Schulnotenverwaltung Version 2

der Gewerbeoberschule Bozen

Name des Autors:.....Markus Windegger
Auftraggeber:.....Gewerbeoberschule Bozen
Ansprechperson:.....Roland Lafogler
Auftragsdatum:.....Schuljahr 2010/2011
Letzte Änderung am:.....12.05.11

Inhaltsverzeichnis

1 Auftrag

Entwickeln einer Anwendung, welche es erlaubt, den Lehrern ihre Noten einzugeben, ohne von anderen Lehrern die Noten zu sehen, und auch die gesamten Noten nach bestimmten Formaten (Poppcorn) auszugeben.

Es soll eine sichere Authentifizierung vorgenommen werden können, sodass auch mehrere Arten von Benutzern, Administratoren und Lehrpersonen Zugriff haben.

Der Zugriff soll so intuitiv wie möglich von sich gehen und soll auch so einfach und mit so wenigen Klicks wie möglich zu bewältigen sein.

Das System soll auch modular geschrieben werden, sodass jederzeit eine Erweiterung der Funktionen einfach zu implementieren ist.

Eine gute Dokumentation und die Lizenzierung unter GPL v1.3 werden vom Programmierer als selbstverständlich angesehen und somit in den Auftrag mit aufgenommen.

2 Theoretischer Hintergrund

Die Eingabe der Noten befreit den Notengabezyklus von Bürokratie, sodass die Sitzungen schneller gehen, sodass keine 3-fache Mitschrift in den Sitzungen mehr Nötig ist.

Vom Entwickler wird aber eingeworfen, dass der Schüler selbst nicht durch diesen vereinfachten Vorgang in den Hintergrund gedrängt wird, und die Benotung so wie vorher vom Klassenrat vorgenommen wird und nicht durch die Lehrperson.

2.1 Php

Php ist eine Skriptsprache, aktuell in der Version 5.3.X, welche bevorzugt zur Implementierung von Webanwendungen gebraucht wird. Mit ihr kann man sei es Objektorientiert als auch funktional arbeiten. Mit dieser Skriptsprache, welcher der Programmiersprache C sehr ähnlich ist, kann man sehr einfach auf Konzepte der Datenbankkommunikation zurückgreifen, was auch der Grund ihrer sehr weiten Verbreitung ist.

2.2 Mysql

Mysql ist eine Open Source Datenbank, welche von Sun Microsystems entwickelt wurde. Aktuell wird sie von Oracel, dem neuen Eigentümer von Sun Microsystems weiterentwickelt. Auch eine grosse Community beteiligt sich an der Entwicklung von Mysql, welche nebem Postgresql zu einer der meist verbreitetsten Dantenbanken gehört. Die Sprache von Mysql ist grossteils SQL-Standart, was einer der grossen Vorteile gegenüber anderen Datenbanken wie z.B. MS SQL usw ist.

2.3 Ajax

Ajax ist eine noch nicht lange bestehende Technik, welche sich vor allem im Web 2.0 entwickelt und verbreitet hat. Es erlaubt sogenannte HTTP_REQUESTS (anfragen an den Webserver) auch dann noch, wenn die Seite schon vollständig geladen ist. Das erlaub noch dynamischere Webseiten, schnellere Reaktion auf Benutzereingaben und vor allem wird der Netzwerkverkehr reduziert.

2.4 Smarty

Smarty ist eine template Engine der neueren Generation in der Version 3, welche es erlaubt, die Darstellung von Webseiten zu generieren und somit das Konzept der strikten Trennung von Verarbeitung und Darstellung gut umzusetzen hilft. Die template Engine selber basiert auf Php, sodass keine alternative Software installiert werden muss, um diese zu verwenden.

2.5 SHA-256

SHA (Secure Hash Algorithm) 256 ist eine Version der SHA-2, dem Nachfolger von SHA 1, welcher auch in der Kryptografie zum Verschlüsseln von Nachrichten verwendet wird. Er wurde von NIST und NSA zusammen entwickelt, nachdem 2006 eine erhebliche Schwäche in SHA1 gefunden und publiziert wurde. SHA - 2 hat die Eigenschaft, dass er 2^{128} Bit grosse Daten auf eine 256 Bit grosse Prüfsumme reduziert, und somit in der Informatik eine sehr Hilfreiche Eigenschaft besitzt (z.B. one-way, claw-free etc.).

Er wird vor allem zum Prüfen von Daten, zum Speichern von Passwörtern und ähnlichem verwendet.

2.6 UTF8

UTF8 ist ein gängiger UNIX Zeichensatz, welcher die meisten Sonderzeichen der europäischen Sprachen beinhaltet. Viele Windows Anwendungen unterstützen diesen Zeichensatz noch nicht.

3 Analyse

Die Dateien werden so abgespeichert, dass mit so wenigen Änderungen wie möglich das Programm auf die Verwendung in einer anderen Schule angepasst werden kann. Da die Gewerbeoberschule durch das Präsentsein von Co-Präsenzen eine sehr komplexe Schule ist, bzw das maximale an Komplexität hat, was eine Schule haben kann, kann das Programm problemlos in anderen Schulen eingesetzt werden.

3.1 Benutzer

3.1.1 Administrator

Ein Administrator ist für die Verwaltung der Webanwendung zuständig, er muss dafür sorgen, dass die Semester geändert werden, dass der Zugang zu Semesterende zur richtigen Zeit gesperrt wird, dass die Dateien alle exportiert werden. Er übernimmt auch die Benutzerverwaltung, indem er Benutzer erstellt, die Verknüpfungen zwischen Schülern, Lehrern, Klassen, Klassenlehrern usw. herstellt.

Der Administrator hat auch das Recht, bei Anfrage das Passwort eines Benutzers zu ändern. Sein Passwort sollte er eigentlich nicht vergessen.

3.1.2 Lehrperson

Eine Lehrperson kann Noten und Absenzstunden eingeben. Die Absenzstunden werden jeweils vom Theorielehrer eingetragen, welcher dann auch die endgültige Note für das 2. Semester eingibt. Der Praxislehrer kann seine Noten nur im ersten Semester eingeben. Diese Eingaben werden von der Eingabedatei gesteuert, die vor dem Eingeben vom Administrator in das Programm eingespeist wird und somit das Programm konfiguriert.

