|  |  |
| --- | --- |
| Dokumentart | **Dokumentation** |
|
| Titel | **IPA Bericht** |
| Produkt | **Inserator** |
| Projekt | **Probe IPA** |

|  |  |  |  |
| --- | --- | --- | --- |
| Erstellt | Alain Keller | MO CDC VT | 10.11.2017 |
|  |  | Org.-Einheit | Datum |

Inhaltsverzeichnis

[Teil 1: Umfeld & Ablauf 4](#_Toc498096531)

[1 Aufgabenstellung 4](#_Toc498096532)

[1.1 Funktionen nicht angemeldeter Nutzer 4](#_Toc498096533)

[1.2 Funktionen angemeldeter Nutzer 4](#_Toc498096534)

[1.3 Funktionen Admin 4](#_Toc498096535)

[1.4 Technologien 4](#_Toc498096536)

[1.5 Wahlkriterien 5](#_Toc498096537)

[2 Projektorganisation 5](#_Toc498096538)

[2.1 Beteiligte Personen 5](#_Toc498096539)

[3 Ausgangslage 5](#_Toc498096540)

[3.1 Vorkenntnisse 5](#_Toc498096541)

[4 Mittel & Methoden 5](#_Toc498096542)

[4.1 Software 5](#_Toc498096543)

[4.2 Style Guide and Coding Conventions 5](#_Toc498096544)

[5 Firmenstandards 6](#_Toc498096545)

[5.1 Dokumentationsvorlage 6](#_Toc498096546)

[6 Zeitplan 7](#_Toc498096547)

[6.1 Meilensteine 8](#_Toc498096548)

[7 Arbeitsprotokoll 9](#_Toc498096549)

[7.1 Mittwoch, 01.11.2017 9](#_Toc498096550)

[7.2 Donnerstag, 02.11.2017 10](#_Toc498096551)

[7.3 Freitag, 03.11.2017 12](#_Toc498096552)

[7.4 Mittwoch, 08.11.2017 13](#_Toc498096553)

[7.5 Freitag, 10.11.2017 15](#_Toc498096554)

[8 Kurzzusammenfassung 16](#_Toc498096555)

[8.1 Ausgangssituation 16](#_Toc498096556)

[8.2 Projektbeschrieb 16](#_Toc498096557)

[8.3 Umsetzung 16](#_Toc498096558)

[8.4 Resultat 16](#_Toc498096559)

[Teil 2: Projekt 17](#_Toc498096560)

[9 Projektmanagement 17](#_Toc498096561)

[9.1 Versionierung 17](#_Toc498096562)

[9.2 Backupsystem 17](#_Toc498096563)

[9.3 Entwicklungsumgebungen 17](#_Toc498096564)

[10 Informieren 17](#_Toc498096565)

[10.1 Ist-Analyse 17](#_Toc498096566)

[10.2 Soll-Analyse 17](#_Toc498096567)

[10.3 Systemgrenzen 18](#_Toc498096568)

[11 Planen 19](#_Toc498096569)

[11.1 Funktionale Anforderungen 19](#_Toc498096570)

[11.2 Priorisieren 20](#_Toc498096571)

[11.3 Architektur 20](#_Toc498096572)

[11.4 Testkonzept 25](#_Toc498096573)

[11.5 Testfälle 26](#_Toc498096574)

[12 Entscheiden 28](#_Toc498096575)

[12.1 Benutzeridentifizierung 28](#_Toc498096576)

[12.2 Datenbankanbindung 28](#_Toc498096577)

[13 Realisieren 29](#_Toc498096578)

[13.1 Verzeichnisstruktur 29](#_Toc498096579)

[13.2 Klassen 30](#_Toc498096580)

[13.3 Gui Screenshots 31](#_Toc498096581)

[14 Kontrollieren 34](#_Toc498096582)

[14.1 Testübersicht 34](#_Toc498096583)

[14.2 Testfazit 34](#_Toc498096584)

[14.3 Testprotokoll 35](#_Toc498096585)

[14.4 Soll – Ist Vergleich 36](#_Toc498096586)

[15 Auswerten 37](#_Toc498096587)

[15.1 Reflexion 37](#_Toc498096588)

[15.2 Schlusswort 37](#_Toc498096589)

[Anhang 38](#_Toc498096590)

[16 Abbildungsverzeichnis 38](#_Toc498096591)

[17 Tabellenverzeichnis 39](#_Toc498096592)

[18 Literaturverzeichnis 40](#_Toc498096593)

[19 Quellenverzeichnis 40](#_Toc498096594)

[20 Glossar 40](#_Toc498096595)

[21 Source-Code 41](#_Toc498096596)

Teil 1: Umfeld & Ablauf

# Aufgabenstellung

Es soll eine Webseite erstellt werden, über welche man Inserate erstellen kann. Auf der Webseite können Benutzer Inserate für Produkte und Dienstleistungen erstellen und ansehen. Die Webseite richtet sich an Desktop und Laptop-User. Ein Responsive-Design ist nicht Pflicht.

Die Benutzer der Webseite können sich über ein Login mit Benutzernamen und Passwort anmelden. Hat ein User noch kein Login, kann er sich registrieren. Es werden folgende Daten für die Registration benötigt: Name, Vorname, Benutzername, E-Mail, Telefonnummer und das Passwort. Die Registration muss nicht über E-Mail bestätigt werden. Nur angemeldete Benutzer können Inserate erstellen. Es können auch die eigenen Inserate angesehen werden, und bei Bedarf deaktiviert werden, falls diese nicht mehr aktuell sind. Deaktivierte Inserate werden für die anderen Benutzer nicht mehr angezeigt. Das Login wird nicht gespeichert und der Benutzer muss sich bei Neustart des Browsers neu anmelden. Der Benutzer kann sich auch während dem Benutzen der Website abmelden. Zusätzlich gibt es ein Admin Account. Dieser kann alle Inserate ansehen und deaktivieren. Jedoch darf er keine Inserate erfassen.   
Auf der Mainpage sieht der Benutzer die Inserate der anderen Personen. Es werden pro Seite 10 Inserate angezeigt und über eine Seitennavigation am unteren Ende der Seite kann man zwischen den Seiten navigieren. Die Inserate werden nach dem Erstelldatum Sortiert. Weitere Filter, und Sortiermöglichkeiten sind nicht vorhanden. Öffnet man ein Inserat wird man auf eine detailliertere Seite weitergeleitet, welche weitere Informationen enthält. Ein Inserat enthält folgende Informationen: Titel, Beschreibung, Vorname Ersteller, E-Mail-Adresse Ersteller, Telefonnummer Ersteller, Ort und Art des Angebotes. Auf der Hauptübersicht sieht man den Titel, die Art und der Ort.

## Funktionen nicht angemeldeter Nutzer

Nicht anmeldete Nutzer können die Inserate der anderen Nutzer ansehen. Sie dürfen keine Inserate erfasse oder Deaktivieren.

## Funktionen angemeldeter Nutzer

Angemeldete Nutzer dürfen Inserate erfassen, ihre eigenen deaktivieren und die Inserate anderer Nutzer einsehen. Zusätzlich hat der angemeldete Nutzer die gleichen Funktionen wie der nicht angemeldete Nutzer.

## Funktionen Admin

Der Admin darf Inserate von anderen Nutzern deaktivieren und somit aus der öffentlichen Anzeige entfernen. Er darf keine Inserate erstellen.

## Technologien

Zur Entwicklung der Seite wird HTML, CSS, Javascript, JQuery & PHP verwendet. Der Server wird in PHP 7 objektorientiert implementiert. Für das Frontend wird HTML und CSS verwendet. Zur Steigerung der Benutzerfreundlichkeit werden zusätzliche Funktionen mit Javascript und JQuery verwendet. Die Datenbank wird mit MySQL entwickelt. Da die Webseite von mehreren Nutzern gleichzeitig benutzt werden kann, muss die Transaktionssicherheit gewährleistet sein.

## Wahlkriterien

225 – Versionsverwaltung

166 – Codingstyle, Lesbarer Code

167 – Coding Dokumentiert

123 – Kommentare im Quellcode

164 – Codierung: Fehlerbehandlung

125 – Gliederung des Programms

130 – Vollständiges ERM

# Projektorganisation

## Beteiligte Personen

|  |  |  |
| --- | --- | --- |
| Name | Funktion |  |
| Alain Keller | IPA-Kandidat |  |
| Remo Steinmann | Experte |  |

Tabelle 1: Beteiligte Personen

# Ausgangslage

## Vorkenntnisse

* HTML
* CSS
* Javascript
* PHP 7
* C#

# Mittel & Methoden

## Software

* Windows 10
* Word 2016
* Excel 2016
* PHP Storm 2017.1.3
* Google Chrome V 61.0.3163.100
* Mozilla Firefox

## Style Guide and Coding Conventions

# Firmenstandards

## Dokumentationsvorlage

Die verwendete Dokumentationsvorlage entspricht dem Siemens Firmenstandard und wurde mithilfe des Tools «Docugate» generiert.

# Zeitplan



## Meilensteine

|  |  |  |
| --- | --- | --- |
| **Meilenstein** | **Beschreibung** | **Datum** |
| Meilenstein 1: Start | Start des Projektes | 01.11.2017 |
| Meilenstein 2: Planung | Planung vollständig beendet, Start der Realisierung | 03.11.2017 |
| Meilenstein 3: Realisierung | Realisierung vollständig beendet, Start der Tests | 08.11.2017 |
| Meilenstein 4: Abgabe | Abgabe der IPA | 10.11.2017 |

Tabelle 2: Projektmeilensteine

# Arbeitsprotokoll

## Mittwoch, 01.11.2017

|  |  |
| --- | --- |
| **Tätigkeitsliste** | |
| **Nr./Thema** | **Beschreibung ausgeführte Tätigkeit** |
| **Teil 1 IPA Bericht** | Ich habe den ersten Teil des IPA-Berichters erstellt. Dazu gehörte der Projektauftrag, die Projektorganisation und meine Vorkenntnisse |
| **Teil 2 VCS** | Damit ich meine Dateien von Beginn an Versionisiert habe, erstellte ich heute das GIT-Repository. Dieses ist in GutHub unter folgendem Link erreichbar: <https://github.com/LordMcFungus/Probe-IPA>. |
| **Teil 2: Informieren** | Ich habe heute den Informieren-Teil von IPERKA abgeschlossen. Zum Informieren gehörte die Analyse der Ausgangslage und die Soll-Situation. |
| **Teil 2: Use Cases** | Anhand der Soll-Analyse erstellte ich die UseCase-diagramme für das Projekt. Diese erstellte ich mit Hilfe von PlantUml, einem Tool, um Textbasierte UMLS zu erstellen. Ein Vorteil dieses Tools ist es, dass man sich nicht mit dem Design auseinandersetzten muss. Dafür sollte man den Syntax kennen um Diagramme zu schreiben |
| **Teil 2: Komponenten** | Ich startete mit der Architektur. Dafür machte ich mir Gedanken welche Komponenten im Programm existieren werden, und was für Verantwortlichkeiten diese haben sollen. Ich habe schon damit begonnen |
| **Journal** | |
| **Erreichte & unerreichte Ziele** | * Informieren abgeschlossen ✔ * UseCase abgeschlossen ✔ * Architektur angefangen ✔ |
| **Aufgetretene Probleme & erarbeitete Lösungen** | Mir ist noch nicht ganz klar wie das Komponentendiagramm aussehen soll. Im Internet werden verschiedene Meinungen vertreten |
| **Reflexion** | Der Heutige Tag verlief recht gut. Ich konnte mich gut an den Zeitplan halten und konnte ohne grössere Probleme das Projekt starten. Es ist teils schwer in einem Raum mit 6 anderen Lernenden zu arbeite, ohne dass man teils abgelenkt ist. Jedoch war es oft still und ich konnte mich gut konzentrieren. |

Tabelle 3: Arbeitsprotokoll 01.11.2017

## Donnerstag, 02.11.2017

|  |  |
| --- | --- |
| **Tätigkeitsliste** | |
| **Nr./Thema** | **Beschreibung ausgeführte Tätigkeit** |
| **Teil 2: Komponenten** | Heute habe ich das Komponentendiagram fertiggestellt. Ich habe nun einen groben Plan, wie das Frontend mit dem Backend kommunizieren wird und wie ich die Daten von der Datenbank bekomme. Zusätzlich habe ich eine kurze Beschreibung erstellt, die das Komponentendiagram näher erläutert |
| **Teil 2: Anforderungen** | Bis jetzt hatte ich die Anforderungen noch nicht wirklich schön gegliedert aufgeschrieben. Dies habe ich heute gemacht und kann nun meine Informationen schneller holen und habe auch allenfalls auftretende Problemquellen gefunden. |
| **Teil 2: ERM** | Als nächsten Task habe ich das ERM erstellt. Damit ich dieses Fachgerecht erstellen kann, und das ERD direkt in Code exportieren kann, installierte ich mir noch die MySql Workbench. Die Grundstruktur des ERDs hatte ich relativ schnell, doch es gab immer wieder noch kleine Verfeinerungen. Anschliessend habe ich die Beziehungen und Tabellen in der Dokumentation näher beschrieben. |
| **Teil 2: Testcases** | Anschliessend an das ERM erstellte ich die Testcases. Als Grundlage dafür nahm ich die Use-Cases, welche ich gestern erstellt haben. Aufgrund der in denen beschrieben Funktionalitäten erstellte ich Testgruppen in denen ich jeweils mehrere Postiv-, sowie auch Negativtestfälle beschreibe. |
| **Expertenbesuch** | Heute fand der erste Expertenbesuch statt. Ich sollte meine Reflexion ausführlicher machen. Ansonsten ist es gut. |
| **Teil 2:Prioriseren** | Gegen Ende des Tages begann ich die Tasks für die Realisierung zu priorisieren. Damit soll ich sicherstellen, dass ich nicht bei einem unwichtigen Task zu lange hängenbleibe und somit die Hauptfunktionalitäten nicht implementieren kann. |
| **Journal** | |
| **Erreichte und Unerreichte Ziele** | * Architekturteil der Planung abschliessen ✔ * Anforderungen detailliert und verständlich dokumentieren ✔ * ERM erstellen und abschliessen ✔ * Testcases Erstellen und abschliessen ✖ * Priorisierung abschliessen ✔ |
| **Aufgetretene Probleme & erarbeitete Lösungen** | Die meisten Probleme sind heute beim Erstellen des ERDs und der Aufsetzung der Datenbank aufgetreten. Zuerst konnte ich die Workbench nicht installieren, weil ich angeblich den Visual C++ Treiber von 2015 nicht installiert habe, ich hatte denjenigen von 2017. Den erforderlichen Treiber konnte ich dann auch nicht installieren, weil ich schon eine neuere Version haben. Nachdem ich meinen aktuellen Treiber deinstallierte, konnte ich den alten installieren und die Workbench erfolgreich aufsetzen. Weiter Probleme sind bei den CREATE-Scripts von MySql aufgetaucht. Immer wieder hatte ich Fehler in der generierten Abfrage. Die meisten konnte ich dann aber damit beheben, dass ich die bestehende Datenbank löschte und neu aufsetzte, anstatt die bestehende zu überschrieben. |
| **Reflexion** | Heute war ein strenger Tag. Vor allem die Arbeit mit MySql war mühsam. Jedoch bin ich auch mehr selber schuld, da ich die nötige Software nicht im Voraus installiert habe sondern erst heute. Das nächste Mal werde ich da sicher besser aufpassen. Die Arbeit im Sitzungszimmer verlief auch mehr oder weniger ok. Es gab wieder Situationen, in denen es schwer war sich zu konzentrieren, doch mehrheitlich war es gut.  Ich bin mit meinen tageszielen nicht ganz durchgekommen. Ich hatte aber auch einige unproduktive Minuten, die ich sicherlich sinnvoller hätte einsetzten können. Ich hoffe, dass ich am Freitagmorgen schnell mit meinen Pendenzen durch bin und mit der Implementierung beginnen kann. |

