Vordiplomarbeit

TSU

4. Semester

LoeGe v0.9

Erstellt am:
7. Oktober 2002 – 2. März 2003

Erstellt von:
Thomas Gemperli
Reto Loepfe

Inhaltsverzeichnis

1. Übersicht

1. Kapitel (Übersicht)
In der Übersicht sind sämtliche Kapitel beschrieben. Es wird kurz erklärt, was den Leser in den einzelnen Kapitel erwartet.

2. Kapitel (Pflichtenheft)
Beschreibt die Aufgabenstellung die wir von unserem Betreuer erhalten haben und unsere Zielsetzung.

3. Kapitel (Analyse)
Beschreibt, welche Ideen wir verfolgt haben. Es zeigt gewisse Problematiken und Lösungsvarianten. Weiter wird in diesem Kapitel unser Konzept beschrieben.

4. Kapitel (Systemvoraussetzungen)
Beschreibt die Systemvoraussetzungen für LoeGe.

5. Kapitel (Projektplanung)
Zeigt auf, in welcher Phase wir wie lange fuer was eingeplant haben, sowie die Arbeitsteilung in der Gruppe.

6. Kapitel (Design des GUI)
Beschreibt, welche Funktionen über das GUI aufgerufen werden und wie das GUI aufgebaut ist.

7. Kapitel (Design der Software)
Beschreibt das Prinzip der Software sowie den Programmablauf und die einzelnen Prozesse.

8. Kapitel (Implementation)
Beschreibt wie das Programm aufgebaut ist und die einzelnen Klassen und Methoden.

9. Kapitel (Test)
Beschreibt die Vorgehensweise bei den Tests.

10. Kapitel (Bedienungsanleitung)
Beschreibt, wie das Programm installiert wird und wie man es bedient.

11. Kapitel (Schlusswort)
Kurzes Schlusswort der Entwickler.

12. Kapitel (Unterschriften)
Unterschriften der einzelnen Mitglieder der Gruppe.

13. Kapitel (Anhang)
Im Anhang befindet sich der Sourcecode in gedruckter Form.

2. Pflichtenheft

2.1 Aufgabenstellung der Schule/Betreuer

Erstellen eines Programms, welches zum Lernen von englischen Wörtern verwendet werden kann.

Das Programm soll Multi-User fähig sein und die Vokabeln in einer Datenbank ablegen. Den Vokabeln soll ein Lektionenflag zugeordnet werden können, um sie gegebenenfalls mit spezifischen Lektionen aus einem Kurs zu verknüpfen. Verschiedene Lernmodi sollen implementiert werden (Random, Alphabetisch, Wortarten, ganze Datenbank, Lektionenorientiert etc.). Zudem sollte man in beide Richtungen (De-En, En-De) lernen können.

Die Aufgabe umfasst folgende Punkte:

- Projektplanung (Ablauf, Arbeitsteilung, Sitzungen, Dokumentation
- Einarbeitung in die Themen vokabularorientierte Lernsysteme und Datenbanken
- Analyse und theoretischer Entwurf der Software
- Implementation der einzelnen Module
- Aufbau einer Testdatenbank
- Test der Software hinsichtlich Funktion, Fehler und Anwendertauglichkeit
- Evtl. Erstellen einer kleinen Bedienungsanleitung
- Dokumentation aller Arbeitsschritte

Daten, Anzustrebendes:

- Plattform — Lauffähig auf Unix, MacOS X und Windows
- Softwarefunktionen — Multi-User, Verschiedene Lernmodi
- Datenbanken — Erstellung, Einbindung, Modifikation
- Ausgabe — Userfreundliches GUI

2.2 Zielsetzung

Ziel	Muss	Soll	Wünscheswert
Lauffähig auf Linux	☒		
Lauffähig auf MacOS X		☒	
Lauffähig auf Windows	☒		
Multiuserfähig	☒		
Lernmodus „Random“	☒		
Lernmodus „Alphabetisch“	☒		
Lernmodus „Lektionen“	☒		
Deutsch – Englisch	☒		
Englisch – Deutsch	☒		
Graphische Oberfläche (GUI)		☒	
Neue Wörter eingeben, löschen, ändern	☒		
Spiele		☒	
Hilfemenu			☒
Tabelle updaten			☒
Programm für Deutsche und Englische Anwender			☒
Lernstatistik			☒
Dix-Funktion		☒	

3. Analyse

Nachfolgend werden einige wichtige Punkte einzeln analysiert, um noch gezieltere Abklärungen zu treffen und um einschneidende Probleme frühzeitig zu erkennen.

3.1 Allgemein

Das Programm soll auf Linux, Windows und MacOS X laufen.

Das Programm muss Multiuserfähig sein. Mehr dazu im Kapitl 3.2 (Multiuserfähig)

Das Programm ist eigentlich eine Konsolenanwendung. Dies ist für unsere Anforderung ungeeignet, die heutigen Benutzer wollen eine bedienerfreundliche Oberfläche. Sie möchten auf einer übersichtlich dargestellten graphischen Oberfläche arbeiten, navigieren und nur das Mindeste von Hand eingeben. Da wir im Unterricht bis jetzt nur mit Konsolenanwendungen gearbeitet haben, haben wir uns entschlossen, die Oberfläche mit Hilfe von HTML zu erstellen. Mehr dazu im Kapitel 3.3 (GUI)

Die Wörter (Englisch und Deutsch) müssen in tabellarischer Form gespeichert werden. Es gibt eigentlich zwei Varianten. Die erste und einfachere ist, die Wörter in ein File abzuspeichern. Die etwas aufwendigere Variante ist, die Wörter in einer Datenbank zu verwalten. Mehr dazu im Kapitel 3.4 (Tabellen)

3.2 Multiuserfähig

Das Programm müssen wir Multiuser- und somit auch Netzwerkfähig gestallten. Einerseits weil unser Programm noch eine Wörterbuch-Funktion intergriert hat und andererseits weil jeder Benutzer auf einem anderen Lern-Level ist. Er muss somit die Wörter markieren, die ihn interessieren. Dazu muss jedes Wort einem Benutzer übergeben werden. Neue Wörter müssen aufgenommen und geändert werden können. Durch das Anmelden beim Starten des Programmes wird der Benutzer identifiziert oder er kann neu erstellt werden. Es können auch mehrere User gleichzeitig angemeldet sein.

3.3 GUI

Wie schon Oben erwähnt, möchten wir eine graphische Oberfläche erstellen, die das Programm möglichst einfach und bedienerfreundlich macht.
Thomas Gemperli hat aus seiner früheren Tätigkeit Erfahrung im Erstellen von Webseiten. Das hat uns auf die Idee gebracht, die Benutzeroberfläche in HTML zu erstellen. Dazu benötigen wir, wenn unser Programm nicht in einem Netzwerk mit Webserver installiert wird, zusätzlich eine Installation eines Webservers lokal auf dem Rechner.

Vorteile:
- Bedienerfreundlich
- Optisch auf aktuellem Stand

Nachteile:
- zusätzliche installation eins Webserver

Entscheid:
Auf dem Markt gibt es unzählige Webserver. Wir haben uns für Apache entschieden, weil die Software gratis, einfach zu installieren und sehr verbreitet ist.

3.4 Tabellen (Datenbank)

Wie wir schon Oben erwähnt haben gibt es zwei für uns sinnvolle Varianten, um die Deutsch- und Englischwörter zu verwalten. In den nächsten zwei Abschnitten haben wir die Möglichkeiten etwas genauer unter die Lupe genommen und versucht die Vor- und Nachteile aufzuzeigen.

3.4.1 ohne Datenbank

Die Wörter werden tabellarisch in einem File abgespeichert. Wir benötigen fogende Tabellen:

- Haupttabelle(n), in der sich alle bekannten Wörter befinden
- Usertabelle(n), in der alle User mit Username und Passwort eingetragen sind
- Usereigene Tabelle, jeder User kann sich seine eigene Lerntabelle erstellen. In dieser Tabelle werden die Wörter (deutsche und englische) für den User gespeichert. Pro User wird eine solche Tabelle angelegt.

Vorteile:

- Keine zusätzliche Datenbanksoftware
- Wir lernen, wie man mit der Programmiersprache C++ auf Dateien zugreift und sie manipuliert

Nachteile:

- Weniger flexibel
- Relativ langsam
- Aufwendiger Realisation

3.4.2 mit Datenbank

Die Wörter werden in eine Tabelle der Datenbank gespeichert. Wir benötigen voraussichtlich folgende Tabellen:

- Haupttabelle, in der sich alle bekannten Wörter und Redewendungen befinden
- Usertabelle, in der alle User mit Username und Passwort befinden
- Verbindungstabelle (Kreuztabelle)

Vorteile:

- Sehr schnell
- Flexibel für spätere Erweiterungen

Nachteile:

- zusätzliche Datenbanksoftware wird benötigt

Entscheid:
Wir haben uns für die Lösung ohne Datenbank entschieden. Vorallem aus dem Grund, dass keine zusätzliche Software eingesetzt werden muss.

3.4.3 Login-Tabelle

Die Login-Tabelle enthält die Informationen über jeden einzelnen Benutzer von LoeGe. Das Einloggen mit Username und Passwort ist notwendig um den User zu Identifizieren, damit er auf seine Tabellen zugreifen kann. Es hat somit nichts mit Security.

Die Login-Tabelle ist wie folgt aufgebaut.

Benutzername	Passwort
gempi	test1
loepfe	test2

Die Benutzerdaten werden in einem File mit dem Namen login.txt abgespeichert. Als Feldtrenner wird ein „;“ verwendet.

Beispiel für ein login.txt:

gempi;test1
loepfe;test2

3.4.4 Master-Tabelle

Die Master-Tabelle ist, wie der Name schon sagt, die Haupttabelle oder mit anderen Worten das Wörterbuch. Sie wird mit dem Namen LG_mastertab.txt abgelegt. Die Wörter werden durch verschiede Feldtrenner und Unterfeldtrenner voneinander getrennt.

Beispiel für ein LG_mastertab.txt

grundsätzlich {adv} :: strictly
gruppieren; in Gruppen einteilen :: to group
gruselig; unheimlich {adj} | gruseliger | am gruseligsten :: creepy | creepier | creepiest

Trenner	Bedeutung	Beispiel
;	Der Sinn der Wörter ist gleich	gruselig; unheimlich {adj}
|	Damit werden Redewendungen oder Steigerungsformen des eigentlichen Wortes angegeben	gruseliger | am gruseligsten
::	Mit diesem Trenner werden die Deutsch-wörter von den Englischen getrennt.	grundsätzlich {adv} :: strictly

3.4.5 User-Tabelle

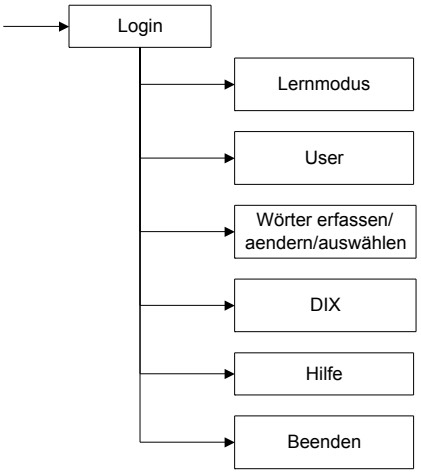
Die User-Tabelle enthält die benutzerspezifischen Wörter. Sie wird vom User generiert. Es werden die Informationen ID, Lektion, deutsches Wort, englisches Wort, Flag (siehe Tabelle).

Bezeichnung	Bedeutung
ID	Fortlaufende Nummer, die pro Unit eindeutig ist.
Lektion	Zugehörigkeit des Wortes
Deutsches Wort	Deutsches Wort
Englisches Wort	Englisches Wort
Flag	TRUE : Der User beherscht dieses Wort FALSE: Der User muss noch üben

3.5 Aufbau der Menüsteuerung

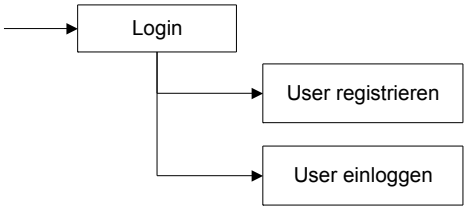
3.5.1 Hauptmenü

Im Hauptmenü sollen alle wichtigen Funktionalitäten ersichtlich sein.



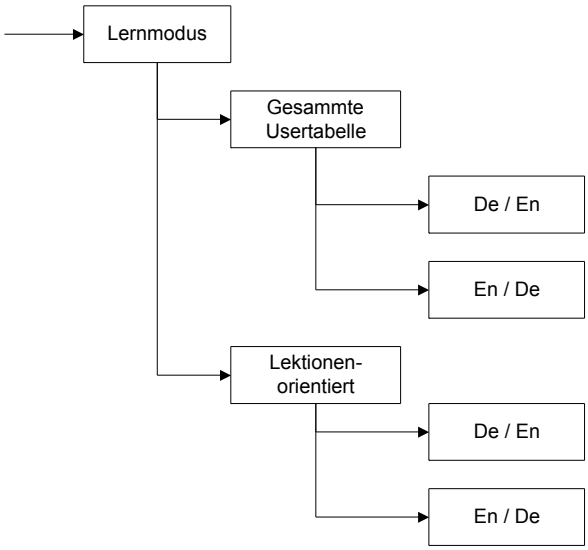
3.5.2 Loginmenü

Im Loginmenü wird ein registrierter Benutzer identifiziert, oder ein Neuer eingetragen.



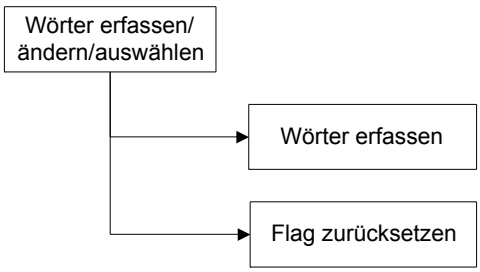
3.5.3 Lernmodusmenü

Im Lernmodusmenü kann man zwischen den einzelnen Modi ausgewählt werden.



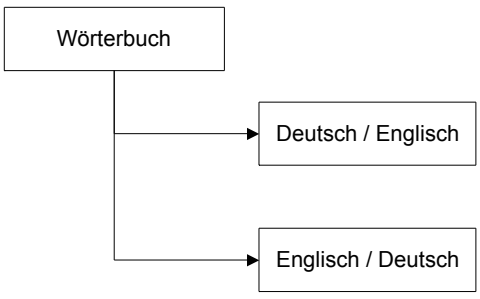
3.5.4 Wortlistemenü

In diesem Menü müssen neue Wörter in das Userfile abgespeichert, verändert und zurückgesetzt werden können.



3.5.5 Wörterbuchmenü

Im Wörterbuch können englische in deutsche Wörter übersetzt werden und umgekehrt. Die eingegebenen Wörter werden mit der Haupttabelle verglichen und alle gefundenen Wörter werden ausgegeben.



4. Systemvoraussetzungen

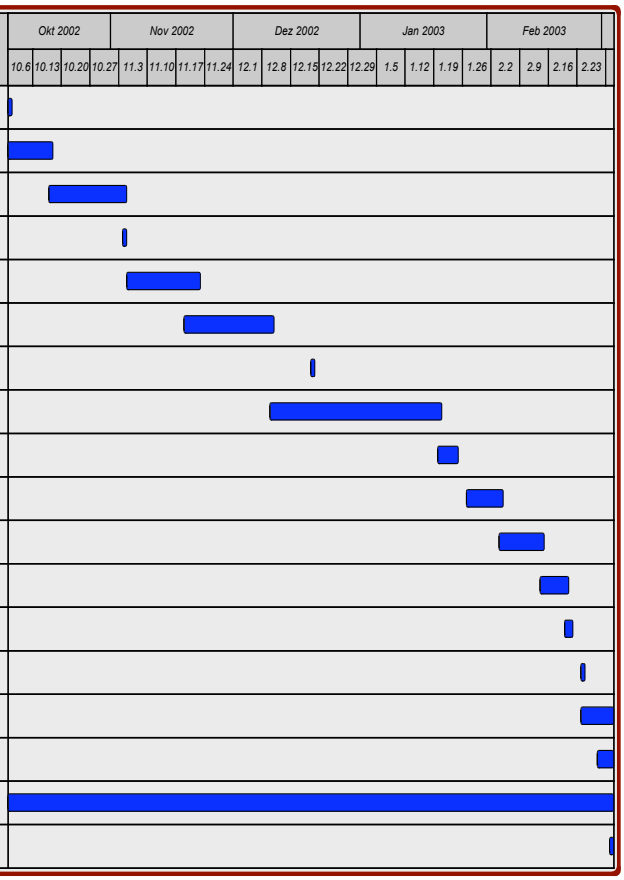
Das ausführbare Programm LoeGe muss auf einem Webserver installiert werden. Der Webserver ist „Schnittstelle“ zwischen dem Browser und dem Programm, welches sich im cgi-bin Verzeichnis befindet. Ansonsten werden keine speziellen Systemanforderungen gestellt. LoeGe ist lauffähig auf einem Linux- , Windows- oder Unixbasierten Betriebssystem.

Zusatzsoftware: Sofern der Anwender keinen Zugriff auf einen Webserver hat, oder keinen auf seinem Computer installiert hat, muss er dies noch nachholen. Wir empfehlen den Webserver von Apache. Er steht kostenlos im Internet zur Verfügung (http://www.apache.org/) Ausserdem unterstützt Apache ziemlich alle heute gebräuchlichen Betriebssystemen

5. Projektplanung

5.1 Projektplan

ID	Aufgabenname	Anfang	Ende	Dauer
1	Empfang der Aufgabenstellung	07.10.2002	07.10.2002	1t
2	Analyse, Pflichtenheft	07.10.2002	17.10.2002	9t
3	Einarbeitung, Informationsbeschaffung	17.10.2002	04.11.2002	13t
4	1. Betreuersitzung	04.11.2002	04.11.2002	1t
5	Design des GUIs	05.11.2002	22.11.2002	14t
6	Design der Software	19.11.2002	10.12.2002	16t
7	2. Betreuersitzung	20.12.2002	20.12.2002	1t
8	Implementation der Phase I	10.12.2002	20.01.2003	30t
9	Enzeltest	20.01.2003	25.01.2003	5t
10	Implementation der Phase II	25.01.2003	04.02.2003	7t
11	Gesamttest	04.02.2003	14.02.2003	9t
12	Korrekturen, Anpassungen	14.02.2003	20.02.2003	5t
13	Abschlusstest	20.02.2003	23.02.2003	2t
14	3. Betreuersitzung	23.02.2003	23.02.2003	1t
15	Vorbereitung der Präsentation	23.02.2003	03.03.2003	6t
16	Unvorhergesehenes	28.02.2003	03.03.2003	2t
17	Dokumentation	07.10.2002	03.03.2003	106t
18	Abgabetermin	03.03.2003	03.03.2003	1t



5.2 Zeitplan

Aufgabe	Maximal geplante Zeit (TEAM)
Analyse, Erstellung des Pflichtenheftes	24 Std
Einarbeitung, Informationsbeschaffung	30 Std
1. Betreuersitzung	4 Std
Design des GUIs	34 Std
Design der Software	48 Std
2. Betreuersitzung	4 Std
Implementation der Phase I	20 Std
Einzeltest	10 Std
Implementation der Phase II	20 Std
Gesamttest	20 Std
Korrekturen, Anpassungen	12 Std
Abschlusstest	6 Std
3. Betreuersitzung	4 Std
Vorbereitung der Präsentation	16 Std
Unvorhergesehenes	4 Std
Dokumentation	44 Std
Gesamtzeit	300 Stunden

5.3 Arbeitsteilung

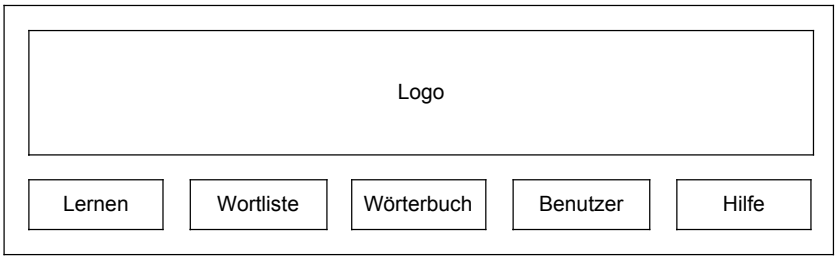
Aufgabe	Thomas Gemperli	Reto Loepfe
Analyse, Erstellung des Pflichten-heftes	12 Std	12 Std
Einarbeitung, Informantionsbeschaffung	15 Std	15 Std
Alle Betreuersitzungen	6 Std	6 Std
Design GUI	30 Std	4 Std
Design Software	18 Std	30 Std
Implementation der Phase I	10 Std	10 Std
Einzeltest	5 Std	5 Std
Implementation der Phase II	10 Std	10 Std
Korrekturen, Anpassungen	6 Std	6 Std
Abschlusstest	3 Std	3 Std
Vorbereitung der Präsentation	8 Std	8 Std
Dokumentation	10 Std	32 Std
Gesamtzeit	150 Std	150 Std

6. Design des GUIs

6.1 Allgemein

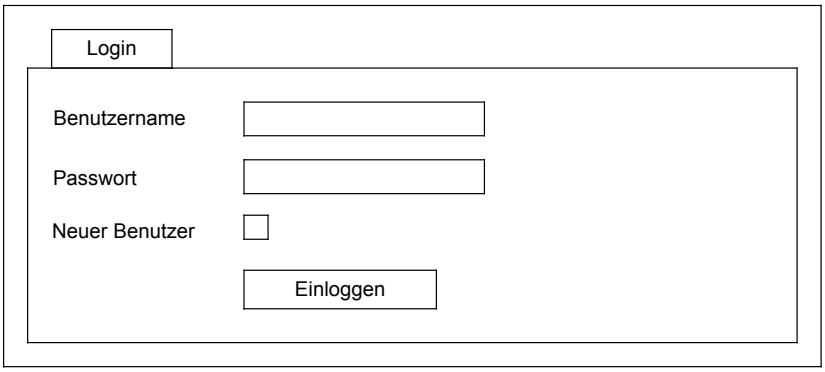
Die graphische Oberfläche soll sehr übersichtlich und für jedermann einfach bedienbar sein. Es soll möglich sein von jedem Menüpunkt jederzeit zu jedem Anderen zu wechseln. Das heisst, wir brauchen einen fixen Header, der alle Menüpunkte beinhaltet und bei jedem Aufruf eines Menüs dargestellt wird.
Der variable Teil, je nachdem wo sich der Benutzer gerade befindet, wird unterhalb der Buttons dargestellt.