3.2 Datenbank

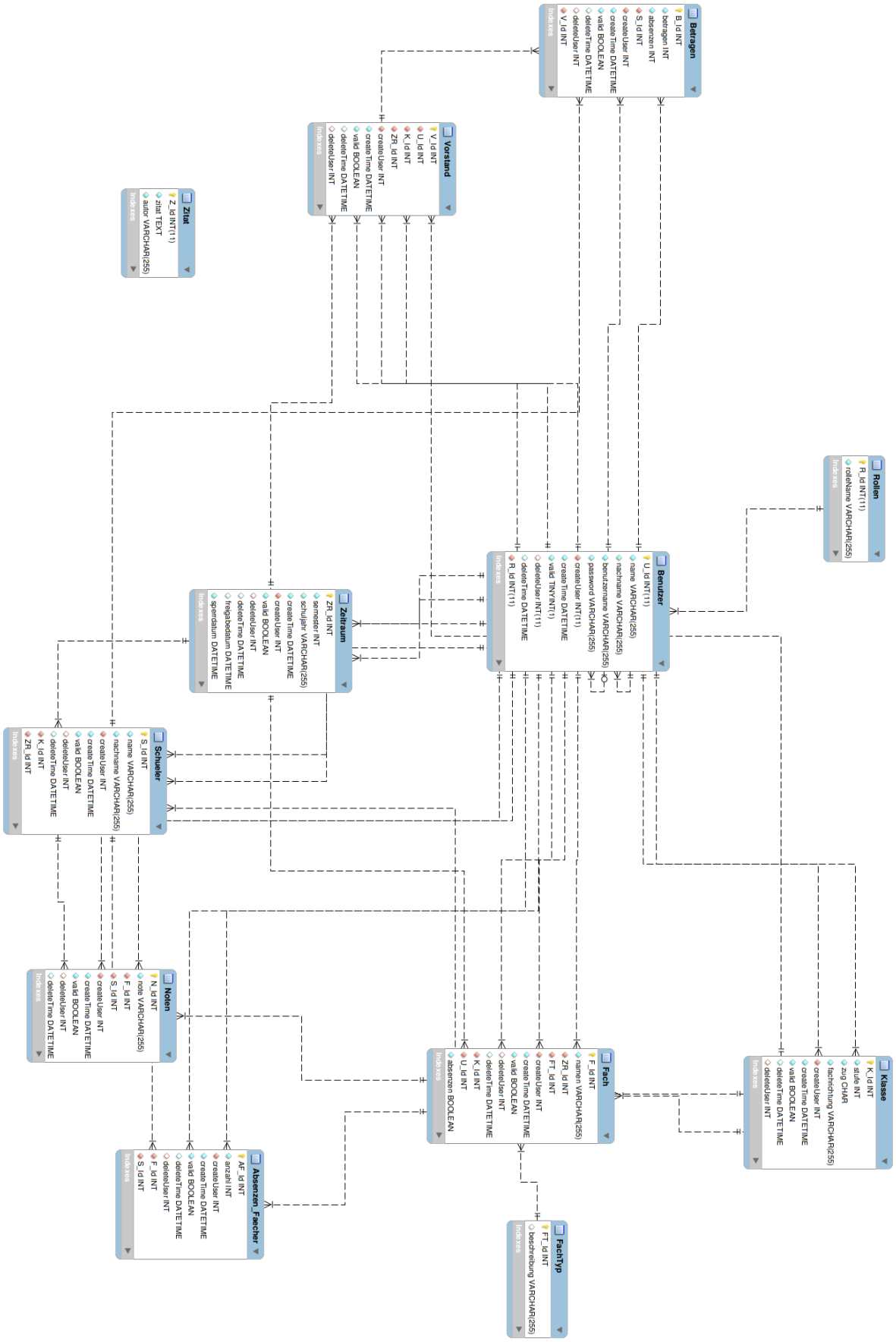
Die Datenbank wird mehrere Tabellen umfassen, in welcher die verschiedensten Entitäten abgebildet werden. Auch die Konfigurationen werden in der Datenbank abgespeichert, da auf das Dateisystem kein Schreibzugriff der Sicherheit wegen passieren darf. Das Passwort wird als Hash abgespeichert, sodass nur ein Vergleich mit dem eingegebenen konvertiertem Passwort ein Ergebnis bringen kann, das Rückkonvertieren des Passworts ist nicht möglich.

4 Planung

4.1 Datenbank

Die Datenbank wird wie im folgenden Schema aufgezeigt realisiert. Diese Realisierung bringt mit sich, dass man eine konsistente Datenbank besitzt, und dass jeder Eintrag eindeutig einem Benutzer zugeordnet werden kann. Somit ist gewährleistet, dass jede Änderung im System nachvollzogen werden kann.

Die Datenbank läuft standardmäßig mit dem Zeichensatz UTF8, so wie auch die Verbindung immer in UTF8 geöffnet wird. Damit wird gewährleistet, dass Sonderzeichen wie Umlaute usw korrekt dargestellt werden können.


Betragen
B_Id INT
betragen INT
absenzen INT
S_Id INT
create User INT
create Time DATETIME
valid BOOLEAN
deleteTime DATETIME
deleteUser INT
V_Id INT
Vorstand
V_Id INT
U_Id INT
K_Id INT
ZR_Id INT
createUser INT
createTime DATETIME
valid BOOLEAN
deleteTime DATETIME
deleteUser INT
Ziltat
Z_Id INT(11)
ziltat TEXT
autor VARCHAR(255)
Rollen
R_Id INT(11)
roleName VARCHAR(255)
Benutzer
U_Id INT(11)
name VARCHAR(255)
nachname VARCHAR(255)
benutzername VARCHAR(255)
passwort VARCHAR(255)
createUser INT(11)
createTime DATETIME
valid TINYINT(1)
deleteUser INT(11)
deleteTime DATETIME
R_Id INT(11)
Zeitraum
ZR_Id INT
semester INT
schuljahr VARCHAR(255)
createTime DATETIME
valid BOOLEAN
createUser INT
deleteUser INT
deleteTime DATETIME
freigabedatum DATETIME
sperrdatum DATETIME
Schueler
S_Id INT
name VARCHAR(255)
nachname VARCHAR(255)
createUser INT
createTime DATETIME
valid BOOLEAN
deleteUser INT
deleteTime DATETIME
K_Id INT
ZR_Id INT
Noten
N_Id INT
note VARCHAR(255)
F_Id INT
S_Id INT
createUser INT
createTime DATETIME
valid BOOLEAN
deleteUser INT
deleteTime DATETIME
Fach
F_Id INT
namen VARCHAR(255)
ZR_Id INT
FT_Id INT
createUser INT
createTime DATETIME
valid BOOLEAN
deleteUser INT
deleteTime DATETIME
K_Id INT
U_Id INT
absenzen BOOLEAN
Klasse
K_Id INT
stufe INT
zug CHAR
fachrichtung VARCHAR(255)
createUser INT
createTime DATETIME
valid BOOLEAN
deleteTime DATETIME
deleteUser INT
Absenzen_Faecher
AF_Id INT
anzahl INT
createUser INT
createTime DATETIME
valid BOOLEAN
deleteTime DATETIME
deleteUser INT
F_Id INT
S_Id INT
FachTyp
FT_Id INT
beschreibung VARCHAR(255)