Tabelle 4: Arbeitsprotokoll 2.11.2017

## Freitag, 03.11.2017

|  |  |
| --- | --- |
| **Tätigkeitsliste** | |
| **Nr./Thema** | **Beschreibung ausgeführte Tätigkeit** |
| **Teil 2 Testcases** | Da ich gestern noch nicht fertig wurde mit den Testcases, stellte ich diese heute Morgen noch fertig. |
| **Teil 2: Entscheidungen** | Heute Morgen habe ich den Entscheiden-Teil von IPERKA gemacht. Die Entscheidungen bezogen sich hauptsächlich auf lösungsvarianten von Problemen wie die Benutzerauthentifizierung. |
| **Teil 2 HTML und CSS** | Nach dem Entscheiden konnte ich endlich mit der Realisierung beginnen. Zuerst habe ich das Grundgerüst im HTML gemacht. Da ich aber die meisten Daten dynamisch lade, sieht die Seite momentan noch leer aus. Zusätzlich habe ich noch die verschiedenen Formulare zu Regsitrieren und Einloggen implementiert. |
| **Teil 2 Registrierung** | Nach dem Design begann ich mit der Implementation der registrierung. Für diese brauchte ich jedoch länger als geplant, da ich auf einige Probleme gestossen bin. Ich habe es dennoch fertig geschafft, wenn auch mit Verzug |
| **Teil 2 Login** | Mit dem Login konnte ich heute nur knapp anfangen. Geplant war aber, dass ich fertig werde. |
|  |  |
| **Journal** | |
| **Erreichte und nicht erreichte Ziele** | * Testcases ✔ * Entscheidungen ✔ * HTML & CSS ✔ * Registrierung ✔ * Login ✖ * Errorhandling ✖ |
| **Probleme** | Die meisten Probleme hatte ich heute während der Implementation. Zuerst funktionierte Ajax nicht so wie ich wollte. Doch nach ein Bisschen Googeln und Pascal fragen hat es dann funktioniert. Ajax ist für mich eine neue Technologie. Mein fehlendes Fachwissen ist wahrscheinlich auch der Hauptgrund für meinen Verzug. Danach hatte ich noch ein paar kleine Logikprobleme bei der Implementation der Registrierung. Doch auch diese konnte ich beheben und die Registrierung funktioniert jetzt. Jedoch habe ich einen Teil noch nicht erledigt, das Errorhandling. Server intern funktioniert sie zwar, der Benutzer bekommt aber noch kein brauchbares Feedback |
| **Reflexion** | Die grösste Lektion von Heute war wohl, dass ich zu wenig Erfahrung habe in der Webentwicklung. Ich habe viel Zeit durch den Kampf mit den Technologien verloren, die ich sinnvoller hätte nutzen können. Dafür habe ich heute viel neues gelernt, was ich vorher nicht wusste. Durch die vielen Verzögerungen in der Implementation konnte ich auch meinen Zeitplan nicht ganz einhalten. Da werde ich am nächsten Mittwoch sicher noch viel zu tun haben. |

Tabelle 5: Arbeitsprotokoll 23.03.2017

## Mittwoch, 08.11.2017

|  |  |
| --- | --- |
| **Tätigkeitsliste** | |
| **Nr./Thema** | **Beschreibung ausgeführte Tätigkeit** |
| **Login** | Heute habe ich das Login fertig erstellt. Da ich am Vortag nicht den Zeitplan einhalten konnte, musste ich dies entgegen des Zeitplanes heute erledigen, |
| **Inputvalidierung** | Nach der Implementation des Logins versuchte ich die Inputvalidierung der Registrationen des Logins zu verbessern. Ich versuchte eine Überprüfung einzubauen, damit ein Username nicht zweimal erstellt werden kann. |
| **Inserat Erfassen** | Danach habe ich die Logik für das Erfassen eines Inserates implementiert. Dabei konnte ich viel von der Registrierung übernehmen |
| **Inserat anzeigen** | Nach dem Erfassen implementierte ich die Anzeige des Inserates. Dies verlief recht gut. Einzig hatte ich Probleme mit der Typisierung von Javascript. |
| **Errorhandling** | Ich hätte noch geplant, dass ich das Errorhandling verbessere und ausbaue, jedoch fehlte mir am Schluss die Zeit und ich musste diesen Task auslassen. |
| **Expertengespräch** | Im Verlauf des Tages hatte ich das Expertengespräch mit Remo. Er hat mir noch ein paar Hinweise gegeben, wo ich meine Dokumentation noch verbessern könnte. |
| **Journal** | |
| **Erreichte & unerreichte Ziele** | * Login✔ * Inputvalidierung✖ * Inserat Erfassen✔ * Inserat anzeigen✔ * Errorhandling✖ * Dokumentation Realisierung✔ |
| **Aufgetretene Probleme und Lösungen** | Am meisten Probleme hatte ich heute mit PDO. Ich versuchte aus einer Abfrage die Anzahl gefundener Datensätze auszulesen, ohne dass ich durch jeden durchiterieren muss. Jedoch habe ich dies nicht wirklich geschafft. Da ich das brauchte, um zu überprüfen ob ein Username schon erfasst ist, habe ich nach einer neuen Lösung gesucht. Schlussendlich habe ich das Feld in der Datenbank auf Unique geschaltet und den Fehler über die Insert Abfrage abgefangen. Des Weiteren hatte ich Probleme mit der Typisierung von Javascript. Da man einer Variable keinen bestimmten Datentyp zuweisen kann, gab das Resultat einer einfachen 1+1 Rechnung 11 statt 2. Diesen Fehler konnte ich mit einem einfachen castInt() beheben. |
| **Reflexion** | Heute war ein strenger Tag. Ich konnte leider nicht alle Tasks erfüllen, die ich mir vorgenommen habe. Ich musste die Input-Validierungsmeldungen auf das Frontend kürzen und kann keine Usermeldungen vom Server aus schicken. Ich habe diese Anforderungen absichtlich nicht implementiert, da es mir wichtiger war die Grund Funktionalität des Programmes zu implementieren. Die Realisierung ist jedoch noch nicht abgeschlossen, da ich noch nicht dokumentiert habe. Ich werde am kommenden Freitag zusätzlich noch die Realisierungsdoku schreiben müssen. |

Tabelle 6: Arbeitsprotokoll 24.03.2017

## Freitag, 10.11.2017

|  |  |  |
| --- | --- | --- |
| **Tätigkeitsliste** | | |
| **Nr./Thema** | | **Beschreibung ausgeführte Tätigkeit** |
| **Realisierung Dokumentieren** | | Heute Morgen habe ich die Realisierung Dokumentiert. Dabei ging ich auf die Dateistruktur und die einzelnen Klassen ein. |
| **Testen** | | Nach der Realisierung habe ich gemäss meinen Testfällen die Applikation getestet. Die Implementierten Funktionen haben grössten Teils die Tests bestanden. Einzig die fehlenden haben logischerweise nicht bestanden |
| **Kontrollieren** | | Zusätzlich zum Testen erstellte ich einen Soll – Ist vergleich, in welchem ich mich auf die vergangen Soll-Analyse beziehe. |
| **Auswerten** | | In der Auswertung stellte ich die Reflexion und das Fazit des Projektes |
| **Dokumentanhang erstellen** | | Für die Projektabgabe habe ich den Anhang mit meinem geschriebenen Code befüllt. |
| **Journal** | | |
| **Erreicht & Unerreichte Ziele** | * Realisierung dokumentieren✖ * Testen✔ * Kontrollieren✔ * Auswerten✔ * Dokumentanhang erstellen✔ * Dokument nach Kriterien überprüfen✖ | |
| **Aufgetretene Probleme und Lösungen** | Das heutige Hauptproblem war Word. Nachdem ich den ganzen Morgen an der Dokumentation gearbeitet habe, und dieses jeweils zwischenspeicherte, schloss ich mein Word. Als ich die Doku wieder öffnete war die ganze Arbeit vom Morgen verschwunden. Ich versuchte die alte Version zu wiederherstellen doch es funktionierte nicht. Ich habe danach Remo gefragt, ob ich am Samstag dem 11.11 meine finale Version abgeben kann. Ich werde nun heute eine Version abgeben und die überarbeitete Version am Samstag bis um 18:00 Uhr abgeben. Sonst hatte ich heute keine weiteren Probleme. | |
| **Reflexion** | Bis am Mittag lief der heutige Tag eigentlich recht gut. Doch nachdem mir Word einen Strich durch die Rechnung machte, wurde ich unruhig. Ich konnte jedoch die verlorenen Teile nochmals schreiben, wenn auch nicht ganz so detailliert. Ich habe vor allem die Dokumentation der Realisierung nicht so genau gemacht wie am Morgen. Da ich am Nachmittag die Auswertung und die Kontrolle eingeplant hatte, legte ich vor allem auf diese beiden Teile mehr Gewicht | |

Tabelle 7: Arbeitsprotokoll 28.03.2017

# Kurzzusammenfassung

## Ausgangssituation

Für das Modul 223 wurde ein Kleinprojekt gesucht, welches Objektorientiert implementiert wird und über eine Datenbank verfügt. Das Projekt sollte vom Kandidaten innerhalb von 5 Tagen realisierbar sein, wobei zu beachten war, dass von diesen 5 Tagen mindestens 2 Tage dokumentiert wird.

## Projektbeschrieb

Das Projekt ist eine Webseite, auf der man Inserate erfassen kann. Diese können von anderen Benutzern angesehen werden. Der Benutzer muss angemeldet sein um ein Inserat zu erfassen und seine eigenen Inserate zu löschen. Für das Projekt werden folgende Techniken verwendet: HTML, CSS, JavaScript, PHP und MySql. Ausserdem wird die Javascriptlibrary JQuery verwendet, damit über Ajax mit dem Server kommuniziert werden kann.

## Umsetzung

Das aufgetragene Kleinprojekt wurde mit der Projektmanagementmethode IPERKA realisiert. Innerhalb von 5 Tagen wurde das Projekt abgewickelt und abgeschlossen.

## Resultat

Das Projekt konnte in dieser Zeit nicht komplett fertiggestellt werden. Die aktuelle Version ist jedoch lauffähig.

Teil 2: Projekt

# Projektmanagement

Als Projektmanagementmethode verwende ich in diesem Projekt IPERKA. Es ist einfach aufgebaut und eignet sich gut für Einzel Projekte über eine kurze Laufzeit

## Versionierung

Als VCS verwende ich Git über die Server von Github. Ich habe mich für Git entschieden, da ich in vergangenen Schulprojekten schon Git über GitHub verwendet habe. Pro Tag werde ich mindestens einen Commit ausführen. Zusätzlich werde ich mindestens nach jedem Implementierten Feature, oder bevor ich meinen Laptop für mehr als eine viertel Stunde verlasse einen Commit durchführen.

## Backupsystem

Neben dem regelmässigen speichern auf GitHub wird am Ende eines jeden Tages eine Kopie auf Dropbox gespeichert. Dies stellt die Verfügbarkeit der Arbeit sicher, falls der Laptop und GitHub gleichzeitig nicht erreicht werden können.

## Entwicklungsumgebungen

Den Code entwickle ich in PHP-Storm. Es ist eine IntelliJ basierte IDE, welche ich auch schon von vergangenen Projekten kenne. Den Apache und MySql-Server verwalte ich über Xampp, da sich dieses gut für solche Projekte eignet und gratis verfügbar ist.

# Informieren

In diesem Teil trage ich die Informationen zusammen und beschreibe diese.

## Ist-Analyse

Zurzeit besteht noch nichts von der gewünschten Applikation.

## Soll-Analyse

Es soll eine WebApp erstellt werden, über welche Benutzer Inserate erstellen und ansehen können.

### Frontend

Das Frontend wird mit Hilfe von HTML, CSS und Javascript erstellt.

Der Benutzer sieht auf der Startseite die schon erstellten Inserate der anderen Benutzer. Diese werden in mehrere Seiten aufgeteilt. Je Seite werden 10 Inserate angezeigt. Auf der Inserat-Liste werden nicht alle Informationen angezeigt. Wählt der Nutzer ein Inserat aus, wird dieses Geöffnet und man sieht mehr Details. Der Benutzer kann sich auf der Webseite regsitrieren und Anmelden. Als angemeldeter Nutzer kann er ein eigenes Inserat erstellen, und die erstellten Inserate deaktivieren.

### Backend

Das Backend ist vor allem dazu da, die Schnitstelle zwischen Frontend und Datenbank zu machen. Die Logindaten werden übers Backend validiert, die Inserate werden ausgelesen und gespeichert. Das Backend soll objektorientiert mit Hilfe von PHP 7.1 geschrieben werden.

### Datenbank

Die Datenbank wird mit MySql gemacht. Die Transaktionssicherheit muss jederzeit gewährleistet sein. In der Datenbank werden die Benutzerdaten und die Inserate gespeichert.

### Design und Layout

Es stehen keine konkreten Anforderungen zum Design. Die Webseite muss jedoch nicht Responsive implementiert sein.

### Errorhandling

Fehler im Programm sollten erkannt werden und entsprechend damit umgegangen werden.

### Source Code

Der Source-Code soll verständlich sein und kommentiert. Es sollen die Clean-Code Guidelines eingehalten werden.

### Tests

Die Applikation wird nach der Implementation ausführlich gegen die Anforderungen getestet. Dabei wird auf Blackboxtesting gesetzt.

## Systemgrenzen

Die Applikation ist ein in sich geschlossenes System, welches in einem Browser angezeigt wird. Alles was ausserhalb dieser Applikation ist, gehört nicht zum System.