Header:



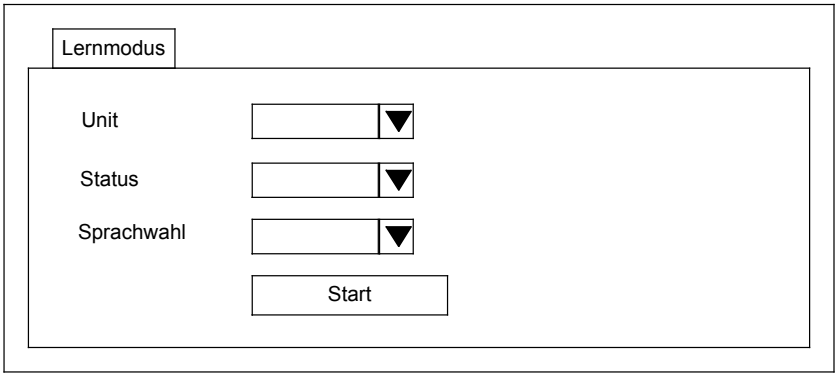
Gleich unterhalb des Kopfes, wird der variable Teil dargestellt. Je nach Menüpunkt wird eine andere Darstellung gewählt. In Kapitel 6.2 – 6.6 wird der variable Teil näher beschrieben.

6.2 Login

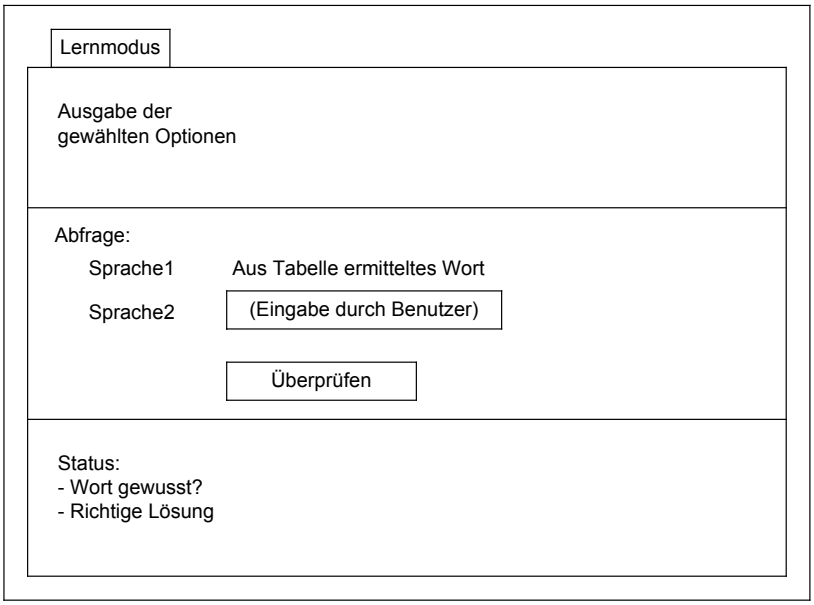


6.3 Lernmodus

Im ersten Schritt des Lernmodus sollen die Optionen auswählbar sein.

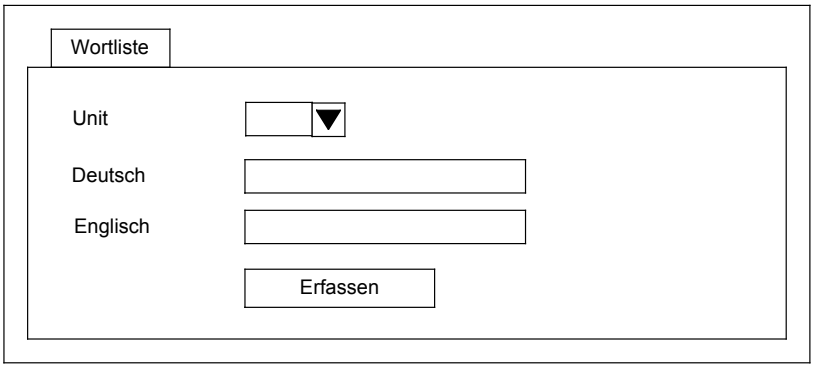


Nach dem absenden der Ausgewählten Optionen ist man im Lernmodus.



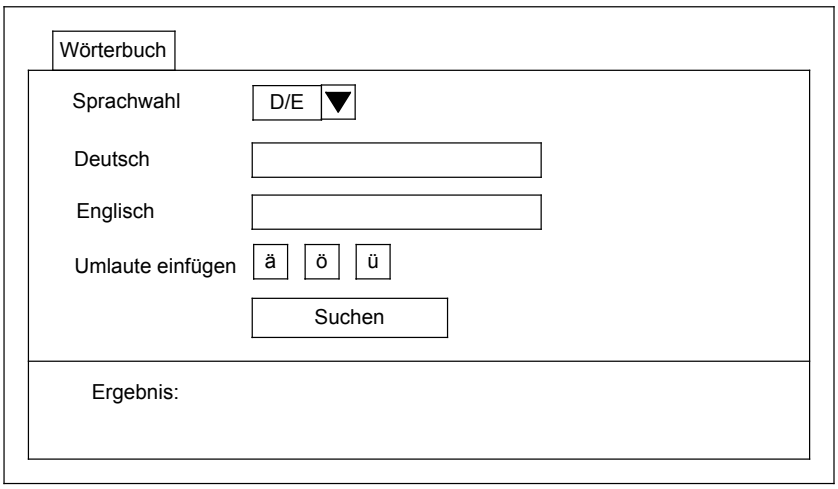
6.4 Wortliste

Hier sollen für jeden Benutzer individuell die zu lernenden Wörter abgespeichert werden.



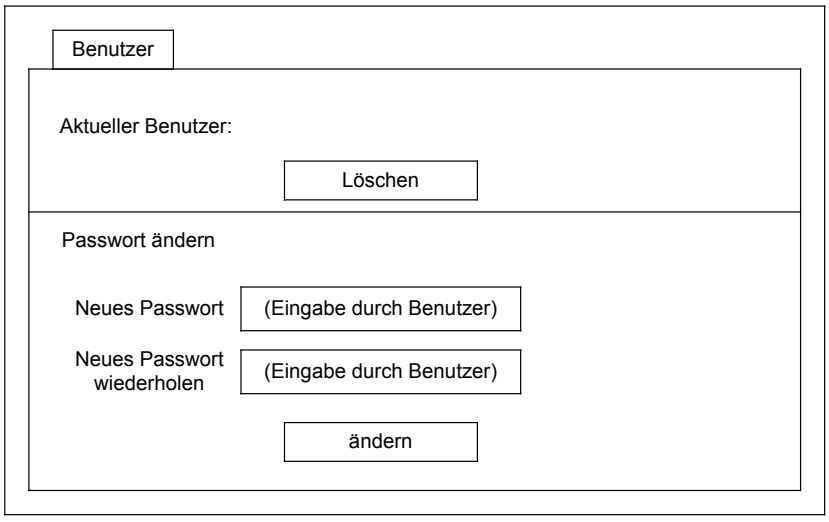
6.5 Wörterbuch

Unter dem Menü Wörterbuch soll, wie der Name schon sagt, ein Englisch-Deutsch Wörterbuch implementiert werden.



6.6 Benutzer

Der Benutzer soll die Möglichkeit erhalten, sein Passwort zu ändern oder sein eigenes Konto zu löschen.



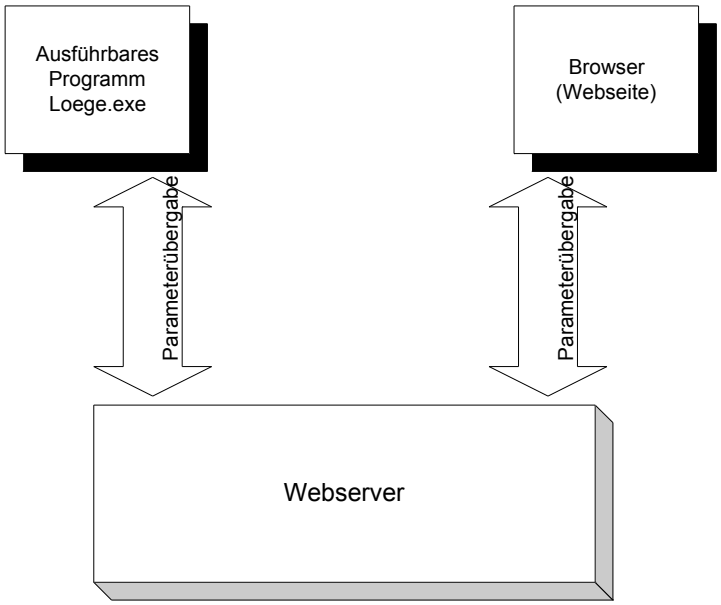
7. Design der Software

7.1 Prinzip der Software

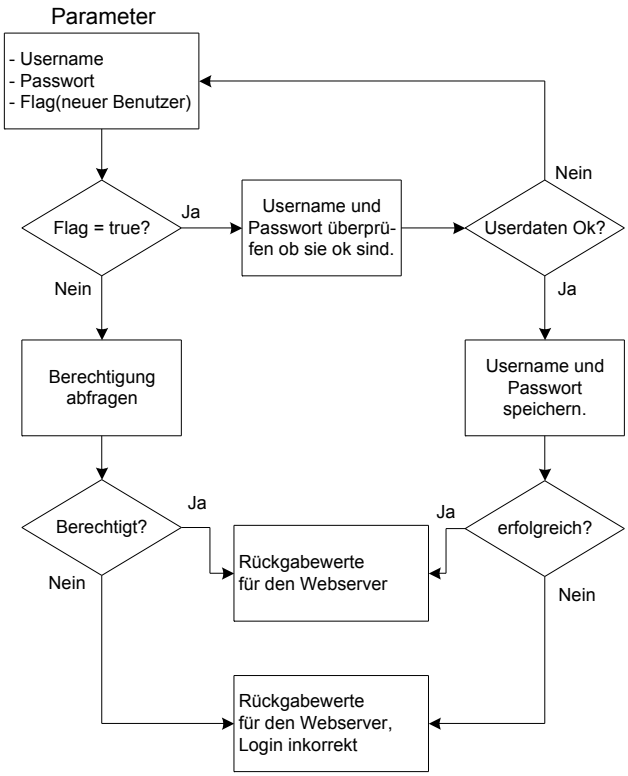
Die graphische Bedieneroberfläche wird mit HTML erstellt. Das Programm soll aber in C++ geschrieben werden. Aufgerufen wird es über einen Browser. Damit alles zusammenspielt, muss sich das ausführbare Programm auf einem Webserver im cgi-bin Verzeichnis befinden. Über einen Browser wir das Programm schliesslich gestartet.
Um das Programm auch auf einem Standalone PC zu verwenden muss zuerst ein Webserver installiert werden.

7.2 Ablauf

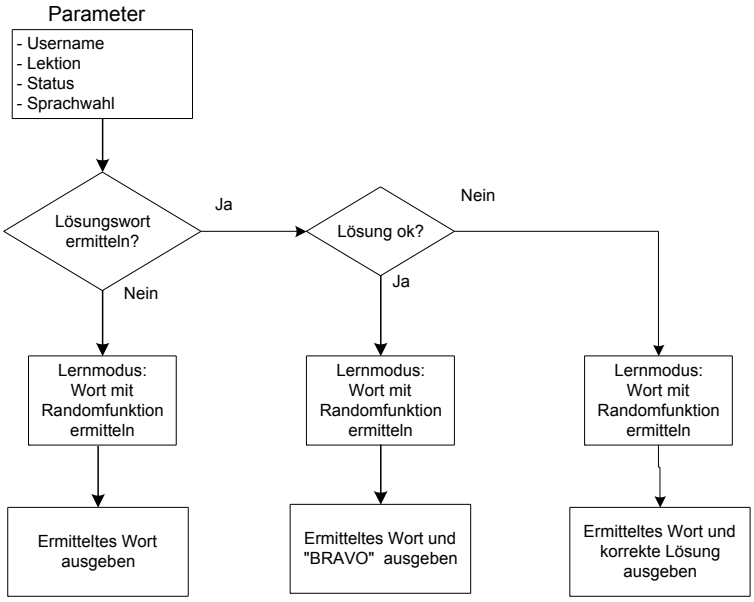
Das ausführbare Programm wird vom Browser über den Webserver gestartet. Das Programm erzeugt einen HTML-Seite. Der Benutzer kann sich nun einloggen und seine Einstellungen machen oder mit dem Lernen beginnen.



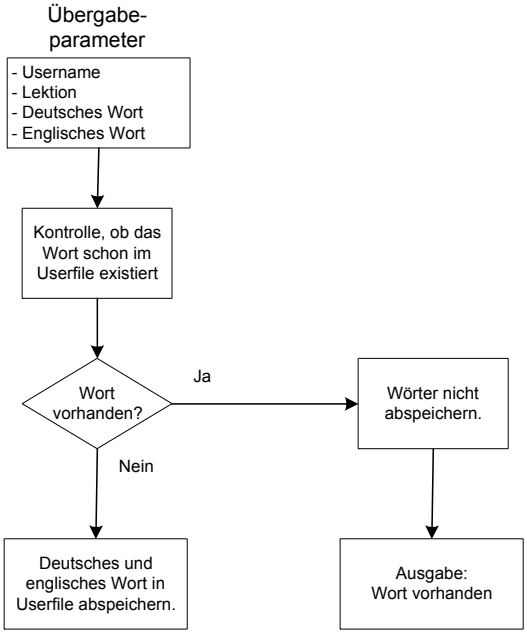
7.3 Login-Prozess



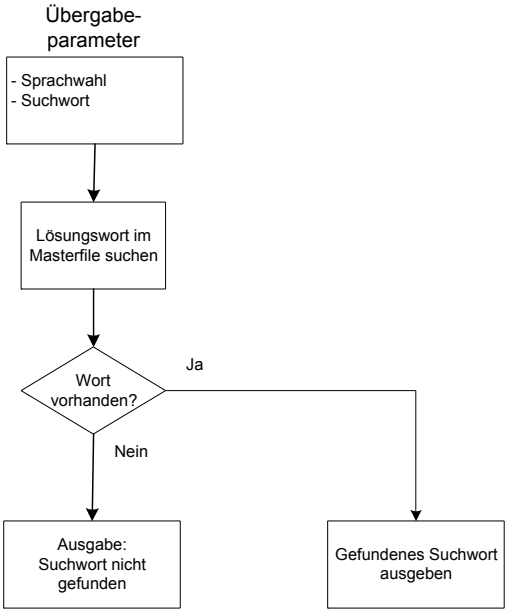
7.4 Lernmodus-Prozess



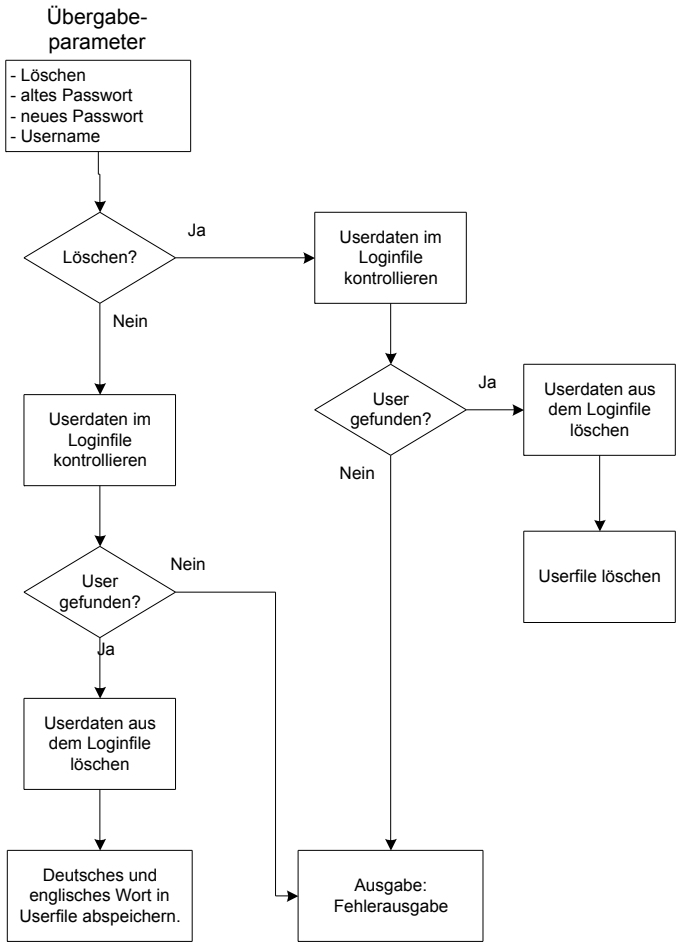
7.5 Wortliste-Prozess



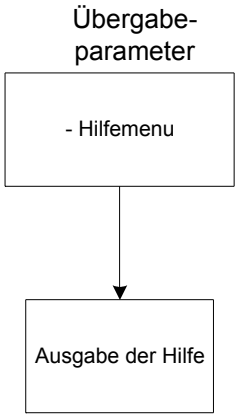
7.6 Wörterbuch-Prozess



7.7 Benutzermodus-Prozess



7.8 Hilfemodus-Prozess



8. Implementation

Beim Einarbeiten in die cgi-Programmierung sind wir im Internet auf eine Klasse cgicc gestossen. Mit Hilfe dieser Klasse können relativ einfach ENVIRONMENT–Variablen oder Elemente vom Webserver ermittelt werden, die wir für unsere Anwendung dringend benötigen. Alle anderen Klassen haben wir selber erstellt. Die cgicc–Klasse wird unter 8.1 kurz erklärt.