4.2 Webanwendung

5 Softwaretechnische Umsetzung

Die gesamte Webanwendung wird in PHP ausprogrammiert, als Datenbank wird MySQL verwendet, wegen seiner Kompatibilität zu PHP und aufgrund seiner Open Source Lizenz, welche es erlaubt, den Server uneingeschränkt und ohne Kosten zu nutzen.

Das bewährte Konzept der Trennung von Verarbeitung und Darstellung wird mit Hilfe von Smarty implementiert (www.smarty.net).

Kurzum, die Anwendung besteht aus 4 Teilbereiche, die Datenbank, die Klassen, die Verarbeitungsskripte und die Smarty-Templates.

5.1 Datenbank

```
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';


CREATE SCHEMA IF NOT EXISTS `notenverwaltung_v2_1` DEFAULT CHARACTER SET utf8 COLLATE
utf8_general_ci ;
USE `notenverwaltung_v2_1` ;


-- -----------------------------------------------------
-- Table `notenverwaltung_v2_1`.`Rollen`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `notenverwaltung_v2_1`.`Rollen` (
  `R_Id` INT(11) NOT NULL AUTO_INCREMENT ,
  `rolleName` VARCHAR(255) NOT NULL ,
  PRIMARY KEY (`R_Id`) )
ENGINE = InnoDB
AUTO_INCREMENT = 4
DEFAULT CHARACTER SET = utf8
COLLATE = utf8_general_ci;



-- -----------------------------------------------------
-- Table `notenverwaltung_v2_1`.`Benutzer`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `notenverwaltung_v2_1`.`Benutzer` (
  `U_Id` INT(11) NOT NULL AUTO_INCREMENT ,
  `name` VARCHAR(255) NOT NULL ,
  `nachname` VARCHAR(255) NOT NULL ,
  `benutzername` VARCHAR(255) NOT NULL ,
```



```
  `password` VARCHAR(255) NOT NULL ,
  `createUser` INT(11) NOT NULL ,
  `createTime` DATETIME NOT NULL ,
  `valid` TINYINT(1) NOT NULL ,
  `deleteUser` INT(11) NULL DEFAULT NULL ,
  `deleteTime` DATETIME NULL DEFAULT NULL ,
  `R_Id` INT(11) NOT NULL ,
  PRIMARY KEY (`U_Id`) ,
  INDEX `fk_Benutzer_1` (`R_Id` ASC) ,
  INDEX `fk_Benutzer_2` (`createUser` ASC) ,
  INDEX `fk_Benutzer_3` (`deleteUser` ASC) ,
  UNIQUE INDEX `fk_Benutzer_4` (`benutzername` ASC, `valid` ASC, `deleteUser` ASC, `deleteTime`
ASC) ,
  CONSTRAINT `fk_Benutzer_1`
    FOREIGN KEY (`R_Id` )
    REFERENCES `notenverwaltung_v2_1`.`Rollen` (`R_Id` )
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Benutzer_2`
    FOREIGN KEY (`createUser` )
    REFERENCES `notenverwaltung_v2_1`.`Benutzer` (`U_Id` )
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Benutzer_3`
    FOREIGN KEY (`deleteUser` )
    REFERENCES `notenverwaltung_v2_1`.`Benutzer` (`U_Id` )
    ON UPDATE CASCADE)
ENGINE = InnoDB
AUTO_INCREMENT = 7
DEFAULT CHARACTER SET = utf8
COLLATE = utf8_general_ci;



-- -----------------------------------------------------
-- Table `notenverwaltung_v2_1`.`Zitat`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `notenverwaltung_v2_1`.`Zitat` (
  `Z_Id` INT(11) NOT NULL AUTO_INCREMENT ,
  `zitat` TEXT NOT NULL ,
  `autor` VARCHAR(255) NOT NULL ,
  PRIMARY KEY (`Z_Id`) )
ENGINE = InnoDB
AUTO_INCREMENT = 10707
DEFAULT CHARACTER SET = utf8
COLLATE = utf8_general_ci;
```



```
-- -----------------------------------------------------
-- Table `notenverwaltung_v2_1`.`FachTyp`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `notenverwaltung_v2_1`.`FachTyp` (
  `FT_Id` INT NOT NULL AUTO_INCREMENT ,
  `beschreibung` VARCHAR(255) NULL ,
  PRIMARY KEY (`FT_Id`) )
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `notenverwaltung_v2_1`.`Zeitraum`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `notenverwaltung_v2_1`.`Zeitraum` (
  `ZR_Id` INT NOT NULL AUTO_INCREMENT ,
  `semester` INT NOT NULL ,
  `schuljahr` VARCHAR(255) NOT NULL ,
  `createTime` DATETIME NOT NULL ,
  `createUser` INT NOT NULL ,
  `valid` TINYINT(1)  NOT NULL ,
  `deleteUser` INT NULL ,
  `deleteTime` DATETIME NULL ,
  `freigabedatum` DATETIME NULL ,
  `sperrdatum` DATETIME NOT NULL ,
  PRIMARY KEY (`ZR_Id`) ,
  INDEX `fk_Zeitraum_1` (`createUser` ASC) ,
  INDEX `fk_Zeitraum_2` (`deleteUser` ASC) ,
  UNIQUE INDEX `fk_Zeitraum_3` (`semester` ASC, `schuljahr` ASC, `valid` ASC, `deleteUser` ASC,
`deleteTime` ASC) ,
  CONSTRAINT `fk_Zeitraum_1`
    FOREIGN KEY (`createUser` )
    REFERENCES `notenverwaltung_v2_1`.`Benutzer` (`U_Id` )
    ON DELETE RESTRICT
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Zeitraum_2`
    FOREIGN KEY (`deleteUser` )
    REFERENCES `notenverwaltung_v2_1`.`Benutzer` (`U_Id` )
    ON DELETE RESTRICT
    ON UPDATE CASCADE)
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `notenverwaltung_v2_1`.`Klasse`
-- -----------------------------------------------------
```



```
CREATE  TABLE IF NOT EXISTS `notenverwaltung_v2_1`.`Klasse` (
  `K_Id` INT NOT NULL AUTO_INCREMENT ,
  `stufe` INT NOT NULL ,
  `zug` CHAR NOT NULL ,
  `fachrichtung` VARCHAR(255) NOT NULL ,
  `createUser` INT NOT NULL ,
  `createTime` DATETIME NOT NULL ,
  `valid` TINYINT(1)  NOT NULL ,
  `deleteTime` DATETIME NULL ,
  `deleteUser` INT NULL ,
  PRIMARY KEY (`K_Id`) ,
  INDEX `fk_Klasse_1` (`createUser` ASC) ,
  INDEX `fk_Klasse_2` (`deleteUser` ASC) ,