# Planen

## Funktionale Anforderungen

Die Soll-Analyse wird nun nochmals verfeinert und in funktionale- und nichtfunktionale Anforderungen gegliedert.

|  |  |
| --- | --- |
| Name | Beschreibung |
| Registrieren | Der Benutzer kann sich auf der Seite Registrieren. Für die Registration muss der Benutzer folgende Daten angeben:   * Name * Vorname * Benutzername * E-Mail * Telefonnummer * Passwort   Es gibt keine Bestätigungsemail bei der Registration  Bei Falscheingaben bekommt der Benutzer sinnvolle Rückmeldungen. |
| Login | Ein registrierter Nutzer kann sich mit seinen Daten einloggen. Für das Login braucht er seinen Benutzernamen und das Passwort.  Bei Falscheingaben bekommt der Benutzer sinnvolle Rückmeldungen. |
| Logout | Ein eingeloggter Nutzer kann sich ausloggen. Seine aktuelle Sitzung wird somit beendet und er hat nur noch Zugriff auf die Funktionen eines nicht angemeldeten Nutzer. |
| Inserat erstellen | Ein angemeldeter Nutzer kann ein Inserat erstellen. Dazu muss in einem Inserat folgende Daten angeben:   * Titel des Inserates * Beschreibung * Vorname des Erstellers * Die E-Mail des Erstellers * Die Telefonnummer des Erstellers * Ort des Angebotes * Art des Angebotes   Hat ein Nutzer ein Inserat erstellt, wird dieses in der Übersicht auf der Mainpage angezeigt.  Bei Falscheingaben bekommt der Benutzer sinnvolle Rückmeldungen. |
| Inserat Übersicht | Auf der Startseite werden die Inserate in einer Liste angezeigt. Die Liste enthält nicht alle Informationen eines Inserates, sondern nur den Titel, die Art des Inserates und der Ort. Nach 10 Inseraten muss man über eine Navigation auf die nächste Seite gehen, um die folgenden Inserate zu sehen. Die Inserate werden nach der Erfassungszeit absteigend sortiert; die neusten sind zuoberst. |
| Inserat öffnen | Auf der Übersichtsseite kann man die Einzelnen Inserat öffnen um weitere Details zu erfahren. Die angezeigten Details entsprechen den selben Daten wie beim Inserat erstellen. |
| Inserat Löschen | Ein angemeldeter Nutzer kann seine Inserate Löschen, falls diese nicht mehr aktuell sind. Hat ein Nutzer ein Inserat gelöscht, kann es von anderen Nutzern nicht mehr gesehen werden. |
| Adminbereich | Ein als Admin gekennzeichneter User kann sich anmelden und hat die Möglichkeit alle Inserate zu löschen, auch wenn sie nicht von ihm stammen |

Tabelle 8 Funktionale Anforderung

### Berechtigungen

|  |  |  |  |
| --- | --- | --- | --- |
| Funktion | Nicht angemeldeter Nutzer | Angemeldeter Nutzer | Admin |
| Inserate Ansehen | ✔ | ✔ | ✔ |
| Inserate Öffnen | ✔ | ✔ | ✔ |
| Inserat erstellen | ✖ | ✔ | ✖ |
| Eigene Inserate Löschen | ✖ | ✔ | ✖ |
| Beliebiges Inserat Löschen | ✖ | ✖ | ✔ |

Tabelle 9 Berechtungungsmatrix

## Priorisieren

Alle Funktionalen Anforderungen haben höchste Priorität. Die Reihenfolge der Implementierung erfolgt gleich wie die Auflistung der funktionalen Anforderungen.

## Architektur

### Use Cases

|  |  |
| --- | --- |
| Use Case Nr. | 1 |
| Use Case Bezeichnung | User Registrierung |
| Akteur | Benutzer |
| Vorbedingung | Benutzer hat die Seite geöffnet und ist nicht eingeloggt |
| Use Case Beschreibung | Nutzer öffnet Registrationsseite  Nutzer gibt valide Daten ein  Nutzer wird Registriert |
| Resultat | Nutzer ist Eingeloggt auf der Startseite |

Tabelle 10: Use Case #1

|  |  |
| --- | --- |
| Use Case Nr. | 2 |
| Use Case Bezeichnung | User Login |
| Akteur | Benutzer |
| Vorbedingung | Benutzer hat die Seite geöffnet und hat schon einen Account |
| Use Case Beschreibung | Nutzer öffnet das Loginfenster  Nutzer gibt seine Daten ein  Nutzer wird Eingeloggt |
| Resultat | Nutzer ist Eingeloggt auf der Startseite |

Tabelle 11: Use Case #2

|  |  |
| --- | --- |
| Use Case Nr. | 3 |
| Use Case Bezeichnung | Logout |
| Akteur | Benutzer |
| Vorbedingung | Benutzer hat sich eingeloggt |
| Use Case Beschreibung | Nutzer loggt sich aus |
| Resultat | Der Nutzer ist abgemeldet. Er kann nicht mehr auf seine Daten zugreifen. |

Tabelle 12: Use Case #3

|  |  |
| --- | --- |
| Use Case Nr. | 4 |
| Use Case Bezeichnung | Inserat ansehen |
| Akteur | Benutzer |
| Vorbedingung | Benutzer befindet sich auf der Startseite |
| Use Case Beschreibung | Nutzer öffnet ein Inserat |
| Resultat | Es wird die Detailseite des Inserates angezeigt. |

Tabelle 13: Use Case #4

|  |  |
| --- | --- |
| Use Case Nr. | 5 |
| Use Case Bezeichnung | Inserat erstellen |
| Akteur | Benutzer |
| Vorbedingung | Benutzer hat sich eingeloggt |
| Use Case Beschreibung | Nutzer öffnet Seite zum neues Inserat erstellen  Nutzer gibt valide Daten ein  Inserat wird gespeichert |
| Resultat | Inserat ist auf der Datenbank gespeichert und kann von anderen Nutzern auf der Startseite angesehen werden. |

Tabelle 14: Use Case #5

|  |  |
| --- | --- |
| Use Case Nr. | 6 |
| Use Case Bezeichnung | Inserat Löschen |
| Akteur | Benutzer |
| Vorbedingung | Benutzer hat sich eingeloggt, Nutzer hat schon ein Inserat erstellt |
| Use Case Beschreibung | Nutzer öffnet Seite seiner Inserate  Nutzer löscht ein Inserat |
| Resultat | Das Inserat wird nicht mehr angezeigt. |

Tabelle 15: Use Case #6

|  |  |
| --- | --- |
| Use Case Nr. | 7 |
| Use Case Bezeichnung | Inserat erstellen |
| Akteur | Administrator |
| Vorbedingung | Administrator hat sich eingeloggt, Es wurden schon Inserate erstellt |
| Use Case Beschreibung | Administrator öffnet Inserateseite  Administrator Löscht beliebiges Inserat |
| Resultat | Das Inserat wird nicht mehr angezeigt. |

Tabelle 16: Use Case #7

### Use Case Diagramm

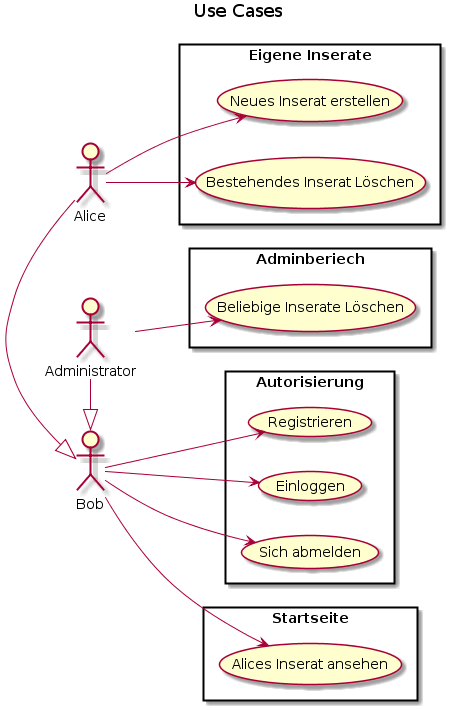


Abbildung 1 Usecase Diagramm

### Komponenten

Das Programm ist in drei Systemen aufgeteilt, welche miteinander kommunizieren müssen. Diese drei Systeme umfassen das Frontend, das Backend und die Datenbank. Um den Datenaustausch zwischen den einzelnen Dokumenten genauer zu erläutern habe ich folgendes Komponentendiagramm erstellt.

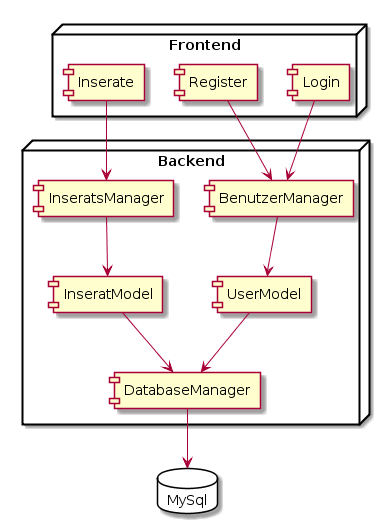


Abbildung 2 Komponentendiagramm

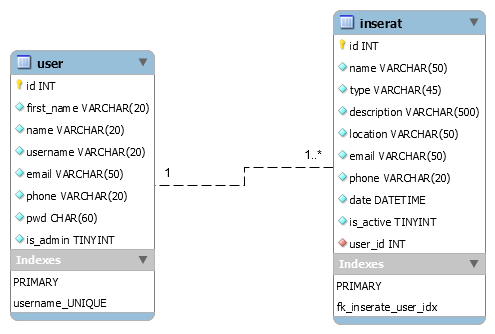
### Frontend

Das Frontend besitzt über mehrere Javascriptkomponenten, welche die Daten an den Server weiterleiten. Die Kommunikation zum Server erfolgt über das http und wird mit Ajax implementiert. Dies ermöglicht es, dass auf einer Seite verschiedene Controller auf dem Server unabhängig voneinander aufgerufen werden können. Zusätzlich kann das Frontend Serveranfragen machen, ohne dass die Seite danach neu geladen werden muss.

### Backend

Im Backend werden die Daten aufbereitet um sie entweder in der Datenbank zu speichern, oder um die Daten von der Datenbank auszulesen, um diese dann auf der Webseite anzeigen zu können. Das Backend wird in Controller und Models aufgeteilt, damit eine klare Trennung der Funktionen vorhanden ist. Die Datenbank wird über einen Datenbank Manager angesprochen. Dieser enthält die Instanz, um die Datenbank ansprechen zu können. Somit soll sichergestellt sein, dass die Datenbankanbindung nur dann steht, wenn sie auch wirklich gebraucht wird.

### Datenbank



|  |  |
| --- | --- |
| User |  |
| Beschreibung | Diese Tabelle speichert die Benutzerdaten ab. Diese entsprechen denjenigen, welche in den Anforderungen gewünscht sind. |
| Relationen | Ein Benutzer kann keine oder mehrere Inserate haben |
| Felder | |  |  |  |  | | --- | --- | --- | --- | | Feldname | Datentyp | Beschreibung |  | | Id | INT | Dieses Feld ist der Primärschlüssel der Tabelle. Es ist ein Integer, welcher automatisch Inkrementiert. | Not Null  Auto Increment  PK | | first\_name | Varchar(20) | Der Vorname des Benutzers | Not Null | | name | Varchar(20) | Der Nachname des Benutzers | Not Null | | username | Varchar(20) | Der Benutzername des Benutzers | Not Null | | email | Varchar(50) | Die E-Mail des Benutzers | Not Null | | phone | Varchar(20) | Die Telefonnummer des Benutzers | Not Null | | pwd | Char(60) | Das Passwort des Benutzers. Dieses wird von Bycrypt gehasht und abgespeichert. Daher wird auch kein zusätzliches Feld für einen Salt benötigt. | Not Null | | Is\_admin | TINYINT | Enthält den Wert, ob ein User ein Abmin ist oder ein normaler User. | Not Null | |

Tabelle 17 User Datenbank

|  |  |
| --- | --- |
| Inserat |  |
| Beschreibung |  |
| Relationen |  |
| Felder | |  |  |  |  | | --- | --- | --- | --- | | Feldname | Datentyp | Beschreibung |  | | Id | INT | Dieses Feld ist der Primärschlüssel der Tabelle. Es ist ein Integer, welcher automatisch Inkrementiert. | Not Null  Auto Increment  PK | | name | Varchar(50) | Dieses Feld enthält die Überschrift des Inserates | Not Null | | type | Varchar(45) | Diese Feild enthält den Inserattyp | Not Null | | description | Varchar(500) | Diese Feld enthält die Inseratbeschreibung | Not Null | | location | Varchar(50) | Dieses Feld enthält den Ort, wo das im Inserat beworbene Produkt verfügbar ist | Not Null | | email | Varchar(50) | Dieses Fled enthält die Kontaktemail für das Inserat | Not Null | | phone | Varchar(20) | Dieses Feld enthält die Kontakttelefonnummer des Inserates | Not Null | | date | Datetime | Diese Feld enthält das Datum an dem das Inserat erstellt wurde | Not Null | | is\_active | Tinyint | Dieses Feld enthält den Wert, ob das Inserat noch verfügbar ist oder nicht. | Not Null | | user\_id | Int | Fremdschlüssel für den User, der dieses Inserat erstellt hat. | Not Null | |

Tabelle 18 Inserat Datenbank

## Testkonzept

### Testmethoden/Testmittel

Das Programm wird mit Blackboxtesting getetstet.

### Testumgebung

Die Testumgebung setzt sich folgendermassen zusammen:

* Software:
* Windows 10 Professional
* XAMPP
* Hardware:
* Dell XPS 15

## Testfälle

### Registrierung

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Vorbedingungen | Testschritte | Erwartetes Ergebnis |
| 1.1 | Startseite ist Geöffnet und der Benutzer ist noch nicht eingeloggt | |  |  | | --- | --- | | 1. | Registrierungsknopf drücken | | Oben rechts erscheint das Registrierungsformular. Der Rest der Seite beliebt gleich |
| 1.2 | Registrierungsformular ist geöffnet, User ist noch nicht Angemeldet oder Registriert. | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 1. | Folgende Daten eingeben:   |  |  | | --- | --- | | **Name:** | Hans | | **Vornamen:** | Zimmer | | **Benutzernamen:** | Hansli | | **E-Mail:** | haz@hans.ch | | **Telefonnummer:** | 076 666 66 66 | | **Passwort:** | Schlachtlauch | | **Passwort**  **Wiederholen:** | Schlachtsellerie | | | 2. | Registrieren drücken | | Registrierung Fehlgeschlagen. Passwörter werden gelöscht und rot umrahmt |
| 1.3 | Registrierungsformular ist geöffnet, User ist noch nicht Angemeldet oder Registriert. | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 1. | Folgende Daten eingeben:   |  |  | | --- | --- | | **Name:** | Hans | | **Vornamen:** | Zimmer | | **Benutzernamen:** | Hansli | | **E-Mail:** | haz@hans.ch | | **Telefonnummer:** | 076 666 66 66 | | **Passwort:** | Schlachtlauch | | **Passwort**  **Wiederholen:** | Schlachtlauch | | | 2. | Registrieren drücken | | Registrierung ist erfolgreich Hans ist eingeloggt |
| 1.4 | Registrierungsformular ist geöffnet, User ist noch nicht Angemeldet oder Registriert. Hansli hat sich schon registriert | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 1. | Folgende Daten eingeben:   |  |  | | --- | --- | | **Name:** | Hans | | **Vornamen:** | Zimmer | | **Benutzernamen:** | Hansli | | **E-Mail:** | haz@hans.ch | | **Telefonnummer:** | 076 666 66 66 | | **Passwort:** | Schlachtzwiebel | | **Passwort**  **Wiederholen:** | Schlachtzwiebel | | | 2. | Registrieren drücken | | Registrierung ist nicht erfolgreich. Es besteht schon ein User mit demselben Namen. Das Feld Username wird rot umrahmt |