8.1 Klassen

Die cgicc Klasse wurde entwickelt, um einfach mit C++ CGI Applikationen für das World Wide Web zu schreiben.

Sie beinhaltet Funktionen wie z.B.:

- Parsen von GET und POST (form)
- Unterstützt Cookies
- Support HTTP File upload
- Kompatibel mit FastCGI
- Beinhaltet Methoden um CGI-Environment zu ermitteln oder ändern.
- .....

8.1.1 cl_printhtml

Die Aufgabe der cl_printhtml-Klasse ist es, die HTML-Seite darzustellen. Sie besteht aus folgenden Methoden:

cl_printhtml(const string& version)

- Ist der Defaultkonstruktor. Hier werden bei jedem Programmaufruf zuerst ein Cgicc Objekt erstellt und die einzelnen CGI-Form Elemente gesetzt
- ~cl_printhtml()
  Der Destruktor
- void print()
  Diese Methode wird aus dem main aufgerufen. Baut die Seite je nach Wert der CGI-Form Daten auf.
- void print_line()
  Erstelle eine horizontale Linie. Zwecks Vereinfachung und Codeersparniss.
- void print_header()
  Ausgeben des HTML-Headers.
- void print_nav()
  Die HTML Ausgabe des Menus.
- void print_foot()
  Die HTML Ausgabe des Fussteiles der Website
- void print_contentWelcome()
  Die HTML Ausgabe des Willkommenseite. Erscheint nach erfolgreichem Login.
- void print_contentError()
  Die HTML Ausgabe bei einem Fehler. Darf nicht vorkommen.
- void print_contentHilfe()
  Die HTML Ausgabe der Hilfe Texte.
- void print_contentLernen()
  Die HTML Ausgabe der Optionenseite der Funktion "Woerter lernen"

- void print_contentLernenSet()
  Die HTML Ausgabe der Maske "Woerter lernen"
- void print_contentErfassen()
  Die HTML Ausgabe der Maske "Wortliste"
- void print_contentDictionary()
  Die HTML Ausgabe der Maske "Woerterbuch"
- void print_contentEinstellungen()
  Die HTML Ausgabe der Maske "Benutzer"
- void print_contentLogin()
  Die HTML Ausgabe der Login Maske
- void add_user()
  Ruft die Methode setUser aus cl_login auf
- void del_user()
  Ruft die Methode delUser aus cl_login auf
- void change_Pw()
  Ruft die Methode changePw aus cl_login auf
- void check_user()
  Login Daten des Benutzers ueberpruefen. Wird bei jedem Aufruf von loege ausgefuehrt.
- void query_dict()
  Ruft getEnWord rsp. getDeWord aus cl_search auf
- void set_word()
  Ruft setDEWord aus cl_learn auf
- void check_word()
  Wort aus Methode getWord2Learn "holen"
- void get_word()
  Worter fuer Maske lernen vergleichen und Status ausgeben.

8.1.2 cl_login

In der cl_login-Klasse befinden sich alle Methoden um Benutzerinformationen wie Benutzername und Passwort zu setzen, löschen und abfragen.

- cl_login()
  Der Defaultkonstruktor. Hier wird der Pfad gesetzt, wo sich das Login-File befindet.
- ~cl_login()
  Der Destruktor.
- bool setUser(const string& Username, const string& Password)
  Username und Password eines neuen Benutzers setzen
- bool getLogin(const string& Username, const string& Password)
  Password abfragen
- bool delUser(const string& Username, const string& Password)
  User löschen
- bool changePw(const string& Username, const string& Password,
  const string& newPassword)
  Passwort aendern

8.1.3 cl_userfile

In der cl_userfile-Klasse befinden sich alle Methoden um im Userfile manipulationen vornehmen zu können.

- cl_userfile()
  Der Konstruktor
- ~cl_userfile()

Der Destruktor

- string setLearnFlag(const string& str_language, const string& str_username, const string& str_Word, const string& str_Flag, const string& str_resAll)
  Setzt das Flag eines Wortes auf true oder false. Das ganze Userfile kann geresetet werden, indem man der Methode. einen String (str_resAll) übergibt.
- bool delWord(const string& str_language, const string& str_username, const string str_Word)
  Wort, resp. eine ganze Zeile, aus dem Userfile löschen
- int anzUnit(const string& str_username)
  Ermittelt die Anzahl verschiedener Units in einem Userfile. Wird in dieser Klasse aufgerufen.

8.1.4 cl_search

In der cl_search-Klasse befinden sich alle Methoden für die Wörterbuch Funktion.

- cl_search()
  Konstruktor
- ~cl_search()
  Destruktor
- string getEnWord(const string& str_DeWord);
  Englisches Word ermitteln
- string getDeWord(const string& str_EnWord)
  Deutsches Word ermitteln
- string getPathName(const string& str_DeWord, const string& str_learnTyp)
  Anhand des ersten Buchstaben Pfad und Name der übersetzungstabelle ermitteln

8.1.5 cl_learn

In der cl_learn-Klasse befinden sich alle Methoden, um im Userfile Wörter abzufragen, zu löschen und zu ermittlen.

- cl_learn()
  Konstruktor
- ~cl_learn()
  Destruktor
- string cl_learn::getWord2Learn(const string& str_Unit, const string& str_learnTyp, const string& str_username, const string& str_WordTrueFalse)
  Wort zum Lernen aus dem Userfile ermitteln
- bool cl_learn::setDEWord(const string& str_DWord, const string& str_EWord, const string& str_Unit, const string& str_username)
  Wörter in Userfile abspeichern
- int cl_learn::getMaxID(const string& str_Unit, const string& str_DeWord, const string& str_WordTrueFalse)
  Hoechste ID der übergebenen Unit ermitteln
- string cl_learn::getResult(const string& str_SprachWahl, const string& str_Word, const string& str_RWord, const string& str_username);

- string cl_learn::getRightWord()

9. Test

Die Idee ist es, das lauffähige Programm auf einem Webserver zu platzieren. Das heisst der Webserver ruft unser Programm auf. Wir können darum unsere Tests in zwei Kategorien einteilen.

- Kategorie 1 ist die tiefste Ebene unserer Test. Hier werden die einzelnen Methoden durchgetestet. Das zusammenspiel mit dem Webserver kommt hier noch nicht zu tragen.
- Kategiorie 2 ist eigentlich der Gesamttest. Das heisst, unser Exe wird auf dem Webserver platziert. Die Funktionen werden mit dem GUI getestet.

Im folgenden wird nur der Gesamttest dokumentiert, aber es ist natürlich klar, dass die einzelnen Module ausgetestet wurden bevor sie in das produktive LoeGe-Schema inportiert wurden.

9.1 Login

Situation	Tätigkeit	Ergebnis	
Vor dem Start	Browsereingabe der Adresse	Loginfenster wird geladen	☐
Nach dem Start	Drücken der anderen Buttons	Keine Änderung, das Loginfenster bleibt bestehen	☐
Nach dem Start	User und Passwort eingeben	Loginfenster mit der Fehlermeldung, dass Username und Passwort falsch sind	☐
Nach dem Start	User und Passwort eingeben und Flag „Neu“ setzten.	User wird eröffnet und ist angemeldet	☐
Nach dem Start	Bekannter User und Passwort eingeben	User erfolgreich angemeldet	☐
Nach dem Anmelden	Drücken der Anderen Buttons	Es kann frei über alle Buttons navigiert werden.	☐
Nach dem Anmelden	Auf das Logo klicken	User wird abgemeldet und Loginfenster wird geladen	☐

9.2 Wortliste

Situation	Tätigkeit	Ergebnis	
Nach dem Login	Klick auf den Button „Wortliste“	Wortlistefenster wird geladen	☐
Nach Start Wortliste	Unit auswählen, deutsches und englisches Wort eingeben und ok	Wörter und Unit werden erfolgreich im Userfile gespeichert.	☐

9.3 Lernen

Situation	Tätigkeit	Ergebnis	
Nach dem Login	Klick auf den Button „Lernen“	Fenster mit den Lernoptionen wird geladen	☐
Nach Start Lernoptionen	Alle Varianten von Optionene testen	Lernenfenster mit dem ersten zu lernenden Wort wird geladen	☐
Nach Start Lernen	Lösungswort eingeben und Button anklicken	Lösungswort wird gesucht und ausgegeben. Meldung ob korrekt erscheint. Neues Wort wird geladen. Im Userfile wird das Flag auf TRUE gesetz wenn Eingabe korrekt war.	☐
Nach Über-prüfen	Neues Lösungswort eingeben und Button anklicken	Lösungswort wird gesucht und ausgegeben. Meldung ob korrekt erscheint. Neues Wort wird geladen. Im Userfile wird das Flag auf TRUE gesetz wenn Eingabe korrekt war.	☐

9.4 Wörterbuch

Situation	Tätigkeit	Ergebnis	
Nach dem Login	Klick auf den Button „Wörterbuch“	Wörterbuchfenster wird geladen	☐
Nach Start Wörterbuch	Sprachflag setzen und Suchwort eingeben	Gefundene Wörter ausgeben	☐
Nach Start Suchen	Sprachflag setzen und Suchwort eingeben	Gefundene Wörter ausgeben	☐

9.5 Benutzer

Situation	Tätigkeit	Ergebnis	
Nach dem Login	Klick auf den Button „Benutzer“	Benutzerfenster wird geladen. Aktueller Benutzer wird dargestellt.	☐
Nach Start Benutzer	Neues Passwort 2 mal richtig eingeben.	Benutzerfenster mit ok Meldung wird dargestellt. Passwort wird im Userfile.geändert	☐
Nach Start Benutzer	Neues Passwort 1 mal falsch eingeben.	Benutzerfenster mit Fehler (PW sind nicht identisch) Meldung wird dargestellt.	☐
Nach Start Benutzer	Klick auf den Button „Löschen“	Loginfenster wird geladen mit der Meldung dass User gelöscht wurde.	☐

9.6 Hilfe

Situation	Tätigkeit	Ergebnis	
Nach dem Login	Klick auf den Button „Hilfe“	Hilfefenster wird geladen.	☐

10. Bedienungsanleitung

10.1 Installation

Bitte beachten sie folgende Installationsschritte:

1. Webserver installieren,sofern noch keiner installiert ist, oder man keinen Zugriff auf einen Webserver hat. Dazu kann man den mitgelieferten Webserver von Apache installieren, oder von http://www.apache.org/ herunterladen.

2. Das ausführbare Programm muss man auf den Webserver ins Verzeichnis cgi-bin kopieren.

3. Den ganzen Ordner loege samt Inhalt und Unterverzeichnisse auf den Webserver ins Verzeichnis htdocs kopieren.

4. Zugriffsrecht für das Verzeichnis loege vergeben!!!

5. Browser öffnen und folgende eingabe machen: http://name-webserver/cgi-bin/loege.exe

6. ... und los gehts, viel Spass beim lernen.

10.2 Menü

Mittels Klick der Buttons werden die verschiedenen Funktionen von LoeGe erreicht. Vor dem Anmelden sind die Buttons gesperrt. Durch ein Klick auf das Logo loggen Sie sich aus.

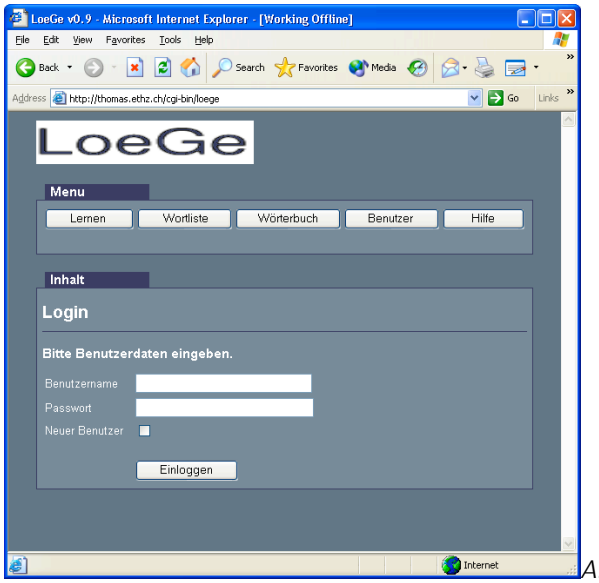


Abbildung 1

10.3 Login oder neuer User anlegen

Durch eingeben des Benutzernamens und des Passwortes kann man sich einloggen. Will sich ein neuer Benutzer anmelden, muss er nur noch das Flag „Neuer Benutzer“ setzen.

10.4 Lernen

Zu Beginn erscheint eine Maske auf der folgende Optionen gesetzt werden müssen:

- **Unit** Selektieren Sie einzelne oder alle Lektionen aus Ihrer persönlichen Wortliste.
- **Status** Wählen Sie entweder alle Wörter, die, die Sie beherrschen oder jene, die noch nicht ganz sitzen.
- **Sprachwahl** Lernen Sie Deutsch-English oder English-Deutsch

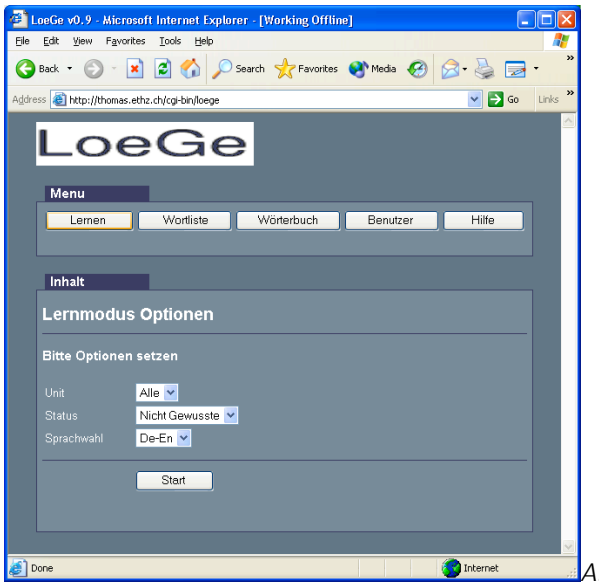


Abbildung 2

Danach folgt das Abfragefenster. Im oberen Teil werden nochmals die gesetzten Suchoptionen angezeigt.

Im Mittelteil geschieht die eigentliche Abfrage. Falls Sie die "Umlaute einfügen" Funktion benutzen wollen, muss ihr Browser Java-Script aktiviert haben.

Nachdem auf Überprüfen geklickt wurde, wird das Ergebniss Ihrer Übersetzung im untersten Teil, dem Resultat Teil, ausgegeben. LoeGe zeigt Ihnen die richtige Lösung in jedem Fall an.

Wenn Sie eine Abfrage richtig beantwortet haben, wird deren Status in Ihrer Wortliste automatisch auf "sitzt" gesetzt.

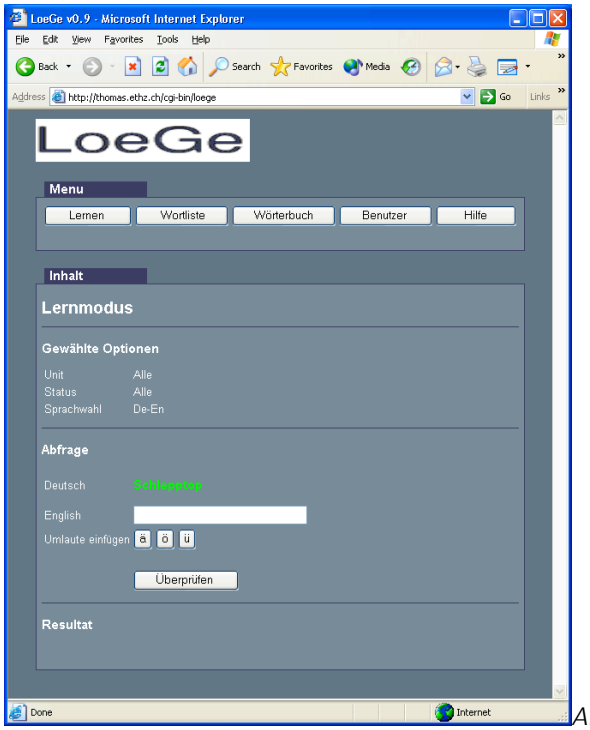


Abbildung 3

10.5 Wortliste

Diese Maske dient zum Erfassen von Vokabeln in die persönliche Wortliste. Die Wortliste ist, wie die meisten Englisch Kurse auch, nach Lektionen sortiert.
Die hier erfassten Vokabeln stehen Ihnen im "Lernen Modus" zur Verfügung.

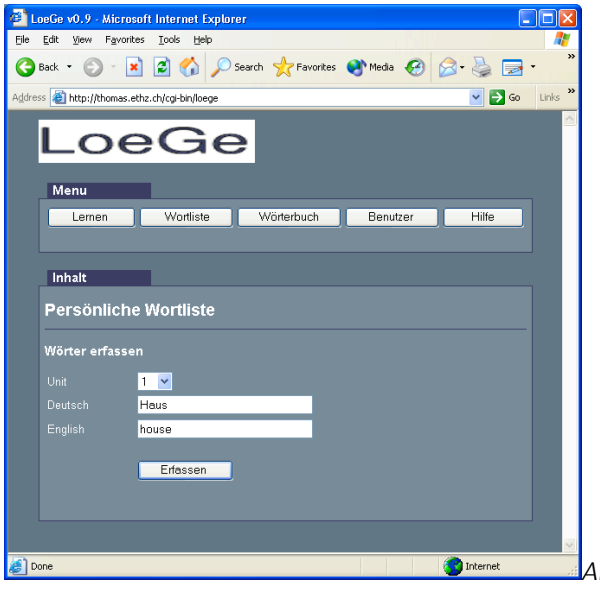


Abbildung 4

10.6 Wörterbuch

Dies ist ein simpler, elektronischer Übersetzer. Suchen Sie nach deutschen oder englischen Wörtern.

Falls Sie die "Umlaute einfügen" Funktion benutzen wollen, muss ihr Browser Java-Script aktiviert haben.

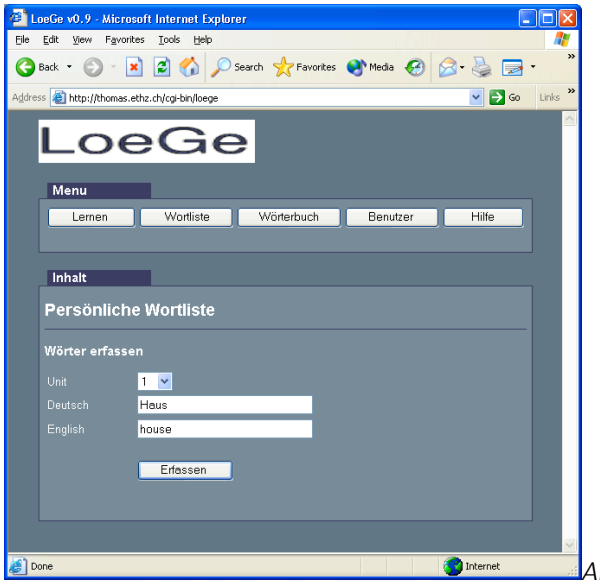


Abbildung 5

10.7 Benutzer

Diese Maske erlaubt Ihnen den aktuellen Benutzer zu löschen sowie dessen Passwort neu zu setzen.

