wenn eine Zahl im Eingabefeld eingtragen wird, sonst ist er inaktiv. Unter dem „Weiter“-Button gibt es zusätzlich noch einen „Hauptmenu“-Button, der den Nutzer zurück auf die Startseite (index.html) führt.

Auf der Seite showData.php erhält der Proband einen Überblick über alle Daten, die im Laufe der Studie in der Datenbank abgespeichert wurden (Abbildung 3.15). Ganz oben steht die ID mit den dazugehörigen Informationen wie Name, Geschlecht, Alter und Status, welche aus der Tabelle „proband“ ausgelesen werden. Unterhalb dieser Daten sind drei Tabellen für die Optionen „Einfaches Bild/Nichts“, „Blickpunkt“

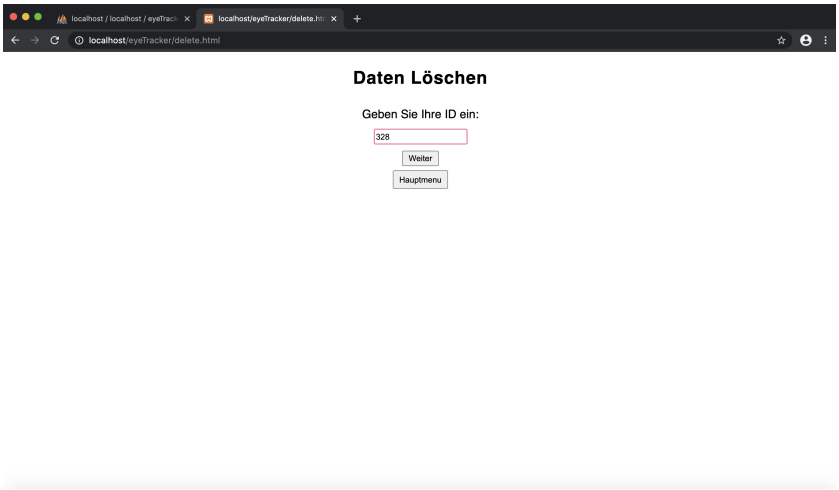


Abbildung 3.14: Grafische Benutzeroberfläche der "delete.html"

und „Emotion/Smiley“ aufgelistet. In der Tabelle „Normal“ liegen alle Bilder vor, die der Proband unter der Variante „Einfaches Bild/Nichts“ aufgenommen hat. Die Tabelle „Emotion“ listet alle Bilder mit der dazugehörigen Emotion auf, die unter der Variante „Emotion/Smiley“ aufgenommen worden sind. In der dritten Tabelle „Blickposition“ sind alle Bilder inklusive der x- und y-Koordinate der Blickposition sowie die x- und y-Koordinate der Bildschirmauflösung zusammengetragen, die unter der Variante „Blickposition“ gespeichert worden sind.

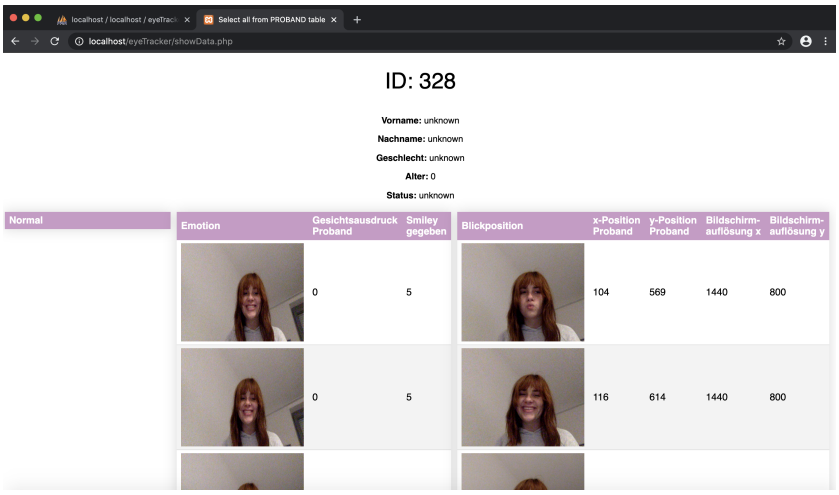


Abbildung 3.15: Grafische Benutzeroberfläche um Daten einzusehen

Entscheidet sich der Proband dazu, seine gesamten Daten löschen zu wollen, geschieht dies über den „Delete“-Button am Ende der Webseite (Abbildung 3.16). Dieser führt dazu, dass alle Daten über den Probanden inklusive seiner Bilder aus der

Datenbank entfernt werden. Zum Beispiel hat ein Proband die ID 333 und entschließt sich dazu alle seine Daten, die er im Laufe der Studie erfasst hat, zu löschen. Nun wird einerseits in der Tabelle „proband“ die Reihe mit der ID 333 entfernt, sowie alle Reihen in den Tabellen „pbImage“, „pbEmotion“ und „pbViewingPos“, die mit einer 333 unter dem Attribut „proband_id“ zu finden sind. Nachdem der Proband aus der Datenbank gelöscht wurde, wird der Nutzer automatisch auf die Startseite der Webplattform zurückgeleitet.
Gibt man jedoch in dem Textfeld in der „delete.html“ eine falsche ID ein, erhält der Nutzer die Meldung „Cannot find any record with this ID“ und wird über den „Zurück“-Button auf die Seite „delete.html“ zurückgeleitet.

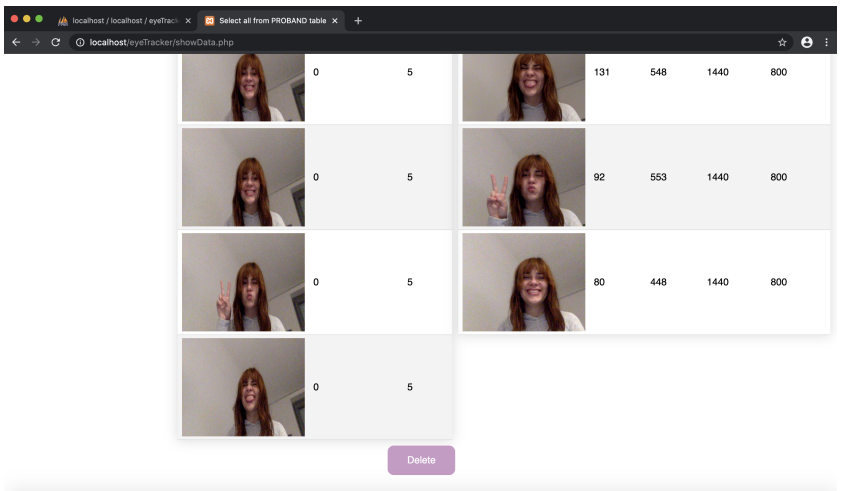


Abbildung 3.16: Grafische Benutzeroberfläche um Daten zu löschen

3.5 Implementierung und Datenstruktur

In diesem Abschnitt geben wir einen kleinen Einblick auf die technische Entwicklung der Datenhaltungssoftware. Neben der UML, zur Visualisierung der Software, werden in diesem Kapitel herausfordernde und „interessante“ Bestandteile der Implementierung vorgestellt.

3.5.1 Unified Modelling Language (UML)

Anhand einer UML soll die Visualisierung, Konstruktion und Dokumentation unserer Software grafisch dargestellt werden. Dies ermöglicht ein besseres Verständnis über die Funktionsweise des Systems. Die UML ist im Folgenden dargestellt (Abbildung: 3.17).