Tabelle 19 Registrierung Testfall

### Login

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Vorbedingungen | Testschritte | Erwartetes Ergebnis |
| 2.1 | Startseite ist Geöffnet | |  |  | | --- | --- | | 1. | Loginknopf drücken | | Oben rechts erscheint das Loginsformular. Der Rest der Seite beliebt gleich |
| 2.2 | Registrierungsformular ist geöffnet, Hansli ist registriert, Hansli ist noch nicht angemeldet | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | 1. | Folgende Daten eingeben:   |  |  | | --- | --- | | **Benutzername:** | Hansli | | **Passwort:** | Schlachtlauch | | | 2. | Login drücken | | Login erfolgreich, Hansli ist eingelogt |
| 2.3 | Registrierungsformular ist geöffnet, Hansli ist registriert, Hansli ist noch nicht angemeldet | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | 1. | Folgende Daten eingeben:   |  |  | | --- | --- | | **Benutzername:** | Peterli | | **Passwort:** | Schlachtlauch | | | 2. | Login drücken | | Login nicht erfolgreich, Meldung wird Angezeigt |

Tabelle 20: Login Testfall

### Logout

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Vorbedingungen | Testschritte | Erwartetes Ergebnis |
| 3.1 | Startseite ist Geöffnet | |  |  | | --- | --- | | 1. | Loginknopf drücken | | Oben rechts erscheint das Loginsformular. Der Rest der Seite beliebt gleich |

Tabelle 21: Logout Testfall

### Inserate erstellen

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Vorbedingungen | Testschritte | Erwartetes Ergebnis |
| 4.1 | Startseite ist Geöffnet, Hansli ist angemeldet | |  |  | | --- | --- | | 1. | «Inserat erstellen» Knopf drücken | | Das Inserat Erstellen Formular wird geöffnet |
| 4.2 | Registrierungsformular ist geöffnet, Hansli ist registriert, Hansli ist noch nicht angemeldet | |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 1. | Folgende Daten eingeben:   |  |  | | --- | --- | | **Titel:** | Mostmaschine | | **Art:** | Verkaufen | | **Beschreibung** | Eine Alte Moschtmaschine in gutem Zustand steht zum Verkauf. Preis ab 500 CHF. | | **Email** | most@hansli.ch | | **Telefon** | 076 666 66 66 | | **Ort** | Schönholzerswilen | | | 2. | Inserat speichern drücken | | Inserat wurde erfolgreich erstellt. Hansli wird zurück zur Hauptseite geleitet und das Inserat sollte bei den neusten angezeigt werden. |
| 4.3 | Registrierungsformular ist geöffnet, Hansli ist registriert, Hansli ist noch nicht angemeldet | |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 1. | Folgende Daten eingeben:   |  |  | | --- | --- | | **Titel:** | Mostmaschine | | **Art:** | Verkaufen | | **Beschreibung** | Eine Alte Moschtmaschine in gutem Zustand steht zum Verkauf. Preis ab 500 CHF. | | **Email** | @. | | **Telefon** | 076 666 66 66 | | **Ort** | Schönholzerswilen | | | 2. | Inserat speichern drücken | | Inserat wurde nicht abgespeichert. E-Mail Feld wird Rot umrahmt. |

Tabelle 22 Inserate erstellen Testfall

### Inserate ansehen

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Vorbedingungen | Testschritte | Erwartetes Ergebnis |
| 5.1 | Startseite ist Geöffnet, Hansli ist nicht angemeldet | |  |  | | --- | --- | | 1. | Inserat mit dem Titel Mostmaschine öffnen | | Die Detailseite des Inserates wird geöffnet |

Tabelle 23 Inserate ansehen Testfall

### Inserat Löschen

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Vorbedingungen | Testschritte | Erwartetes Ergebnis |
| 6.1 | Startseite ist Geöffnet, Hansli ist angemeldet | |  |  | | --- | --- | | 1. | «Meine Inserate» Knopf drücken | | Die Meine Inserate Seite wird angezeigt. |
| 5.2 | Meine Inserate Seite ist Geöffnet, Hansli ist angemeldet | |  |  | | --- | --- | | 1. | «Löschen» Knopf neben dem Inserat Mostmaschine drücken | | Inserat Mostmaschine wird gelöscht |

Tabelle 24 Inserat Löschen Testfall

# Entscheiden

## Benutzeridentifizierung

Für die Benutzerautorisierung bestehen zwei Möglichkeiten. Die eine erfolgt über Cookies und die andere über die PHP-Session. Ich implementiere die Authentifizierung über die Session, da ich dies schon gemacht habe und weniger aufwändig ist zu implementieren.

## Datenbankanbindung

Die Datenbankanbindung erfolgt über PDO. Ich entschied mich für PDO weil es unabhängig vom benutzten DBMS benutzt werden kann

# Realisieren

## Verzeichnisstruktur

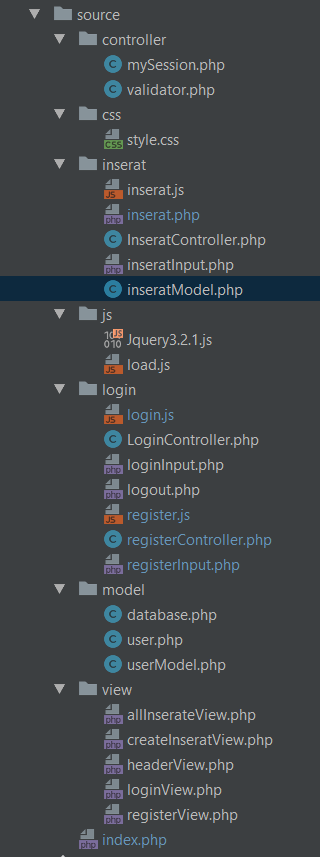
Der Source-code wird in verschiedene Ordner unterteilt. Diese Ordner richten sich jeweils nach den Funktionalitäten der darin enthaltenen Klassen.

Abbildung 3 Ordnerstruktur

|  |  |
| --- | --- |
| Verzeichnis | Beschreibung |
| Controller | Enthält generelle Controllerklassen wie den mySession Singelton |
| Css | Enthält die Stylesheetdateien |
| Inserat | Enthält alle Klassen, welche für die Aufbereitung und Speicherung von Inseraten zu tun haben |
| Js | Enthält allgemeine Javascripts und Libraries |
| Login | Enthält alle Klassen, welche benötigt werden um den User zu registrieren, Anzumelden und abzumelden. |
| Model | Enthält die Dateien, welche direkt mit der Datenbank zu tun haben. |
| View | Enthält die verschiedenen View-Dateien, welche im Frontend angezeigt werden |
| Source | Enthält die index.php Datei, sowie die Unterordner, welche oben erwähnt wurden. |

## Klassen

### mySession

MySession ist ein Singelton, welcher die Session verwaltet. Er ist dafür verantwortlich, dass die Session zentral verwaltet wird. Man kann über den Singelton die Sessionvariable ‘user’ setzten, diese auslesen oder die Session zerstören.

### Database

Database ist ein Singelton, welcher die Datenbankanbindung verwaltet. Er stellt sicher, dass nur eine Datenbankverbindung steht und nicht mehrere gleichzeitig existieren. Über eine Funktion kann von aussen die Verbindung beantragt werden.

### RegisterController

Über den Registercontroller werden die Daten des Registrierungsformular aufbereitet, damit sie in die Datenbank geschrieben werden können.

### LoginController

Der Logincontroller enthält die Funktionen zum Einloggen und abmelden eines Nutzers.

### Validator

Der Validator überprüft die Inputs vom Registrierungsformular und Inseratformular. Ist mindestens ein Eintrag falsch, gibt er false zurück.

### InseratController

Der Inseratcontroller stellt die Verbindung zwischen dem Model und der View her. Der Controller verfügt über drei Methoden. Über eine werden die eingegebenen Daten vom Inseratformular für das Model aufbereitet. Über die zweite Methode werden die Seiten auf der Startseite geladen. Diese enthalten maximal 10 Previews von Inseraten. Über die dritte Funktion kann die Anzahl Seiten an 10 Inseraten abgefragt werden.

### InseratModel

Das InseratModel liest die Inserat-Daten aus der Datenbank heraus, oder schreibt neue oder geänderte Daten in die Datenbank. Inserate können direkt im zehnerpack geladen werden. Dafür muss man angeben, von welcher Position an gesucht werden soll. Eine weiter Funktion ermöglicht es ein Inserat anhand seiner ID zu laden. Des Weiteren gibt es noch die Funktion um ein neues Inserat in die Datenbank zu speichern.