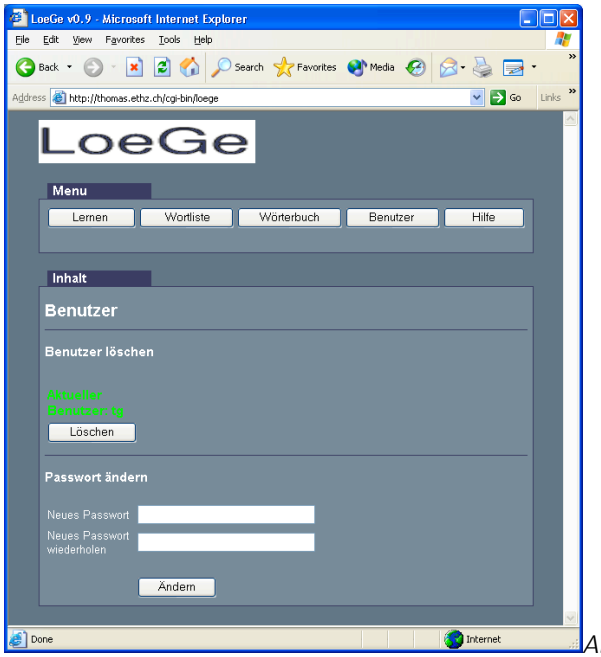


Abbildung 6

10.8 Hilfe

Informationen und die Bedienungsanleitung zu LoeGe.

11. Erfahrungsbericht/Erkenntnisse

11.1 Negatives

Während des ganzen Projekts mussten wir uns zum Teil gewisse Ziele, die wir uns gesteckt haben, neu Definieren. Wir haben gemerkt, dass unsere Analyse in gewissen Punkten etwas Optimistisch war, da unsere Erfahrung und unser Wissen in C++ noch nicht so gross ist. Anbei sind tabellarisch die Abweichungen zur Analyse aufgezeigt.

Analyse	Realität	Grund
Masterfile	Wir mussten mehrere Masterfiles machen. Genauer gesagt für jeden Buchstaben eines.	Wir hatten Probleme mit der Geschwindigkeit. Man musste bis zu 20 Sekunden warten bis ein Resultat erschien..

11.2 Positives

Wir sind beide der Meinung, dass wir sehr viel gelernt habe. Wir mussten uns wirklich in die Problematik mit dem Umgang mit HTML, Webserver und C++ auseinandersetzen.
Ausserdem war es eine Erfahrung ein solches Projekt durch zu ziehen, Rückschläge wegzustecken und andere Lösungsansätze zu suchen.

11.3 Fazit

Es müsste mehr Zeit in eine gute Analyse und Zeitplan investiert werden. Denn das ist die Basis für ein Erfolgreiches Projekt.

Der Lerneffekt war aber sehr gross und eine sehr gute Erfahrung im Hinblick auf unsere Diplomarbeit. Wir können bestimmt auch einiges im geschäftlichen Alltag wieder verwenden.

Natürlich haben wir unsere Defizite und Schwächen erkannt und werden in Zukunft versuchen die Fehler nicht mehr zu begehen.

12. Unterschriften

Das Programm und die dazugehörige Dokumentation wurden von der Thomas Gemperli und Reto Loepfe erstellt.

Thomas Gemperli

Reto Loepfe

13. Anhang

13.1 Source Code

Loege wird weiterentwickelt.
Aktuelle Versionen des Source Codes koennen unter
http://loege.ethz.ch -> Hilfe -> Download
bezogen werden.

13.1.1 main

```
/***************************************************************************
                          main.cpp  -  description
                             -------------------
    begin                : Sat Jan 04 2003
    copyright            : (C) 2003 by Reto Loepfe & Thomas Gemperli
    email                : loege@gemperli.net
 ***************************************************************************/

/***************************************************************************
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 2 of the License, or     *
 *   (at your option) any later version.                                   *
 *                                                                         *
 ***************************************************************************/

#ifdef HAVE_CONFIG_H
#include <config.h>
#endif

#include "cl_printhtml.h"

using namespace std;
using namespace cgicc;

string Version = "0.9";

int main(int argc, char *argv[])
{

        cl_printhtml eine_page(Version);
        eine_page.print();

        return EXIT_SUCCESS;

}
```

13.1.2 cl_printhtml.h

```
/***************************************************************************
                          cl_printhtml.h  -  description
                             -------------------
    begin                : Sat Jan 04 2003
```

```
   copyright          : (C) 2003 by Reto Loepfe & Thomas Gemperli
   email              : loege@gemperli.net
 ***************************************************************************/

/***************************************************************************
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 2 of the License, or     *
 *   (at your option) any later version.                                   *
 *                                                                         *
 ***************************************************************************/

#ifndef CL_PRINTHTML_H
#define CL_PRINTHTML_H

#include <iostream>
#include <fstream>
#include <string>

#include "cgicc/Cgicc.h"
#include "cgicc/HTTPHTMLHeader.h"
#include "cgicc/HTMLClasses.h"

#include "cl_login.h"
#include "cl_search.h"
#include "cl_learn.h"


using namespace std;
using namespace cgicc;

class cl_printhtml
{

public:
        cl_printhtml(const string& version);
        ~cl_printhtml();

        void print();
        void print_line();
        void print_header();
        void print_nav();
        void print_foot();
        void print_contentWelcome();
        void print_contentError();
        void print_contentHilfe();
        void print_contentLernen();
        void print_contentLernenSet();
        void print_contentErfassen();
        void print_contentDictionary();
        void print_contentEinstellungen();
        void print_contentLogin();
        void add_user();
        void del_user();
        void change_Pw();
        void check_user();
        void query_dict();
        void set_word();
        void check_word();
        void get_word();

private:
        string m_version;
        string m_showform;
        string m_formLernenUnit;
```

```
        string m_formLernenLang;
        string m_formLernenWord;
        string m_formLernenQuestion;
        string m_formLernenAnswer;
        string m_formErfassenUnit;
        string m_formErfassenDeutsch;
        string m_formErfassenEnglish;
        string m_formDictionaryTolerant;
        string m_formDictionarySearch;
        string m_formDictionaryResult;
        string m_formUsername;
        string m_formPassword;
        string m_formNewUser;
        string m_formDelUser;
        string m_formSearchWord;
        string m_formDixLang;
        string m_formSetWord;
        string m_formGetWord;
        string m_formchangePw;
        string m_formchangePwW1;
        string m_formchangePwW2;
        string m_formCheckWord;
        string m_formLernenStatusRes;
        string m_tmpErfassenFile;

};

#endif
```