Die zentrale Datei der Software ist die „index.html“, welche ebenfalls für die Interaktion mit dem Nutzer verantwortlich ist. Die Inhalte und Informationen

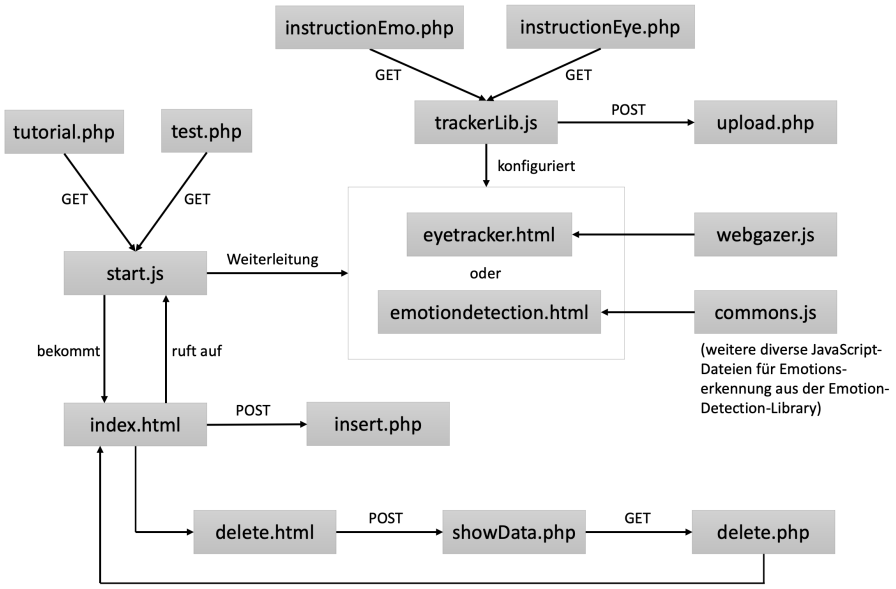


Abbildung 3.17: Visualisierung der Software anhand einer UML

bekommt die „index.html“ über die „start.js“ vermittelt. Dazu werden die Inhalte durch die „test.php“ und „tutorial.php“ aus der Datenbank geholt und über ein HTTP-Request GET übertragen. Die personenbezogenen Angaben des Probanden sowie die ID werden durch eine POST-Methode an die „insert.php“ auf dem Server übersendet und anschließend in der Datenbank gespeichert. Für die Anwendung mit dem Eye-Tracker bzw. der Emotion-Detection findet eine Weiterleitung über die „start.js“ entweder zur „eyetracker.html“ oder zur „emotiondetection.html“ statt. Der Eye-Tracker und die Emotion-Detection wurden dazu mithilfe der „webgazer.js“ und der „commons.js“ aufgesetzt. Für die Funktionsweise der Emotion-Detection ruft beispielsweise die„commons.js“ diverse JavaScript-Dateien zur Emotionserkennung aus der Emotion-Detection-Library auf. Diese Dateien wurden nicht im UML dargestellt. Andererseits wird die Funktionsweise des Eye-Trackers bzw. der Emotion-Detection über die „trackerLib.js“ konfiguriert. Auch hier erhalten sie die Inhalte aus der Datenbank über die „instructionEye.php“ und „instructionEmo.php“, welche mittels GET-Variablen gesendet werden. Um die aufgenommenen Bilder des Nutzers anschließend in der Datenbank zu speichern, werden die Bilder erst über die POST-Methode an die „upload.php“ übermittelt und daraufhin mit einem INSERT-Befehl in die jeweiligen Tabellen eingefügt. Zum Einsehen und Löschen der Daten leitet die „index.html“ erst an die „delete.html“ weiter, wo der Nutzer seine ID eingibt. Die ID wird daraufhin über eine POST-Variable an die „showData.php“ gesendet. Die „showData.php“ nimmt sich alle benötigten Daten aus der Datenbank und visualisert sie für den Nutzer. Für das Löschen der Daten ruft „showData.php“ die „delete.php“ auf und leitet den Nutzer zum Schluss wieder auf die „index.html“. Die Dateien, in denen der Verbindungsaufbau mit der Datenbank sowie das Aufsetzen

der Datenbank und Tabellen implementiert ist, finden im Hintergrund der Software statt und wurden ebenfalls nicht in diesem UML-Modell dargestellt.

3.5.2 Bilder speichern

MySQL ist ein relationales Datenbanksystem, das in erster Linie zu Verwaltung von Datentypen wie Zahlen und Zeichenketten dient. Komplexe Objekte, wie zum Beispiel die in der Studie aufgenommenen Bilder, können zwar ebenfalls in der MySQL-Datenbank verwaltet werden (anhand eines Binary Large Object (BLOB)), jedoch ist dies eine unübliche Methode und wird so in der Praxis selten realisiert. Dieses Problem lösen wir, indem der Zugriff und Aufruf der Bilder auf andere Weise umgesetzt wird. Wie es auch in der allgemeinen Praxis üblich ist, speichern wir die Bilder in Verzeichnissen auf dem Dateisystem und erzeugen Verweise, wie z.B. der Pfad zum Bild, auf diese Bilder, welche daraufhin in der Datenbank abgespeichert werden. Die Bilder werden dazu in einem geschützten Bereich der Festplatte bzw. Servers gespeichert, wo sie vor unberechtigtem Zugriff geschützt sind. Dazu haben wir auf dem Server zwei gesicherte Ordner angelegt. Einmal einen Ordner „smileys“, wo die Bilder der Smiley gespeichert sind, die bei der Variante „Emotion/Smiley“ angezeigt werden. Dazu wird ein Pfad von der Tabelle „emotions“ in der Datenbank zu diesem Verzeichnis angelegt. Der zweite Ordner „uploads“ speichert alle Bilder der Probanden ab, die im Laufe der Studie aufgenommen werden. Hierzu werden in den Tabellen „pbImage“, „pbViewingPos“ und „pbEmotion“ in der Datenbank Pfade zu diesem Verzeichnis verwaltet, um so das Speichern dieser Bilder zu ermöglichen und im Nachhinein Zugriff auf die Probanden-Bilder zu bekommen.

Bei den vom Probanden aufgenommenen Bildern wird jedes Mal ein Canvas erzeugt, auf dem das Bild draufgelegt wird. Um diese Bilder, welche mithilfe des clientseitigen JavaScript aufgenommen wurden, in der Datenbank zu speichern, müssen sie an den Server übertragen werden. Dazu wird das Bild zuerst in das URL-Format base64 codiert, dieser Code wird an den Server gesendet und dort wird das URL-Format wieder in ein Bild decodiert und abgespeichert.
Die Codierung des Bildes in base64 haben wir mit Hilfe der Methode *HTMLCanvas-Element.toDataURL().toString()* umgesetzt, sodass man zunächst eine Repräsentation des aufgenommenen Bildes als DataURL erhält und diese anschließend in ein String umwandelt. Dieser String wird daraufhin durch AJAX unter dem Einsatz der XMLHttpRequest-Methode *send()* an den Server gesendet. Diese Methode sendet einen Request an einen Server. Die Implementierung dieses Codes erfolgt über die Skriptsprache JavaScript.
Die Decodierung des URL-Formats sowie das Speichern des Bildes auf dem Server findet serverseitig statt und läuft dementsprechend mit PHP ab. Die Datenübertragung der DataURL findet hierzu mittels $_POST statt, d.h. die $_POST-Variablen werden nicht per URL (wie bei $_GET) übertragen, sondern per Formular unsichtbar im Hintergrund des Clients. Daraufhin kann der empfangene base64-Code nun

serverseitig anhand der *base64_decode()*-Methode decodiert werden. Die Methode *file_put_contents()* erzeugt dann eine Datei mit dem decodierten Bild und speichert sie im „uploads“-Ordner auf dem XAMPP-Server ab. Mit Hilfe des MySQL Statements INSERT INTO kann der Pfad zum Bild gemeinsam mit weiteren Informationen wie der Emotion oder den Blickpositionen in der Datenbank gesichert und bei Bedarf abgerufen werden.