## Gui Screenshots

### Startseite

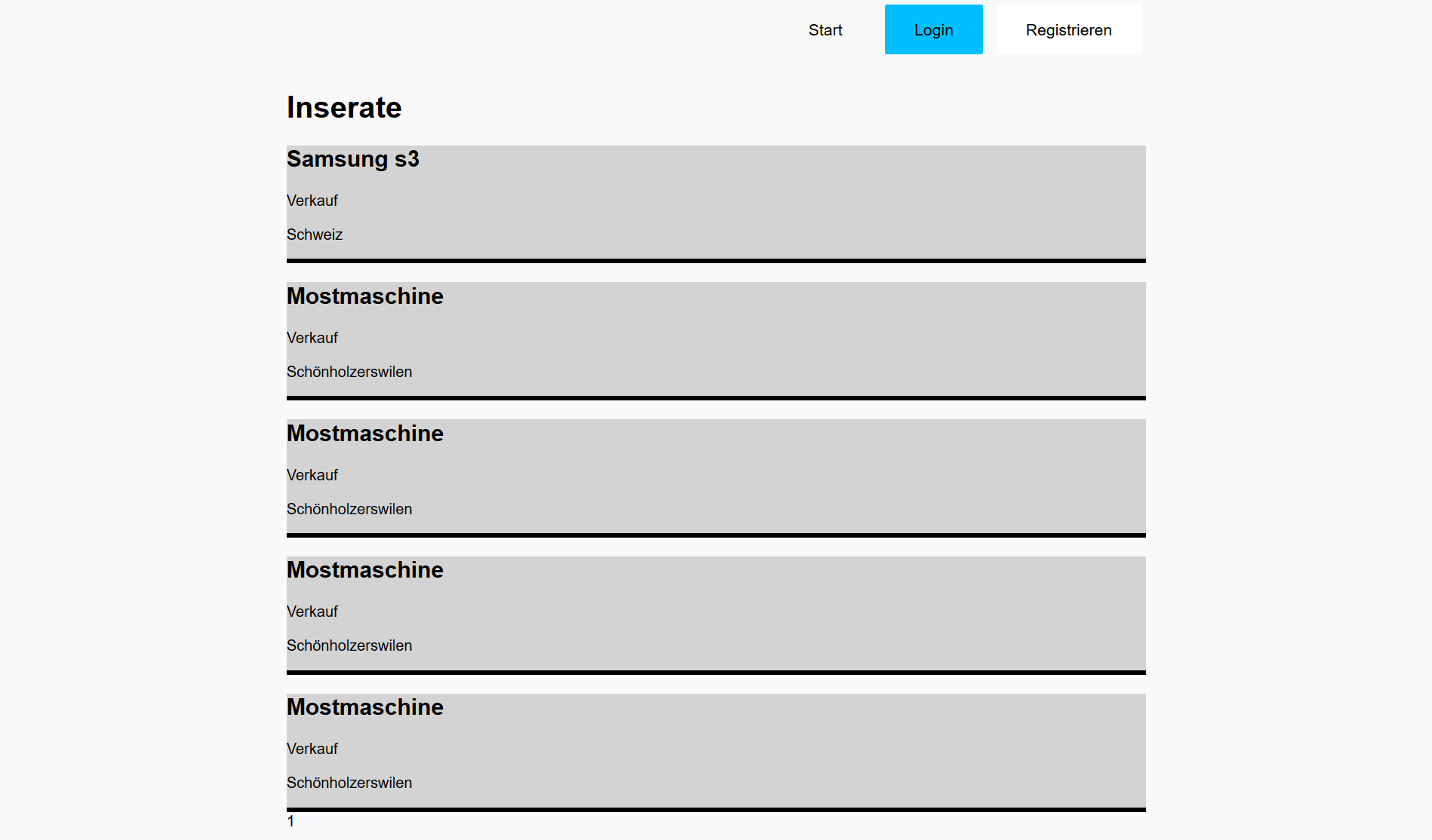


Abbildung 4 Startseite Screenshot

### Registrierung

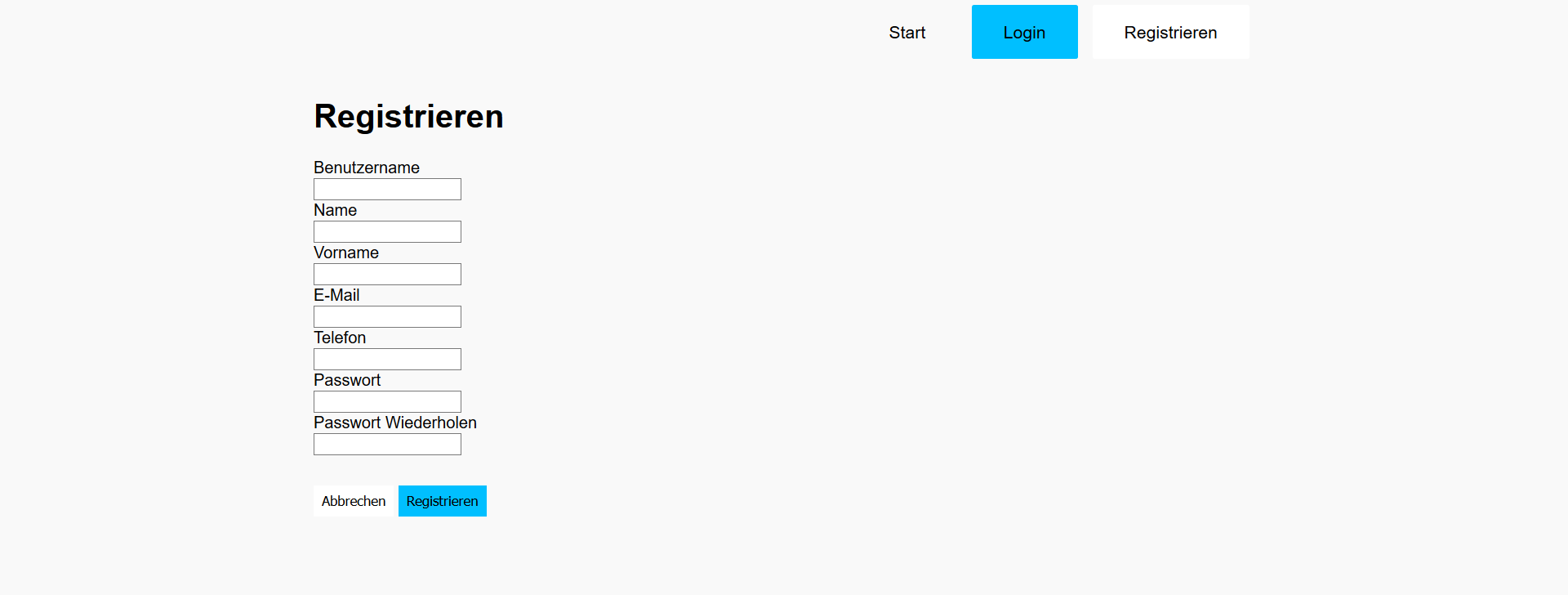


Abbildung 5 Registrierungsseite Screenshot

### Login

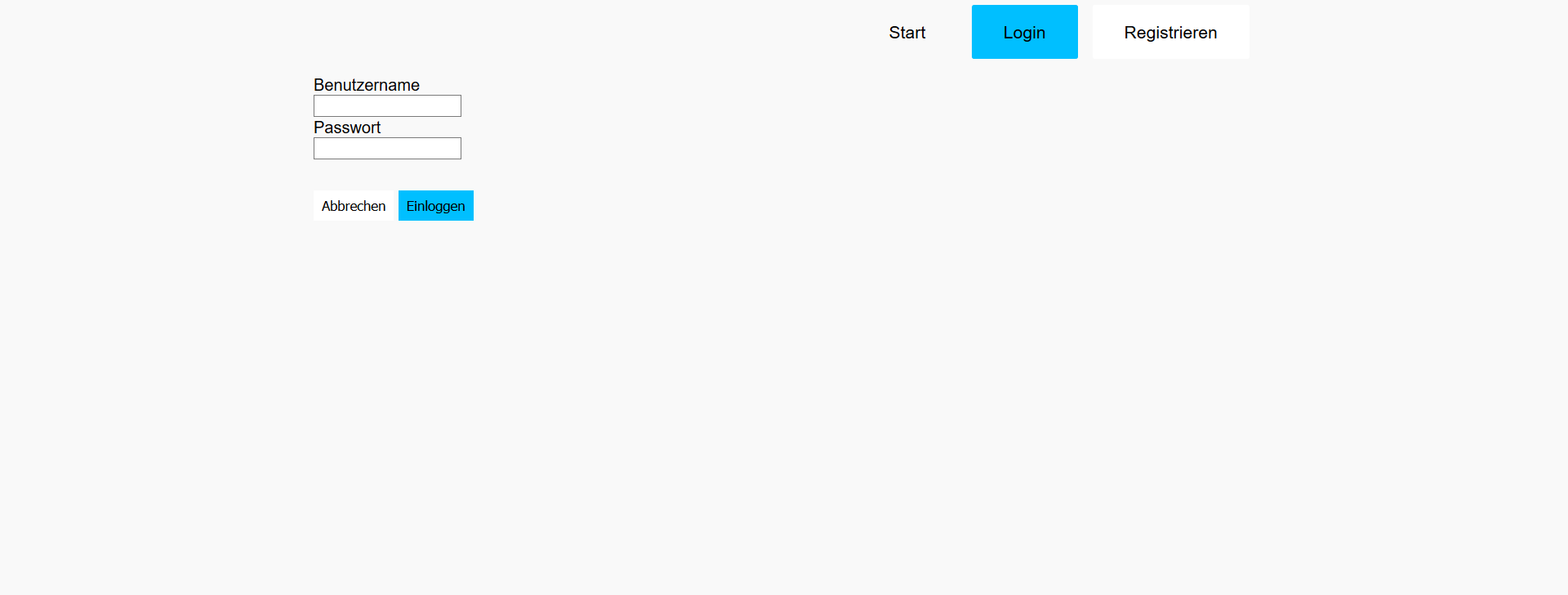


Abbildung 6 Loginseite Screenshot

### Inserat erstellen

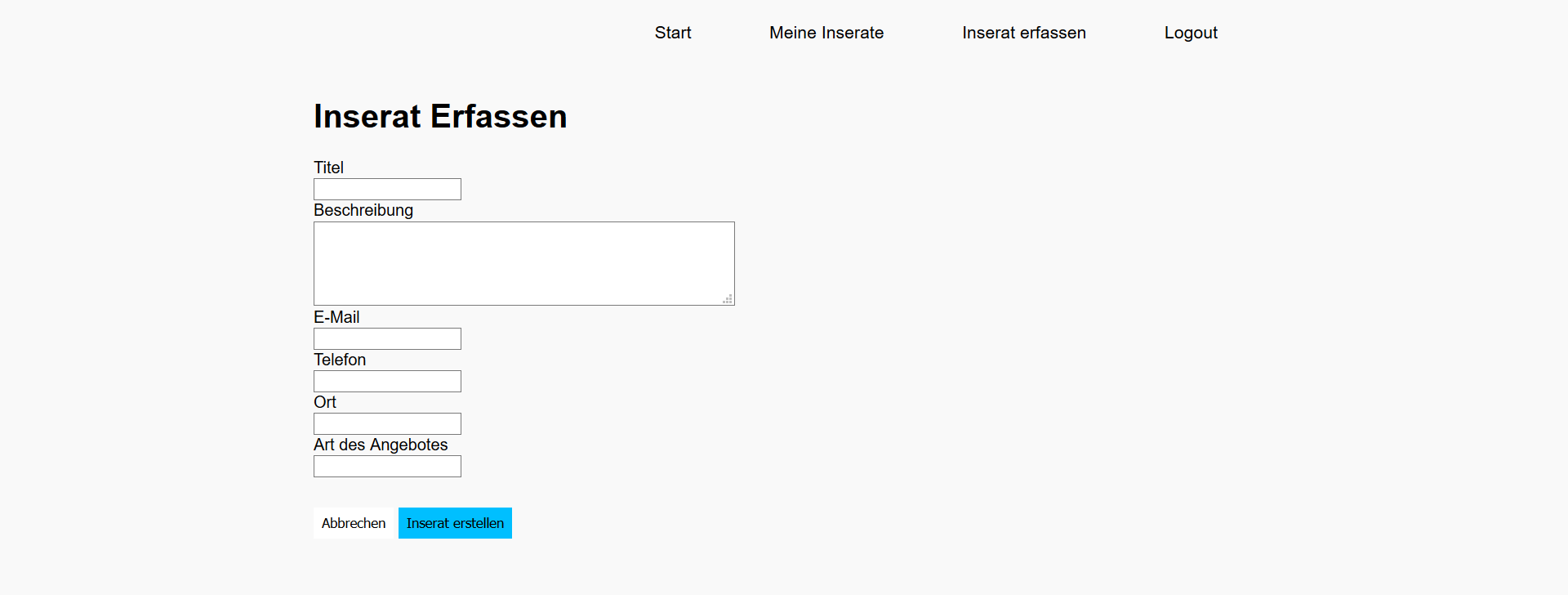


Abbildung 7 Inserat erstellen Seite

### Inserat



Abbildung 8 Inseratseite

# Kontrollieren

## Testübersicht

|  |  |
| --- | --- |
| Kategorie | Ergebnis Testfälle |
| Erfolgreich | 10 |
| Teilweise Erfolgreich | 2 |
| Nicht erfolgreich | 2 |

Tabelle 25: Ergebnis Testfälle

## Testfazit

Das Produkt kann im aktuellen Status nicht veröffentlicht werden. Es mehr als ein Viertel der Testfälle sind nicht oder nur Teilweise bestanden. Da die nicht bestandenen Fällen zu explizit gewünschten Anforderungen gehören, gelten diese als kritisch. Zuerst sollten die fehlenden Funktionen implementiert werden, bevor die Applikation wieder getestet wird und gegebenenfalls veröffentlicht wird.

## Testprotokoll

### Registrierung

|  |  |  |
| --- | --- | --- |
| Test Nummer | Fazit | Abweichungen |
| 1.1 |  | - |
| 1.2 |  | Passwörter werden nicht gelöscht und nicht rot umrahmt |
| 1.3 |  | - |
| 1.4 |  | Das Feld Username wird nicht rot umrahmt |

Tabelle 26 Registrierung Testprotokoll

### Login

|  |  |  |
| --- | --- | --- |
| Test Nummer | Fazit | Abweichungen |
| 2.1 |  | - |
| 2.2 |  | - |
| 2.3 |  | - |

Tabelle 27 Login Testprotokoll

### Logout

|  |  |  |
| --- | --- | --- |
| Test Nummer | Fazit | Abweichungen |
| 3.1 |  | - |

Tabelle 28 Logout Testprotokoll

### Inserate erstellen

|  |  |  |
| --- | --- | --- |
| Test Nummer | Fazit | Abweichungen |
| 4.1 |  | - |
| 4.2 |  | - |
| 4.3 |  | - |

Tabelle 29 Inserate erstellen Testprotokoll

### Inserate ansehen

|  |  |  |
| --- | --- | --- |
| Test Nummer | Fazit | Abweichungen |
| 5.1 |  | - |

Tabelle 30 Inserate ansehen Testprotokoll

### Inserate Löschen

|  |  |  |
| --- | --- | --- |
| Test Nummer | Fazit | Abweichungen |
| 6.1 |  | Seite öffnet nicht |
| 6.2 |  | Kann nicht getestet werden, da die Seite nicht existiert. |

Tabelle 31 Inserate Löschen Testprotokoll

## Soll – Ist Vergleich

Für die Funktionalen Anforderungen siehe: 11.1

|  |  |
| --- | --- |
| Name | Ist-Abweichung |
| Registrieren | Bei Falscheingaben wird keine Rückmeldungen zurückgegeben wenn diese vom Server erkannt wurden. |
| Login | Bei Falscheingaben wird keine Rückmeldungen zurückgegeben wenn diese vom Server erkannt wurden. |
| Logout | - |
| Inserat erstellen | Bei Falscheingaben wird keine Rückmeldungen zurückgegeben wenn diese vom Server erkannt wurden. |
| Inserat Übersicht | - |
| Inserat öffnen | - |
| Inserat Löschen | Die Inserat Löschen Funktion wurde wegen Zeitmängel nicht implementiert und sind daher nicht vorhanden. |
| Adminbereich | Der Adminbereich und seine weiteren Funktionen wurden wegen Zeitmangel nicht implementiert und sind daher nicht vorhanden. |

Tabelle 32 Soll Ist vergleich

Die Applikation konnte in der gegeben Zeit nicht vollständig implementiert werden. Wegen Zeitmängel musste ich gewisse Funktionen streichen, oder konnte sie nicht vollständig implementieren.

# Auswerten

Das Projekt wird Ausgewertet und über die Arbeitsweise reflektiert.

## Reflexion

Das Projekt ist nun abgeschlossen. Es war mein erstes Projekt seit längerem, welches ich alleine von Anfang bis Ende durchführte. Ich bin jedoch nur mässig zufrieden mit dem Resultat. Ich konnte meinen Auftrag nicht komplett abschliessen, habe jedoch eine lauffähige Applikation realisiert. Ich habe in diesem Projekt einige neue Aspekte der Webentwicklung kennengelernt. Ich habe zum Beispiel das erste Mal mit Ajax gearbeitet. Jedoch hat mir Ajax auch einige kleinere Probleme beschert. Mit meiner Architektur bin ich am wenigsten zufrieden. Diese hatte ich in der Planung zu wenig hinterfragt und war dann in der Realisierung nicht sicher ob dies wirklich sinnvoll ist. Dies liegt sicher auch daran, dass ich nie wirklich Produktiverfahrungen im Objektorientiertem Programmieren, sowie Server – Client Kommunikation habe.

## Schlusswort

Wenn ich nachträglich auf die Aufgabenstellung zurückblicke, hätte ich schon früher merken sollen, dass ich das Projekt mit meinem vorherigen Wissen nicht hätte komplett abschliessen können. Es war jedoch eine gute Erfahrung, denn ich kann nun sicher besser einschätzen was ich innerhalb von zwei Tagen realisieren kann. Ich habe während dieser Probe-IPA einige Erfahrungen sammeln können, die mir sowohl in der richtigen IPA als auch in anderen zukünftigen Projekten nützlich sein können.

.

Anhang

# Abbildungsverzeichnis

Jegliche Abbildungen in diesem Dokument wurden selber erstellt.

[Abbildung 1 Usecase Diagramm 23](#_Toc498096161)

[Abbildung 2 Komponentendiagramm 24](#_Toc498096162)

[Abbildung 3 Ordnerstruktur 30](file:///C:\Users\Alain\PhpstormProjects\IPA\dokumentation\IPA-Dokuuuu.docx#_Toc498096163)

[Abbildung 4 Startseite Screenshot 32](#_Toc498096164)

[Abbildung 5 Registrierungsseite Screenshot 33](#_Toc498096165)

[Abbildung 6 Loginseite Screenshot 33](#_Toc498096166)

[Abbildung 7 Inserat erstellen Seite 34](#_Toc498096167)

[Abbildung 8 Inseratseite 34](#_Toc498096168)

# Tabellenverzeichnis

[Tabelle 1: Beteiligte Personen 6](#_Toc498096122)

[Tabelle 2: Projektmeilensteine 9](#_Toc498096123)

[Tabelle 3: Arbeitsprotokoll 01.11.2017 10](#_Toc498096124)

[Tabelle 4: Arbeitsprotokoll 2.11.2017 12](#_Toc498096125)

[Tabelle 5: Arbeitsprotokoll 23.03.2017 13](#_Toc498096126)

[Tabelle 6: Arbeitsprotokoll 24.03.2017 15](#_Toc498096127)

[Tabelle 7: Arbeitsprotokoll 28.03.2017 16](#_Toc498096128)

[Tabelle 8 Funktionale Anforderung 20](#_Toc498096129)

[Tabelle 9 Berechtungungsmatrix 21](#_Toc498096130)

[Tabelle 10: Use Case #1 21](#_Toc498096131)

[Tabelle 11: Use Case #2 22](#_Toc498096132)

[Tabelle 12: Use Case #3 22](#_Toc498096133)

[Tabelle 13: Use Case #4 22](#_Toc498096134)

[Tabelle 14: Use Case #5 22](#_Toc498096135)

[Tabelle 15: Use Case #6 22](#_Toc498096136)

[Tabelle 16: Use Case #7 23](#_Toc498096137)

[Tabelle 17 User Datenbank 25](#_Toc498096138)

[Tabelle 18 Inserat Datenbank 26](#_Toc498096139)

[Tabelle 19 Registrierung Testfall 27](#_Toc498096140)

[Tabelle 20: Login Testfall 28](#_Toc498096141)

[Tabelle 21: Logout Testfall 28](#_Toc498096142)

[Tabelle 22 Inserate erstellen Testfall 29](#_Toc498096143)

[Tabelle 23 Inserate ansehen Testfall 29](#_Toc498096144)

[Tabelle 24 Inserat Löschen Testfall 29](#_Toc498096145)

[Tabelle 25: Ergebnis Testfälle 35](#_Toc498096146)

[Tabelle 26 Registrierung Testprotokoll 36](#_Toc498096147)

[Tabelle 27 Login Testprotokoll 36](#_Toc498096148)

[Tabelle 28 Logout Testprotokoll 36](#_Toc498096149)

[Tabelle 29 Inserate erstellen Testprotokoll 36](#_Toc498096150)

[Tabelle 30 Inserate ansehen Testprotokoll 36](#_Toc498096151)

[Tabelle 31 Inserate Löschen Testprotokoll 36](#_Toc498096152)

[Tabelle 32 Soll Ist vergleich 37](#_Toc498096153)