13.1.3 cl_printhtml.cpp

```
/***************************************************************************
                cl_printhtml.cpp  -  description
                  -------------------
  begin            : Sat Jan 04 2003
  copyright        : (C) 2003 by Reto Loepfe & Thomas Gemperli
  email            : loege@gemperli.net
 ***************************************************************************/

/***************************************************************************
 *                                                                         *
 *  This program is free software; you can redistribute it and/or modify  *
 *  it under the terms of the GNU General Public License as published by  *
 *  the Free Software Foundation; either version 2 of the License, or     *
 *  (at your option) any later version.                                    *
 *                                                                         *
 ***************************************************************************/

#include "cl_printhtml.h"

// Default Konstruktor
cl_printhtml::cl_printhtml(const string& version)
{
        // Ein neues Cgicc Objekt erstellen. Enthaelt die CGI-Form Daten.
        Cgicc cgi;

        // Zeiger(form_iterator) auf die einzelnen CGI-Form Elemente setzen.
        form_iterator showform = cgi.getElement("showform");
        form_iterator formLernenUnit = cgi.getElement("formLernenUnit");
        form_iterator formLernenLang = cgi.getElement("formLernenLang");
        form_iterator formLernenWord = cgi.getElement("formLernenWord");
        form_iterator formLernenQuestion = cgi.getElement("formLernenQuestion");
        form_iterator formLernenAnswer = cgi.getElement("formLernenAnswer");
        form_iterator formErfassenUnit = cgi.getElement("formErfassenUnit");
        form_iterator formErfassenDeutsch = cgi.getElement("formErfassenDeutsch");
```

```
form_iterator formErfassenEnglish = cgi.getElement("formErfassenEnglish");
form_iterator formDictionaryTolerant = cgi.getElement("formDictionaryTolerant");
form_iterator formDictionarySearch = cgi.getElement("formDictionarySearch");
form_iterator formDictionaryResult = cgi.getElement("formDictionaryResult");
form_iterator formUsername = cgi.getElement("formUsername");
form_iterator formPassword = cgi.getElement("formPassword");
form_iterator formNewUser = cgi.getElement("formNewUser");
form_iterator formDelUser = cgi.getElement("formDelUser");
form_iterator formSearchWord = cgi.getElement("formSearchWord");
form_iterator formDixLang = cgi.getElement("formDixLang");
form_iterator formSetWord = cgi.getElement("formSetWord");
form_iterator formGetWord = cgi.getElement("formGetWord");
form_iterator formchangePw = cgi.getElement("formchangePw");
form_iterator formchangePwW1 = cgi.getElement("formchangePwW1");
form_iterator formchangePwW2 = cgi.getElement("formchangePwW2");
form_iterator formCheckWord = cgi.getElement("formCheckWord");
form_iterator formLernenStatusRes = cgi.getElement("formLernenStatusRes");

// CGI-Formdaten in private Variablen abfuellen. Nur bei gueltigem Wert.
if(showform != cgi.getElements().end())
{       m_showform = **showform; }
if(formLernenUnit != cgi.getElements().end())
{       m_formLernenUnit = **formLernenUnit; }
if(formLernenLang != cgi.getElements().end())
{       m_formLernenLang = **formLernenLang; }
if(formLernenWord != cgi.getElements().end())
{       m_formLernenWord = **formLernenWord; }
if(formLernenQuestion != cgi.getElements().end())
{       m_formLernenQuestion = **formLernenQuestion; }
if(formLernenAnswer != cgi.getElements().end())
{       m_formLernenAnswer = **formLernenAnswer; }
if(formErfassenUnit != cgi.getElements().end())
{       m_formErfassenUnit = **formErfassenUnit; }
if(formErfassenDeutsch != cgi.getElements().end())
{       m_formErfassenDeutsch = **formErfassenDeutsch; }
if(formErfassenEnglish != cgi.getElements().end())
{       m_formErfassenEnglish = **formErfassenEnglish; }
if(formDictionaryTolerant != cgi.getElements().end())
{       m_formDictionaryTolerant = **formDictionaryTolerant; }
if(formDictionarySearch != cgi.getElements().end())
{       m_formDictionarySearch = **formDictionarySearch; }
if(formDictionaryResult != cgi.getElements().end())
{       m_formDictionaryResult = **formDictionaryResult; }
if(formUsername != cgi.getElements().end())
{       m_formUsername = **formUsername; }
if(formPassword != cgi.getElements().end())
{       m_formPassword = **formPassword; }
if(formNewUser != cgi.getElements().end())
{       m_formNewUser = **formNewUser; }
if(formDelUser != cgi.getElements().end())
{       m_formDelUser = **formDelUser; }
if(formSearchWord != cgi.getElements().end())
{       m_formSearchWord = **formSearchWord; }
if(formDixLang != cgi.getElements().end())
{       m_formDixLang = **formDixLang; }
if(formSetWord != cgi.getElements().end())
{       m_formSetWord = **formSetWord; }
if(formGetWord != cgi.getElements().end())
{       m_formGetWord = **formGetWord; }
if(formchangePw != cgi.getElements().end())
{       m_formchangePw = **formchangePw; }
if(formchangePwW1 != cgi.getElements().end())
{       m_formchangePwW1 = **formchangePwW1; }
if(formchangePwW2 != cgi.getElements().end())
{       m_formchangePwW2 = **formchangePwW2; }
if(formCheckWord != cgi.getElements().end())
```

```
        {       m_formCheckWord = **formCheckWord; }
        if(formLernenStatusRes != cgi.getElements().end())
        {       m_formLernenStatusRes = **formLernenStatusRes; }

        // Pfad zu den Datenfiles feststellen und in private Variable sichern.
        if(getenv("DOCUMENT_ROOT"))
        {
                m_tmpErfassenFile = getenv("DOCUMENT_ROOT");
                m_tmpErfassenFile.append("/loege/data/tmperfassen.txt");
        }

        // Einfache Versionsaenderung
        m_version = version;

}


// Default Destruktor. Der Vollstaendigkeit halber.
cl_printhtml::~cl_printhtml()
{
}


// Diese Methode wird aus dem main aufgerufen.
// Baut die Seite je nach Wert der CGI-Form Daten auf.
void cl_printhtml::print()
{
        // Header der html Seite ausgeben.
        print_header();

        // Navigation ausgeben
        print_nav();

        // Neuen Benutzer anlegen
        add_user();

        // Benutzer loeschen
        del_user();

        // Passwort neu setzen
        change_Pw();

        // Hat User eingeloggt und ist er gueltig?
        check_user();

        // Footer der html Seite ausgeben.
        print_foot();
}


// Eine horizontale Linie. Zwecks Vereinfachung und Codeersparniss.
void cl_printhtml::print_line()
{
  cout << "<table width=\"100%\" border=\"0\" \
                cellpadding=\"0\" cellspacing=\"0\">" << endl;
  cout << "<tr><td class=\"menu\">\
                <img src=\"/loege/pix/onepixel.gif\" height=\"11\" alt=\"\"></td></tr>" << endl;
  cout << "<tr><td class=\"border\">\
                <img src=\"/loege/pix/onepixel.gif\" height=\"1\" alt=\"\"></td></tr>" << endl;
  cout << "<tr><td class=\"menu\">\
                <img src=\"/loege/pix/onepixel.gif\" height=\"6\" alt=\"\"></td></tr>" << endl;
  cout << "</table>" << endl;
}

// Ausgeben des HTML-Headers
void cl_printhtml::print_header()
```

```
{
        // HTML Dokument Typ festlegen
  cout << HTTPHTMLHeader() << HTMLDoctype(HTMLDoctype::eTransitional) << endl;
  cout << html().set("lang", "en").set("dir", "ltr") << endl;

  // Header oeffnen
  cout << head() << endl;

        // Style sheet Direktiven. Vereinfachen das graphische Managament der
        // Website durch zentrale Speicherung von Darstellungsinformationen.
        cout << style() << comment() << endl;
        cout << "body{margin-left:0px;margin-right:0px;margin-top:\
                        0px;margin-bottom:0px;font-family:helvetica,arial,sans-serif;\
                        background-color:#667788;color:#ffffff;}" << endl;
        cout << "a:link{color:#d0e0f0;}" << endl;
        cout << "a:visited{color:#d0e0f0;}" << endl;
        cout << "a:active{color:#d0e0f0;}" << endl;
        cout << "p{font-family:helvetica,arial,sans-serif;\
                        font-size:12pt;color:#ffffff;}" << endl;
        cout << "p.title{margin-top:10px;margin-bottom:0px;\
                        font-family:helvetica,arial,sans-serif;\
                        font-weight:bold;font-size:14pt;color:#ffffff;}" << endl;
        cout << "p.subtitle{margin-top:10px;margin-bottom:10px;\
                        font-family:helvetica,arial,sans-serif;\
                        font-weight:bold;font-size:11pt;color:#ffffff;}" << endl;
        cout << "p.good{margin-top:10px;margin-bottom:10px;\
                        font-family:helvetica,arial,sans-serif;\
                        font-weight:bold;font-size:11pt;color:#00ff00;}" << endl;
        cout << "p.error{margin-top:10px;margin-bottom:10px;\
                        font-family:helvetica,arial,sans-serif;\
                        font-weight:bold;font-size:11pt;color:#ee7733;}" << endl;
        cout << "pre{font-family:courier,sans-serif;font-size:10pt;\
                        text-align:left;color:#ffffff;}" << endl;
        cout << "td{font-family:helvetica,arial,sans-serif;\
                        font-size:10pt;color:#ffffff;}" << endl;
        cout << "td.menutop{font-family:helvetica,arial,sans-serif;\
                        font-size:11pt;background-color:#3e4266;color:#ffffff;}" << endl;
        cout << "td.content{font-family:helvetica,arial,sans-serif;\
                        font-size:11pt;background-color:#ffffff;color:#3e4266;}" << endl;
        cout << "td.border{background-color:#3e4266;color:#ffffff;}" << endl;
        cout << "td.menu{font-family:helvetica,arial,sans-serif;font-size:10pt;\
                        background-color:#7a8a9a;color:#ffffff;}" << endl;
        cout << "td.menumain{font-family:helvetica,arial,sans-serif;font-size:12pt;\
                        background-color:#7a8a9a;color:#ffffff;}" << endl;
        cout << "td.background{background-color:#667788;color:#667788;}" << endl;
  cout << comment() << style() << endl;

        // Titel ausgeben und Header schliessen
        cout << title() << "LoeGe v" << m_version << title() << endl;
  cout << head() << endl;
}

// Die HTML Ausgabe des Menus.
void cl_printhtml::print_nav()
{
  // HTML body oeffnen
  cout << body() << endl;

        // Ausgeben des Navigationsteiles der Website
        cout << "<!-- main table -->" << endl;
  cout << "<table border=\"0\" width=\"90%\" cellspacing=\"0\" \
                  cellpadding=\"10\" align=\"center\">" << endl;
  cout << "<!-- logo -->" << endl;
  cout << "<tr><td>" << endl;

  // Laden des Logos
```

```
    cout << "<a href=\"loege\"><img src=\"/loege/pix/logo-loege.png\" border=\"0\" \
                                alt=\"LoeGe\"></a>" << endl;
    cout << "</td></tr>" << endl;
    cout << "<!-- menu -->" << endl;
    cout << "<tr><td>" << endl;
    cout << "<table width=\"130\" border=\"0\" cellpadding=\"0\" \
                        cellspacing=\"0\">" << endl;
    cout << "<tr valign=\"top\">" << endl;
    cout << "<td class=\"background\" width=\"10\">\
                        <img src=\"/loege/pix/onepixel.gif\" alt=\"\"></td>" << endl;
    cout << "<td class=\"menutop\" width=\"120\">\
                        <img src=\"/loege/pix/onepixel.gif\" width=\"6\" alt=\"\"><b>Menu</b></td>" << endl;
    cout << "</tr>" << endl;
    cout << "</table>" << endl;
    cout << "<table width=\"100%\" border=\"0\" cellpadding=\"0\" \
                        cellspacing=\"0\">" << endl;
    cout << "<tr>" << endl;
    cout << "<td class=\"border\" colspan=\"3\">\
                        <img src=\"/loege/pix/onepixel.gif\" height=\"1\" alt=\"\"></td>" << endl;
    cout << "</tr>" << endl;
    cout << "<tr>" << endl;
    cout << "<td class=\"border\">\
                        <img src=\"/loege/pix/onepixel.gif\" width=\"1\" alt=\"\"></td>" << endl;
    cout << "<td width=\"100%\">" << endl;
    cout << "<table width=\"100%\" border=\"0\" \
                        cellpadding=\"6\" cellspacing=\"0\">" << endl;
    cout << "<tr>" << endl;
    cout << "<td class=\"menu\">" << endl;
            cout << "<table><tr>" << endl;

            // Zielseite, Username und Passwort werden durch Senden des (Navigations) Formulars uebergeben
            cout << "<td width=\"20%\"><form action=\"loege\" method=\"Post\" name=\"gotoLernen\">" << endl;
            cout << "<input type=\"hidden\" name=\"formUsername\" value=\"" << m_formUsername << "\">" <<
endl;
            cout << "<input type=\"hidden\" name=\"formPassword\" value=\"" << m_formPassword << "\">" <<
endl;
            cout << "<input type=\"hidden\" name=\"showform\" value=\"lernen\">" << endl;
            cout << "<input type=\"submit\" name=\"action\" value=\"   Lernen    \">" << endl;
            cout << "</form></td>" << endl;
            cout << "<td width=\"20%\"><form action=\"loege\" method=\"Post\" name=\"gotoErfassen\">" << endl;
            cout << "<input type=\"hidden\" name=\"formUsername\" value=\"" << m_formUsername << "\">" <<
endl;
            cout << "<input type=\"hidden\" name=\"formPassword\" value=\"" << m_formPassword << "\">" <<
endl;
            cout << "<input type=\"hidden\" name=\"showform\" value=\"erfassen\">" << endl;
            cout << "<input type=\"submit\" name=\"action\" value=\"   Wortliste   \">" << endl;
            cout << "</form></td>" << endl;
            cout << "<td width=\"20%\"><form action=\"loege\" method=\"Post\" name=\"gotoDictionary\">" <<
endl;
            cout << "<input type=\"hidden\" name=\"formUsername\" value=\"" << m_formUsername << "\">" <<
endl;
            cout << "<input type=\"hidden\" name=\"formPassword\" value=\"" << m_formPassword << "\">" <<
endl;
            cout << "<input type=\"hidden\" name=\"showform\" value=\"dictionary\">" << endl;
            cout << "<input type=\"submit\" name=\"action\" value=\"  W&ouml;rterbuch  \">" << endl;
            cout << "</form></td>" << endl;
            cout << "<td width=\"20%\"><form action=\"loege\" method=\"Post\" name=\"gotoEinstellungen\">" <<
endl;
            cout << "<input type=\"hidden\" name=\"formUsername\" value=\"" << m_formUsername << "\">" <<
endl;
            cout << "<input type=\"hidden\" name=\"formPassword\" value=\"" << m_formPassword << "\">" <<
endl;
            cout << "<input type=\"hidden\" name=\"showform\" value=\"einstellungen\">" << endl;
            cout << "<input type=\"submit\" name=\"action\" value=\"   Benutzer    \">" << endl;
            cout << "</form></td>" << endl;
            cout << "<td width=\"20%\"><form action=\"loege\" method=\"Post\" name=\"gotoHilfe\">" << endl;
```

```
            cout << "<input type=\"hidden\" name=\"formUsername\" value=\"" << m_formUsername << "\">" <<
endl;
            cout << "<input type=\"hidden\" name=\"formPassword\" value=\"" << m_formPassword << "\">" <<
endl;
            cout << "<input type=\"hidden\" name=\"showform\" value=\"hilfe\">" << endl;
            cout << "<input type=\"submit\" name=\"action\" value=\"     Hilfe     \">" << endl;
            cout << "</form></td>" << endl;
            cout << "</tr></table>" << endl;
   cout << "</td>" << endl;
   cout << "</tr>" << endl;
   cout << "</table>" << endl;
   cout << "</td>" << endl;
   cout << "<td class=\"border\">\
                        <img src=\"/loege/pix/onepixel.gif\" width=\"1\" alt=\"\"></td>" << endl;
   cout << "</tr>" << endl;
   cout << "<tr>" << endl;
   cout << "<td class=\"border\" colspan=\"3\">\
                                <img src=\"/loege/pix/onepixel.gif\" height=\"1\" alt=\"\"></td>" << endl;
   cout << "</tr>" << endl;
   cout << "</table>" << endl;
   cout << "</td></tr>" << endl;

  // Der "Inhaltsbereich" beginnt
   cout << "<!-- content -->" << endl;
   cout << "<tr><td>" << endl;
   cout << "<table width=\"130\" border=\"0\" \
                        cellpadding=\"0\" cellspacing=\"0\">" << endl;
   cout << "<tr valign=\"top\">" << endl;
   cout << "<td class=\"background\" width=\"10\">\
                        <img src=\"/loege/pix/onepixel.gif\" alt=\"\"></td>" << endl;
   cout << "<td class=\"menutop\" width=\"120\">\
                        <img src=\"/loege/pix/onepixel.gif\" width=\"6\" \
                        alt=\"\"><b>Inhalt</b></td>" << endl;
   cout << "</tr>" << endl;
   cout << "</table>" << endl;
   cout << "<table width=\"100%\" border=\"0\" \
                        cellpadding=\"0\" cellspacing=\"0\">" << endl;
   cout << "<tr>" << endl;
   cout << "<td class=\"border\" colspan=\"3\">\
                        <img src=\"/loege/pix/onepixel.gif\" height=\"1\" \
                        alt=\"\"></td>" << endl;
   cout << "</tr>" << endl;
   cout << "<tr>" << endl;
   cout << "<td class=\"border\">\
                        <img src=\"/loege/pix/onepixel.gif\" width=\"1\" alt=\"\"></td>" << endl;
   cout << "<td width=\"100%\">" << endl;
   cout << "<table width=\"100%\" border=\"0\" \
                        cellpadding=\"6\" cellspacing=\"0\">" << endl;
   cout << "<tr>" << endl;
   cout << "<td class=\"menumain\">" << endl;
}

// Die HTML Ausgabe des Fussteiles der Website
void cl_printhtml::print_foot()
{
   cout << "</td>" << endl;
   cout << "</tr>" << endl;
   cout << "</table>" << endl;
   cout << "</td>" << endl;
   cout << "<td class=\"border\">\
                        <img src=\"/loege/pix/onepixel.gif\" width=\"1\" alt=\"\"></td>" << endl;
   cout << "</tr>" << endl;
   cout << "<tr>" << endl;
   cout << "<td class=\"border\" colspan=\"3\">\
                        <img src=\"/loege/pix/onepixel.gif\" height=\"1\" alt=\"\"></td>" << endl;
   cout << "</tr>" << endl;
```

```
   cout << "</table>" << endl;
   cout << "</td></tr>" << endl;
   cout << "</table>" << endl;
   // HTML Body und Dokument schliessen
            cout << body() << html() << endl;
}

// Die HTML Ausgabe des Willkommenseite. Erscheint nach erfolgreichem Login.
void cl_printhtml::print_contentWelcome()
{
            cout << "<p class=\"title\">LoeGe</p>" << endl;
            print_line();
   cout << "<p class=\"subtitle\">English - Deutsch Lernprogramm</p>" << endl;
   cout << "<img src=\"/loege/pix/lazy.gif\" alt=\"under construction\">" << endl;
   cout << "<p>Einfacher Englisch lernen.</p>" << endl;
   cout << "<p> </p>" << endl;
}

// Die HTML Ausgabe bei einem Fehler. Darf nicht vorkommen.
void cl_printhtml::print_contentError()
{
            cout << "<p class=\"title\">Fehler!</p>" << endl;
            print_line();
   cout << "<p>Gib dem Programmierer einen Tritt in den Arsch.</p>" << endl;
   cout << "<p> </p>" << endl;
}

// Die HTML Ausgabe der Hilfe Texte.
void cl_printhtml::print_contentHilfe()
{
            cout << "<p class=\"title\">Hilfe</p>" << endl;
            print_line();
   cout << "<p class=\"subtitle\">Zweck</p>" << endl;
   cout << "<p>LoeGe ist ein Vokabular Lernprogramm, das " << endl;
   cout << "zum Erlernen von englischen W&ouml;rtern benutzt wird.<br>" << endl;
   cout << "Weiter kann es als normales W&ouml;rterbuch verwendet werden.\
                        </p><p>LoeGe ist das Resultat einer Vordiplomarbeit zum Informatiker TS. <br>\
                        Rahmenbedingungen der Arbeit waren die Implementation in C++ und \
            die Verwendung der Konsole zur Daten Ein- und Ausgabe.</p><p>\
         Wir erweiterten die Ein- und Ausgabe um HTML Steuerelemente und \
         konnten so ein GUI realisieren. </p><p>Weiter verwendeten wir bei der \
         Implementation nur ANSI Standard C++. <br>Wir erreichen so ein Maximum an \
         Portabilit&auml;t (und Lerneffekt f&uuml;r uns 8).</p>" << endl;
            print_line();
   cout << "<p class=\"subtitle\">Menu</p>" << endl;
   cout << "<p>Mittels Klick der Buttons werden die verschiedenen Funktionen von LoeGe " << endl;
   cout << "erreicht.<br>Durch ein Klick auf das Logo loggen Sie sich aus.</p>" << endl;
            print_line();
   cout << "<p class=\"subtitle\">Lernen</p>" << endl;
   cout << "<p>Dies ist der Vokabeltrainer. <br>Zu Beginn erscheint eine Maske auf der " << endl;
   cout << "folgende Optionen gesetzt werden m&uuml;ssen:</p>" << endl;
   cout << "<ul><li><b>Unit</b> Selektieren Sie einzelne oder alle Lektionen aus Ihrer\
                        pers&ouml;nlichen Wortliste.</li><li><b>Status</b> W&auml;hlen Sie entweder \
                        alle W&ouml;rter, die, die Sie beherrschen oder jene, die noch nicht ganz \
                        sitzen.</li><li><b>Sprachwahl</b> Lernen Sie Deutsch-English oder English-Deutsch\
            </li></ul>" << endl;
   cout << "<p>Danach folgt das Abfragefenster. Im oberen Teil werden nochmals die gesetzten \
                        Suchoptionen angezeigt. </p><p> Im Mittelteil geschieht die eigentliche Abfrage. \
                        Falls Sie die \"Umlaute einf&uuml;gen\" Funktion benutzen wollen, muss ihr \
            Browser Java-Script aktiviert haben. </p><p>Nachdem auf &Uuml;berpr&uuml;fen geklickt \
            wurde, wird das Ergebniss Ihrer &Uuml;bersetzung im untersten, dem Resultat Teil \
            ausgegeben. <br>LoeGe zeigt Ihnen die richtige L&ouml;sung in jedem Fall an. \
                        </p><p>Wenn Sie eine Abfrage richtig beantwortet haben, wird deren Status in \
                        Ihrer Wortliste automatisch auf \"sitzt\" gesetzt." << endl;
            print_line();
   cout << "<p class=\"subtitle\">Wortliste</p>" << endl;
```

```
   cout << "<p>Diese Maske dient zum Erfassen von Vokabeln in die pers&ouml;nliche Wortliste. <br>\
                        Die Wortliste ist, wie die meisten Englisch Kurse auch, nach Lektionen\
                        sortiert. </p><p>Die hier erfassten Vokabeln stehen Ihnen im \"Lernen Modus\" \
            zur Verf&uuml;gung.  </p>" << endl;
   print_line();
   cout << "<p class=\"subtitle\">W&ouml;rterbuch</p>" << endl;
   cout << "<p>Dies ist ein simpler, elektronischer &Uuml;bersetzer.<br>\
                        Suchen Sie nach deutschen oder englischen W&ouml;rtern. </p><p>\
                        Falls Sie die \"Umlaute einf&uuml;gen\" Funktion benutzen wollen, muss ihr \
            Browser Java-Script aktiviert haben. </p>" << endl;
   print_line();
   cout << "<p class=\"subtitle\">Benutzer</p>" << endl;
   cout << "<p>Diese Maske erlaubt Ihnen den aktuellen Benutzer zu \
                        l&ouml;schen sowie dessen Passwort neu zu setzen.</p>" << endl;
   print_line();
   cout << "<p class=\"subtitle\">Hilfe</p>" << endl;
   cout << "<p>Dieses Dokument. <br>Informationen und die Bedienungsanleitung zu LoeGe.</p>" << endl;
   print_line();
   cout << "<p class=\"subtitle\">About</p>" << endl;
   cout << "<p><a href=\"http://thomas.ethz.ch/loege/\">LoeGe</a> \
                        Version " << m_version << "</p>" << endl;
            cout << "<p>(C) 2003 by Reto Loepfe & Thomas Gemperli<br>" << endl;

            // loege@gemperli.net ist ein Mailverteiler an die Projektbeteiligten
   cout << "Kontakt: <a href=\"mailto:loege@gemperli.net\">\
                        loege@gemperli.net</a></p>" << endl;
   cout << "<p> </p>" << endl;
}

// Die HTML Ausgabe der Optionenseite der Funktion "Woerter lernen"
void cl_printhtml::print_contentLernen()
{
            cout << "<p class=\"title\">Lernmodus Optionen</p>" << endl;
            print_line();
   cout << "<p class=\"subtitle\">Bitte Optionen setzen</p>" << endl;
            cout << "<form action=\"loege\" method=\"Post\" name=\"formLernenParam\">" << endl;
            cout << "<table><tr>" << endl;
            cout << "<td width=\"100\">Unit</td>" << endl;
            cout << "<td><select name=\"formLernenUnit\">" << endl;
            cout << "<option selected=\"Alle\">Alle" << endl;
            for(int i = 1; i < 100; ++i)
            {
                        cout << "<option value=\"" << i << "\"" << ">" << i << endl;
            }
            cout << "</select></td></tr>" << endl;
            cout << "<tr><td width=\"100\">Status</td>" << endl;
            cout << "<td><select name=\"formLernenWord\">" << endl;
            cout << "<option value=\"Alle\">Alle" << endl;
            cout << "<option selected=\"Nicht Gewusste\">Nicht Gewusste" << endl;
            cout << "<option value=\"Gewusste\">Gewusste" << endl;
            cout << "</select></td></tr>" << endl;
            cout << "<tr><td width=\"100\">Sprachwahl</td>" << endl;
            cout << "<td><select name=\"formLernenLang\">" << endl;
            cout << "<option selected=\"De-En\">De-En" << endl;
            cout << "<option value=\"En-De\">En-De" << endl;
            cout << "</select></td>" << endl;
            cout << "</tr></table>" << endl;
            print_line();
            cout << "<table><tr>" << endl;
            cout << "<input type=\"hidden\" name=\"formUsername\" value=\"" << m_formUsername << "\">" <<
endl;
            cout << "<input type=\"hidden\" name=\"formPassword\" value=\"" << m_formPassword << "\">" <<
endl;
            cout << "<input type=\"hidden\" name=\"formGetWord\" value=\"on\">" << endl;
            cout << "<td width=\"100\"><input type=\"hidden\" name=\"showform\" value=\"lernenSet\"></td>" <<
endl;
```

```
		cout << "<td><input type=\"submit\" name=\"action\" value=\"    Start     \"></td>" << endl;
		cout << "</tr></table></form>" << endl;
		cout << "<p> </p>" << endl;
}

// Die HTML Ausgabe der Maske "Woerter lernen"
void cl_printhtml::print_contentLernenSet()
{
		cout << "<p class=\"title\">Lernmodus</p>" << endl;
		print_line();
  cout << "<p class=\"subtitle\">Gew&auml;hlte Optionen</p>" << endl;
		cout << "<table><tr>" << endl;
		cout << "<form action=\"loege\" method=\"Post\" name=\"formLernenSet\">" << endl;
		cout << "<td width=\"100\">Unit</td>" << endl;
		cout << "<td width=\"100\">" << m_formLernenUnit << "</td></tr>" << endl;
		cout << "<tr><td width=\"100\">Status</td>" << endl;
		cout << "<td width=\"100\">" << m_formLernenWord << "</td></tr>" << endl;
		cout << "<tr><td width=\"100\">Sprachwahl</td>" << endl;
		cout << "<td width=\"100\">" << m_formLernenLang << "</td></tr></table>" << endl;
		print_line();

		// Der Abfragebereich der Maske "Woerter lernen"
  cout << "<p class=\"subtitle\">Abfrage</p>" << endl;
		cout << "<table><tr>" << endl;

		// Status des letzten Aufrufes der Seite abfragen, default Optionen
		// entsprechend setzen und ausgeben
  if(m_formLernenLang == "De-En")
  {
		cout << "<td width=\"100\">Deutsch</td><td>" << endl;
  }
  else
  {
		cout << "<td width=\"100\">English</td><td>" << endl;
  }

  // Wort zum Abfragen holen.
  get_word();

		// Status des letzten Aufrufes der Seite abfragen, default Optionen
		// entsprechend setzen und ausgeben
  if(m_formLernenLang == "En-De")
  {
		cout << "</td></tr><tr><td width=\"100\">Deutsch</td>" << endl;
  }
  else
  {
		cout << "</td></tr><tr><td width=\"100\">English</td>" << endl;
  }
		cout << "<td><input type=\"text\" name=\"formLernenAnswer\" size=\"30\">" << endl;
  cout << "</td></tr><tr>" << endl;
		cout << "<td width=\"100\">Umlaute einf&uuml;gen</td>" << endl;
  cout << "<td><input type=\"button\" value=\" &auml; \" onClick=\"this.form.formLernenAnswer.value\
                  += 'ä';\">" << endl;
  cout << "<input type=\"button\" value=\" &ouml; \" onClick=\"this.form.formLernenAnswer.value\
                  += 'ö';\">" << endl;
  cout << "<input type=\"button\" value=\" &uuml; \" onClick=\"this.form.formLernenAnswer.value\
                        += 'ü';\">" << endl;
  cout << "</td></tr><tr>" << endl;
		cout << "<td width=\"100\"> </td></tr><tr>" << endl;
		cout << "<td width=\"100\"><input type=\"hidden\" name=\"showform\" value=\"lernenSet\">" << endl;
		cout << "<input type=\"hidden\" name=\"formGetWord\" value=\"on\">" << endl;
  cout << "<input type=\"hidden\" name=\"formCheckWord\" value=\"on\">" << endl;
		cout << "<input type=\"hidden\" name=\"formUsername\" value=\"" << m_formUsername << "\">" <<
endl;
```

```
		cout << "<input type=\"hidden\" name=\"formPassword\" value=\"" << m_formPassword << "\">" <<
endl;
		cout << "<input type=\"hidden\" name=\"formLernenUnit\" value=\"" << m_formLernenUnit << "\">" <<
endl;
		cout << "<input type=\"hidden\" name=\"formLernenWord\" value=\"" << m_formLernenWord << "\">"
<< endl;
		cout << "<input type=\"hidden\" name=\"formLernenLang\" value=\"" << m_formLernenLang <<
"\"></td>" << endl;
		cout << "<td><input type=\"submit\" name=\"action\" value=\"  &Uuml;berpr&uuml;fen  \"></td>" <<
endl;
		cout << "</form></tr></table>" << endl;
		print_line();

		// Der Statusbereich der Maske "Woerter lernen"
   cout << "<p class=\"subtitle\">Resultat</p>" << endl;

		// Rueckmeldung der Resulte an den Benutzer
		check_word();

		cout << "<p> </p>" << endl;
}

// Die HTML Ausgabe der Maske "Wortliste"
void cl_printhtml::print_contentErfassen()
{
		// Falls vorhanden, Wort erfassen.
		set_word();

		cout << "<p class=\"title\">Pers&ouml;nliche Wortliste</p>" << endl;
		print_line();
		cout << "<p class=\"subtitle\">W&ouml;rter erfassen</p>" << endl;
		cout << "<table><tr>" << endl;
		cout << "<form action=\"loege\" method=\"Post\" name=\"formErfassen\">" << endl;
		cout << "<td width=\"100\">Unit</td>" << endl;
		cout << "<td><select name=\"formErfassenUnit\">" << endl;

		for(int i = 1; i < 100; ++i)
		{
			ofstream outFile(m_tmpErfassenFile.c_str());
			if(outFile.is_open())
			{
				outFile << i << ";" << endl;
				outFile.close();
			}

			string g;
			ifstream inFile(m_tmpErfassenFile.c_str());
			if(inFile.is_open())
			{
				getline(inFile,g,';');
				inFile.close();
			}

			if(m_formErfassenUnit == g)
			{
				cout << "<option selected=\"" << i << "\"" << ">" << i << endl;
			}
			else
			{
				cout << "<option value=\"" << i << "\"" << ">" << i << endl;
			}
		}
		cout << "</select></td></tr><tr>" << endl;
		cout << "<td width=\"100\">Deutsch</td>" << endl;
		cout << "<td><input type=\"text\" name=\"formErfassenDeutsch\"\
				size=\"30\"></td></tr><tr>" << endl;
```

```
		cout << "<td width=\"100\">English</td>" << endl;
		cout << "<td><input type=\"text\" name=\"formErfassenEnglish\"\
				size=\"30\"></td></tr><tr>" << endl;
		cout << "<td width=\"100\"> </td></tr><tr>" << endl;
		cout << "<input type=\"hidden\" name=\"formSetWord\" value=\"on\">" << endl;
		cout << "<input type=\"hidden\" name=\"formUsername\" value=\"" << m_formUsername << "\">" <<
endl;
		cout << "<input type=\"hidden\" name=\"formPassword\" value=\"" << m_formPassword << "\">" <<
endl;
		cout << "<td width=\"100\"><input type=\"hidden\" name=\"showform\" value=\"erfassen\"></td>" <<
endl;
		cout << "<td><input type=\"submit\" name=\"action\" value=\"  Erfassen  \"></td>" << endl;
		cout << "</form></tr></table>" << endl;
  cout << "<p> </p>" << endl;
}

// Die HTML Ausgabe der Maske "Woerterbuch"
void cl_printhtml::print_contentDictionary()
{
		cout << "<p class=\"title\">W&ouml;rterbuch</p>" << endl;
		print_line();
		cout << "<form action=\"loege\" method=\"Post\" name=\"formDictionary\">" << endl;
		cout << "<table><tr>" << endl;

		cout << "<td width=\"100\">Sprachwahl</td>" << endl;
		cout << "<td><select name=\"formDixLang\">" << endl;
		if(m_formDixLang == "De-En")
		{
			cout << "<option selected=\"De-En\">De-En" << endl;
			cout << "<option value=\"En-De\">En-De" << endl;
		}
		else
		{
			cout << "<option value=\"De-En\">De-En" << endl;
			cout << "<option selected=\"En-De\">En-De" << endl;
		}
		cout << "</select></td>" << endl;

		cout << "</tr><tr>" << endl;
		cout << "<td width=\"100\">Suchwort</td>" << endl;
		cout << "<td><input type=\"text\" name=\"formDictionarySearch\"\
				size=\"30\"></td></tr><tr>" << endl;

		cout << "<td width=\"100\">Umlaute einf&uuml;gen</td>" << endl;
  cout << "<td><input type=\"button\" value=\" &auml; \" onClick=\"this.form.formDictionarySearch.value\
			+= 'ä';\">" << endl;
  cout << "<input type=\"button\" value=\" &ouml; \" onClick=\"this.form.formDictionarySearch.value\
			+= 'ö';\">" << endl;
  cout << "<input type=\"button\" value=\" &uuml; \" onClick=\"this.form.formDictionarySearch.value\
			+= 'ü';\"></td></tr><tr>" << endl;


		cout << "<td width=\"100\"><p> </p></td>" << endl;
		cout << "</tr><tr><td width=\"100\"> </td></tr><tr>" << endl;
		cout << "<input type=\"hidden\" name=\"formSearchWord\" value=\"on\">" << endl;
		cout << "<td width=\"100\"><input type=\"hidden\" name=\"showform\" value=\"dictionary\"></td>" <<
endl;
		cout << "<input type=\"hidden\" name=\"formUsername\" value=\"" << m_formUsername << "\">" <<
endl;
		cout << "<input type=\"hidden\" name=\"formPassword\" value=\"" << m_formPassword << "\">" <<
endl;
		cout << "<td><input type=\"submit\" name=\"action\" value=\"  Suchen  \"></td>" << endl;
		cout << "</form></tr></table>" << endl;

  print_line();
  cout << "<table><tr>" << endl;
```

```
   cout << "<td width=\"100\">Ergebniss(e)</td><td>" << endl;

            // Dictionary Funktion aufrufen, Suchwort abfragen
            query_dict();

            cout << "</td></tr></table>" << endl;
            cout << "<br>" << endl;
}

// Die HTML Ausgabe der Maske "Benutzer"
void cl_printhtml::print_contentEinstellungen()
{
            cout << "<p class=\"title\">Benutzer</p>" << endl;
            print_line();
   cout << "<p class=\"subtitle\">Benutzer l&ouml;schen</p>" << endl;
            cout << "<form action=\"loege\" method=\"Post\" name=\"delUser\">" << endl;
            cout << "<table><tr>" << endl;
            cout << "<td width=\"100\"><p class=\"good\">Aktueller Benutzer: " << m_formUsername << "</p>" <<
endl;
            cout << "<input type=\"hidden\" name=\"formUsername\" value=\"" << m_formUsername << "\">" <<
endl;
            cout << "<input type=\"hidden\" name=\"formPassword\" value=\"" << m_formPassword << "\">" <<
endl;
            cout << "<input type=\"hidden\" name=\"showform\" value=\"einstellungen\">" << endl;
            cout << "<input type=\"hidden\" name=\"formDelUser\" value=\"on\"></td></tr>" << endl;
            cout << "<tr><td width=\"100\"><input type=\"submit\" name=\"action\" \
                              value=\"   L&ouml;schen   \"></td></tr>" << endl;
            cout << "</form></table>" << endl;
            print_line();

            // Paaswort aendern
   cout << "<p class=\"subtitle\">Passwort &auml;ndern</p>" << endl;
            cout << "<form action=\"loege\" method=\"Post\" name=\"changePw\">" << endl;
            cout << "<table><tr>" << endl;
            cout << "<td width=\"100\">Neues Passwort</td>" << endl;
            cout << "<td><input type=\"password\" name=\"formchangePwW1\"\
                               size=\"30\"></td></tr><tr>" << endl;
            cout << "<td width=\"100\">Neues Passwort wiederholen</td>" << endl;
            cout << "<td><input type=\"password\" name=\"formchangePwW2\"\
                               size=\"30\"></td></tr><tr><td> </td></tr><tr>" << endl;
            cout << "<td width=\"100\"><input type=\"hidden\" \
                              name=\"formUsername\" value=\"" << m_formUsername << "\">" << endl;
            cout << "<input type=\"hidden\" name=\"formPassword\"\
                              value=\"" << m_formPassword << "\">" << endl;
            cout << "<input type=\"hidden\" name=\"showform\" value=\"einstellungen\">" << endl;
            cout << "<input type=\"hidden\" name=\"formchangePw\" value=\"on\"></td>" << endl;
            cout << "<td width=\"100\"><input type=\"submit\" name=\"action\" \
                              value=\"   &Auml;ndern   \"></td></tr>" << endl;
            cout << "</form></table>" << endl;

}

// Die HTML Ausgabe der Login Maske
void cl_printhtml::print_contentLogin()
{
            cout << "<p class=\"title\">Login</p>" << endl;
            print_line();
   cout << "<p class=\"subtitle\">Bitte Benutzerdaten eingeben.</p>" << endl;
            cout << "<table><tr>" << endl;
            cout << "<form action=\"loege\" method=\"Post\" name=\"Login\">" << endl;
            cout << "<td width=\"100\">Benutzername</td>" << endl;
            cout << "<td><input type=\"text\" name=\"formUsername\" size=\"30\"></td></tr>" << endl;
            cout << "<tr><td width=\"100\">Passwort</td>" << endl;
            cout << "<td><input type=\"password\" name=\"formPassword\" size=\"30\"></td></tr>" << endl;
   cout << "<tr><td width=\"100\">Neuer Benutzer</td>" << endl;
```

```
   // Falls die folgende Checkbox true ist, wird ein Neuer Benutzer angelegt
            cout << "<td><input type=\"checkbox\" name=\"formNewUser\"></td></tr>" << endl;
            cout << "<tr><td width=\"100\"> </td></tr><tr>" << endl;
            cout << "<td width=\"100\"><input type=\"hidden\" name=\"showform\" value=\"\"></td>" << endl;
            cout << "<td><input type=\"submit\" name=\"action\" value=\"   Einloggen   \"></td></tr>" << endl;
            cout << "</form></table>" << endl;
}

// Ruft die Methode setUser aus cl_login auf
void cl_printhtml::add_user()
{
            // Objekt ein_login der Klasse cl_login erstellen
             cl_login ein_login;

            // Nur Ausfuehren, wenn ein neuer Benutzer angelegt werden soll.
   if(m_formNewUser == "on" )
   {
            // Ungueltige Zeichen im Benutzername und Passwort abfangen (" ; usw)
                        if(m_formUsername == "" || m_formPassword == "")
                        {
                        cout << "<p class=\"error\">Benutzername/Passwort ung&uuml;ltig \
                        oder Benutzer bereits vorhanden.</p>" << endl;
            print_line();
                        }
                        else
                        {
                                    //setUser gibt bei Erfolg true zurueck
                        if(ein_login.setUser(m_formUsername,m_formPassword))
                        {
                                    cout << "<p class=\"good\">Benutzer \
                                                " << m_formUsername << " wurde erfolgreich angelegt.</p>" << endl;
                         print_line();
              }
                 else
                 {
                                    cout << "<p class=\"error\">Benutzername/Passwort ung&uuml;ltig \
                                    oder Benutzer bereits vorhanden.</p>" << endl;
                        print_line();
                        }
              }
   }
}

// Ruft die Methode delUser aus cl_login auf
void cl_printhtml::del_user()
{
            // Objekt ein_login der Klasse cl_login erstellen
            cl_login ein_login;

            // Nur Ausfuehren, wenn ein Benutzer geloescht werden soll.
            if(m_formDelUser == "on" )
  {
            //delUser gibt bei Erfolg true zurueck
            if(ein_login.delUser(m_formUsername,m_formPassword))
            {
                                    cout << "<p class=\"good\">Benutzer \
                        " << m_formUsername << " wurde erfolgreich gel&ouml;scht.</p>" << endl;
              }
              else
              {
                                    cout << "<p class=\"error\">Benutzer \
                        " << m_formUsername << " konnte nicht gel&ouml;scht werden.</p>" << endl;
              }
   }
}
```

```
// Ruft die Methode changePw aus cl_login auf
void cl_printhtml::change_Pw()
{
	// Objekt ein_login der Klasse cl_login erstellen
	cl_login ein_login;

	// Nur Ausfuehren, wenn ein Passwort gesetzt werden soll.
	if(m_formchangePw == "on")
  {
	if(m_formchangePwW1 == m_formchangePwW2 && m_formchangePwW1 != "")
	{
		if(ein_login.changePw(m_formUsername,m_formPassword,m_formchangePwW1))
		{
			cout << "<p class=\"good\">Passwort von Benutzer \
				" << m_formUsername << " wurde erfolgreich gesetzt.</p>" << endl;
			print_line();
		}
		else
		{
				cout << "<p class=\"error\">Passwort von Benutzer \
			" << m_formUsername << " konnte nicht gesetzt werden.</p>" << endl;
			}
		 }
		 else
		 {
			cout << "<p class=\"error\">Passw&ouml;rter stimmen nicht &uuml;berein\
					oder sind ung&uuml;ltig.</p>" << endl;
		 }
   }
}

// Login Daten des Benutzers ueberpruefen. Wird bei jedem Aufruf von loege ausgefuehrt.
void cl_printhtml::check_user()
{
	cl_login ein_login;

	//getLogin gibt bei Erfolg true zurueck
	if(ein_login.getLogin(m_formUsername,m_formPassword))
	{
		// Falls User authentifiziert ist, gewuenschte Seite ausgeben
		if(m_showform == "")
		{
			print_contentWelcome();
		}
		else if(m_showform == "lernen")
	  {
		print_contentLernen();
	  }
		else if(m_showform == "lernenSet")
	  {
		print_contentLernenSet();
	  }
	  else if(m_showform == "erfassen")
	{
		print_contentErfassen();
	  }
	  else if(m_showform == "dictionary")
	  {
		print_contentDictionary();
	  }
	  else if(m_showform == "einstellungen")
	  {
		print_contentEinstellungen();
	  }
	  else if(m_showform == "hilfe")
	  {
```

```
                print_contentHilfe();
            }
                else
                {
                        print_contentError();
                }
        }
        else
        {
                print_contentLogin();
        }
}

// Ruft getEnWord rsp. getDeWord aus cl_search auf
void cl_printhtml::query_dict()
{
        cl_search eineSuche;

        // Nur ausfuehren, wenn User den Dictionary benutzt
        if(m_formSearchWord == "on" )
  {
        // Zu gewuenschter Spache Suchfunktion aufrufen
        if(m_formDixLang == "De-En")
        {
                        m_formDictionaryResult = eineSuche.getEnWord(m_formDictionarySearch);
                }
                else
                {
                        m_formDictionaryResult = eineSuche.getDeWord(m_formDictionarySearch);
                }

                // Resultate der Suche ausgeben.
                if(m_formDictionaryResult == "Suchwort nicht gefunden")
                {
                        cout << "<p class=\"error\">" << m_formDictionaryResult << "</p>" << endl;
                }
                else
                {
                        cout << "<p class=\"good\">" << m_formDictionaryResult << "</p>" << endl;
                }
        }
}

// Ruft setDEWord aus cl_learn auf
void cl_printhtml::set_word()
{
        // Objekt einLearn der Klasse cl_learn erstellen.
        cl_learn einLern;

        // Nur ausfuehren, wenn User ein Wort setzen will
        if(m_formSetWord == "on" )
  {
        // Erfassen von leeren Werten verhindern
        if(m_formErfassenDeutsch != "" && m_formErfassenEnglish != "")
        {

        if(einLern.setDEWord(m_formErfassenDeutsch,m_formErfassenEnglish,m_formErfassenUnit,m_formU
sername))
                        {
                                cout << "<p class=\"good\">Das Wortpaar wurde erfolgreich erfasst.</p>" <<
endl;
                        }
                        else
                        {
                                cout << "<p class=\"error\">Fehler beim Schreiben des Wortpaares.</p>" <<
endl;
```

```
                        }
                }
                else
                {
                        cout << "<p class=\"error\">Ung&uuml;ltige Eingabe.</p>" << endl;
                }
        }
}


// Wort aus Methode getWord2Learn "holen"
void cl_printhtml::get_word()
{
        // Objekt einLearn der Klasse cl_learn erstellen.
        cl_learn einLearn;

        if(m_formGetWord == "on" )
        {
                // Bugfix. Einige der Werte der &Uuml;bergabeparameter sind nicht kompatibel.
                string str_unit = m_formLernenUnit;
                string str_lang;
                string str_user = m_formUsername;
                string str_stat;

                if(str_unit != "Alle")
                {
                        // Leerzeichen von HTML Form entfernen
                        str_unit.erase(1,1);
                }

                if(m_formLernenLang == "De-En")
                {str_lang = "DE_EN"; }
                if(m_formLernenLang == "En-De")
                {str_lang = "EN_DE"; }

                if(m_formLernenWord == "Nicht Gewusste")
                {str_stat = "false"; }
                if(m_formLernenWord == "Gewusste")
                {str_stat = "true"; }
                if(m_formLernenWord == "Alle")
                {str_stat = ""; }

                m_formLernenQuestion = einLearn.getWord2Learn(str_unit,str_lang,str_user,str_stat);

                if(m_formLernenQuestion != "")
                {
                        cout <<   "<p class=\"good\">" << m_formLernenQuestion << "</p>" << endl;
                }
                else
                {
                        cout <<   "<p class=\"error\">Mit diesen Optionen ist kein Wort verf&uuml;gbar.</p>"
<< endl;
                }
        }
}


// Worter fuer Maske lernen vergleichen und Status ausgeben.
void cl_printhtml::check_word()
{
        // Objekt einLearn der Klasse cl_learn erstellen.
        cl_learn einLearn;

        if(m_formCheckWord == "on" )
        {
                // Bugfix. Einige der Werte der &Uuml;bergabeparameter sind nicht kompatibel.
```

```
                string str_lang;
                string str_user = m_formUsername;
                string str_Word = m_formLernenQuestion;
                string str_RWord = m_formLernenAnswer;

                if(m_formLernenLang == "De-En")
                {str_lang = "DE_EN"; }
                if(m_formLernenLang == "En-De")
                {str_lang = "EN_DE"; }

                m_formLernenStatusRes = einLearn.getResult(str_lang,str_Word,str_RWord,str_user);

                cout << "<table><tr>" << endl;
                cout << "<td width=\"100\">Wort gewusst</td><td>" << endl;

                if(m_formLernenStatusRes == "SUCCESS")
                {
                        cout <<   "<p class=\"good\">Ja. <font color=\"ffffff\">Das Wort </font>"
                                << m_formLernenQuestion << " <font color=\"ffffff\">stimmt \
                                        mit dem Wort </font>" << m_formLernenAnswer
                                        << " <font color=\"ffffff\">&uuml;berein.</font></p>" << endl;
                }
                else
                {
                        cout <<   "<p class=\"error\">Nein. <font color=\"ffffff\">Das Wort </font>"
                                << m_formLernenQuestion << " <font color=\"ffffff\">stimmt \
                                        mit dem Wort </font>" << m_formLernenAnswer
                                        << " <font color=\"ffffff\">nicht &uuml;berein.</font></p>" << endl;
                }

                cout << "</td></tr><tr><td width=\"100\">Richtige L&ouml;sung</td>" << endl;
                cout << "<td><p class=\"good\">" << einLearn.getRightWord() << "</p></td>" << endl;
                cout << "</tr></table>" << endl;

                get_word();
        }
}
```