3.5.3 Eintrag in der Datenbank einsehen/löschen

Jeder Proband der Studie kann durch Angabe seiner ID alle Daten (Persönliche Angaben, Bilder, Emotionen und Blickposition), die im Laufe der Teilnahme gesammelt wurden einsehen und bei Bedarf löschen. Die Eingabe der ID erfolgt auf der „delete.html“ in einem Textfeld.
Als erstes prüft die Software, ob die eingegebene ID überhaupt gültig ist, sprich es wird überprüft, ob es für die ID einen entsprechenden Eintrag in der Tabelle „proband“ gibt. Das funktioniert folgendermaßen:
Die in das HTML-Formular eingetippte ID muss vom Client an den Server gesendet werden. Dies erfolgt durch den Einsatz einer HTTP Request-Variable und wandelt den Inhalt der Variable in einen String zur Verwendung in einer SQL-Anweisung.

```
<?php
$deleteID = $db->real_escape_string($_REQUEST['deleteID']);
$records = fetchid($db, $deleteID);
?>
```

Nun prüfen wir serverseitig, ob die ID sich in der Datenbank befindet und speichern alle geeigneten Datensätze in der Variable $records. Die Umsetzung dazu findet durch den Aufruf einer Funktion (*fetchid()*) statt, die alle Datensätze aus der Tabelle „proband“ ausgibt, welche einer übergebenen ID entsprechen. Die Funktion *fetchid()* sieht wie folgt aus:

```
<?php
function fetchid(mysqli $db, $id) {
    $data = [];
    $sql = "SELECT␣*␣FROM␣proband␣WHERE␣proband.id␣=␣'".$id."'";
    $results = $db->query($sql);
    if ($results->num_rows > 0) {
        while ($row = $results->fetch_object()) {
            $data[] = $row;
        }
    }
    return $data;
}
?>
```

Als Parameter erhält die Funktion einmal die Datenbank (mysqli $db), welches eine Instanz von MySQLi ist, und als zweiten Parameter die vom Nutzer eingegebene ID

($id). Da wir aus dieser Funktion ein Array aus Daten zurückbekommen, definieren wir ein leeres Array, in welchem die Daten am Ende gespeichert werden sollen. Anschließend erzeugen wir ein SQL SELECT-Befehl, mit dem wir die Daten aus der Datenbank auslesen können. Dazu sollen alle Felder aus der Tabelle „proband“ ausgegeben werden, wo der Primärschlüssel des Attributs „ID“ dem übergebenen Parameter $id der Funktion entspricht. Dieser SELECT-Befehl kann anschließend durch den *query()*-Befehl ausgeführt werden und ist in der Variable $results abgespeichert. Nun überprüfen wir durch *num_rows*, ob die $results-Variable größer 0 ist, da es sonst keinen passenden Eintrag in der Tabelle geben würde. Wenn dies der Fall ist, können alle Felder aus der Reihe abgerufen werden und sie anschließend in dem leeren Array speichern. Das Anzeigen der Daten für den Probanden findet auf der showData.php statt. Hier wird zuerst anhand einer if-Abfrage geprüft, ob die Variable $records größer als 0 ist. Wenn nicht, erhält der Nutzer die Rückmeldung „Cannot find any record with this ID“ und kann durch den „Zurück“-Button auf die delete.html zurückgeleitet werden. Sind unter der ID aber Datensätze vorhanden, werden alle Informationen aus der Datenbank unter dieser ID ausgegeben. Dazu können sich Daten in den vier Tabellen „proband“, „pbImage“, „pbEmotion“ und „pbViewingPos“ befinden. Für jede dieser vier Tabellen gibt es jeweils eine SELECT-Abfrage, welcher wieder durch den query-Befehl ausgeführt wird.

```
<?php
$sql = "SELECT * FROM proband WHERE id = '".$_SESSION["deleteid"]."'";
$results = $db->query($sql);
?>
```

Um die Daten aus der Datenbank für den Nutzer sichtbar zu machen, werden diese in HTML-Tags eingebettet. Die Daten des Probanden aus der Tabelle „proband“ lassen sich beispielsweise wie folgt auf der Webseite darstellen:

```
<?php
    $record = $results->fetch_object();
?>
<p id="ID">ID: <?php echo $record->id; ?></p>
<p><b>Vorname:</b> <?php echo $record->f_name; ?></p>
<p><b>Nachname:</b> <?php echo $record->l_name; ?></p>
<p><b>Geschlecht:</b> <?php echo $record->gender; ?></p>
<p><b>Alter:</b> <?php echo $record->age; ?></p>
<p><b>Status:</b> <?php echo $record->status; ?></p>
```

Die Anzeige der aufgenommenen Bilder des Probanden auf der Webseite erfolgt analog, jedoch ist der PHP-Code in HTML-Tabellen-Tags eingebettet. Dies ermöglicht eine getrennte und strukturierte Anzeige der Bilder für die verschiedenen Varianten.

Das Löschen der Daten findet über den Delete-Button am Ende der showData.php statt. Durch das Betätigen dieses Buttons erfolgt serverseitig eine Weiterleitung auf die delete.php. Hier wird die Funktion deleteRecord() aufgerufen, die anschließend

alle Reihen in der Datenbank löscht, welche unter der ID des Probanden gespeichert sind. Die Funktion deleteRecord() sieht wie folgt aus:

```
<?php
function deleteRecord(mysqli $db, $id) {
    $sql = "DELETE
␣␣␣␣␣␣␣␣proband.*,␣pbImage.*␣,pbEmotion.*,␣pbViewingPos.*
␣␣␣␣␣␣␣␣FROM␣proband
␣␣␣␣␣␣␣␣LEFT␣JOIN␣pbImage␣ON␣proband.id␣=␣pbImage.proband_id
␣␣␣␣␣␣␣␣LEFT␣JOIN␣pbEmotion␣ON␣proband.id␣=␣pbEmotion.proband_id
␣␣␣␣␣␣␣␣LEFT␣JOIN␣pbViewingPos␣ON␣proband.id␣=␣pbViewingPos.proband_id
␣␣␣␣␣␣␣␣WHERE␣proband.id␣=␣'".$id."'";

    $result = $db->query($sql);

    if(!$result) {
        throw new Exception('Cannot␣delete␣record');
    }
}
?>
```

Als Eingabe erhält *deleteRecord()* dieselben Parameter wie die Funktion *fetchid()* von oben, also die Datenbank (mysqli $db) und die eingegebene ID ($id). Anschließend konstruieren wir ein SQL DELETE-Befehl, um Datensätze aus der Datenbank zu löschen. Das Ziel ist es, alle Daten und Informationen eines Probanden zu löschen, obwohl diese Daten in mehreren Tabellen gespeichert sind. Um jetzt nicht für alle Tabellen jeweils einen SQL-Befehl zu schreiben, besteht die Möglichkeit anhand des JOIN-Befehls Reihen aus zwei oder mehreren Tabellen miteinander zu vereinen. Die Voraussetzung dafür ist, dass diese Tabellen durch eine gemeinsame Spalte verbunden sind, wie zum Beispiel einer ID. In der Regel unterscheidet man verschiedene Typen von SQL JOIN-Befehlen wie (INNER) JOIN, LEFT (OUTER) JOIN, RIGHT (OUTER) JOIN oder FULL (OUTER) JOIN. Für unsere Datenbank benötigen wir den LEFT JOIN-Befehl, die alle Datensätze aus der linken Tabelle und die passenden Datensätze aus der rechten Tabelle zurückgibt [Joi20]. Damit können wir alle Datensätze aus den Tabellen „pbImage“, „pbEmotion“ und „pbViewingPos“ mit dem passenden Datensatz aus „proband“ verbinden. Am Beispiel für „proband“ und „pbImage“ sieht das folgendermaßen aus: Lösche aus den Tabellen „proband“ und „pbImage“ alle Daten aus der linken Tabelle („proband“) unter einer gegebenen ID sowie die passenden Daten aus der rechten Tabelle („pbImage“), wenn die Probanden-ID mit der gegebenen ID übereinstimmt.
Die SQL-Abfrage kann daraufhin, wie in den obigen Anwendungen, durch den *query()*-Befehl ausgeführt werden, sofern dies möglich ist. Wenn nicht, erhält man eine passende Fehlermeldung.