  CONSTRAINT `fk_Klasse_1`
    FOREIGN KEY (`createUser` )
    REFERENCES `notenverwaltung_v2_1`.`Benutzer` (`U_Id` )
    ON DELETE RESTRICT
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Klasse_2`
    FOREIGN KEY (`deleteUser` )
    REFERENCES `notenverwaltung_v2_1`.`Benutzer` (`U_Id` )
    ON DELETE RESTRICT
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `notenverwaltung_v2_1`.`Fach`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `notenverwaltung_v2_1`.`Fach` (
  `F_Id` INT NOT NULL AUTO_INCREMENT ,
  `namen` VARCHAR(255) NOT NULL ,
  `ZR_Id` INT NOT NULL ,
  `FT_Id` INT NOT NULL ,
  `createUser` INT NOT NULL ,
  `createTime` DATETIME NOT NULL ,
  `valid` TINYINT(1)  NOT NULL ,
  `deleteUser` INT NULL ,
  `deleteTime` DATETIME NULL ,
  `K_Id` INT NOT NULL ,
  `U_Id` INT NOT NULL ,
  `absenzen` TINYINT(1)  NOT NULL DEFAULT 0 ,
  PRIMARY KEY (`F_Id`) ,
  INDEX `fk_Fach_1` (`FT_Id` ASC) ,
  INDEX `fk_Fach_2` (`createUser` ASC) ,
```

```
  INDEX `fk_Fach_3` (`deleteUser` ASC) ,
  INDEX `fk_Fach_4` (`ZR_Id` ASC) ,
  INDEX `fk_Fach_5` (`K_Id` ASC) ,
  INDEX `fk_Fach_6` (`U_Id` ASC) ,
  CONSTRAINT `fk_Fach_1`
    FOREIGN KEY (`FT_Id` )
    REFERENCES `notenverwaltung_v2_1`.`FachTyp` (`FT_Id` )
    ON DELETE RESTRICT
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Fach_2`
    FOREIGN KEY (`createUser` )
    REFERENCES `notenverwaltung_v2_1`.`Benutzer` (`U_Id` )
    ON DELETE RESTRICT
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Fach_3`
    FOREIGN KEY (`deleteUser` )
    REFERENCES `notenverwaltung_v2_1`.`Benutzer` (`U_Id` )
    ON DELETE RESTRICT
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Fach_4`
    FOREIGN KEY (`ZR_Id` )
    REFERENCES `notenverwaltung_v2_1`.`Zeitraum` (`ZR_Id` )
    ON DELETE RESTRICT
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Fach_5`
    FOREIGN KEY (`K_Id` )
    REFERENCES `notenverwaltung_v2_1`.`Klasse` (`K_Id` )
    ON DELETE RESTRICT
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Fach_6`
    FOREIGN KEY (`U_Id` )
    REFERENCES `notenverwaltung_v2_1`.`Benutzer` (`U_Id` )
    ON DELETE RESTRICT
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `notenverwaltung_v2_1`.`Schueler`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `notenverwaltung_v2_1`.`Schueler` (
  `S_Id` INT NOT NULL AUTO_INCREMENT ,
  `name` VARCHAR(255) NOT NULL ,
  `nachname` VARCHAR(255) NOT NULL ,
  `createUser` INT NOT NULL ,
```

```
  `createTime` DATETIME NOT NULL ,
  `valid` TINYINT(1)  NOT NULL ,
  `deleteUser` INT NULL ,
  `deleteTime` DATETIME NULL ,
  `K_Id` INT NOT NULL ,
  `ZR_Id` INT NOT NULL ,
  PRIMARY KEY (`S_Id`) ,
  INDEX `fk_Schueler_1` (`createUser` ASC) ,
  INDEX `fk_Schueler_2` (`deleteUser` ASC) ,
  INDEX `fk_Schueler_3` (`K_Id` ASC) ,
  INDEX `fk_Schueler_4` (`ZR_Id` ASC) ,
  CONSTRAINT `fk_Schueler_1`
    FOREIGN KEY (`createUser` )
    REFERENCES `notenverwaltung_v2_1`.`Benutzer` (`U_Id` )
    ON DELETE RESTRICT
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Schueler_2`
    FOREIGN KEY (`deleteUser` )
    REFERENCES `notenverwaltung_v2_1`.`Benutzer` (`U_Id` )
    ON DELETE RESTRICT
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Schueler_3`
    FOREIGN KEY (`K_Id` )
    REFERENCES `notenverwaltung_v2_1`.`Klasse` (`K_Id` )
    ON DELETE RESTRICT
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Schueler_4`
    FOREIGN KEY (`ZR_Id` )
    REFERENCES `notenverwaltung_v2_1`.`Zeitraum` (`ZR_Id` )
    ON DELETE RESTRICT
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `notenverwaltung_v2_1`.`Vorstand`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `notenverwaltung_v2_1`.`Vorstand` (
  `V_Id` INT NOT NULL AUTO_INCREMENT ,
  `U_Id` INT NOT NULL ,
  `K_Id` INT NOT NULL ,
  `ZR_Id` INT NOT NULL ,
  `createUser` INT NOT NULL ,
  `createTime` DATETIME NOT NULL ,
  `valid` TINYINT(1)  NOT NULL ,
```



```
  `deleteTime` DATETIME NULL ,
  `deleteUser` INT NULL ,
  PRIMARY KEY (`V_Id`) ,
  INDEX `fk_Vorstand_1` (`U_Id` ASC) ,
  INDEX `fk_Vorstand_2` (`createUser` ASC) ,
  INDEX `fk_Vorstand_3` (`deleteUser` ASC) ,
  INDEX `fk_Vorstand_4` (`K_Id` ASC) ,
  INDEX `fk_Vorstand_5` (`ZR_Id` ASC) ,
  CONSTRAINT `fk_Vorstand_1`
    FOREIGN KEY (`U_Id` )
    REFERENCES `notenverwaltung_v2_1`.`Benutzer` (`U_Id` )
    ON DELETE RESTRICT
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Vorstand_2`
    FOREIGN KEY (`createUser` )
    REFERENCES `notenverwaltung_v2_1`.`Benutzer` (`U_Id` )
    ON DELETE RESTRICT
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Vorstand_3`
    FOREIGN KEY (`deleteUser` )
    REFERENCES `notenverwaltung_v2_1`.`Benutzer` (`U_Id` )
    ON DELETE RESTRICT
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Vorstand_4`
    FOREIGN KEY (`K_Id` )
    REFERENCES `notenverwaltung_v2_1`.`Klasse` (`K_Id` )
    ON DELETE RESTRICT
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Vorstand_5`
    FOREIGN KEY (`ZR_Id` )