13.1.4 cl_login.h

```
/***************************************************************************
                cl_login.h  -  description
                  -------------------
  begin            : Sat Jan 04 2003
  copyright        : (C) 2003 by Reto Loepfe & Thomas Gemperli
  email            : loege@gemperli.net
 ***************************************************************************/

/***************************************************************************
 *                                                                         *
 *  This program is free software; you can redistribute it and/or modify   *
 *  it under the terms of the GNU General Public License as published by   *
 *  the Free Software Foundation; either version 2 of the License, or      *
 *  (at your option) any later version.                                    *
 *                                                                         *
 ***************************************************************************/

#ifndef CL_LOGIN_H
#define CL_LOGIN_H

#include <string>
#include <fstream>
#include <iostream>

using namespace std;
```

```
class cl_login
{
	public:
	cl_login();
	~cl_login();

   // Username und Password eines neuen Benutzers setzen
   bool setUser(const string& Username, const string& Password);

	// Password abfragen
	bool getLogin(const string& Username, const string& Password);

	// User löschen
	bool delUser(const string& Username, const string& Password);

	// Passwort aendern
	bool changePw(const string& Username, const string& Password, const string& newPassword);

	private:
	string m_Username;
	string m_Password;
	string m_newPassword;
	string m_loginFile;
	string m_tmploginFile;
	string m_filePath;

};

#endif
```