# Literaturverzeichnis

Das Literaturverzeichnis beinhaltet alle Webseiten, welche während der Realisierung zur Hilfe verwendet wurden.

|  |  |
| --- | --- |
| Was | Quelle |

# Quellenverzeichnis

Das Quellenverzeichnis beinhaltet alle Dokumente & Webseiten, von welchen Texte/Bilder/Vorlagen/Programmcode übernommen und/oder angepasst wurden.

|  |  |
| --- | --- |
| Was | Quelle |
| PHP Problembehebungen | <http://php.net/manual/de/> |

# Glossar

|  |  |
| --- | --- |
| Abkürzung | Definition |
| Ajax | Asynchrone Dateiübertragung zwischen Client und Server, welcher es ermöglicht, Daten vom Server zu laden ohne die Seite neu zu laden. |
| Plant UML | Textbasiertes Tool um UMLs zu erstellen |

# Source-Code

### Index.php

<!DOCTYPE **html**>  
<**html lang="en"**>  
<**head**>  
 <**meta charset="UTF-8"**>  
 <**title**>Inserator</**title**>  
 <**link rel="stylesheet" href="css/style.css"**>  
 <**script src="inserat/inserat.js"**></**script**>  
 <**script src="login/login.js"**></**script**>  
 <**script src="login/register.js"**></**script**>  
 <**script src="js/load.js"**></**script**>  
 <**script src="js/Jquery3.2.1.js"**></**script**>  
</**head**>  
<**body onload="***loadContent*()**"**>  
 <**div id="container"**>  
 <**header**>  
 <**div id="headerContent"** >  
 <**div id="headerButtonContainer"**>  
 <**a onclick="***showLogin*()**" class="headerButton" id="loginButton"**>Login</**a**>  
 <**a onclick="***showRegister*()**" class="headerButton" id="registerButton"** >Registrieren</**a**>  
 </**div**>  
 <**div class="floatClear"**></**div**>  
 </**div**>  
 </**header**>  
  
 <**main id="mainContent"**>  
  
 </**main**>  
 </**div**>  
  
</**body**>  
</**html**>

### registerView.php

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: Alain  
 \* Date: 03.11.2017  
 \* Time: 09:31  
 \*/***echo '<div id="LoginContent">  
 <h1 id="RegisterTitle">Registrieren</h1>  
  
 <form onsubmit="***register***(); return false;" id="registerform">  
 <label for="RegisterUsername" class="RegisterLabel">Benutzername</label> <br />  
 <input type="text" id="RegisterUsername" name="RegisterUsername" class="RegisterInput" required="required"> <br />  
  
 <label for="Name" class="RegisterLabel">Name</label> <br />  
 <input type="text" id="Name" name="Name" class="RegisterInput" > <br />  
  
 <label for="Surname" class="RegisterLabel">Vorname</label> <br />  
 <input type="text" id="Surname" name="Surname" class="RegisterInput" > <br />  
  
 <label for="Mail" class="RegisterLabel">E-Mail</label> <br />  
 <input type="email" id="Mail" name="Mail" class="RegisterInput"> <br />  
  
 <label for="Phone" class="RegisterLabel">Telefon</label> <br />  
 <input type="tel" id="Phone" name="Phone" class="RegisterInput"> <br />  
  
 <label for="RegisterPassword" class="RegisterLabel">Passwort</label> <br />  
 <input pattern="^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)(?=.\*[$@!%\*?&\_-])[A-Za-z\d$@!%\*?&\_-]{8,}"  
 title="Mindestens 8 Zeichen lang. Inklusive Sonderzeichen, Grossbuchstaben und Zahlen." type="password"  
 id="RegisterPassword" name="RegisterPassword" class="RegisterInput"> <br/>  
  
 <label for="RepPassword" class="RegisterLabel">Passwort Wiederholen</label> <br />  
 <input pattern="^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)(?=.\*[$@!%\*?&\_-])[A-Za-z\d$@!%\*?&\_-]{8,}"  
 title="Mindestens 8 Zeichen lang. Inklusive Sonderzeichen, Grossbuchstaben und Zahlen." type="password"  
 id="RepPassword" name="RepPassword" class="RegisterInput" > <br/>  
  
 <input type="reset" name="Abbrechen" class="ResetButton" id="Reset" value="Abbrechen">  
 <input type="submit" name="Registrieren" class="SubmitButton" id="Submit" value="Registrieren">  
  
 </form>  
</div>'**;

### loginView.php

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: Alain  
 \* Date: 03.11.2017  
 \* Time: 10:56  
 \*/***echo '<form id="loginform" onsubmit="***login***(); return false;">  
  
 <label for="Username" class="LoginLabel">Benutzername</label> <br />  
 <input type="text" id="Username" name="Username" class="LoginInput" required="required"> <br />  
  
 <label for="Password" class="LoginLabel">Passwort</label> <br />  
 <input type="password" id="Password" name="Password" class="LoginInput" required="required"> <br />  
  
 <input type="reset" class="ResetButton" id="ResetLogin" value="Abbrechen">  
 <input type="submit" class="SubmitButton" id="SubmitLogin" value="Einloggen">  
  
 </form>  
 <div id="loginErrorContainer"></div>'**;

### HeaderView

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: Alain  
 \* Date: 03.11.2017  
 \* Time: 09:31  
 \*/***require\_once "../model/user.php"**;  
**require\_once "../controller/mySession.php"**;  
  
$currentSession = mySession::*getInstance*();  
  
$user = $currentSession->getCurrentUser();  
  
**if**($user) {  
 **echo "  
 <div>  
   
 </div>  
 <div id=\"headerButtonContainer\">  
 <a onclick=\"***loadContent***()\" class=\"headerButton\" id=\"myInserateButton\">Start</a>  
 <a onclick=\"showMyInserate()\" class=\"headerButton\" id=\"myInserateButton\">Meine Inserate</a>  
 <a onclick=\"***openInseratForm***()\" class=\"headerButton\" id=\"makeInseratButton\">Inserat erfassen</a>  
 <a onclick=\"***logout***()\" class=\"headerButton\" id=\"logoutButton\">Logout</a>  
 </div>  
 <div class=\"floatClear\"></div>"**;  
} **else** {  
 **echo '<div id="headerButtonContainer">  
 <a onclick="***loadContent***()" class="headerButton" id="myInserateButton">Start</a>  
 <a onclick="***showLogin***()" class="headerButton" id="loginButton">Login</a>  
 <a onclick="***showRegister***()" class="headerButton" id="registerButton">Registrieren</a>  
 </div>  
 <div class="floatClear"></div>'**;  
}

### createInserateView.php

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: Alain  
 \* Date: 03.11.2017  
 \* Time: 09:31  
 \*/***echo '<div id="LoginContent">  
 <h1 id="RegisterTitle">Inserat Erfassen</h1>  
  
 <form onsubmit="***createInserat***(); return false;" id="inseratForm">  
 <label for="Title" class="RegisterLabel">Titel</label> <br />  
 <input type="text" id="Title" name="Title" class="RegisterInput" required="required"> <br />  
  
 <label for="Description" class="RegisterLabel">Beschreibung</label> <br />  
  
 <textarea rows="4" cols="50" id="Description" name="Description" maxlength="500">  
 </textarea><br />  
  
 <label for="Mail" class="RegisterLabel">E-Mail</label> <br />  
 <input type="email" id="Mail" name="Mail" class="RegisterInput"> <br />  
  
 <label for="Phone" class="RegisterLabel">Telefon</label> <br />  
 <input type="tel" id="Phone" name="Phone" class="RegisterInput"> <br />  
  
 <label for="Place" class="RegisterLabel">Ort</label> <br />  
 <input type="text" id="Place" name="Place" class="RegisterInput"> <br />  
  
 <label for="Type" class="RegisterLabel">Art des Angebotes</label> <br />  
 <input type="text" id="Type" name="Type" class="RegisterInput"> <br />  
  
  
  
 <input type="reset" name="Abbrechen" class="ResetButton" id="Reset" value="Abbrechen">  
 <input type="submit" name="Registrieren" class="SubmitButton" id="Submit" value="Inserat erstellen">  
  
 </form>  
</div>'**;

### allInserateView.php

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: Alain  
 \* Date: 03.11.2017  
 \* Time: 09:33  
 \*/***require\_once '../inserat/InseratController.php'**;  
  
  
$currentPage = (int)*filter\_input*(***INPUT\_GET***, **'CurrentPage'**, ***FILTER\_SANITIZE\_NUMBER\_INT***) ?? 1;  
  
$controller = **new** InseratController();  
  
$maxPages = $controller->getNumberOfPages();  
  
$inserate = $controller->showInserate($currentPage);  
  
$buttons = **""**;  
  
$buttons = $currentPage == 1 ? **""** : **"<div> <a class='navigationButton' onclick='***previousPage***()'>Vorherige Seite</a></div>"**;  
$buttons = $currentPage == $maxPages ? $buttons.**""** : $buttons.**"<div> <a class='navigationButton' onclick='***nextPage***()'>Nächste Seite</a></div>"**;  
  
  
**echo " <div >  
 <h1>Inserate</h1>  
 </div>  
 <div id=\"inserateContainer\"/>** $inserate **</div>  
 <div>** $buttons $maxPages**</div>"**;

### userModul.php

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: Alain  
 \* Date: 03.11.2017  
 \* Time: 08:48  
 \*/***require\_once "database.php"**;  
**require\_once "user.php"**;  
**require\_once "../controller/mySession.php"**;  
  
**class** userModel  
{  
 **private** $connection;  
 **private $mySession** ;  
  
 **public function** \_\_construct()  
 {  
 $this->connection = database::*getConnection*();  
 $this->**mySession** = mySession::*getInstance*();  
 }  
  
 **public function** registerUser(string $username, string $password, string $surname, string $name, string $mail, string $phone) {  
 $hashedpassword = *password\_hash*($password, ***PASSWORD\_BCRYPT***);  
  
 **try** {  
 $this->connection->beginTransaction();  
 $sql = **"INSERT INTO `inserator`.`user` (`first\_name`, `name`, `username`, `email`, `phone`, `pwd`) VALUES ('**$surname**', '**$name**', '**$username**', '**$mail**', '**$phone**', '**$hashedpassword**')"**;  
 $stmt = $this->connection->prepare($sql); *// Prevent MySQl injection. $stmt means statement* $stmt->execute();  
 $this->connection->commit();  
  
 } **catch** (PDOException $e) {  
 $this->connection->rollBack();  
 **echo "ERROR"**;  
 **return false**;  
 }  
  
 *//$user = new user($name, $surname, $username, $mail, $phone);* $user = $this->loadUserByUsername($username);  
  
 $this->**mySession**->setCurrentUser($user);  
  
  
 **return true**;  
 }  
  
  
 **public function** isUserExisting(string $username) {  
 */\* $sql = "SELECT id FROM user WHERE username ='$username'";  
 $stmt = $this->connection->prepare($sql); // Prevent MySQl injection. $stmt means statement  
 $stmt->execute();  
\*/  
  
 /\*  
 if ( $stmt->() >=1)  
 {  
 return true;  
 }  
\*/* **return true**;  
  
 }  
  
 **public function** loadUserByUsername($username)  
 {  
 $sql = **"SELECT** *\** **FROM user WHERE username ='**$username**'"**;  
 $stmt = $this->connection->prepare($sql); *// Prevent MySQl injection. $stmt means statement* $stmt->execute();  
  
 **if** ($stmt->rowCount() <= 0) {  
 **echo 'Error'**;  
 **return null**;  
 } **else** {  
 **return** $stmt->fetch();  
 }  
  
  
  
 }  
}

### database.php

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: Alain  
 \* Date: 03.11.2017  
 \* Time: 11:05  
 \*/***class** database  
{  
 **private static** *$instance*;  
 **private $conn**;  
  
 */\*\*  
 \* Database constructor.  
 \*/* **private function** \_\_construct()  
 {  
 **try** {  
 $this->**conn** = **new** PDO(**'mysql:host=localhost;dbname=inserator'**, **'root'**, **''**);  
  
 }  
 **catch** (PDOException $e)  
 {  
 **echo 'Error: '** . $e->getMessage();  
 **exit**();  
 }  
 }  
  
 **public static function** getConnection()  
 {  
 **return self**::*getInstance*()->**conn**;  
 }  
  
 **public static function** getInstance()  
 {  
 **if** (!**self**::*$instance*) { *// If no instance then create one* **self**::*$instance* = **new self**();  
 }  
 **return self**::*$instance*;  
 }  
}

### registerInput.php

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: Alain  
 \* Date: 03.11.2017  
 \* Time: 09:43  
 \*/***require\_once "registerController.php"**;  
  
*/\*\*  
 \* Saintizes user Inputs, andgive it to the regsiterController  
 \*/*$username = *filter\_input*(***INPUT\_GET***, **'Username'**, ***FILTER\_SANITIZE\_STRING***) ?? **""**;  
$name = *filter\_input*(***INPUT\_GET***, **'Name'**, ***FILTER\_SANITIZE\_STRING***) ?? **""**;  
$surname = *filter\_input*(***INPUT\_GET***, **'Surname'**, ***FILTER\_SANITIZE\_STRING***) ?? **""**;  
$mail = *filter\_input*(***INPUT\_GET***, **'Mail'**, ***FILTER\_SANITIZE\_EMAIL***) ?? **""**;  
$phone = *filter\_input*(***INPUT\_GET***, **'Phone'**, ***FILTER\_SANITIZE\_STRING***) ?? **""**;  
$password = *filter\_input*(***INPUT\_GET***, **'Password'**) ?? **""**;  
$repeatPassword = *filter\_input*(***INPUT\_GET***, **'RepPassword'**) ?? **""**;  
  
  
$controller = **new** registerController();  
  
$controller->registerUser($username, $password, $repeatPassword, $surname, $name, $mail, $phone);

### registerController

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: Alain  
 \* Date: 03.11.2017  
 \* Time: 09:41  
 \*/***require\_once "../model/userModel.php"**;  
**require\_once "../controller/validator.php"**;  
  
*/\*\*  
 \* Class registerController  
 \* Manages methods used for register an User  
 \*/***class** registerController  
{  
 **private $model**;  
 **private $validator**;  
  
 */\*\*  
 \* registerController constructor.  
 \*/* **public function** \_\_construct()  
 {  
 $this->**model** = **new** userModel();  
 $this->**validator** = **new** validator();  
 }  
  
 */\*\*  
 \* Manages the Userinput to register User  
 \** ***@param*** *$username  
 \** ***@param*** *$password  
 \** ***@param*** *$repeatPassword  
 \** ***@param*** *$surname  
 \** ***@param*** *$name  
 \** ***@param*** *$mail  
 \** ***@param*** *$phone  
 \*/* **public function** registerUser($username, $password, $repeatPassword, $surname, $name, $mail, $phone)  
 {  
  
 $isInputValid = $this->**validator**->registerInputValid($username, $password, $repeatPassword, $surname, $name, $mail, $phone);  
  
 **if**(!$isInputValid){  
 **echo "FEHLER"**;  
 **return**;  
 }  
  
 **if**($this->**model**->registerUser($username, $password, $surname, $name, $mail, $phone)) {  
 **echo 'Success'**;  
 }  
  
