Projektarbeit für das Modul 151

Technische Dokumentation

Einleitung

Vergleiche ich das Projekt, wie es nun abgegeben wird, mit meiner ursprüunglichen Projekteingabe, so habe ich zwar einiges erreicht, aber vieles auch nicht. Um die folgenden Ausführungen besser zu verstehen, folgt hier eine kurze Übersicht über den Stand der Webapplikation.

1. Benutzer ruft die Seite auf → er sieht auf dem Kopf der Seite ein Menu mit den Links “Forum”, “Tactical Map”, einem Login-Forumar und einem Link zur Registrierung.

2. Benutzer registrert sich und loggt sich ein → Das Menu enthält jetzt auch einen Drop-Down-Bereich “MySquad”. Anstatt dem Login-Formular und dem Registrierungslink sieht er nun einen Logout-Link und einen Link zu seinem Profil.

3. Ein frisch registrierter Benutzer kann noch nicht viel machen: Er kann sein Profil ändern (Name, Passwort etc). Sämtliche Seiten unter “MySquad” werden ihm aber die Meldung einbringen, dass er noch nicht in einem Squad ist. Er bekommt einen Link zu einem Formular, mit dem er ein eigenes Squad beantragen kann.

4. Für den Antrag auf ein eigenes Squad benötigt ein Benutzer zwei weitere Mitspieler. Sobald er das Formular abgeschickt hat, ist der Antrag im Adminbereich sichtbar. Hat der Admin dem Antrag stattgegeben, hat der Benutzer nun vollen Zugriff auf die Seite.

5. Im momentanen Stand kann ein Benutzer, sofern er Squadleader ist, unter “MySquad → Marketplace” Inventar kaufen und verkaufen. Unter “MySquad → InventoryOverview” kann er Inventar von seinem Camp in das Spiel schieben und umgekehrt. Spieler, die Mitfglied in einem Squad, aber nicht Anführer sind, können diese Seiten ansehen, aber keine Transaktionen tätigen.

6. Es existiert zudem ein Adminbereich unter “/admin”. Admins können Squadanträge bewilligen, neues Inventar in die Datenbank hinzufügen und bestehendes Inventar verändern.

Registrierung und Login

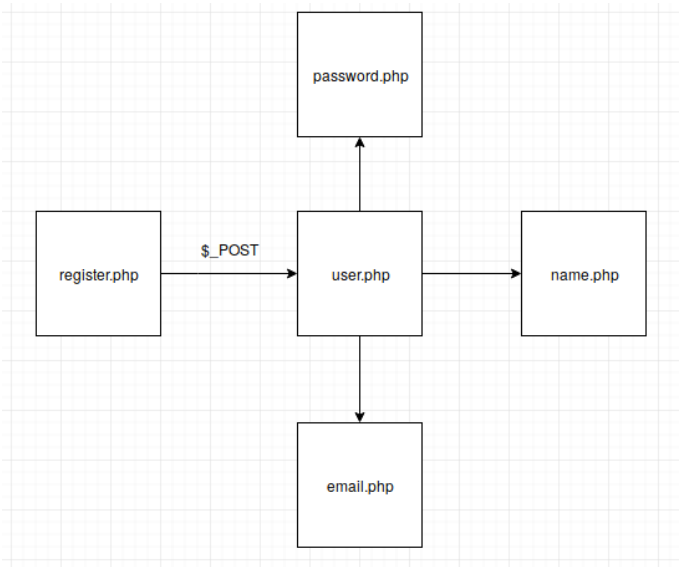
1. Clientseitige Validierung
   Die clientseitige Validierung geschieht mit HTML5. Verwendet wird unter anderem die Spezifizierung eines Inputtyps: <input type = ‘text’, = ‘email’ und andere), das Markieren

von Eingabefeldern als ‘required’ oder das Setzen einer Mindestlänge mit ‘minlength’. Auf ein Passwortmuster habe ich verzichtet.

Siehe Script “includes/register.php” für die konkrete Umsetzung.

2. Serversetige Validierung
   Sämtliche Benutzereingaben werden auch serverseitig validiert. Eingaben werden von Leerzeichen am Anfang und Ende befreit (php: trim()) und nochmals auf ihre Länge überprüft. Alle Eingaben – auch das Passwort – werden zudem mit dem php-Befehl “htmlspecialchars()” gefiltert, um Script-Injections zu verhindern.