13.1.5 cl_login.cpp

```
/***************************************************************************
                  cl_login.cpp  -  description
                    -------------------
   begin            : Sat Jan 04 2003
   copyright        : (C) 2003 by Reto Loepfe & Thomas Gemperli
   email            : loege@gemperli.net
 ***************************************************************************/

/***************************************************************************
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 2 of the License, or     *
 *   (at your option) any later version.                                   *
 *                                                                         *
 ***************************************************************************/

#include "cl_login.h"

// Default Konstruktor
cl_login::cl_login()
{
	// Pfad zu den Datenfiles feststellen und in private Variable sichern.
	if(getenv("DOCUMENT_ROOT"))
	{
		m_loginFile = getenv("DOCUMENT_ROOT");
		m_loginFile.append("/loege/data/login.txt");
		m_tmploginFile = getenv("DOCUMENT_ROOT");
		m_tmploginFile.append("/loege/data/tmplogin.txt");
		m_filePath = getenv("DOCUMENT_ROOT");
		m_filePath.append("/loege/data/");
	}
```

```
}

// Default Destruktor. Der Vollstaendigkeit halber.
cl_login::~cl_login()
{
}

//User erstellen
bool cl_login::setUser(const string& Username, const string& Password)
{
	m_Username = Username;
	m_Password = Password;

	bool user = false;
	bool success = false;

	ifstream inFile(m_loginFile.c_str());

	// Sicherstellen dass Quellfile offen ist
	if(inFile.is_open())
	{
		string text;

		//Zeilen auslesen bis das Ende des Files erreicht ist.
		do
		{
			getline(inFile,text,';'); //String bis zum ";" einlesen (User)

			if(m_Username == text && text != "")
			{
				getline(inFile,text,';'); //String bis zum ";" einlesen (Password)
				getline(inFile,text,'\n'); //String bis zur naechsten newline einlesen

				user = true;
			}
			else
			{
				getline(inFile,text,';'); //String bis zum ";" einlesen (Password)
				getline(inFile,text,'\n'); //String bis zur naechsten newline einlesen
			}
		}
		while(!inFile.eof());
	}

	// Wenn der User noch nicht existiert
	if(user != true)
	{
		//Datei erzeugen und öffnen
		ofstream outFile(m_loginFile.c_str(),ios::app);
		//Ist die Datei wirklich offen?
		if(outFile.is_open())
		{
			//Userangaben ins File schreiben
			outFile << m_Username << ";" << m_Password << ";\n";
		}
		//Datei schliessen
		outFile.close();
		success = true;
	}
	else
	{
		success = false;
	}
	inFile.close();

	return success;
```

```
}


//Berechtigung abfragen
bool cl_login::getLogin(const string& Username, const string& Password)
{
	// Username und Password übergeben
	m_Username = Username;
	m_Password = Password;

	bool login = false;


	ifstream inFile(m_loginFile.c_str());

	if(inFile.is_open())
	{
		string text;

		//Zeilen auslesen bis das Ende des Files erreicht ist.
		do
		{
			getline(inFile,text,';'); //String bis zum ";" einlesen (User)

			if(m_Username == text && text != "")
			{
				getline(inFile,text,';'); //String bis zum ";" einlesen (Password)

				if(m_Password == text)
				{
					login = true;
				}
			}
			else
			{
				getline(inFile,text,';'); //String bis zum ";" einlesen (Password)
				getline(inFile,text,'\n'); //String bis zur naechsten newline einlesen
			}
		}
		while(!inFile.eof());
	}
	inFile.close();

	//Rückgabewert, true -> User gefunden, false -> User nicht gefunden
	return login;

}

//User löschen
bool cl_login::delUser(const string& Username, const string& Password)
{
	// Username und Password übergeben
	m_Username = Username;
	m_Password = Password;

	bool del = false;

	ifstream inFile(m_loginFile.c_str());
	ofstream outFile(m_tmploginFile.c_str());


	if(inFile.is_open())
	{
		string text;

		//Zeilen auslesen und scheiben bis das Ende des Files erreicht ist.
```

```
		do
		{
			getline(inFile,text,';'); //String bis zum ";" einlesen (User)

			// Alle User ausser dem zu Loeschenden in neues File schreiben.
			if(m_Username != text && outFile.is_open() && text != "")
			{
				outFile << text << ";"; //Userangaben ins File schreiben
				getline(inFile,text,';'); //String bis zum ";" einlesen (Password)
				outFile << text << ";\n"; //Userangaben ins File schreiben
				getline(inFile,text,'\n'); //String bis zur naechsten newline einlesen
			}
			else
			{
				getline(inFile,text,';'); //String bis zum ";" einlesen (Password)
				getline(inFile,text,'\n'); //String bis zur naechsten newline einlesen
				del = true;
			}
		}
		while(!inFile.eof());
	}
	inFile.close(); //Datei schliessen
	outFile.close(); //Datei schliessen

	// File zurueckschreiben
	ifstream inFileNew(m_tmploginFile.c_str());
	ofstream outFileNew(m_loginFile.c_str());

	if(inFileNew.is_open())
	{
		string text;

		getline(inFileNew,text,';'); //String bis zum ";" einlesen (User)

		//Zeilen auslesen und scheiben bis das Ende des Files erreicht ist.
		do
		{
			if(outFileNew.is_open() && text != "")
			{
				outFileNew << text << ";"; //Daten ins File schreiben
				getline(inFileNew,text,';'); //String bis zum ";" einlesen (User)
			}
		}
		while(!inFileNew.eof());

		outFileNew << endl;
	}
	inFileNew.close(); //Datei schliessen
	outFileNew.close(); //Datei schliessen

	// Userdaten file leoschen
	m_filePath.append(m_Username);
	m_filePath.append(".dat");
	unlink(m_filePath.c_str());
	delete("/tmp/bla");

	// Tmpfile loeschen
	unlink(m_tmploginFile);

	return del;
}

bool cl_login::changePw(const string& Username, const string& Password, const string& newPassword)
{
	// Username und Password übergeben
	m_Username = Username;
```

```
m_Password = Password;
m_newPassword = newPassword;

bool change = false;

ifstream inFile(m_loginFile.c_str());
ofstream outFile(m_tmploginFile.c_str());


if(inFile.is_open())
{
	string text;

	//Zeilen auslesen und scheiben bis das Ende des Files erreicht ist.
	do
	{
		getline(inFile,text,';'); //String bis zum ";" einlesen (User)

		// Alle User ausser dem mit der Passwort Aenderung in neues File schreiben.
		if(m_Username != text && outFile.is_open() && text != "")
		{
			outFile << text << ";"; //Userangaben ins File schreiben
			getline(inFile,text,';'); //String bis zum ";" einlesen (Password)
			outFile << text << ";\n"; //Userangaben ins File schreiben
			getline(inFile,text,'\n'); //String bis zur naechsten newline einlesen
		}
		else
		{
			getline(inFile,text,';'); //String bis zum ";" einlesen (Password)
			getline(inFile,text,'\n'); //String bis zur naechsten newline einlesen
			change = true;
		}
	}
	while(!inFile.eof());
}
inFile.close(); //Datei schliessen
outFile.close(); //Datei schliessen

// File zurueckschreiben
ifstream inFileNew(m_tmploginFile.c_str());
ofstream outFileNew(m_loginFile.c_str());

if(inFileNew.is_open())
{
	string text;

	getline(inFileNew,text,';'); //String bis zum ";" einlesen (User)

	//Zeilen auslesen und scheiben bis das Ende des Files erreicht ist.
	do
	{
		if(outFileNew.is_open() && text != "")
		{
			outFileNew << text << ";"; //Daten ins File schreiben
			getline(inFileNew,text,';'); //String bis zum ";" einlesen (User)
		}
	}
	while(!inFileNew.eof());

	// User mit dem neuen Passwort wieder ins File zurueckschreiben
	outFileNew << endl;
	outFileNew << m_Username << ";";
	getline(inFileNew,text,';');
	outFileNew << m_newPassword << ";";
	getline(inFileNew,text,';');
```

```
                outFileNew << endl;
        }
        inFileNew.close(); //Datei schliessen
        outFileNew.close(); //Datei schliessen
        // Tmpfile loeschen
        unlink(m_tmploginFile);

        return change;
}
```

13.1.6 cl_userfile.h

```
/***************************************************************************
                cl_userfile.h  -  description
                  -------------------
   begin            : Mon Jan 21 2003
   copyright        : (C) 2003 by Reto Loepfe & Thomas Gemperli
   email            : loege@gemperli.net
 ***************************************************************************/

/***************************************************************************
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 2 of the License, or     *
 *   (at your option) any later version.                                   *
 *                                                                         *
 ***************************************************************************/

#ifndef CL_USERFILE_H
#define CL_USERFILE_H

#include <string>

using namespace std;


class cl_userfile
{
public:

        cl_userfile(); //Konstruktor

        ~cl_userfile(); //Destruktor

        //Setzt das Flag eines Wortes auf true oder false.
        //Das ganze Userfile kann geresetet werden, indem man der Methode
        //einen String (str_resAll) übergibt
        string setLearnFlag(const string& str_language,
                                                const string& str_username,
                                                const string& str_Word,
                                                const string& str_Flag,
                                                const string& str_resAll);

        //Wort, resp. eine ganze Zeile, aus dem Userfile löschen
        bool delWord(const string& str_language,
                                                const string& str_username,
                                                const string str_Word);

        //Ermittelt die Anzahl verschiedener Units in einem Userfile
        int anzUnit(const string& str_username);

private:

        string m_unit;
```

```
        string m_username;
        string m_DWord;
        string m_EWord;
        string m_Word;
        string m_language;
        string m_Flag;
        string m_Dummy;
        string m_ID;
        string m_resAll;
        string m_Path;
        bool m_ok;
        int m_anzUnit;

};

#endif
```

13.1.7 cl_userfile.cpp

```
/***************************************************************************
                cl_userfile.cpp  -  description
                  -------------------
  begin            : Mon Jan 21 2003
  copyright        : (C) 2003 by Reto Loepfe & Thomas Gemperli
  email            : loege@gemperli.net
 ***************************************************************************/

/***************************************************************************
 *                                                                         *
 *  This program is free software; you can redistribute it and/or modify  *
 *  it under the terms of the GNU General Public License as published by  *
 *  the Free Software Foundation; either version 2 of the License, or     *
 *  (at your option) any later version.                                    *
 *                                                                         *
 ***************************************************************************/

#include <fstream>
#include <iostream>
#include <string>
#include <map>

#include "cl_userfile.h"

using namespace std;

typedef multimap<int, string> MULTI_MAP;
typedef MULTI_MAP::iterator ITERATOR;


cl_userfile::cl_userfile()
{
        if(getenv("DOCUMENT_ROOT"))
        {
                m_Path = getenv("DOCUMENT_ROOT");
                m_Path.append("/loege/data/");
        }
}

cl_userfile::~cl_userfile()
{
}


//Setzt das Flag eines Wortes auf true oder false.
//Das ganze Userfile kann geresetet werden, indem man der Methode
```

```
//einen String (str_resAll) übergibt
string cl_userfile::setLearnFlag(const string& str_language,
                                 const string& str_username,
                                 const string& str_Word,
                                 const string& str_Flag,
                                 const string& str_resAll)
{
        string str_Path(m_Path);
        m_username = str_username;
        m_language = str_language;

        str_Path.append(m_username);

        string str_tmpPath = str_Path;
        str_tmpPath.append(".tmp");
        str_Path.append(".dat");

        m_resAll = str_resAll;
        m_Word = str_Word;
        string *m_tmpWord;

        if(m_language == "DE_EN")
        {
                m_tmpWord = &m_DWord;
        }

        else
        {
                m_tmpWord = &m_EWord;
        }


        ifstream inFile(str_Path.c_str());
        ofstream outFile(str_tmpPath.c_str());

        if(inFile.is_open())
        {
                //Zeilen auslesen und schreiben bis das Ende des Files erreicht ist.
                do
                {
                        //String bis zum ";" einlesen (ID)
                        getline(inFile,m_ID,':');
                        getline(inFile,m_unit,':');
                        getline(inFile,m_DWord,':');
                        getline(inFile,m_EWord,':');
                        getline(inFile,m_Flag,'\n');

                        if(m_ID != "")
                        {
                                outFile << m_ID << ":";                 //Userangaben ins File schreiben
                                outFile << m_unit << ":";       //Userangaben ins File schreiben
                                outFile << m_DWord << ":";  //Userangaben ins File schreiben
                                outFile << m_EWord << ":";  //Userangaben ins File schreiben

                                if(m_resAll != "")
                                {
                                        //Setz alle Flags im gesammten Userfile auf false (RESET)
                                        outFile << "false\n" << "\n";
                                }
                                else
                                {
                                        //Wenn Flag == str_Flag, muss das Flag gesetzt werden ganze file
false
                                        if(outFile.is_open() && m_Flag != str_Flag && m_Word ==
m_DWord)
```

```
                                        {
                                                outFile << str_Flag << "\n";//Userangaben ins File
schreiben
                                        }
                                        else
                                        {
                                                outFile << m_Flag//Userangaben ins File
schreiben
                                        }
                                }
                        }

                }
                while(!inFile.eof());
        }
        inFile.close(); //Datei schliessen
        outFile.close(); //Datei schliessen

        // File zurueckschreiben
        ifstream inFileNew(str_tmpPath.c_str());
        ofstream outFileNew(str_Path.c_str());

        if(inFileNew.is_open())
        {
                string text;

                //Zeilen auslesen und scheiben bis das Ende des Files erreicht ist.
                do
                {
                        getline(inFileNew,text,'\n'); //String bis zum ";" einlesen (User)
                        if(outFileNew.is_open() && text != "")
                        {
                                outFileNew << text << "\n"; //Daten ins File schreiben
                        }
                }
                while(!inFileNew.eof());

        }
        inFileNew.close(); //Datei schliessen
        outFileNew.close(); //Datei schliessen

        return m_Flag;

}


//Zeile aus dem Userfile löschen
bool cl_userfile::delWord(const string& str_language,
                                                const string& str_username,
                                                const string str_Word)
{
        m_language = str_language;
        m_username = str_username;
        m_Word = str_Word;
        m_ok = false;

        string str_Path(m_Path);
        str_Path.append(m_username);
        str_Path.append(".dat");

        string *m_tmpWord;

        if(m_language == "DE_EN")
        {
                m_tmpWord = &m_DWord;
        }
```

```
	else
	{
		m_tmpWord = &m_EWord;
	}

	ifstream inFile(str_Path.c_str());

	ofstream outFile(str_Path.c_str());
	if(inFile.is_open())
	{
		//Zeilen auslesen und schreiben bis das Ende des Files erreicht ist.
		do
		{
			//String bis zum ";" einlesen (ID)
			getline(inFile,m_ID,':');
			getline(inFile,m_unit,':');
			getline(inFile,m_DWord,':');
			getline(inFile,m_EWord,':');
			getline(inFile,m_Flag,'\n');

			if(m_ID != "")
			{
				outFile << m_ID << ":";			//Userangaben ins File schreiben
				outFile << m_unit << ":";	//Userangaben ins File schreiben
				outFile << m_DWord << ":";  //Userangaben ins File schreiben
				outFile << m_EWord << ":";  //Userangaben ins File schreiben

				if(m_resAll != "")
				{
					//Setz alle Flags im gesammten Userfile auf false (RESET)
					outFile << "false\n" << "\n";
				}
				else
				{
					string str_Flag;
					//Wenn Flag == str_Flag, muss das Flag gesetzt werden ganze file
false
					if(outFile.is_open() && m_Flag != str_Flag && m_Word ==
m_DWord)
					{
						outFile << str_Flag << "\n";//Userangaben ins File
schreiben
					}
					else
					{
						outFile << m_Flag //Userangaben ins File
schreiben
					}
				}
			}

		}
		while(!inFile.eof());
	}
	inFile.close(); //Datei schliessen
	outFile.close(); //Datei schliessen

	// File zurueckschreiben
	ifstream inFileNew(str_Path.c_str());
	ofstream outFileNew(str_Path.c_str());

	if(inFileNew.is_open())
	{
		string text;

		//Zeilen auslesen und scheiben bis das Ende des Files erreicht ist.
```

```
                do
                {
                        getline(inFileNew,text,'\n'); //String bis zum ";" einlesen (User)
                        if(outFileNew.is_open() && text != "")
                        {
                                outFileNew << text << "\n"; //Daten ins File schreiben
                        }
                }
                while(!inFileNew.eof());

        }
        inFileNew.close(); //Datei schliessen
        outFileNew.close(); //Datei schliessen

        return m_ok;
}

//Ermittelt die Anzahl verschiedener Units in einem Userfile
int cl_userfile::anzUnit(const string& str_username)
{
        string str_Path(m_Path);
        m_username = str_username;
        str_Path.append(m_username);
        str_Path.append(".dat");

        MULTI_MAP m;
//      ITERATOR pos;
        int counter = 0;

        ifstream inFile(str_Path.c_str());
        if(inFile.is_open())
        {
                //Zeilen auslesen und schreiben bis das Ende des Files erreicht ist.
                do
                {
                        //String bis zum ";" einlesen (ID)
                        getline(inFile,m_ID,':');
                        getline(inFile,m_unit,':');
                        getline(inFile,m_Dummy,'\n');
                        cout << m_unit << endl;


                        //erster Durchgang
                        if(counter == 0)
                        {
                                m.insert(pair<int, string>(counter, m_unit));
                                counter++;
                        }
/*                      else
                        {
                                //if()

                        }

                        if(m_ID != "")
                        {
                                outFile << m_ID << ":";                 //Userangaben ins File schreiben
                                outFile << m_unit << ":";       //Userangaben ins File schreiben
                                outFile << m_DWord << ":";  //Userangaben ins File schreiben
                                outFile << m_EWord << ":";  //Userangaben ins File schreiben

                                if(m_resAll != "")
                                {
                                        //Setz alle Flags im gesammten Userfile auf false (RESET)
                                        outFile << "false\n" << "\n";
                                }
```

```
                                        else
                                        {
                                                string str_Flag;
                                                //Wenn Flag == str_Flag, muss das Flag gesetzt werden ganze file
false
                                                if(outFile.is_open() && m_Flag != str_Flag && m_Word ==
m_DWord)
                                                {
                                                        outFile << str_Flag << "\n";//Userangaben ins File
schreiben
                                                }
                                                else
                                                {
                                                        outFile << m_Flag//Userangaben ins File
schreiben
                                                }
                                        }
                                }
*/
                        }
                        while(!inFile.eof());
                }
                inFile.close(); //Datei schliessen

                return m_anzUnit;
}
```

13.1.8 cl_search.h

```
/***************************************************************************
                cl_search.h  -  description
                  -------------------
   begin            : Mon Jan 21 2003
   copyright        : (C) 2003 by Reto Loepfe & Thomas Gemperli
   email            : loege@gemperli.net
 ***************************************************************************/

/***************************************************************************
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 2 of the License, or     *
 *   (at your option) any later version.                                   *
 *                                                                         *
 ***************************************************************************/

#ifndef CL_SEARCH_H
#define CL_SEARCH_H

#include <string>

using namespace std;


class cl_search
{
        public:

        cl_search(); //Konstruktor

        ~cl_search(); //Destruktor

   // Englisches Word ermitteln
        string getEnWord(const string& str_DeWord);
```

```
   // Deutsches Word ermitteln
            string getDeWord(const string& str_EnWord);

            //Anhand des ersten Buchstaben Pfad und Name der übersetzungstabelle ermitteln
            string getPathName(const string& str_DeWord, const string& str_learnTyp);

            private:

            string m_Word;
            string m_DeWord;
            string m_EnWord;
            string m_EnText;
            string m_DeText;
            string m_Path;
            string m_dummy;

            int m_mtDeLen;
            int         m_DeLen;
            int m_mtEnLen;
            int m_EnLen;
            int m_count;
            int m_anz_counter;
            bool m_found_flag;

            string m_DePath;
            string m_EnPath;

};

#endif
```