4 Evaluation

Dieses Kapitel beinhaltet die Evaluierung der Datenhaltungs- und Bewertungssoftware für Eye-Tracking- und Emotion-Detection-Studien, die im Rahmen dieser Arbeit entwickelt wurde. Das Ergebnis dieser Evaluierung bietet einen Überblick über verschiedene Kriterien, die im Rahmen dieser Arbeit überprüft und bewertet werden können. Dazu betrachten wir verschiedene Anforderungen bezüglich der Qualität der Datenbank und ihrer Einträge in den Tabellen. Anschließend analysieren wir die Qualität und Genauigkeit der Blick- und Emotionserkennung, die als Anwendung für diese Arbeit aufgesetzt wurden.

4.1 Datenbank und Datenbankeinträge

Für die Bewertung unserer implementierten Software muss unter anderem die Qualität der entwickelten Datenbank analysiert werden. Diese Analyse findet unter Berücksichtigung folgender Kriterien statt:

- **Logischer Aufbau und Struktur:** Die Datenbank und seine Einträge haben einen sinnvollen und logischen Aufbau und Struktur, sowie gut nachvollziehbare Zusammenhänge zwischen den Tabellen
- **Redundanzarmut:** Es gibt keine ungeordnete Mehrfachspeicherung von Datenwerten
- **Integritätssicherung:** Daten werden auf Korrektheit überprüft und Fehlmanipulation verhindert
- **Speicherplatz:** Platzbedarf pro Eintrag in einer Tabelle
- **Geschwindigkeit der Datenbank:** Zeit, die benötigt wird, um Daten aus der Datenbank auszulesen bzw. neue Daten in die Datenbank einzufügen

Logischer Aufbau und Struktur

Viele Anforderungen für eine effiziente Datenbank können nur durch einen logischen Aufbau und Struktur umgesetzt werden. Anhand einer gründlichen Analyse der Struktur und einem ausführlichen Datenbankentwurf lassen sich beispielsweise Redundanzen vermeiden und Integrität sichern. Für unsere Software haben wir einen logischen Aufbau durch die Anwendung eines relationalen Datenbankmodells

ermöglicht. Dementsprechend speichern wir die verschiedenen Informationen in mehreren, zusammenhängenden Tabellen. Ein Proband wird mit seinen personenbezogenen Daten in einer eigenen Tabelle angelegt. Ebenso sind die Bildaufnahmen der verschiedenen Studienvarianten in jeweils separaten Tabellen gespeichert. Mithilfe einer eindeutigen ID, die in jeder Tabelle mit abgespeichert wird, können die Bilder schließlich dem Probanden zugeordnet werden. Die Inhalte, die auf der Webseite angezeigt werden, sind ebenfalls in einer separaten und unabhängigen Tabelle gesichert.

Redundanzarmut

Bei der Entwicklung einer Datenbank für eine Software ist es wichtig ein redundantes Anlegen gleicher Daten so gut wie möglich zu vermeiden. Das bedeutet einerseits, dass die Daten, die angelegt werden, sich nicht ständig wiederholen. Andererseits sollte man die Datenbank so strukturieren, dass beim Anlegen eines neuen Datensatzes, keine bzw. so wenig Felder wie möglich mit NULL-Werten gefüllt werden. Unter Berücksichtigung dieser Voraussetzungen können wir die Speichergröße unserer Datenbank bestmöglich reduzieren und eine effiziente Datenhaltungssoftware gewährleisten.
Eine Redundanzarmut wurde erreicht, indem wir die Verwaltung der Probandendaten getrennt von den resultierenden Studiendaten speichern. Jeder Proband wird einmalig in der Probanden-Tabelle angelegt und kann anschließend für die verschiedenen Optionen von Eye-Tracking bzw. Emotion-Detection mehrere Bilder nach Belieben aufnehmen. Für jedes neue Bild muss dementsprechend eine neue Zeile in der Tabelle angelegt werden. Die Trennung der Daten in relationale Tabellen ermöglicht, dass für jedes neu aufgenommene Bild nur die notwendigen Informationen in einer getrennten Tabelle verwaltet werden und trotzdem der Bezug zu den Probanden-Daten gesichert werden kann. Die Trennung der Bilddaten zu den Probanden-Daten führt dazu, dass bei mehreren Bildern die Angaben des Probanden nicht erneut gespeichert werden müssen.
Das separate Abspeichern der Studiendaten reduziert ebenfalls die Verwaltung von NULL-Werten in der Datenbank. Da jeder Proband sowohl an allen drei Varianten als auch nur an einer oder zwei Varianten in der Studie teilnehmen kann, reduzieren wir das Anlegen von NULL-Werten, indem für jede Variante eine eigene Tabelle angelegt wurde. Es muss keine große Tabelle erzeugt werden, die alle Anforderungen an allen drei Varianten erfüllt. Nimmt ein Proband beispielsweise nur an der Option „Emotion/Smiley“ teil, können wir die Emotionen in einer Tabelle abspeichern, ohne die Spalte für die Koordinaten der Blickposition bei der Option „Blickposition“ mit NULL-Werten zu belegen. Jedoch kann ein vollständiger Einsatz von NULL-Werten in der gesamten Datenbank nicht verhindert werden. Da jeder Proband personenbezogene Angaben freiwillig angeben kann, werden die Spalten in der Probanden-Tabelle mit NULL-Werten gefüllt, sobald eine anonyme Teilnahme stattfindet.

Integritätssicherung

Die Integrität ist eine wesentliche und selbstverständliche Forderung an eine Datenbank, die sich in der Praxis jedoch nur sehr schwer umsetzen lässt. Einerseits müssen die Daten physisch korrekt gespeichert sein. Die physische Form der Integrität kann jedoch vom Datenbankprogrammierer nicht beeinflusst werden, da sie auf dem Betriebssystem und der Hardware basiert. Jedoch kann sie beispielsweise durch die richtige Auswahl einer Datenbank beeinflusst werden, wie bei unserer Software durch den Einsatz von MySQL. [Sch17]
Eine andere Form der Integrität ist die semantische Integrität, also die Übereinstimmung der Daten in der Datenbank mit der realen Welt. Hier ist die Datenbank auf die Eingabe durch den Menschen angewiesen, was problematisch sein kann. Das System kann nämlich nur schwer entscheiden, ob die eingegebenen Daten der realen Welt entsprechen oder ob ein Tipp- und Eingabefehler vorliegt [Sch17]. Dieses Problem spiegelt sich in unserer Software unter anderem bei der Eingabe der personenbezogenen Angaben wider. Das System kann diese Angaben nie auf komplette Vollständigkeit überprüfen. Jedoch kann die semantische Integrität beispielsweise durch den Einsatz von Datentyp-Definition und Eingabe-Masken gewährleistet werden. Das Alter eines Probanden kann beispielsweise nur als Zahl in einem Wertebereich zwischen 15 und 75 angegeben werden. Durch die passende Eingabemaske verhindern wir die Eingabe von Zeichenketten und dadurch, dass der Datentyp für das Alter in der Datenbank einem Integer entspricht, können keine Fließkommazahlen oder Characters gespeichert werden. Eine ähnliche Vorgehensweise haben wir für die Angabe des Geschlechts oder Status angewendet, wo der Nutzer eine Auswahl in Form eines Drop-Down-Menüs mit vorgegebenen Möglichkeiten durchführen kann.