    REFERENCES `notenverwaltung_v2_1`.`Zeitraum` (`ZR_Id` )
    ON DELETE RESTRICT
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `notenverwaltung_v2_1`.`Absenzen_Faecher`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `notenverwaltung_v2_1`.`Absenzen_Faecher` (
  `AF_Id` INT NOT NULL AUTO_INCREMENT ,
  `anzahl` INT NOT NULL ,
  `createUser` INT NOT NULL ,
  `createTime` DATETIME NOT NULL ,
  `valid` TINYINT(1)  NOT NULL ,
```



```
  `deleteTime` DATETIME NULL ,
  `deleteUser` INT NULL ,
  `F_Id` INT NOT NULL ,
  `S_Id` INT NOT NULL ,
  PRIMARY KEY (`AF_Id`) ,
  INDEX `fk_Absenzen_Faecher_1` (`createUser` ASC) ,
  INDEX `fk_Absenzen_Faecher_2` (`deleteUser` ASC) ,
  INDEX `fk_Absenzen_Faecher_3` (`F_Id` ASC) ,
  INDEX `fk_Absenzen_Faecher_4` (`S_Id` ASC) ,
  CONSTRAINT `fk_Absenzen_Faecher_1`
    FOREIGN KEY (`createUser` )
    REFERENCES `notenverwaltung_v2_1`.`Benutzer` (`U_Id` )
    ON DELETE RESTRICT
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Absenzen_Faecher_2`
    FOREIGN KEY (`deleteUser` )
    REFERENCES `notenverwaltung_v2_1`.`Benutzer` (`U_Id` )
    ON DELETE RESTRICT
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Absenzen_Faecher_3`
    FOREIGN KEY (`F_Id` )
    REFERENCES `notenverwaltung_v2_1`.`Fach` (`F_Id` )
    ON DELETE RESTRICT
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Absenzen_Faecher_4`
    FOREIGN KEY (`S_Id` )
    REFERENCES `notenverwaltung_v2_1`.`Schueler` (`S_Id` )
    ON DELETE RESTRICT
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `notenverwaltung_v2_1`.`Noten`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `notenverwaltung_v2_1`.`Noten` (
  `N_Id` INT NOT NULL AUTO_INCREMENT ,
  `note` VARCHAR(255) NOT NULL ,
  `F_Id` INT NOT NULL ,
  `S_Id` INT NOT NULL ,
  `createUser` INT NOT NULL ,
  `createTime` DATETIME NOT NULL ,
  `valid` TINYINT(1)  NOT NULL ,
  `deleteUser` INT NULL ,
  `deleteTime` DATETIME NULL ,
```



```
  PRIMARY KEY (`N_Id`) ,
  INDEX `fk_Noten_1` (`createUser` ASC) ,
  INDEX `fk_Noten_2` (`deleteUser` ASC) ,
  INDEX `fk_Noten_3` (`F_Id` ASC) ,
  INDEX `fk_Noten_4` (`S_Id` ASC) ,
  CONSTRAINT `fk_Noten_1`
    FOREIGN KEY (`createUser` )
    REFERENCES `notenverwaltung_v2_1`.`Benutzer` (`U_Id` )
    ON DELETE RESTRICT
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Noten_2`
    FOREIGN KEY (`deleteUser` )
    REFERENCES `notenverwaltung_v2_1`.`Benutzer` (`U_Id` )
    ON DELETE RESTRICT
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Noten_3`
    FOREIGN KEY (`F_Id` )
    REFERENCES `notenverwaltung_v2_1`.`Fach` (`F_Id` )
    ON DELETE RESTRICT
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Noten_4`
    FOREIGN KEY (`S_Id` )
    REFERENCES `notenverwaltung_v2_1`.`Schueler` (`S_Id` )
    ON DELETE RESTRICT
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `notenverwaltung_v2_1`.`Betragen`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `notenverwaltung_v2_1`.`Betragen` (
  `B_Id` INT NOT NULL AUTO_INCREMENT ,
  `betragen` INT NOT NULL ,
  `absenzen` INT NOT NULL ,
  `S_Id` INT NOT NULL ,
  `createUser` INT NOT NULL ,
  `createTime` DATETIME NOT NULL ,
  `valid` TINYINT(1)  NOT NULL ,
  `deleteTime` DATETIME NULL ,
  `deleteUser` INT NULL ,
  `V_Id` INT NOT NULL ,
  PRIMARY KEY (`B_Id`) ,
  INDEX `fk_Betragen_1` (`createUser` ASC) ,
  INDEX `fk_Betragen_2` (`deleteUser` ASC) ,
```

```
    INDEX `fk_Betragen_3` (`S_Id` ASC) ,
    INDEX `fk_Betragen_4` (`V_Id` ASC) ,
    CONSTRAINT `fk_Betragen_1`
      FOREIGN KEY (`createUser` )
      REFERENCES `notenverwaltung_v2_1`.`Benutzer` (`U_Id` )
      ON DELETE RESTRICT
      ON UPDATE CASCADE,
    CONSTRAINT `fk_Betragen_2`
      FOREIGN KEY (`deleteUser` )
      REFERENCES `notenverwaltung_v2_1`.`Benutzer` (`U_Id` )
      ON DELETE RESTRICT
      ON UPDATE CASCADE,
    CONSTRAINT `fk_Betragen_3`
      FOREIGN KEY (`S_Id` )
      REFERENCES `notenverwaltung_v2_1`.`Schueler` (`S_Id` )
      ON DELETE RESTRICT
      ON UPDATE CASCADE,
    CONSTRAINT `fk_Betragen_4`
      FOREIGN KEY (`V_Id` )
      REFERENCES `notenverwaltung_v2_1`.`Vorstand` (`V_Id` )
      ON DELETE RESTRICT
      ON UPDATE CASCADE)
  ENGINE = InnoDB;



  SET SQL_MODE=@OLD_SQL_MODE;
  SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
  SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

5.2 Webanwendung

5.2.1 data

In data werden eventuell benötigte Dokumente abgelegt, die einen geschützten Zugriff benötigen, und somit nicht im Ordner www (dem vom Web aus Zugänglichem) abgelegt werden können.

5.2.2 etc

In etc befinden sich die Konfigurationsdateien der Webanwendung. Durch diese kann man das Verhalten der Webanwendung beinflussen.

5.2.2.1 conf.xml

In dieser Datei werden im XML – Format die Parameter der Webanwendung festgehalten. Unter anderem werden hier die Länge der Session, das Logintimeout, der Datenbankzugriff usw. eingetragen.

5.2.3 includes

5.2.3.1 classes