13.1.9 cl_search.cpp

```
/***************************************************************************
                  cl_search.cpp  -  description
                    -------------------
   begin             : Mon Jan 21 2003
   copyright         : (C) 2003 by Reto Loepfe & Thomas Gemperli
   email             : loege@gemperli.net
 ***************************************************************************/

/***************************************************************************
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 2 of the License, or     *
 *   (at your option) any later version.                                   *
 *                                                                         *
 ***************************************************************************/

#include <fstream>
#include <iostream>
#include <string>


#include "cl_search.h"

using namespace std;

cl_search::cl_search()
{
            if(getenv("DOCUMENT_ROOT"))
            {
                        m_DePath = getenv("DOCUMENT_ROOT");
                        m_DePath.append("/loege/data/DE_EN/");
```

```
                m_EnPath = getenv("DOCUMENT_ROOT");
                m_EnPath.append("/loege/data/EN_DE/");
        }

        m_DeLen = 0;
        m_EnLen = 0;
        m_mtDeLen = 0;
        m_mtEnLen = 0;
        m_anz_counter = 0;
        m_Word = "";
        m_EnWord = "";    //Variable für das Englische Wort
        m_DeWord = "";          //Variable für das Deutsche Wort
        m_EnText = "";          //Variable für englisches Wort aus den Masterfiles
        m_DeText = "";          //Variable für deutsches Wort aus den Masterfiles
        m_dummy = "";           //Dummyvariable, wenn Deutsches Wort nicht gefunden wurde
        m_found_flag = false;   //Flag welches auf True gesetzt wird, wenn das Wort im Masterfile
gefunden wurde

}


cl_search::~cl_search()
{
}


//Anhand des ersten Buchstaben Pfad und Name der übersetzungstabelle ermitteln
string cl_search::getPathName(const string& str_Word, const string& str_learnTyp)
{
        m_Word = str_Word;
        int i_fChar = m_Word[0];
        string str_fName;

        if(str_learnTyp == "DE_EN")
        {
                m_Path = m_DePath;
        }
        else
        {
                m_Path = m_EnPath;
        }

        switch (i_fChar)
        {
        case 'A':
                str_fName = "m_tabA.dat";
                break;
        case 'a':
                str_fName = "m_tabA.dat";
                break;
        case 'B':
                str_fName = "m_tabB.dat";
                break;
        case 'b':
                str_fName = "m_tabB.dat";
                break;
        case 'C':
                str_fName = "m_tabC.dat";
                break;
        case 'c':
                str_fName = "m_tabC.dat";
                break;
        case 'D':
                str_fName = "m_tabD.dat";
                break;
        case 'd':
```

```
                str_fName = "m_tabD.dat";
                break;
        case 'E':
                str_fName = "m_tabE.dat";
                break;
        case 'e':
                str_fName = "m_tabE.dat";
                break;
        case 'F':
                str_fName = "m_tabF.dat";
                break;
        case 'f':
                str_fName = "m_tabF.dat";
                break;
        case 'G':
                str_fName = "m_tabG.dat";
                break;
        case 'g':
                str_fName = "m_tabG.dat";
                break;
        case 'H':
                str_fName = "m_tabH.dat";
                break;
        case 'h':
                str_fName = "m_tabH.dat";
                break;
        case 'i':
                str_fName = "m_tabI.dat";
                break;
        case 'I':
                str_fName = "m_tabI.dat";
                break;
        case 'J':
                str_fName = "m_tabJ.dat";
                break;
        case 'j':
                str_fName = "m_tabJ.dat";
                break;
        case 'K':
                str_fName = "m_tabK.dat";
                break;
        case 'k':
                str_fName = "m_tabK.dat";
                break;
        case 'L':
                str_fName = "m_tabL.dat";
                break;
        case 'l':
                str_fName = "m_tabL.dat";
                break;
        case 'M':
                str_fName = "m_tabM.dat";
                break;
        case 'm':
                str_fName = "m_tabM.dat";
                break;
        case 'N':
                str_fName = "m_tabN.dat";
                break;
        case 'n':
                str_fName = "m_tabN.dat";
                break;
        case 'O':
                str_fName = "m_tabO.dat";
                break;
        case 'o':
```

```
                str_fName = "m_tabO.dat";
                break;
        case 'P':
                str_fName = "m_tabP.dat";
                break;
        case 'p':
                str_fName = "m_tabP.dat";
                break;
        case 'Q':
                str_fName = "m_tabQ.dat";
                break;
        case 'q':
                str_fName = "m_tabQ.dat";
                break;
        case 'R':
                str_fName = "m_tabR.dat";
                break;
        case 'r':
                str_fName = "m_tabR.dat";
                break;
        case 'S':
                str_fName = "m_tabS.dat";
                break;
        case 's':
                str_fName = "m_tabS.dat";
                break;
        case 'T':
                str_fName = "m_tabT.dat";
                break;
        case 't':
                str_fName = "m_tabT.dat";
                break;
        case 'U':
                str_fName = "m_tabU.dat";
                break;
        case 'u':
                str_fName = "m_tabU.dat";
                break;
        case 'V':
                str_fName = "m_tabV.dat";
                break;
        case 'v':
                str_fName = "m_tabV.dat";
                break;
        case 'W':
                str_fName = "m_tabW.dat";
                break;
        case 'w':
                str_fName = "m_tabW.dat";
                break;
        case 'X':
                str_fName = "m_tabX.dat";
                break;
        case 'x':
                str_fName = "m_tabX.dat";
                break;
        case 'Y':
                str_fName = "m_tabY.dat";
                break;
        case 'y':
                str_fName = "m_tabY.dat";
                break;
        case 'Z':
                str_fName = "m_tabZ.dat";
                break;
        case 'z':
```

```
                str_fName = "m_tabZ.dat";
                break;
        default:
                str_fName = "m_tabOth.dat";
                break;
        }

        m_Path.append(str_fName);

        return m_Path;
}


//Englisches Wort suchen
string cl_search::getEnWord(const string& str_DeWord)
{
        m_DeWord.erase();
        m_EnWord.erase();
        m_anz_counter = 0;
        //übergebenes Wort in Variable m_DeWord abspeichern
        m_DeWord = str_DeWord;
        //länge des übergebenen Wortes ermitteln
        m_DeLen = m_DeWord.length();

        string str_learnTyp = "DE_EN";
        getPathName(m_DeWord,str_learnTyp);

        ifstream inFile(m_Path.c_str());

        if(inFile.is_open())
        {
                bool ok_flag = false;

                //Zeilen auslesen bis das Ende des Files erreicht ist.
                do
                {
                        m_count = 0;                    //Zählvariable


                        getline(inFile,m_DeText,':');


                        m_mtDeLen = m_DeText.length();

                        if( ( m_DeLen <= m_mtDeLen))
                        {
                                //Schleife mit der Anzahl der länge des eingegebenen Wortes (deutsch)
                                //Kontrolle ob Wort identisch mit dem Suchbegriff!
                                for(int i = 0; i < m_mtDeLen; ++i)
                                {

                                        ok_flag = true;

                                        if((m_DeWord[i] == m_DeText[i]))
                                        {
                                                //Buchstabe identisch
                                                m_count++;
                                        }
                                        else
                                        {
                                                ok_flag = false;
                                                i = m_mtDeLen;
                                        }
                                }
                        }
```

```
                        if(ok_flag)
                        {
                                //String bis zum "\n" einlesen, d.h. englische Seite einlesen
                                getline(inFile,m_EnText,'\n');
                                if(m_anz_counter > 0)
                                {
                                        m_EnWord.append ("<br>");
                                        m_EnWord.append (m_EnText);
                                }
                                else
                                {
                                        m_EnWord.append (m_EnText);
                                }
                                m_found_flag = true;
                                m_anz_counter++;
                                ok_flag = false;
                        }
                        else
                        {
                                //Dummy-Variable abfüllen
                                //String bis zum "\n" einlesen, d.h. englische Seite einlesen
                                getline(inFile,m_dummy,'\n');
                        }
                }
                while(!inFile.eof());

                //File wieder schliessen
                inFile.close();

        }

        //Kontrolle, ob im Masterfile das deutsche Wort gefunden wurde
        if(!m_found_flag)
        {
                m_EnWord = "Suchwort nicht gefunden";
        }
        //Rückgabewert, das gefundene Wort wird zurückgegeben
        return m_EnWord;

}

//Deutsches Wort suchen
string cl_search::getDeWord(const string& str_EnWord)
{
        //übergebenes Wort in Variable m_DeWord abspeichern

        m_DeWord.erase();
        m_EnWord.erase();
        m_anz_counter = 0;
        m_EnWord = str_EnWord;
        m_EnLen = m_EnWord.length();
        string str_learnTyp = "EN_DE";
        getPathName(m_EnWord,str_learnTyp);

        ifstream inFile(m_Path.c_str());

        if(inFile.is_open())
        {
                bool ok_flag = false;

                //Zeilen auslesen bis das Ende des Files erreicht ist.
                do
                {
                        //Zählvariable
                        m_count = 0;
```

```
                    getline(inFile,m_DeText,':');
                    getline(inFile,m_EnText,'\n');

                    m_mtEnLen = m_EnText.length();

                    if( ( m_EnLen <= m_mtEnLen))
                    {
                            //Schleife mit der Anzahl der länge des eingegebenen Wortes (deutsch)
                            //Kontrolle ob Wort identisch mit dem Suchbegriff!
                            for(int i = 0; i < m_mtEnLen; ++i)
                            {

                                    ok_flag = true;

                                    if((m_EnWord[i] == m_EnText[i]))
                                    {
                                            //Buchstabe identisch
                                            m_count++;
                                    }
                                    else
                                    {
                                            ok_flag = false;
                                            i = m_mtEnLen;
                                    }
                            }
                    }

                    if(ok_flag)
                    {
                            if(m_anz_counter > 0)
                            {
                                    m_DeWord.append ("<br>");
                                    m_DeWord.append (m_DeText);
                            }
                            else
                            {
                                    m_DeWord.append (m_DeText);
                            }
                            m_found_flag = true;
                            m_anz_counter++;
                            ok_flag = false;
                    }
            }
            while(!inFile.eof());

            //File wieder schliessen
            inFile.close();

    }

    //Kontrolle, ob im Masterfile das deutsche Wort gefunden wurde
    if(!m_found_flag)
    {
            m_DeWord = "Suchwort nicht gefunden";
    }

    //Rückgabewert, das gefundene Wort wird zurückgegeben
    return m_DeWord;

}
```

13.1.10 cl_learn.h

```
/*************************************************************************
```

```
                cl_learn.h  -  description
                  -------------------
  begin            : Mon Jan 21 2003
  copyright        : (C) 2003 by Reto Loepfe & Thomas Gemperli
  email            : loege@gemperli.net
 ***************************************************************************/

/***************************************************************************
 *                                                                         *
 *  This program is free software; you can redistribute it and/or modify  *
 *  it under the terms of the GNU General Public License as published by  *
 *  the Free Software Foundation; either version 2 of the License, or     *
 *  (at your option) any later version.                                    *
 *                                                                         *
 ***************************************************************************/

#ifndef CL_LEARN_H
#define CL_LEARN_H

#include <string>
#include "cl_userfile.h"

using namespace std;


class cl_learn
{
public:
        cl_learn(); //Konstruktor

        ~cl_learn(); //Destruktor

        //Wort zum Lernen aus dem Userfile ermitteln
        string cl_learn::getWord2Learn(const string& str_Unit,
                                                        const string& str_learnTyp,
                                                        const string& str_username,
                                                        const string& str_WordTrueFalse);

        //Wörter in Userfile abspeichern
        bool cl_learn::setDEWord(const string& str_DWord,
                                                        const string& str_EWord,
                                                        const string& str_Unit,
                                                        const string& str_username);

        //Hoechste ID der übergebenen Unit ermitteln
        int cl_learn::getMaxID(const string& str_Unit,
                                        const string& str_DeWord,
                                                        const string& str_WordTrueFalse);


        string cl_learn::getResult(const string& str_SprachWahl,
                                                        const string& str_Word,
                                                        const string& str_RWord,
                                                        const string& str_username);

        string cl_learn::getRightWord();


private:
        string m_unit;
        string m_m_Unit;
        string m_learnTyp;

        string m_m_learnTyp;
        string m_username;
        string m_m_username;
```

```
        string m_WordTrueFalse;
        string m_m_WordTrueFalse;
        string m_Word2Learn;
        string m_m_Word2Learn;
        string m_DWord;
        string m_m_DWord;
        string m_EWord;
        string m_m_EWord;
        string m_RWord;
        string m_Dummy;
        string m_filePath;
        string m_rightWord;
        bool m_flag;
        string m_ID;
        int m_m_ID;
        int m_fPointer;
};

#endif
```

13.1.11 cl_learn.cpp

```
/***************************************************************************
                cl_learn.cpp  -  description
                  -------------------
  begin            : Mon Jan 21 2003
  copyright        : (C) 2003 by Reto Loepfe & Thomas Gemperli
  email            : loege@gemperli.net
 ***************************************************************************/

/***************************************************************************
 *                                                                         *
 *  This program is free software; you can redistribute it and/or modify   *
 *  it under the terms of the GNU General Public License as published by   *
 *  the Free Software Foundation; either version 2 of the License, or      *
 *  (at your option) any later version.                                    *
 *                                                                         *
 ***************************************************************************/

#include <fstream>
#include <iostream>
#include <string>
#include <map>


#include "cl_learn.h"

using namespace std;

typedef multimap<int, string> MULTI_MAP;
typedef MULTI_MAP::iterator ITERATOR;

cl_learn::cl_learn()
{
        // Pfad zu den Datenfiles feststellen und in private Variable sichern.
        if(getenv("DOCUMENT_ROOT"))
        {
                m_filePath = getenv("DOCUMENT_ROOT");
                m_filePath.append("/loege/data/");
        }

        m_unit = "";
        m_m_Unit = "";
        m_learnTyp = "";
        m_m_learnTyp = "";
```

```
	m_username = "";
	m_m_username = "";
	m_WordTrueFalse = "";
	m_WordTrueFalse = "";
	m_Word2Learn = "";
	m_Word2Learn = "";
	m_flag = true;
}


cl_learn::~cl_learn()
{
}


//Hoechste ID der übergebenen Unit ermitteln
int cl_learn::getMaxID(const string& str_Unit, const string& str_DeWord, const string& str_WordTrueFalse)
{
	string str_ID;
	m_unit = str_Unit;

	//Zähler um die höchste ID zu ermitteln, ist auch der Rückgabewert
	int count = 0;

	//Wort zur überprüfung, ob es schon im User-File vorhanden ist
	string Word;
	m_DWord = str_DeWord;
	m_WordTrueFalse = str_WordTrueFalse;

	string str_Path = m_filePath;
	str_Path.append(m_username);
	str_Path.append(".dat");

	ifstream inFile(str_Path.c_str());

	if(inFile.is_open())
	{
		//Zeilen auslesen bis das Ende des Files erreicht ist.
		do
		{
			getline(inFile,str_ID,':');

			if(m_unit == "Alle")
			{
				m_m_Unit = "Alle";
				getline(inFile,m_Dummy,':');
			}
			else
			{
				getline(inFile,m_m_Unit,':');
			}

			getline(inFile,Word,':');
			getline(inFile,m_Dummy,':');
			getline(inFile,m_m_WordTrueFalse,'\n');

			//Kontrolle, ob das zu speichernde Wort schon in der Tabelle ist mit der gleichen
Lektion
			if(m_DWord == Word && m_DWord != "")
			{
				//Wort schon einmal gefunden,
				//Loop bis zum Dateiende
				do
				{
					getline(inFile,m_Dummy,'\n');
				}
```

```
                                        while(!inFile.eof());

                                        //-99 heisst, das zu speichernde Wort ist bereits vorhanden
                                        count = -99;
                                }
                                else
                                {
                                        if((m_m_Unit == str_Unit) && (m_m_WordTrueFalse ==
m_WordTrueFalse))
                                        {
                                                //Gefundene Unit identisch mit gesuchter
                                                str_ID = m_ID;
                                                count++;
                                        }
                                        else
                                        {
                                                if((m_WordTrueFalse == "") && (m_m_Unit == str_Unit))
                                                {
                                                        count++;
                                                }
                                        }
                                }

                        }
                        while(!inFile.eof());

        }
        else
        {
//                      cout << "Kein File vorhanden";
        }

        return count;
}

//Word zum lernen mit random ermitteln
string cl_learn::getWord2Learn(const string& str_Unit,
                                                        const string& str_learnTyp,
                                                        const string& str_username,
                                                        const string& str_WordTrueFalse)
{
        m_unit = str_Unit;
        m_learnTyp = str_learnTyp;
        m_username = str_username;
        m_WordTrueFalse = str_WordTrueFalse;

        int id = 0;
        int m_random = 0;

        id = getMaxID(m_unit,"",m_WordTrueFalse);

        // Division durch Null verhindern
        if(id == 0)
        {
                id = 1;
        }

        for(int i=0; i < 20; ++i)
        {
                m_random = rand() % id;
        }

        string str_Path = m_filePath;
        str_Path.append(m_username);
        str_Path.append(".dat");
```

```
int map_ID = 0;
string flag,Unit,Word;

ifstream inFile(str_Path.c_str());

MULTI_MAP m;
ITERATOR pos;

if(inFile.is_open())
{
	//Zeilen auslesen bis das Ende des Files erreicht ist.
	do
	{
		getline(inFile,m_Dummy,':');

		if(m_unit == "Alle")
		{
			Unit = "Alle";
			getline(inFile,m_Dummy,':');
		}
		else
		{
			getline(inFile,Unit,':');
		}

		getline(inFile,m_DWord,':');
		getline(inFile,m_EWord,':');
		getline(inFile,flag,'\n');

		if(m_learnTyp == "EN_DE")
		{
			Word = m_EWord;
		}
		else
		{
			Word = m_DWord;
		}


		//Es werden nur Wörter mit der richtigen Auswahl in den Container gespeichert.
		if(flag == m_WordTrueFalse && m_unit == Unit)
		{
			m.insert(pair<int, string>(map_ID,Word));
			map_ID++;
		}
		else if(m_WordTrueFalse == "" && m_unit == Unit)
		{
			m.insert(pair<int, string>(map_ID,Word));
			map_ID++;
		}

	}
	while(!inFile.eof());
}

inFile.close();


pos = m.find(m_random);

if(pos != m.end())
{
	m_Word2Learn = pos->second;
}

return m_Word2Learn;
```

```
}


bool cl_learn::setDEWord(const string& str_DWord,
                                                        const string& str_EWord,
                                                        const string& str_Unit,
                                                        const string& str_username)
{
        m_DWord = str_DWord;
        m_EWord = str_EWord;
        m_unit = str_Unit;
        m_username = str_username;

        string str_Path = m_filePath;
        str_Path.append(m_username);
        str_Path.append(".dat");

        m_m_ID = getMaxID(m_unit,m_DWord,"");

        ofstream outFile(str_Path.c_str(),ios::app);

        if(outFile.is_open() && m_m_ID != -99)
        {

                outFile << ++m_m_ID << ":" << m_unit << ":" << m_DWord << ":" << m_EWord << ":" << "false" <<
"\n";
                m_flag = true;

        }
        else
        {
                m_flag = false;
        }

        //Datei schliessen
        outFile.close();

        return m_flag;
}

//Vom User eingegebene Lösung kontrollieren
string cl_learn::getResult(const string& str_SprachWahl,
                                                        const string& str_Word,
                                                        const string& str_RWord,
                                                        const string& str_username)
{
        m_username = str_username;
        m_learnTyp = str_SprachWahl;
        string result,unit;

        if(m_learnTyp == "DE_EN")
        {
                //Englisches Wort ermitteln
                m_DWord = str_Word;
                m_RWord = str_RWord;
        }
        else
        {
                //Deutsches Wort ermitteln
                m_EWord = str_Word;
                m_RWord = str_RWord;
        }

        string str_Path = m_filePath;
        str_Path.append(m_username);
```

```
str_Path.append(".dat");

ifstream inFile(str_Path.c_str());


if(inFile.is_open())
{
	//Zeilen auslesen bis das Ende des Files erreicht ist.
	do
	{
		getline(inFile,m_Dummy,':');
		getline(inFile,m_unit,':');
		getline(inFile,m_m_DWord,':');
		getline(inFile,m_m_EWord,':');
		getline(inFile,m_Dummy,'\n');


		if(m_DWord == m_m_DWord && m_learnTyp == "DE_EN")
		{
			if(m_RWord == m_m_EWord)
			{
				do
				{
					getline(inFile,m_Dummy,'\n');
				}
				while(!inFile.eof());
				result = "SUCCESS";
			}
			else
			{
				do
				{
					getline(inFile,m_Dummy,'\n');
				}
				while(!inFile.eof());
				result = "FAILED";
			}
			m_rightWord = m_m_EWord;
			unit = m_unit;
		}
		else if(m_EWord == m_m_EWord && m_learnTyp == "EN_DE")
		{
			if(m_RWord == m_m_DWord)
			{
				do
				{
					getline(inFile,m_Dummy,'\n');
				}
				while(!inFile.eof());
				result = "SUCCESS";
			}
			else
			{
				do
				{
					getline(inFile,m_Dummy,'\n');
				}
				while(!inFile.eof());
				result = "FAILED";
			}
			m_rightWord = m_m_DWord;
			unit = m_unit;
		}

	}
	while(!inFile.eof());
```

```
		}
		inFile.close();

		//Bei richtiger Eingabe des Users!
		if(result == "SUCCESS")
		{
			cl_userfile einSetFlagTrue;
			einSetFlagTrue.setLearnFlag(m_learnTyp, m_username, str_Word, "true", "");
		}

		return result;

}

// Rueckgabe der richtigen Loesung an html teil
string cl_learn::getRightWord()
{
		return m_rightWord;
}
```