  
 }  
  
  
}

### register.js

*/\*\*  
 \* Created by Alain on 03.11.2017.  
 \*/  
  
/\*\*  
 \* Sow register form  
 \* Invoked by RegsiterButton in header  
 \*/***function** *showRegister*() {  
 **$**.**ajax**({  
 **type**: **"GET"**,  
 **url**: **'view/registerView.php'**,  
 **cache**: **false**,  
 success: **function**(data){  
 **$**(**'#mainContent'**).empty();  
 **$**(**'#mainContent'**).append(data);  
 },  
 error: **function** (request, status, error) {  
 *//Shake animation effect.* alert(error);  
 }  
 });  
}  
  
*/\*\*  
 \* Sends inputed registerdata to server for further validation  
 \* invoked by registerbutton in registerform  
 \*/***function** *register*() {  
  
 **var** username=**$**(**"#RegisterUsername"**).val();  
 **var** name=**$**(**"#Name"**).val();  
 **var** surname=**$**(**"#Surname"**).val();  
 **var** mail=**$**(**"#Mail"**).val();  
 **var** phone=**$**(**"#Phone"**).val();  
 **var** password=**$**(**"#RegisterPassword"**).val();  
 **var** repPassword=**$**(**"#RepPassword"**).val();  
  
 **var** dataString = **'Username='**+username+**'&Name='**+name+**'&Surname='**+surname+**'&Mail='**+mail+**'&Phone='**+phone+**'&Password='**+password+**'&RepPassword='**+repPassword;  
  
 **$**.**ajax**({  
 **type**: **"GET"**,  
 **url**: **"login/registerInput.php"**,  
 **data**: dataString,  
 **cache**: **false**,  
 success: **function**(data){  
  
 **if**(data == **"Success"**){  
 *loadContent*();  
  
 }  
 },  
 error: **function** (request, status, error) {  
 *//Shake animation effect.* }  
 });  
}

### logout.php

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: Alain  
 \* Date: 08.11.2017  
 \* Time: 08:38  
 \*/***require\_once 'LoginController.php'**;  
  
$loginController = **new** LoginController();  
  
$loginController->logout();  
  
**require\_once '../view/allInserateView.php'**;

### loginInput.php

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: Alain  
 \* Date: 08.11.2017  
 \* Time: 08:49  
 \*/***require\_once "LoginController.php"**;  
  
$username = *filter\_input*(***INPUT\_GET***, **'Username'**, ***FILTER\_SANITIZE\_STRING***) ?? **""**;  
$password = *filter\_input*(***INPUT\_GET***, **'Password'**) ?? **""**;  
  
$controller = **new** LoginController();  
  
$controller->loginPerson($username, $password);

### LoginController.php

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: Alain  
 \* Date: 03.11.2017  
 \* Time: 09:27  
 \*/***require\_once '../controller/mySession.php'**;  
**require\_once '../model/userModel.php'**;  
  
**class** LoginController  
{  
 **private $customSession**;  
 **private $model**;  
  
 **public function** loginPerson($username, $password)  
 {  
 $user = $this->**model**->loadUserByUsername($username);  
  
 **if**($user) {  
 $passwordCorrect = *password\_verify*($password, $user[**'pwd'**]);  
  
 **if** ($passwordCorrect) {  
  
 $this->**customSession**->setCurrentUser($user);  
 **echo 'Success'**;  
 **return**;  
 }  
 }  
  
 *//Code wrong* $this->loginError();  
 **return**;  
 }  
  
 **public function** \_\_construct()  
 {  
 $this->**customSession** = mySession::*getInstance*();  
 $this->**model** = **new** userModel();  
 }  
  
 **public function** logout() {  
 $this->**customSession**->destroySession();  
 }  
  
 **private function** loginError()  
 {  
 **echo 'error'**;  
 }  
}

### Login.js

*/\*\*  
 \* Created by Alain on 03.11.2017.  
 \*/  
  
/\*\*  
 \* Shows Loginformular for the User to login  
 \* Called when Loginbutton is clicked  
 \*/***function** *showLogin*() {  
 **$**.**ajax**({  
 **type**: **"GET"**,  
 **url**: **'view/loginView.php'**,  
 **cache**: **false**,  
 success: **function**(data){  
 **$**(**'#mainContent'**).empty();  
 **$**(**'#mainContent'**).append(data);  
 },  
 error: **function** (request, status, error) {  
 *//Shake animation effect.* alert(error);  
 }  
 });  
}  
  
*/\*\*  
 \* sends inputed data to server for further validation  
 \* Invoked by Loginbutton in Loginform  
 \*/***function** *login*() {  
 **var** username=**$**(**"#Username"**).val();  
 **var** password=**$**(**"#Password"**).val();  
  
 **var** dataString = **'Username='**+username+ **'&Password='**+password;  
  
 **$**.**ajax**({  
 **type**: **"GET"**,  
 **url**: **"login/loginInput.php"**,  
 **data**: dataString,  
 **cache**: **false**,  
 success: **function**(data){  
  
 **if**(data == **"Success"**){  
 *loadContent*();  
 }  
 **else** {  
 **errorPrint** = **"<p>Login Fehlgeschlagen</p>"**;  
 **$**(**'#loginErrorContainer'**).empty();  
 **$**(**'#loginErrorContainer'**).append(**errorPrint**);  
 }  
 },  
 error: **function** (request, status, error) {  
  
  
 }  
 });  
}  
*/\*\*  
 \* Sends request to Server for the User to get logged out  
 \*/***function** *logout*() {  
 **$**.**ajax**({  
 **type**: **"GET"**,  
 **url**: **'login/logout.php'**,  
 **cache**: **false**,  
 success: **function**(data){  
 *loadContent*();  
 },  
 error: **function** (request, status, error) {  
 *//Shake animation effect.* alert(error);  
 }  
 });  
}

### Load.js

*/\*\*  
 \* Created by Alain on 03.11.2017.  
 \*/  
  
/\*\*  
 \* Loads pagecontent on first start or opening the mainpage  
 \*/***function** *loadContent*(){  
 **$**.**ajax**({  
 **type**: **"GET"**,  
 **url**: **"view/headerView.php"**,  
 **cache**: **false**,  
 success: **function**(data){  
 **$**(**'#headerContent'**).empty();  
 **$**(**'#headerContent'**).append(data);  
 },  
 error: **function** (request, status, error) {  
 *//Shake animation effect.* }  
 });  
  
 **sessionStorage**.setItem(**"CurrentPage"**, 1);  
 *// sessionStorage.getItem("CurrentPage");  
 loadInserate*();  
}

### inseratModel.php

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: Alain  
 \* Date: 03.11.2017  
 \* Time: 08:48  
 \*/***require\_once '../model/database.php'**;  
**require\_once '../controller/mySession.php'**;  
  
  
*/\*\*  
 \* Class inseratModel  
 \* Datamodel Responsilbe for getting Inserat-datas from database and return it to InseratController  
 \*/***class** inseratModel  
{  
  
 **private $connection**;  
 **private $mySession**;  
  
 */\*\*  
 \* inseratModel constructor.  
 \*/* **public function** \_\_construct()  
 {  
 $this->**connection** = database::*getConnection*();  
 }  
  
 */\*\*  
 \* Inserts Inserat into Database  
 \** ***@param*** *$title  
 \** ***@param*** *$description  
 \** ***@param*** *$mail  
 \** ***@param*** *$phone  
 \** ***@param*** *$place  
 \** ***@param*** *$type  
 \** ***@return*** *bool  
 \*/* **public function** insertInserat($title, $description, $mail, $phone, $place, $type)  
 {  
  
 $this->**mySession** = mySession::*getInstance*();  
  
 $user = $this->**mySession**->getCurrentUser();  
  
 $date = *date*(**"Y-m-d H:i:s"**);  
 $userId = $user[**'id'**];  
 $isActive = **true**;  
  
 **try** {  
 $this->**connection**->beginTransaction();  
 $sql = **"INSERT INTO `inserator`.`inserat` (`name`, `type`, `description`, `location`, `email`, `phone`, `date`, `is\_active`, `user\_id`) VALUES ('**$title**', '**$type**', '**$description**', '**$place**', '**$mail**', '**$phone**', '**$date**', '**$isActive**', '**$userId**')"**;  
 $stmt = $this->**connection**->prepare($sql); *// Prevent MySQl injection. $stmt means statement* $stmt->execute();  
 $this->**connection**->commit();  
 } **catch** (mysqli\_sql\_exception $e) {  
 $this->**connection**->rollBack();  
 **echo "ERROR"**;  
 **return false**;  
 }  
 **echo "Success"**;  
 }  
  
 */\*\*  
 \* Loads 10 Inserate from a defined Start, ordered by Date  
 \** ***@param*** *int $start  
 \** ***@return*** *string  
 \*/* **public function** loadInserate(int $start)  
 {  
 $sql = **"SELECT inserat.name, inserat.type, inserat.location, inserat.id FROM inserat ORDER BY inserat.date DESC LIMIT** $start**,10"**;  
 $stmt = $this->**connection**->prepare($sql); *// Prevent MySQl injection. $stmt means statement* $stmt->execute();  
  
 $outputString = **""**;  
  
 **while** ($row = $stmt->fetch())  
 {  
 $name = $row[**'name'**];  
 $type = $row[**'type'**];  
 $location = $row[**'location'**];  
 $id = $row[**'id'**];  
  
 $outputString = $outputString.**"<div class='inseratPreview'>  
 <a href='inserat/inserat.php?id=**$id**'>  
 <h2>**$name**</h2>  
 <p>**$type**</p>  
 <p>**$location**</p></a>  
 </div>"**;  
 }  
  
 **return** $outputString;  
  
  
  
 }  
  
 */\*\*  
 \* Returns a specific Inserat from the database, identified by its ID  
 \** ***@param*** *$id  
 \** ***@return*** *mixed  
 \*/* **public function** loadInserateById($id)  
 {  
 $sql = **"SELECT inserat.name, inserat.type, inserat.location, inserat.description, inserat.email, inserat.phone FROM inserat where inserat.id = '**$id**' and is\_active = 1"**;  
 $stmt = $this->**connection**->prepare($sql); *// Prevent MySQl injection. $stmt means statement* $stmt->execute();  
  
 **return** $stmt->fetch();  
  
 }  
  
 */\*\*  
 \* Returns total ammount of active Inserate stored in the Database.  
 \** ***@return*** *mixed  
 \*/* **public function** loadInserateTotal()  
 {  
 $sql = **"SELECT** *count***(id) as 'count' FROM inserat where is\_active = 1"**;  
 $stmt = $this->**connection**->prepare($sql); *// Prevent MySQl injection. $stmt means statement* $stmt->execute();  
  
 **return** $stmt->fetch();  
 }  
}

### inseratInput.php

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: Alain  
 \* Date: 08.11.2017  
 \* Time: 09:48  
 \*/***require\_once 'InseratController.php'**;  
  
*/\*\*  
 \* Sanitizes input from new Inserat form and passing it formward to controller  
 \*/*$title = *filter\_input*(***INPUT\_GET***, **'Title'**, ***FILTER\_SANITIZE\_STRING***) ?? **""**;  
$description = *filter\_input*(***INPUT\_GET***, **'Description'**, ***FILTER\_SANITIZE\_STRING***) ?? **""**;  
$mail = *filter\_input*(***INPUT\_GET***, **'Mail'**, ***FILTER\_SANITIZE\_EMAIL***) ?? **""**;  
$phone = *filter\_input*(***INPUT\_GET***, **'Phone'**, ***FILTER\_SANITIZE\_STRING***) ?? **""**;  
$place = *filter\_input*(***INPUT\_GET***, **'Place'**, ***FILTER\_SANITIZE\_STRING***) ?? **""**;  
$type = *filter\_input*(***INPUT\_GET***, **'Type'**, ***FILTER\_SANITIZE\_STRING***) ?? **""**;  
  
  
$mySession = mySession::*getInstance*();  
**if**($mySession->getCurrentUser()) {  
 $controller = **new** InseratController();  
  
 $controller->createInserat($title, $description, $mail, $phone, $place, $type);  
} **else** {  
 **echo 'not authenticated'**;  
}

### InseratController.php

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: Alain  
 \* Date: 08.11.2017  
 \* Time: 09:48  
 \*/***require\_once 'inseratModel.php'**;  
**require\_once '../controller/validator.php'**;  
  
*/\*\*  
 \* Class InseratController  
 \* Controller for methods concerning Inserate  
 \*/***class** InseratController  
{  
 **private $model**;  
 **private $validator**;  
  
  
 */\*\*  
 \* InseratController constructor.  
 \*/* **public function** \_\_construct()  
 {  
 $this->**model** = **new** inseratModel();  
 }  
  
 */\*\*  
 \* Manages the creation of a new Inserat  
 \** ***@param*** *string $title  
 \** ***@param*** *string $description  
 \** ***@param*** *string $mail  
 \** ***@param*** *string $phone  
 \** ***@param*** *string $place  
 \** ***@param*** *string $type  
 \*/* **public function** createInserat(string $title, string $description, string $mail, string $phone, string $place, string $type)  
 {  
 $this->**validator** = **new** validator();  
 $isInputValid = $this->**validator**->inseratInputValid($title, $description, $mail, $phone, $place, $type);  
 **if**($isInputValid)  
 {  
 $this->**model**->insertInserat($title, $description, $mail, $phone, $place, $type);  
 }  
 }  
  
 */\*\*  
 \* Loads Inserate from Model an pass it to the View  
 \** ***@param*** *$page  
 \** ***@return*** *string  
 \*/* **public function** showInserate($page)  
 {  
 $page = (int)($page \* 10) - 10;  
 $inserate = $this->**model**->loadInserate($page);  
 **return** $inserate;  
 }  
  
 */\*\*  
 \* Returns total number of pages if pagelength is equals 10  
 \** ***@return*** *int  
 \*/* **public function** getNumberOfPages()  
 {  
 $result = $this->**model**->loadInserateTotal();  
  
 $numberOfInserates = $result[**'count'**];  
  
 $numberOfPages = $numberOfInserates % 10 >= 0 ? ($numberOfInserates / 10) + 1 : $numberOfInserates / 10;  
  
 **return** (int)$numberOfPages;  
 }  
  
  
}

### Inserat.php

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: Alain  
 \* Date: 08.11.2017  
 \* Time: 13:55  
 \*/***require\_once 'InseratController.php'**;  
  
$model = **new** inseratModel();  
  
$id = $\_GET[**'id'**];  
  
$data = $model->loadInserateById($id);  
  
**?>**<!DOCTYPE **html**>  
<**html lang="en"**>  
<**head**>  
 <**meta charset="UTF-8"**>  
 <**title**>Inserator</**title**>  
 <**link rel="stylesheet" href="../css/style.css"**>  
 <**script src="inserat.js"**></**script**>  
 <**script src="../login/login.js"**></**script**>  
 <**script src="../login/register.js"**></**script**>  
 <**script src="../js/load.js"**></**script**>  
 <**script src="../js/Jquery3.2.1.js"**></**script**>  
</**head**>  
<**body onload="***loadContent*()**"**>  
 <**header**>  
 <**div id="headerContent"**>  
 <**a class="headerButton" id="loginButton" href="../index.php"**>Zurück</**a**>  
 <**div class="floatClear"**></**div**>  
 </**div**>  
 </**header**>  
  