Geschwindigkeit der Datenbank

Die Eingabe und Abfrage von Daten findet bei der Nutzung einer Datenbank permanent statt und sollte deswegen schnell erfolgen. Solche Ausführungsgeschwindigkeit sollten normalerweise im Bereich des menschlichen Reaktionsvermögens liegen, d.h. Antwortzeiten von mehr als zwei Sekunden haben eine negative Wirkung [Sch17]. Um die Qualität unserer Datenhaltungssoftware zu beurteilen, spielt diese Anforderung in unserer Evaluierung eine wichtige Rolle. Aus diesem Grund haben wir die Laufzeit für eine Datenbankabfrage bzw. für einen neuen Datenbankeintrag je nach Kategorie gemessen (siehe Abbildung 4.1).

Wie man im Schaubild sehen kann, übersteigt keine der aufgezählten Kategorien die Antwortzeit von zwei Sekunden. Vor allem das Auslesen und Anzeigen von Daten aus der Datenbank nimmt sehr wenig Zeit in Anspruch. Im Gegensatz dazu benötigt die Software zum Erzeugen und Abspeichern eines neuen Bildes eine Dauer von etwa 29 bis 382 Millisekunden. Die Zeit variiert hier abhängig von den Daten, die bei der Aufnahme eines Bildes mit abgespeichert werden müssen wie

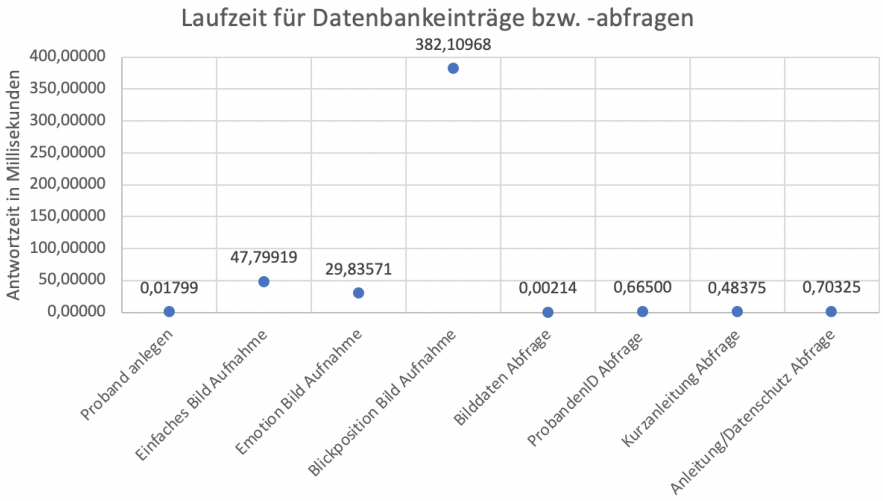


Abbildung 4.1: Laufzeit der Datenbankeinträge

z.B. die Blickposition oder die Emotion. Diese Zeiten sind dennoch schnell genug und für das menschliche Reaktionsvermögen nicht wahrnehmbar. Daraus können wir schließen, dass die Antwortzeit für eine Abfrage aus der Datenbank bzw. für einen neuen Eintrag in die Datenbank sehr gering ist und sich somit positiv auf die Performance der Software auswirkt.

Platzbedarf

Das Ziel bei der Entwicklung einer Datenbank ist es, die Größe der Datenbank so klein wie möglich zu halten, um die Abfragen zu beschleunigen und so die Performance der Software zu verbessern. Deswegen fokussiert man sich bei der Datenbankentwicklung darauf, gut geeignete Datentypen für die Attribute einer Tabelle auszuwählen. Dazu gilt: Man wählt den kleinsten Datentyp, der für das Feld ausreicht [Sto]. Zur Leistungs- und Qualitätsbeurteilung der Datenbank wurde der Platzbedarf pro Eintrag anhand der Größe der Datentypen berechnet. Für die Bewertung wurde jeweils der Platzbedarf eines Eintrags in den Tabellen „proband“, „pbImage“, „pbEmotion“ und „pbViewingPos“ analysiert.
Die Tabellen setzen sich jeweils aus Feldern mit den Datentypen Integer und Varchar zusammen. Ein Integer hat eine Speichergröße von 4 Bytes pro Feld, während die Speichergröße eines Varchars je nach Größe des Strings in Abhängigkeit zur deklarierten Spaltenlänge variiert. Die Speichergröße für einen Varchar lässt sich anhand der Formel $L+1$ Bytes (L = aktuelle Länge in Bytes für einen gegebenen String Value) berechnen [Sto20]. Folglich ergeben sich für die Tabellen folgende Speichergrößen pro Eintrag (siehe Tabelle 4.1):

Obwohl der Platzbedarf für einen Eintrag im Rahmen liegt, könnte man diesen weiter

Tabelle 4.1: Speichergröße in den Tabellen.

Tabelle in Datenbank	**Speichergröße pro Eintrag in Bytes**
proband	27 – 92 Bytes (abhängig von den personenbezogenen Daten des Probanden)
pbImage	38 Bytes
pbEmotion	46 Bytes
pbViewingPos	54 Bytes

verringern und die Performance der Software verbessern. Anstatt alle Datenwerte in den Tabellen als Integer zu speichern, wäre eine Speicherung als Tinyint (nimmt Werte zwischen 0 bis 255) oder Smallint (nimmt Werte zwischen 0 bis 65.535 oder -32.768 bis 32.767) ebenfalls ausreichend gewesen. Diese Datentypen haben eine Speichergröße pro Feld von 1 Byte bzw. 2 Bytes und führen zu einem kleineren Platzbedarf pro Eintrag in den Tabellen.

4.2 Genauigkeit der Emotions- und Blickpunkterkennung

Die Datenhaltungssoftware wurde für eine Anwendung im Rahmen einer Eye-Tracking- bzw. Emotion-Detection-Studie entwickelt. Deswegen wollen wir in der Evaluierung der Software die Qualität der Emotions- und Blickpunkterkennung analysieren. Zur Bewertung des Eye-Tracking und der Emotion-Detection wurden jeweils ca. 25 Bildaufnahmen mit folgenden Ergebnissen durchgeführt.

Emotionserkennung

Die Bewertung der Emotionserkennung fand durch den Vergleich zwischen der vorgegebenen Emotion und der vom System erkannten Emotion statt. Das Ergebnis sieht wie folgt aus:
Die Emotionen „glücklich“ und „überrascht“ wurden in allen Fällen richtig erfasst. Mehr Schwierigkeiten resultierten bei der Erkennung der Emotionen „ängstlich“ und „traurig“. Einerseits wurde „ängstlich“ in fast der Hälfte der Fälle mit der Emotion „überrascht“ verwechselt und die Emotion „traurig“ hat das System oft als „neutral“ bewertet. Schlussendlich kamen wir zu dem Ergebnis, dass bei ca. ein Drittel der Fälle die Emotion entweder falsch oder schwierig (d.h. Erkennung schwankt zwischen zwei Emotionen) erkannt wurde. Hierbei sollte jedoch angemerkt werden, dass es sich bei dem angewendeten Emotion-Detector um eine ausschließlich clientseitige Anwendung handelt, weshalb die Emotionserkennung nicht so gut ist wie bei anderen Anwendungen, die von der Nutzung zusätzlicher Hardware profitieren.

Blickpunkterkennung

Die Bewertung der Blickpunkterkennung fand durch den Vergleich der erkannten Blickposition und der Bildschirmauflösung statt. Das Ergebnis sieht wie folgt aus: Die Blickpunktvorhersage fand überwiegend links vom Mauszeiger statt. Dies beruht auf der Annahme, dass die Vorhersage unter der Voraussetzung stattfindet, dass die Augen dem Mauszeiger gewohnheitsmäßig von links nach rechts folgen. Außerdem war die Vorhersage trotz direkten Blick auf den Mauszeiger nie genau richtig. Ebenfalls konnte man sehen, dass für die 25 Bildaufnahmen der vorhergesagte Blickpunkt meist um wenige Pixel (ca. 20-50 Pixel) vom schwarzen Punkt abweicht. Die Blickpunktvorhersage wird in Abbildung 4.2 exemplarisch veranschaulicht.