   Registrierung: Das Script “includes/register.php” leitet die gesamten Benutzereingaben direkt weiter and die Klasse “User”.Sämtliche serverseitigen Validierungen geschehen nun hier (siehe “validations/user.php”) oder werden weiter delegiert: Die Klasse User erstellt einen neuen Namen (“validations/name.php”) oder ein neues Email-Objekt (“validations/email.php”). Diese validieren die jeweilige Eingabe.



   Login: Das Login funktioniert vom Prinzip her ähnlich wie die Registrierung (siehe “index.php”):
   Hier prüft die Klasse Login zunächt die Länge der Eingabe und Filtert sie danach mit htmlspecialchars(). Danach wird wird die statische Methode User::fromLogin() aufgerufen, die nun den Abgleich mit der Datenbank übernimmt und das Passwort überprüft.

3. Passwort
   Passwörter werden gesalzen und gehascht. Dies geschieht mit dem php-Befehl “password_hash(). Siehe script “validations/password.php”. Nur das gehaschte Passwort wird in der Datenbank gespeichert. Zum Speichervorgang siehe User::save() oder User::update().

Datenbank

1. Verbindung
   Die Verbindung zur Datenbank wird in “includes/database.php” über das Objekt “Database” hergestellt. Der Zugang zur Datenbank geschieht über den User “webuser”. Dieser Benutzer kann in der Datenbank “a3” alle Tabellen lesen, Einträge verändern, hinzufügen und löschen. Er kann keine neuen Tabellen erstellen oder ganze Tabellen löschen.

   Die Verbindung zur Datenbank wird im Destruktor der Klasse Database getrennt.

2. Aufbau:
   Für den grundlegenden Aufbau der Datenbank verweise ich auf den Evernote-Eintrag “Skizze der Datenbank”. Einige Details haben sich inwzischen geändert. Das gehaschte und gesalzene Passwort wird unter “Player.password” als Varchar(255) gespeichert.

   Zudem kam eine Tabelle “Admin” hinzu. Diese beinhaltet ebenfalls ein Passwort-Feld: Da ein Admin auch gleichzeitig ein normaler User sein kann, habe ich mich entschieden, hier zwei verschiedene Passwörter zu verwenden. Admins können nicht per Website registriert werden: Sie müssen direkt in die Datenbank eingetragen werden.

3. Ansprechen mit PHP
   Alle Datenbankverbindungen von der Website aus werden mit PDO umgesetzt. Ich arbeite immer mit Prepared Statements, um SQL-Injection zu verhindern, aber auch aus Performancegründen: Auch Datenbankzugriffe, die keine Benutzereingaben verwenden, werden mit Prepared Statements ausgeführt.
   In den meisten Fällen wird die Datenbank über “index.php” geöffnet, und alle inkludierten Unterseiten haben nun Zugriff auf sie.

Sessionhandling

1. Grundlagen
   Fast alle Scripts der Webapplikation starten eine neue Session mit “session_start()”. Sensible Scripts wie die Registrierung, das Ändern der Benutzerdaten (“includes/profile.php”), aber auch der Marktplatz für die Squads setzten zudem die Session-ID neu (“session_regenerate_id()). Diese Funktion wird eher zu oft als zu selten verwendet.

2. Identifizierung der Benutzer und Squads
   Ob ein Benutzer angemeldet ist oder nicht, wird über die Sessionvariable “$_SESSION[‘user’] entschieden. Sämtliche sensiblen Seiten leiten einen nicht-registrierten Benutzer zurück auf den Index. Gesetzt wird die Sessionvariable in der Funktion “User::fromLogin” in “validations/user.php”.

Da meine Website mehrere Benutzerschichten hat (registrierte Benutzer ohne Squad, Squadmitglieder, Squadleader, aber auch Squadmitglieder von einem Squad, das noch nicht von einem Admin validiert wurde) braucht es weitere Prüfungen des Benutzerstatus. Hier habe ich – leider – nicht konsequent mit Sessions gearbeitet. Ebenfalls in einer Session gespeichert wird, ob ein Benutzer Squadleader ist ($_SESSION[‘squadLeader’]). Zu welchem Squad ein Benutzer gehört, wird aber jeweils aus der Datenbank abgefragt (siehe “Squad::load()” in “validations/squad.php”).