 <**main id="mainContent"**>  
 <**h2**>**<?php echo** $data[**'name'**]; **?>**</**h2**>  
 <**h2**>Typ</**h2**>  
 <**p**>**<?php echo** $data[**'type'**]; **?>**</**p**>  
 <**h2**>Beschreibung</**h2**>  
 <**p**>**<?php echo** $data[**'description'**]; **?>**</**p**>  
 <**h2**>Ort</**h2**>  
 <**p**>**<?php echo** $data[**'location'**]; **?>**</**p**>  
 <**h2**>Telefon</**h2**>  
 <**p**>**<?php echo** $data[**'phone'**]; **?>**</**p**>  
 <**h2**>E-Mail</**h2**>  
 <**p**>**<?php echo** $data[**'email'**]; **?>**</**p**>  
  
 </**main**>  
</**body**>  
</**html**>

### Inserat.js

*/\*\*  
 \* Created by Alain on 03.11.2017.  
 \*/  
  
/\*\*  
 \* Calls php method for creating new Inserat  
 \*/***function** *createInserat*() {  
 **var** title=**$**(**"#Title"**).val();  
 **var** description=**$**(**"#Description"**).val();  
 **var** phone=**$**(**"#Phone"**).val();  
 **var** mail=**$**(**"#Mail"**).val();  
 **var** place=**$**(**"#Place"**).val();  
 **var** type=**$**(**"#Type"**).val();  
  
  
 **var** dataString = **'Title='**+title+**'&Description='**+description+**'&Mail='**+mail+**'&Phone='**+phone+**'&Place='**+place+**'&Type='**+type;  
  
 **$**.**ajax**({  
 **type**: **"GET"**,  
 **url**: **"inserat/inseratInput.php"**,  
 **data**: dataString,  
 **cache**: **false**,  
 success: **function**(data){  
  
 **if**(data == **"Success"**){  
 alert(**'Inserat wurde erstellt'**);  
 *loadContent*();  
 } **else** {  
 alert(**"Fehlgeschlagen"**);  
 }  
 },  
 error: **function** (request, status, error) {  
 *//Shake animation effect.* }  
 });  
}  
  
*/\*\*  
 \* Opens php method for showing the new Inserat Formular  
 \*/***function** *openInseratForm*() {  
 **$**.**ajax**({  
 **type**: **"GET"**,  
 **url**: **'view/createInseratView.php'**,  
 **cache**: **false**,  
 success: **function**(data){  
 **$**(**'#mainContent'**).empty();  
 **$**(**'#mainContent'**).append(data);  
 },  
 error: **function** (request, status, error) {  
 *//Shake animation effect.* alert(error);  
 }  
 });  
}  
  
*/\*\*  
 \* Gos to next page in the Mainpage  
 \*/***function** *nextPage*() {  
 **var** currentPage = **sessionStorage**.getItem(**"CurrentPage"**);  
 **sessionStorage**.setItem(**"CurrentPage"**, parseInt(currentPage) + 1);  
 *loadInserate*();  
}  
  
*/\*\*  
 \* Go to previous page in the Mainpage  
 \*/***function** *previousPage*() {  
 **var** currentPage = **sessionStorage**.getItem(**"CurrentPage"**);  
 **sessionStorage**.setItem(**"CurrentPage"**, parseInt(currentPage) - 1);  
 *loadInserate*();  
}  
  
*/\*\*  
 \* Loads the Inserate for one page from the Server  
 \*/***function** *loadInserate*() {  
  
 **var** currentPage = **sessionStorage**.getItem(**"CurrentPage"**);  
  
 **var** dataString = **'CurrentPage='**+currentPage;  
  
 **$**.**ajax**({  
 **type**: **"GET"**,  
 **url**: **"view/allInserateView.php"**,  
 **data**: dataString,  
 **cache**: **false**,  
 success: **function**(data){  
  
 **$**(**'#mainContent'**).empty();  
 **$**(**'#mainContent'**).append(data);  
 },  
 error: **function** (request, status, error) {  
 *//Shake animation effect.* }  
 });  
}

### Style.css

**html** {  
 **background-color**: **#f9f9f9**;  
}  
**#container** {  
 **background-color**: **#f9f9f9**;  
}  
  
**body** {  
 **margin**: 0;  
 **font-family**: **Arial**, **"Century Gothic"**;  
 **background-color**: **#f9f9f9**;  
}  
**a** {  
 **text-decoration**: **none**;  
 **color**: **black**;  
}  
**header** {  
 **width**: 60%;  
 **min-height**: 50**px**;  
 **background**: **#f9f9f9**;  
 **margin**: 0 **auto**;  
}  
  
**main** {  
 **width**: 60%;  
 **min-height**: 500**px**;  
 **background**: **#f9f9f9**;  
 **margin**: 0 **auto**;  
 **padding**:10**px**;  
}  
  
**#headerButtonContainer** {  
 **float**: **right**;  
}  
  
.**headerButton** {  
 **-moz-border-radius**:2**px**;  
 **-webkit-border-radius**:2**px**;  
 **border-radius**:2**px**;  
 **display**:**inline-block**;  
 **cursor**:**pointer**;  
 **color**:**black**;  
 **font-size**:17**px**;  
 **padding**:16**px** 31**px**;  
 **text-decoration**:**none**;  
 **margin**:5**px**;  
}  
  
.**SubmitButton** {  
 **background-color**: **deepskyblue**;  
 **border**: **none**;  
 **height**: 30**px**;  
 **margin-top**: 20**px**;  
}  
  
.**ResetButton** {  
 **background-color**: **white**;  
 **border**: **none**;  
 **height**: 30**px**;  
 **margin-top**: 30**px**;  
}  
  
**#loginButton** {  
 **background**: **deepskyblue**;  
}  
  
**#registerButton** {  
 **background**: **white**;  
}  
  
.**floatClear** {  
 **clear**: **both**;  
}  
  
.**inseratPreview** {  
 **background**: **lightgrey**;  
 **border-bottom**: **solid black** 5**px**;  
}

### Validator.php

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: Alain  
 \* Date: 03.11.2017  
 \* Time: 14:17  
 \*/***require\_once "../model/userModel.php"**;  
  
*/\*\*  
 \* Class validator  
 \* Class to manage the Input Validators  
 \*/***class** validator  
{  
 */\*\*  
 \* Validates the inputs used for register  
 \** ***@param*** *string $username  
 \** ***@param*** *string $password  
 \** ***@param*** *string $repeatPassword  
 \** ***@param*** *string $surname  
 \** ***@param*** *string $name  
 \** ***@param*** *string $mail  
 \** ***@param*** *string $phone  
 \** ***@return*** *bool  
 \*/* **public function** registerInputValid(string $username, string $password, string $repeatPassword, string $surname, string $name, string $mail, string $phone)  
 {  
 $model = **new** userModel();  
 $passwordRegularExpression = **'/^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)(?=.\*[$@!%\*?&\_-])[A-Za-z\d$@!%\*?&\_-]{8,}/'**;  
  
 $usernameValid = *strlen*($username) < 40 && *strlen*($username) > 1;  
  
 $usernameUnique = $model->isUserExisting($username);  
  
 $passwordValid = $password == $repeatPassword; *//&& preg\_match($passwordRegularExpression, $password);* $surnameValid = *strlen*($surname) < 40 && *strlen*($surname) > 1;  
  
 $nameValid = *strlen*($name) < 40 && *strlen*($name) > 1;  
  
 $mailValid = *filter\_var*($mail, ***FILTER\_VALIDATE\_EMAIL***);  
  
 $phoneValid = *strlen*($phone) <= 20 && *strlen*($phone) >= 10;  
  
 $validationResults = **array**(**'Username'** => $usernameValid, **'Username2'** =>$usernameUnique, **'Password'** => $passwordValid, **'Surname'** => $surnameValid, **'Name'** => $nameValid, **'Email'** => $mailValid != **""**, **"Phone"** => $phoneValid);  
  
 **foreach** ($validationResults **as** $key => $value) {  
 **if**(!$value) {  
 **echo** $key;  
 **return false**;  
 *//****TODO: manage the faulty input*** }  
 }  
 **return true**;  
 }  
  
 */\*\*  
 \* Validates the Input used for creating a new Inserat  
 \** ***@param*** *string $title  
 \** ***@param*** *string $description  
 \** ***@param*** *string $mail  
 \** ***@param*** *string $phone  
 \** ***@param*** *string $place  
 \** ***@param*** *string $type  
 \** ***@return*** *bool  
 \*/* **public function** inseratInputValid(string $title, string $description, string $mail, string $phone, string $place, string $type)  
 {  
 $titleValid = *strlen*($title) < 50 && *strlen*($title) > 1;  
 $descriptionValid = *strlen*($description) < 500 && *strlen*($description) > 1;  
 $mailValid = *filter\_var*($mail, ***FILTER\_VALIDATE\_EMAIL***);  
 $phoneValid = *strlen*($phone) <= 20 && *strlen*($phone) >= 10;  
 $placeValid = *strlen*($place) < 50 && *strlen*($place) > 1;  
 $typeValid = *strlen*($type) < 45 && *strlen*($type) > 1;  
  
  
 $validationResults = **array**(**'Title'** => $titleValid, **'Description'** =>$descriptionValid, **'mail'** => $mailValid != **""**, **'Phone'** => $phoneValid, **'Place'** => $placeValid, **'Type'** => $typeValid);  
  
 **foreach** ($validationResults **as** $key => $value) {  
 **if**(!$value) {  
 **echo** $key;  
 **return false**;  
 }  
 }  
 **return true**;  
  
  
 }  
}

### mySession.php

**<?php***/\*\*  
 \* Created by PhpStorm.  
 \* User: Alain  
 \* Date: 04.11.2017  
 \* Time: 18:19  
 \*/  
  
/\*\*  
 \* Class mySession  
 \* Singelton to manage the Session  
 \*/***class** mySession  
{  
 **private static** *$instance*;  
  
 */\*\*  
 \* SessionHandler constructor.  
 \*/* **private function** \_\_construct() {  
 **if** (*session\_status*() == ***PHP\_SESSION\_NONE***) {  
 *session\_start*();  
 }  
 }  
  
 */\*\*  
 \* Returns instance of itself. If an Instance already exists return this  
 \** ***@return*** *mySession  
 \*  
 \*/* **public static function** getInstance()  
 {  
 **if** (!**self**::*$instance*) { *// If no instance then create one* **self**::*$instance* = **new self**();  
 }  
 **return self**::*$instance*;  
 }  
  
 */\*\*  
 \* Set the session-value for current user  
 \** ***@param*** *$user User to be stored in the sessionfield 'user'  
 \*/* **public function** setCurrentUser($user)  
 {  
 $\_SESSION[**'CurrentUser'**] = $user;  
 }  
  
 */\*\*  
 \* Returns the set user of the session. If it doesn't exists return null  
 \** ***@return*** *$\_SESSION['CurrentUser']  
 \*/* **public function** getCurrentUser()  
 {  
 **return isset**($\_SESSION[**'CurrentUser'**]) ? $\_SESSION[**'CurrentUser'**] : **null**;  
 }  
  
 */\*\*  
 \* Destroys current session  
 \*/* **public function** destroySession()  
 {  
 **if** (*session\_status*() == ***PHP\_SESSION\_NONE***) {  
 *session\_start*();  
 }  
  
 *session\_destroy*();  
 }  
}

### Inserator.sql

*-- MySQL Script generated by MySQL Workbench  
-- Wed Nov 8 12:21:59 2017  
-- Model: New Model Version: 1.0  
-- MySQL Workbench Forward Engineering***SET** @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;  
**SET** @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;  
**SET** @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE=**'TRADITIONAL,ALLOW\_INVALID\_DATES'**;  
  
*-- -----------------------------------------------------  
-- Schema Inserator  
-- -----------------------------------------------------  
  
-- -----------------------------------------------------  
-- Schema Inserator  
-- -----------------------------------------------------***CREATE SCHEMA** IF **NOT EXISTS** `Inserator` **DEFAULT CHARACTER SET** utf8 ;  
USE `Inserator` ;  
  
*-- -----------------------------------------------------  
-- Table `Inserator`.`user`  
-- -----------------------------------------------------***DROP TABLE** IF **EXISTS** `Inserator`.`user` ;  
  
**CREATE TABLE** IF **NOT EXISTS** `Inserator`.`user` (  
 `id` **INT NOT NULL** AUTO\_INCREMENT,  
 `first\_name` **VARCHAR**(20) **NOT NULL**,  
 `name` **VARCHAR**(20) **NOT NULL**,  
 `username` **VARCHAR**(20) **NOT NULL**,  
 `email` **VARCHAR**(50) **NOT NULL**,  
 `phone` **VARCHAR**(20) **NOT NULL**,  
 `pwd` **CHAR**(60) **NOT NULL**,  
 `is\_admin` TINYINT **NOT NULL**,  
 **PRIMARY KEY** (`id`),  
 **UNIQUE** INDEX `username\_UNIQUE` (`username` **ASC**))  
ENGINE = InnoDB;  
  
  
*-- -----------------------------------------------------  
-- Table `Inserator`.`inserat`  
-- -----------------------------------------------------***DROP TABLE** IF **EXISTS** `Inserator`.`inserat` ;  
  
**CREATE TABLE** IF **NOT EXISTS** `Inserator`.`inserat` (  
 `id` **INT NOT NULL** AUTO\_INCREMENT,  
 `name` **VARCHAR**(50) **NOT NULL**,  
 `type` **VARCHAR**(45) **NOT NULL**,  
 `description` **VARCHAR**(500) **NOT NULL**,  
 `location` **VARCHAR**(50) **NOT NULL**,  
 `email` **VARCHAR**(50) **NOT NULL**,  
 `phone` **VARCHAR**(20) **NOT NULL**,  
 `date` DATETIME **NOT NULL**,  
 `is\_active` TINYINT **NOT NULL**,  
 `user\_id` **INT NOT NULL**,  
 **PRIMARY KEY** (`id`),  
 INDEX `fk\_inserate\_user\_idx` (`user\_id` **ASC**),  
 **CONSTRAINT** `fk\_inserate\_user`  
 **FOREIGN KEY** (`user\_id`)  
 **REFERENCES** `Inserator`.`user` (`id`)  
 **ON DELETE NO ACTION  
 ON UPDATE NO ACTION**)  
ENGINE = InnoDB;  
  
USE `Inserator` ;  
  
*-- -----------------------------------------------------  
-- Placeholder table for view `Inserator`.`view1`  
-- -----------------------------------------------------***CREATE TABLE** IF **NOT EXISTS** `Inserator`.`view1` (`id` **INT**);  
  
*-- -----------------------------------------------------  
-- View `Inserator`.`view1`  
-- -----------------------------------------------------***DROP VIEW** IF **EXISTS** `Inserator`.`view1` ;  
**DROP TABLE** IF **EXISTS** `Inserator`.`view1`;  
USE `Inserator`;  
  
  
**SET** SQL\_MODE=@OLD\_SQL\_MODE;  
**SET** FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;  
**SET** UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;