Abbildung 4.2: Exemplarische Darstellung der Blickpunktvorhersage

5 Diskussion

Das Ziel der webplattformbasierten Datenhaltungssoftware war es, eine qualitative und effiziente Datenverwaltung und -organisation für Studienergebnisse zu erzielen, die im Rahmen einer Eye-Tracking- und Emotion-Detection-Anwendung gesammelt wurden. Dieses Kapitel beinhaltet eine Diskussion über die Ergebnisse der Evaluation bzw. über die Funktionsweise der implementierten Software. Die Diskussion basiert auf folgender Leitfrage: Welche Aspekte wurden gut bzw. schlecht umgesetzt und was kann folglich verbessert werden?

Wichtige Anforderungen, die eine effiziente Datenhaltungssoftware gewährleisten, wurden in dieser Arbeit vorgestellt und bei der Erstellung der Datenbank berücksichtigt und anschließend evaluiert. Aspekte wie ein logischer Aufbau und Redundanzarmut konnten besonders durch einen ausführlichen Datenbankentwurf realisiert werden. In den verschiedenen Phasen des Datenbankentwurfs konnte der Aufbau der Datenbank mit den notwendigen Zusammenhängen zwischen den Tabellen umgesetzt werden. Wir haben uns am relationalen Datenbankmodell orientiert und die Studienergebnisse jedes Probanden je nach Kategorie separat in Tabellen abgespeichert. Infolgedessen findet eine übersichtlichere Verwaltung der Daten statt und ermöglicht eine sinnvolle Weiterverarbeitung. Zusätzlich vermeidet die Aufspaltung der Daten in mehrere Tabellen das Auftreten von NULL-Werten und Redundanzen und fördert somit die Performance der Software. Zwar lassen sich NULL-Werte und Redundanzen in den Tabellen (wie z.B. bei einer anonymen Teilnahme an der Studie) nicht gänzlich vermeiden, jedoch werden sie durch den Einsatz des relationalen Datenbankmodells verringert.
Weitere Anforderungen wie der Platzbedarf pro Eintrag und die Antwortgeschwindigkeit zum Auslesen bzw. Eingeben von Daten konnten durch eine sinnvolle Datenbankplanung und -struktur geringgehalten werden. Speichergröße und Laufzeit der Software befinden sich in einem Umfang, der für den Einsatz der Software im Rahmen liegt. Dennoch kann die Speichergröße durch die Verwendung von kleineren numerischen Datentypen anstatt eines Integer verringert werden können. Ein wichtiger Punkt bei der Entwicklung der Datenbank war eine effiziente Speicherung von Bilddaten, die für die Durchführung unserer Eye-Tracking- und Emotion-Detection-Studie von großer Bedeutung ist. Dazu wurden die aufgenommenen Bilder der Probanden nicht direkt in der Datenbank gespeichert, sondern in einem geschützten Ordner auf dem Server. Infolgedessen verweisen wir auf die Bilder, in dem der Pfad zum Bild in der Datenbank gesichert wird. Durch diese Vorgehensweise verwalten wir in den Tabellen ausschließlich Varchar-Datentypen, die weniger

Speicherplatz benötigen als zum Beispiel ein BLOB-Datentyp. Dies hat in unserer Software jedoch den Nachteil, dass beim Löschen der Bilder aus der Datenbank, diese weiterhin auf dem Dateisystem vorhanden sind. Der Proband kann somit seine Bilder nicht vollständig aus dem System löschen und würde sowohl hinsichtlich Speicherplatz und Datenschutz problematisch werden. Dies wäre unter anderem ein Punkt, bei dem man bei der Weiterentwicklung der Software ansetzen könnte.
Ein weiteres Problem stellt die Software bezüglich einer flexiblen Teilnahme an der Studie dar. Jeder Proband bekommt zu Beginn einer Session eine ID zugewiesen, unter der er seine Bilder aufnimmt und die für die Weiterverarbeitung der Daten notwendig ist. Dadurch, dass die ID aber für jede Session neu generiert wird, kann ein Teilnehmer zu einem späteren Zeitpunkt kein weiteres Mal mit dieser ID an der Studie teilnehmen. Dies könnte man zum Beispiel durch eine feste Registrierung zu Beginn der Studie lösen, die alle Informationen eines Nutzers einmalig verwaltet und bei erneuter Teilnahme erweitert.
Obwohl unsere entwickelte Software viele Anforderungen an eine effiziente Webplattformbasierte Datenhaltungs- und Bewertungssoftware erfüllt, ist festzustellen, dass weiteres Entwicklungspotenzial in der vorliegenden Arbeit vorhanden ist.

6 Zusammenfassung

Das Ziel dieser Arbeit war die Erstellung mit anschließender Bewertung einer Datenbank, die in einen Webplattformbasierten Eye-Tracking- und Emotion-Detection-Studienbetrieb integriert wurde.
Die Datenbank haben wir in Anlehnung an das relationale Datenbankmodell entworfen und implementiert. Die praktische Umsetzung erfolgte anhand von MySQL und XAMPP sowie durch die Skriptsprache PHP. Hierbei war es wichtig eine Schnittstelle nach außen aufzubauen, um mit JavaScript zu kommunizieren und eine gezielte Verwaltung der Studiendaten zu erreichen. Die Datenbank speichert alle erforderlichen Daten, die zur Durchführung einer webplattformbasierten Eye-Tracking- und Emotion-Detection-Studie benötigt werden, beispielsweise personenbezogene Probandendaten, Bilddaten, Koordinaten der Blickposition oder Emotionen sowie auch spezifische Augenmerkmale [FSK17b, FEH+18] aber auch Interessante Bildbereiche [GFSK17, FKSK18]. Neben der Verwaltung dieser Daten galt hier die Herausforderung, die aufgenommenen Bilder möglichst effizient und in einem stabilen Format in der Datenbank zu organisieren. Indem alle Bilder in einem Verzeichnis auf dem Server gespeichert sind und über die Datenbank schließlich die Verwaltung der Bilddaten durch Pfade zum Verzeichnis erfolgte, konnte vor allem der Platzbedarf für einen neuen Eintrag deutlich reduziert werden. Dieses Kriterium spielt in dieser Hinsicht eine wichtige Rolle, da es bei zu hohem Speicherbedarf sonst zu Inkonsistenzen kommt, welche wiederum die Performance der Software beeinträchtigt.
Neben der Speicherung von Daten wurden in der Evaluierung weitere Anforderungen analysiert, um die Qualität der entwickelten Datenbank zu untersuchen, wie zum Beispiel eine logische Struktur der Datenbank, Redundanzarmut oder Integritätssicherung. Vor allem durch einen gut strukturierten Datenbankentwurf ließen sich diese Punkte so umsetzen, dass die Anzahl an Redundanzen und Fehlinformationen möglichst gering sind.
Trotzdem können alle genannten Anforderungen weiterhin verbessert, um die Effizienz und Qualität der Datenhaltungs- und Bewertungssoftware zu erhöhen. Zum Beispiel die Verwendung von kleineren numerischen Datentypen würde den Speicherplatz für einen Eintrag reduzieren und daraufhin die Performance verbessern. Genauso kann die Software noch um weitere Punkte verbessert werden. Beispielsweise wäre es wichtig beim Löschen von Datensätzen, die Bilder ebenfalls aus dem Verzeichnis auf dem Server zu entfernen. Zusätzlich kann die Eye-Tracking-Software so optimiert werden, dass eine flexiblere Teilnahme für Probanden möglich ist, indem im Nachhinein mit der gleichen ID wieder Bilder aufgenommen werden.

Dies sind Anforderungen an unsere Software, die vor allem in Zukunft ausgearbeitet und optimiert werden können.