In diesem Ordner werden die Klassen abgespeichert. Die Klassen sind Objekte, die großteils ihre Informationen aus der Datenbank holen. Dazu gibt es die Beiden Interfaces DBObject und DBObjectList. Diese haben die Felder indentNumber, createTime, createUser, valid, deleteTime, deleteUser schon implementiert, sodass diese in den anderen Klassen nicht mehr ausprogrammiert werden müssen.

Gleichzeitig finden sich hier auch die Klassen für das Funktionieren der Template-Engine Smarty und die Klasse, welche die Konfigurationsdatei etc/Conf.xml ausliest.

5.2.3.2 modules

In modules werden die Skripte gespeichert, welche das Funktionieren der Webanwendung implementieren. Sie holen sich jeweils die Listen aus der Datenbank mit den verschiedenen Objekte und tragen neue Objekte in die Datenbank ein. Diese Dateien werden von der Anwendung über den Parameter page=<dateinameohne“.inc.php“> aufgerufen. Die Ausgabe der Datei erfolgt über eine .tpl Datei, welche am Ende des Skriptes angegeben wird.

5.2.3.2.1 private

In diesem Ordner sind alle Module gespeichert, welche nach der Anmeldung des Benutzers zur Verarbeitung gebraucht werden. Jedes Modul beginnt mit der Abfrage der Rechte, sodass garantiert wird, dass auch bei einem Aufruf über Parameter die korrekte Funktion der Anwendung garantiert ist. Die Dateiendung der hier enthaltenen Dateien ist .inc.php. Diese Dateien werden von der Webanwendung durch den Parameter page=private.<dateinameohne“.inc.php“> aufgerufen. Auch hier erfolgt die Ausgabe über eine Template Datei, welche am Ende des Skriptes angegeben wird.

5.2.3.2.1.1 ajax

Hierin befinden sich alle Dateien, welche über die Technologie Ajax angesprochen werden können. Diese Dateien enden mit einem exit und geben unterschiedliche Daten zurück, je nachdem für welche Zwecke das Skript gebraucht wird.

5.2.4 lib

Im Ordner lib werden die Bibliotheken welche für die Anwendung gebraucht werden abgelegt. In dieser Anwendung befindet sich hier der Ordner Smarty3, welcher die Dateien der template Engine enthält.

5.2.5 logs

Hier wird die Logdatei des Webservers abgelegt. So kann diese vom Besitzer der Webseite eingesehen werden. Sie birgt oft interessante Details zum Zugriff, aber auch wertvolle Informationen bei der Fehlersuche.

5.2.6 smarty

Dieser Ordner enthält 2 Unterordner, templates und templates_c.

5.2.6.1 templates

Dieser Ordner enthaltet die Templatedateien, in welchen die Darstellung in einer Mischung aus HTML und Smarty-Befehlen geschrieben sind. auch sie sind in Ordnern gegliedert.

Die Datei main.tpl ist die Haupttemplatedatei, welche die Struktur der Seite darstellt.

5.2.6.1.1 errors

In errors sind die Templates abgelegt, welche für das Darstellen der Fehler hergenommen werden.

5.2.6.1.2 private

In private befinden sich die Templates, welche nach dem Anmelden für das Darstellen der Informationen dienen.

Im speziellen gibt es 2 Templates, welche alleine aufgerufen werden müssen, und nicht in main.tpl includiert werden dürfen, nämlich print.tpl und print_betragen.tpl. Diese Beiden erstellen eine Ausgabe, welche extra für den Druck vorbereitet wird.

5.2.6.1.2.1 ajax

In ajax befinden sich die Templates, welche bei Ajaxrequests gebruacht werden. Je nach Anforderung geben sie verschiedene Informationen zurück.

5.2.6.2 templates_c

Hier speichert Smarty die erzeugten Templatedateien ab. Das bringt vor allem Vorteile bei der Geschwindigkeit, welche dadurch optimiert wird. Dieser Ordner muss vom Webserver beschreibbar sein.

5.2.7 www

www ist der Ordner, welche vom Web aus zugänglich ist. er beinhaltet die index.php-Datei, die stylesheets, den Bilderordner, und Dokumente, welche ohne Zugriffsbeschränkungen zugänglich sind.

6 Benutzerschnittstellen

Im folgenden werden die Benutzerschnittstellen beschrieben, und auch mit Bildern unterlegt.

6.1 Allgemein

6.1.1 Startseite

Hier kann sich die Person anmelden, mit ihrem Benutzernamen und mit dem dazugehörigen Passwort. Die Person wird dann je nach Rolle, welche sie von der Administration zugeteilt bekommen hat auf die entsprechende Seite weitergeleitet.

6.2 Administrator

Ein Administrator kann die Teilnehmenden Lehrpersonen, ihre Fächer usw. mittels Dateien in die Datenbank eintragen. Dazu muss aber folgende Reihenfolge unbedingt eingehalten werden:

1. Lehrer importieren
2. Lehrer-Fächer importieren
3. Schüler importieren

Der Aufbau der Import-Dateien ist weiter unten beschrieben.

6.2.1 Menüpunkte und Eintragen

Nachfolgend werden die Aktionen aufgezählt, welche dem Administrator zur Verfügung stehen:

- Lehrer importieren
- Fächer importieren
- Schüler importieren
- Export Klassen
- Benutzerverwaltung
- Schülerliste
- Fächerliste
- Semesterverwaltung

6.2.1.1 Eingangsdatei Lehrer

Diese Datei ist in Zeilen und Spalten aufgebaut. Jede Zeile steht für einen bestimmte Lehrperson mit ihren Eigenschaften. Die verschiedenen Spalten müssen durch ein eindeutiges Trennzeichen getrennt werden, und müssen in folgender Reihenfolge folgende Informationen enthalten:

1. Nachname der Lehrperson
2. Vorname der Lehrperson
3. Zugangsname/Login
4. Passwort

6.2.1.2 Eingangsdatei Lehrer-Fächer

Diese Datei ist in Zeilen und Spalten aufgebaut. Jede Zeile steht für einen bestimmte Lehrperson mit ihrem Fach. Falls die Lehrperson Vorsitzende des Klassenrates ist, so muss bei einem der Fächer in dieser Klasse in der dazugehörigen Spalte anstatt eine 0 oder nichts eine 1 eingetragen werden. Für die Notenvergabe im Fach, ob schriftlich, mündlich praktisch muss jeweils eine 1 für vorhanden, oder eine 0 für nicht vorhanden in der Spalte eingetragen werden. Eine leere Spalte wird als 0 interpretiert. Wenn ein Lehrer Klassenvorstand ist, so genügt einmal das Eintragen einer 1 bei einem Fach in dieser Klasse. Ist der Lehrer in diesem Fach auch zuständig für das Eintragen der Absenzen, so muss auch hier eine 1 eingetragen werden. In der Fächertabelle des 2. Semester steht somit automatisch bei jedem Fach eine 1, im ersten Semester nur bei einigen Lehrern. Dieses Flag wird benötigt , da es informationstechnisch fast unmöglich ist, eine Gesetzmäßigkeit bei dieser Art der Eintragung zu finden.

Die Klasse wird nicht als ganzes Angegeben, sprich eine 1. Klasse, im Zug A wird hier als 1;Biennium;A eingetragen, eine 5. Klasse Informatik im Zug A wird als 5;Informatik;A eingetragen. Die Reihenfolge der Fächer muss aber die Reihenfolge der Fächer in der Poppcorn-Tabelle widerspiegeln, da ansonsten diese willkürliche Reihenfolge nicht eingehalten werden kann.

Die verschiedenen Spalten müssen durch ein eindeutiges Trennzeichen getrennt werden, und müssen in folgender Reihenfolge folgende Informationen enthalten:

1. Nachname der Lehrperson
2. Vorname der Lehrperson
3. Name des Faches
4. schriftlich
5. mündlich
6. praktisch
7. Klassenratsvorsitzender

8. Absenzen eintragen
9. Klassenstufe (Nummer)
10. Fachrichtung (Text)
11. Klassenzug (Buchstabe)

6.2.1.3 Eingangsdatei Schüler

Diese Datei ist in Zeilen und Spalten aufgebaut. Jede Zeile steht für einen bestimmten Schüler mit seinen Eigenschaften. Die verschiedenen Spalten müssen durch ein eindeutiges Trennzeichen getrennt werden, und müssen in folgender Reihenfolge folgende Informationen enthalten:

1. Nachname des Schülers
2. Vorname des Schülers
3. Klassenstufe (Nummer)
4. Fachrichtung (Text)
5. Klassenzug (Buchstabe)

6.2.2 Passwörter von Lehrpersonen ändern

Das Passwort kann einfach geändert werden, indem der Administrator auf den dafür vorgesehenen Link klickt. Es wird dann ein 10-stelliges Passwort zufällig generiert (Klein-, Großbuchstaben und Zahlen), welches dann auch nach der Änderung angezeigt wird. Es wird nur einmal angezeigt, nach diesem Anzeigen ist das Passwort nicht mehr rückführbar, und muss falls es nicht gemerkt oder abgeschrieben wurde wieder neu gesetzt werden. Passwörter von anderen Administratoren können nicht geändert werden.

6.3 Lehrperson

Eine Lehrperson kann Noten und Absenzen eingeben, und falls diese auch noch Klassenvorstand ist, so kann sie auch die Betragensnote und die unentschuldigten Absenzen eingeben.

6.3.1 Passwortänderung

Die Lehrperson kann ihr eigenes Passwort ändern, indem sie das alte Passwort in das dazugehörige Formular unter „Passwort ändern“ eingibt, und anschließend das neue Passwort 2 mal korrekt und übereinstimmend eingibt. Die Änderung wird von derselben Seite entweder bestätigt, oder eine Fehlermeldung wird ausgegeben.

7 Installation des Systems

7.1 Insellösung

7.1.1 Hardware

Pc HP D530

Bildschirm HP LP1985

Drucker HP Laserjet 2015n

7.1.2 Software

Ubuntu 8.10, Grundinstallation mit grafischer Oberfläche

Apache (php5)

Mysql-server

Notenprogramm

Firefox

BMA (Brooklyn Museum Addon)
https://www.mozdevgroup.com/clients/bm/ (Kiosk Mode)

Pessulus und Sabayon

7.1.3 Grundlegende Konfigurationsschritte:

A) Ubuntu installation

B) Apache, php5, php5-mysql und mysql-server installieren

C) Drucker installieren und konfigurieren

D) Installation des Notenprogramms

E) Erstellen eines Mysql-Benutzers, importieren der dazugehörigen Mysql-Tabellen, erstellen eines Benutzers (admin) mit Password (md5) in der Lehrer-Tabelle

F) Testen des Notenprogramms auf vollständige Funktionsweise

G) System absichern:

1. auto login aktivieren auf user ohne sudo-rechte
2. Ssh beschränken auf eine ip/mac, konfiguration testen
3. bma kiosk plugin installieren und konfigurieren
   (a) drucker aktivieren,
   (b) temporäre dateien/cache aktivieren
   (c) file protokoll deaktivieren

(d) popups erlauben für 127.0.0.1

(e) Url auf 127.0.0.1 beschränken

(f) javascript aktivieren (drucken)

4. gconf-editor: /apps/gnome_settings_daemon/keybindings key "power" löschen
(ctrl-alt-entf)
5. Pessulus und Sabayon (http://www.linux-magazin.de/Heft-Abo/Ausgaben/2006/07/Kaefige-nach-Mass)

   nähere Infos und Ressourcen unter
   http://library.gnome.org/admin/deployment-guide/

H) System testen:

1. Import Schüler/Lehrer/Fächer von externen Laptop (http)