Literaturverzeichnis

[API20] PHP/Tutorials/Umstieg von der veralteten MySQL-API. https://wiki.selfhtml.org/wiki/PHP/Tutorials/Umstieg_von_der_veralteten_MySQL-API, 2020. Accessed: 08.11.2020.

[AWNM20] Francisca Adoma Acheampong, Chen Wenyu, and Henry Nunoo-Mensah. Text-based emotion detection: Advances, challenges, and opportunities. Engineering Reports, page e12189, 2020.

[BDK] Olena Bochkor and Veikko Dr. Krypczyk. PHP Tutorial: Datenbankprogrammierung mit PHP und MySQL. https://entwickler.de/online/php/php-tutorial-datenbankprogrammierung-579772374.html. Accessed: 10.10.2020.

[BFG+16] H. Bahmani, W. Fuhl, E. Gutierrez, G. Kasneci, E. Kasneci, and S. Wahl. Feature-based attentional influences on the accommodation response. In Vision Sciences Society Annual Meeting Abstract, 2016.

[Bla13] Christopher Blake. Eye-Tracking: Grundlagen und Anwendungsfelder. In Handbuch standardisierte Erhebungsverfahren in der Kommunikationswissenschaft, pages 367–387. Springer, 2013.

[BvM01] Carmen Barth and Sandra van Marwick. Evaluation der Anbindung einer SQL-Datenbank an einen Apache Webserver. Bachelor Thesis, 2001. Technische Universität München.

[Con16] Conceptual Design in Database Design Process. http://www.myreadingroom.co.in/notes-and-studymaterial/65-dbms/507-conceptual-design.html, 2016. Accessed: 17.10.2020.

[Dat20] Database: Was ist eine Datenbank? https://www.oracle.com/de/database/what-is-database/, 2020. Accessed: 10.10.2020.

[Dvo07] Dalibor D Dvorski. Installing, configuring, and developing with Xampp. Skills Canada, 2007.

[Dwi19] Priya Dwivedi. Face Detection, Recognition and Emotion Detection in 8 lines of code! https://towardsdatascience.com/face-detection-recognition-and-emotion-detection-in-8-lines-of-code-b2ce32d4d5de, April 2019. Accessed: 18.11.2020.

[EFK17] Shahram Eivazi, Wolfgang Fuhl, and Enkelejda Kasneci. Towards intelligent surgical microscopes: Surgeons gaze and instrument tracking. In Proceedings of the 22st International Conference on Intelligent User Interfaces, IUI 2017. ACM, 03 2017.

[EHF+17] S. Eivazi, A. Hafez, W. Fuhl, H. Afkari, E. Kasneci, M. Lehecka, and R. Bednarik. Optimal eye movement strategies: a comparison of neurosurgeons gaze patterns when using a surgical microscope. Acta Neurochirurgica, 2017.

[ESF+17] Shahram Eivazi, Michael Slupina, Wolfgang Fuhl, Hoorieh Afkari, Ahmad Hafez, and Enkelejda Kasneci. Towards automatic skill evaluation in microsurgery. In Proceedings of the 22st International Conference on Intelligent User Interfaces, IUI 2017. ACM, 03 2017.

[FBH+19] Wolfgang Fuhl, Efe Bozkir, Benedikt Hosp, Nora Castner, David Geisler, Thiago C., and Enkelejda Kasneci. Encodji: Encoding gaze data into emoji space for an amusing scanpath classification approach ;). In Eye Tracking Research and Applications, 2019.

[FBK20] Wolfgang Fuhl, Efe Bozkir, and Enkelejda Kasneci. Reinforcement learning for the privacy preservation and manipulation of eye tracking data. arXiv preprint arXiv:2002.06806, 08 2020.

[FCK18a] W. Fuhl, N. Castner, and E. Kasneci. Histogram of oriented velocities for eye movement detection. In International Conference on Multimodal Interaction Workshops, ICMIW, 2018.

[FCK18b] W. Fuhl, N. Castner, and E. Kasneci. Rule based learning for eye movement type detection. In International Conference on Multimodal Interaction Workshops, ICMIW, 2018.

[FCK+19] W. Fuhl, N. Castner, T. C. Kübler, A. Lotz, W. Rosenstiel, and E. Kasneci. Ferns for area of interest free scanpath classification. In Proceedings of the 2019 ACM Symposium on Eye Tracking Research & Applications (ETRA), 06 2019.

[FCZ+18] W. Fuhl, N. Castner, L. Zhuang, M. Holzer, W. Rosenstiel, and E. Kasneci. Mam: Transfer learning for fully automatic video annotation and specialized detector creation. In International Conference on Computer Vision Workshops, ICCVW, 2018.

[FEH+18] W. Fuhl, S. Eivazi, B. Hosp, A. Eivazi, W. Rosenstiel, and E. Kasneci. Bore: Boosted-oriented edge optimization for robust, real time remote pupil center detection. In Eye Tracking Research and Applications, ETRA, 2018.

[FGK20a] W. Fuhl, H. Gao, and E. Kasneci. Neural networks for optical vector and

eye ball parameter estimation. In ACM Symposium on Eye Tracking Research & Applications, ETRA 2020. ACM, 01 2020.

[FGK20b] W. Fuhl, H. Gao, and E. Kasneci. Tiny convolution, decision tree, and binary neuronal networks for robust and real time pupil outline estimation. In ACM Symposium on Eye Tracking Research & Applications, ETRA 2020. ACM, 01 2020.

[FGRK19] W. Fuhl, D. Geisler, W. Rosenstiel, and E. Kasneci. The applicability of cycle gans for pupil and eyelid segmentation, data generation and image refinement. In International Conference on Computer Vision Workshops, ICCVW, 11 2019.

[FGS+18] W. Fuhl, D. Geisler, T. Santini, T. Appel, W. Rosenstiel, and E. Kasneci. Cbf:circular binary features for robust and real-time pupil center detection. In ACM Symposium on Eye Tracking Research & Applications, 06 2018.

[Fie20] Webseite oder Webanwendung? `https://byteq.com/de/b/webseite-webanwendung-unterschied`, 2020. Accessed: 14.11.2020.

[FK18] W. Fuhl and E. Kasneci. Eye movement velocity and gaze data generator for evaluation, robustness testing and assess of eye tracking software and visualization tools. In Poster at Egocentric Perception, Interaction and Computing, EPIC, 2018.

[FK19] W. Fuhl and E. Kasneci. Learning to validate the quality of detected landmarks. In International Conference on Machine Vision, ICMV, 11 2019.

[FK20a] Wolfgang Fuhl and Enkelejda Kasneci. Multi layer neural networks as replacement for pooling operations. arXiv preprint arXiv:2006.06969, 08 2020.

[FK20b] Wolfgang Fuhl and Enkelejda Kasneci. Rotated ring, radial and depth wise separable radial convolutions. arXiv, 08 2020.

[FK20c] Wolfgang Fuhl and Enkelejda Kasneci. Weight and gradient centralization in deep neural networks. arXiv, 08 2020.

[FKB+18] W. Fuhl, T. C. Kübler, H. Brinkmann, R. Rosenberg, W. Rosenstiel, and E. Kasneci. Region of interest generation algorithms for eye tracking data. In Third Workshop on Eye Tracking and Visualization (ETVIS), in conjunction with ACM ETRA, 06 2018.

[FKH+17] W. Fuhl, T. C. Kübler, D. Hospach, O. Bringmann, W. Rosenstiel, and E. Kasneci. Ways of improving the precision of eye tracking data: Controlling the influence of dirt and dust on pupil detection. Journal of Eye Movement Research, 10(3), 05 2017.

[FKRK20] W. Fuhl, G. Kasneci, W. Rosenstiel, and E. Kasneci. Training decision trees as replacement for convolution layers. In Conference on Artificial Intelligence, AAAI, 02 2020.