2. Zugriff über ssh (externen Laptop)
3. Zugang Lehrer, Eingabe Noten, Ausdruck Noten (http)
4. Export Daten als Admin

8 Lizenzbedingungen

8.1 Lizenzbedingungen

Diese Software basiert auf freier Software, und darf nur weitergegeben werden, sofern folgende Bedingungen erfüllt werden:

- Die Software ist gedankliches Eigentum ihrer Entwickler
- Die Software darf von jedem Verwendet werden, sofern die Autoren ihr Einverständnis gegeben haben (gilt auch für Teile der Software wie z.B. Klassen)
- Bei Vorstellung der Software muss immer der Name der Entwickler genannt werden
- Jeder darf erweitern, am Quellcode weiterbauen, optimieren etc. mit der Verpflichtung, diesen Quellcode dokumentiert und weiterhin frei zugänglich zu machen
- Es wird nicht gehaftet für Schäden, welche aufgrund einer fehlerhaften Installation usw auftreten.
- Der Support ist denen vorbehalten, welche einen Vertrag mit den Entwicklern abschließen, in welchem die weiteren Bedingungen festgehalten sind.

9 Aussicht

9.1 Wartung

Die Wartung und Aktualisierung erfolgt entweder automatisiert durch Skripts, welche von der Firma MoWiSo zur Verfügung gestellt werden, oder können auch manuell übernommen werden, wobei hier große Vorsicht geboten ist, dass man nicht die gesamte Webanwendung unstabil macht.

Die Weiterentwicklung dieser Softwareanwendung ist garantiert, solange mindestens eine Schule einen Wartungsvertrag für eine automatisierte Wartung und Aktualisierung mit der Firma MoWiSo aufrecht hat.

9.2 Weiterentwicklung

Für die Zukunft sind weitere Plugins und die effizientere Implementierung der Software geplant. So soll z.B. möglich sein, eine echte Poppcorn-Tabelle erstellen zu lassen, und schon automatisiert in das System des Schulamtes einzuspeisen (erfordert Zusammenarbeit mit Schulamt).

10 Version 1.0

10.1 Entwickler

10.1.1 Alexander Moser

Alexander ist Schüler der Gewerbeoberschule Bozen und wird im Jahr 2010 die Matura hinter sich bringen. Er war vor allem beim Planen der Datenbank am Projekt beteiligt. Er wird in Zukunft vor allem für die Fehlersuche im Quellcode, für das Suchen von Sicherheitslücken und für die Erstellung von automatisierenden Skripts zuständig sein.

10.1.2 Markus Windegger

Markus ist Schüler der Gewerbeoberschule Bozen. Er war für die Ausprogrammierung und für die Erstellung der Dokumentation zuständig, und hat das Projekt eigentlich durchgeführt, da sich Alexander im Projektzeitraum mit einem anderem Projekt beschäftigte. Er wird sich im weiterem Verlauf vor allem auf die Erweiterungen spezialisieren, und diese dann mit automatisierenden Skripts welche die Anwendung updaten zur Verfügung stellen.

10.2 Dank für die Version 1.0

Vor allem gebührt ein riesengroßer Dank dem Systemadministrator Kofler Jürgen, von dem auch die Installationshinweise des Inselrechners übernommen

wurden. Er hat uns durch das Projekt geholfen, und stand uns immer mit Rat und Tat zur Seite. Er hat uns mit seinem Wissen um die Webtechnologien und um die Sicherheitslücken im Web sehr viel mitgegeben, nicht nur für dieses Projekt, sondern auch für unsere weiteren Projekte.

11 Version 2.0

11.1 Entwickler

11.1.1 Markus Windegger

Markus Windegger ist Student an der Universität in Pisa und studiert dort Theoretische Informatik. Neben dem Studium versucht er sich immer wieder an Projekte, wie zuletzt die Notenverwaltung Version 2 oder das Literaturportal Lipo (www.lipo.bz.it). Vor seinem Studium besuchte er die Gewerbeoberschule Bozen, während der er die erste Version der Notenverwaltung zusammen mit Alexander Moser auf die Beine stellte.

11.1.1.1 Persönlicher Kommentar zur Version 2.0

Die Version 2.0 basiert auf meinen Erfahrungen, welche ich vor allem während der Sommermonate bei der SAD gemacht habe. Die Entwicklung der Datenbank, die Verwendung von Smarty als template Engine wurde mir damals im Sommer schmackhaft gemacht und auch beigebracht. Das zusätzlich erarbeitete Wissen an der Universität vor allem im Bereich der Sicherheit von Webanwendungen haben geholfen, die Authentifizierung sicherer zu machen und somit auch die gesamte Anwendung.

Die gesamte Anwendung wurde von Grund auf neu strukturiert, eine Struktur, welche wirklich dynamisch, erweiterbar und leichter Verständlich ist als die alte Struktur.

Auch die Datenbank wurde neu entwickelt, sodass es einfacher ist, mit den Tabellen umzugehen, dass der Programmablauf viel einfacher zu realisieren ist, und die Daten viel transparenter und logischer zu gliedern sind.

11.2 Dank für die Version 2.0

Der Dank geht an die verschiedensten Personen, Personen, die mich hinderten Tag und Nacht zu arbeiten, Personen, die mir immer mit Rat und Tat zur Seite standen und Personen, die mir Teilestücke von Software zur Verfügung stellten, wie zum Beispiel Christoph Mair von der Sad. Er stellte mir eine wunderbare Klasse zur realisierung der template Engine Smarty zur Verfügung, welche wunderbar funktioniert und sehr komfortabel ist.