[FKS+15] W. Fuhl, T. C. Kübler, K. Sippel, W. Rosenstiel, and E. Kasneci. Arbitrarily shaped areas of interest based on gaze density gradient. In European Conference on Eye Movements, ECEM 2015, 08 2015.

[FKSK18] W. Fuhl, T. Kübler, T. Santini, and E. Kasneci. Automatic generation of saliency-based areas of interest. In Symposium on Vision, Modeling and Visualization (VMV), 09 2018.

[FRE20] Wolfgang Fuhl, Yao Rong, and Kasneci Enkelejda. Fully convolutional neural networks for raw eye tracking data segmentation, generation, and reconstruction. In Proceedings of the International Conference on Pattern Recognition, pages 0–0, 2020.

[FRK19] W. Fuhl, W. Rosenstiel, and E. Kasneci. 500,000 images closer to eyelid and pupil segmentation. In Computer Analysis of Images and Patterns, CAIP, 11 2019.

[FRM+20] Wolfgang Fuhl, Yao Rong, Thomas Motz, Michael Scheidt, Andreas Hartel, Andreas Koch, and Enkelejda Kasneci. Explainable online validation of machine learning models for practical applications. In Proceedings of the International Conference on Pattern Recognition, pages 0–0, 2020.

[FSG+16] W. Fuhl, T. Santini, D. Geisler, T. C. Kübler, W. Rosenstiel, and E. Kasneci. Eyes wide open? eyelid location and eye aperture estimation for pervasive eye tracking in real-world scenarios. In ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct publication – PETMEI 2016, 09 2016.

[FSG+17] W. Fuhl, T. Santini, D. Geisler, T. C. Kübler, and E. Kasneci. Eyelad: Remote eye tracking image labeling tool. In 12th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2017), 02 2017.

[FSK17a] W. Fuhl, T. Santini, and E. Kasneci. Fast and robust eyelid outline and aperture detection in real-world scenarios. In IEEE Winter Conference on Applications of Computer Vision (WACV 2017), 03 2017.

[FSK17b] W. Fuhl, T. Santini, and E. Kasneci. Fast camera focus estimation for gaze-based focus control. In CoRR, 2017.

[FSK+18] W. Fuhl, T. Santini, T. Kuebler, N. Castner, W. Rosenstiel, and E. Kasneci. Eye movement simulation and detector creation to reduce laborious parameter adjustments. arXiv preprint arXiv:1804.00970, 2018.

[FSR+16] W. Fuhl, T. Santini, C. Reichert, D. Claus, A. Herkommer, H. Bahmani, K. Rifai, S. Wahl, and E. Kasneci. Non-intrusive practitioner pupil detection for unmodified microscope oculars. Elsevier Computers in Biology and Medicine, 79:36–44, 12 2016.

[FST88] Schkolnick Finkelstein, Mario Schkolnick, and Paolo Tiberio. Physical database design for relational databases. ACM Transactions on Database Systems (TODS), 13(1):91–128, 1988.

[Fuh19] W. Fuhl. Image-based extraction of eye features for robust eye tracking. PhD thesis, University of Tübingen, 04 2019.

[Fuh20] Wolfgang Fuhl. From perception to action using observed actions to learn gestures. User Modeling and User-Adapted Interaction, pages 1–18, 08 2020.

[GFSK17] D. Geisler, W. Fuhl, T. Santini, and E. Kasneci. Saliency sandbox: Bottom-up saliency framework. In 12th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2017), 02 2017.

[Gib16] Robert Gibb. What is a Web Application? https://blog.stackpath.com/web-application/, May 2016. Accessed: 14.11.2020.

[Joi20] SQL Joins. https://www.w3schools.com/sql/sql_join.asp, 2020. Accessed: 15.11.2020.

[Kle06] Stephan Kleuker. Grundkurs Datenbankentwicklung - Von der Anforderungsanalyse zur komplexen Datenbankanfrage. Springer, 2006.

[Kli09] Eine Studie - was ist das? Grundlagen klinischer Studien. 2009.

[Kö00] Manuela König. Client-Server-Technologie mit PHP. https://www.akademie.de/de/wissen/php-lernen/client-server-technologie, 2000. Accessed: 19.10.2020.

[Len97] Richard Lenz. Datenverwaltung aus Anwendungssicht. In Adaptive Datenreplikation in verteilten Systemen, pages 96–147. Springer, 1997.

[Log16] Logical Design. http://www.myreadingroom.co.in/notes-and-studymaterial/65-dbms/508-logical-design.html, 2016. Accessed: 17.10.2020.

[Mar20] Database Design Phase 2: Conceptual Design. https://mariadb.com/kb/en/database-design-phase-2-conceptual-design/, 2020. Accessed: 17.10.2020.

[Php18] PHP | Introduction. https://www.geeksforgeeks.org/php-introduction/, 2018. Accessed: 18.10.2020.

[Php20] PHP: Was ist PHP? https://www.php.net/manual/de/intro-whatis.php, 2020. Accessed: 18.10.2020.

[Phy20] Physischer Datenbankentwurf. https://www.datenbanken-verstehen.de/datenbankentwicklung/datenbank-erstellen/physischer-datenbankentwurf/, 2020. Accessed: 17.10.2020.

[Rei10] Holger Reibold. XAMPP 1.8 kompakt. Holger Reibold, 2010.

[Rei13] Ulf-Dietrich Reips. Internet-Based Studies. In Marc D. Gellman and J. Rick Turner, editors, Encyclopedia of Behavioral Medicine, pages 1097–1102. Springer New York, New York, NY, 2013.

[Ren13] Christian Rengstl. Clustering genetischer Daten auf der Basis eines konsistenten Softwareframeworks zur Datenverwaltung in klinischen Studien. PhD thesis, 2013.

[Sch17] Edwin Schicker. Datenbanken und SQL - Eine praxisorientierte Einführung mit Anwendungen in Oracle, SQL Server und MySQL. Springer, 2017.

[Sql19] SQL | DDL, DQL, DML, DCL and TCL Commands. https://www.geeksforgeeks.org/sql-ddl-dql-dml-dcl-tcl-commands/, 2019. Accessed: 15.10.2020.

[Sto] MySQL Tabellen-Optimierung: Der optimale Datentyp. https://sirmark.de/computer/mysql/mysql-tabellen-optimierung-der-optimale-datentyp-1298.html, mar.

[Sto20] Data Type Storage Requirements. https://dev.mysql.com/doc/refman/8.0/en/storage-requirements.html, 2020. Accessed: 18.11.2020.

[The06] Thomas Theis. Einstieg in PHP 5 und MySQL 5. Galileo Press, 2006.

[Tho19] Ruth Thomas. The benefits of using web-based applications. https://www.geeks.ltd.uk/about-us/blog/details/eQU5Ip/the-benefits-of-using-web-based-applications, January 2019. Accessed: 14.11.2020.

[UM12] Michael Unterstein and Günter Matthiessen. Relationale Datenbanken und SQL in Theorie und Praxis. Springer-Verlag, 2012.

[Web07] The Benefits of Web Based Applications and Systems. http://www.dbnetsolutions.co.uk/articles/BenefitsOfWebBasedApplications.aspx, 2007. Accessed: 14.11.2020.

[Web20a] Webapplikationen und Webanwendungen. https://oneclick-cloud.com/de/blog/trends/webapplikationen-webanwendungen/, May 2020. Accessed: 14.11.2020.

[Web20b] What Is a Web Application? How a Web Application Works, Benefits and Examples. https://www.indeed.com/career-advice/career-development/what-is-web-application, February 2020. Accessed: 14.11.2020.

[WG14] Er Saurabh Walia and Er Satinderjit Kaur Gill. A framework for web based student record management system using PHP. International Journal of Computer Science and Mobile Computing, 3(8):24–33, 2014.

[Zah18] Ira Zahorsky. Was ist eine Studie? https://www.it-business.de/was-ist-eine-studie-a-730732/, May 2018. Accessed: 14.11.2020.