Universität Essen

Wirtschaftsinformatik und Softwaretechnik

Fachbereich 5, Wirtschaftsinformatik

Universitätsstraße 9

D-45141 Essen

Seminararbeit DII:

Entwurf und Entwicklung eines Web-basierten Terminkalenders

(WML - Version)

Vorgelegt dem Fachbereich Wirtschaftswissenschaften der Universität Essen von

Ednan Masovic

Süllenstr. 78

40599 Düsseldorf

Matr-Nr. : 1080774

Betreuer: Reinhold Klapsing

abgegeben am: 10. Dezember 2000 (WS 2000/01)

Inhaltsverzeichnis

1. Einleitung 2
1.1 „Web-Calendar“ 3
1.2 Meilensteine der Arbeit 3
2. Anforderungsanalyse 4
2.1 Abrufen von persönlichen Terminen mittels Web 4
2.2 Schwerpunkt dieser Arbeit 5
2.3 Benötigte Komponenten 5
3. Systemarchitektur 8
3.1 Clients 8
3.2 HTTP und WAP 8
3.3 WWW-Server 9
3.4 HTML vs. WML 9
3.5 Initiierung einer Session - login.rxx & CGI 10
3.6 Ansichtsauswahl - diary.rxx & CGI 10
4. Klassensystem 12
4.1 UseCase-Notation 12
4.2 Klassendokumentation 14
5. Skripte 22
5.1 login.cgi 22
5.2 login.rxx 22
5.3 diary.cgi 23
5.4 diary.rxx 23
6. Datenbankkonzeption 24
6.1 „CalendarStore“ 24
7. Screenshots des Prototypen 25
7.1 Login-Menu 25
7.2 Main-Menu 26
7.3 View-Types 27
8. Ausblick 28
9. Rechercheergebnisse 29
Reference 29
Software / Literatur 30

1. Einleitung

Wenn die Web-Informationen nicht nur in HTML [5] sondern auch in WML [2] über ein WAP-Gateway [3] verfügbar gemacht werden, erreicht man nicht nur die klassischen Internet-User, die lokale Endgeräte (PCs, Fernsehapparate,...) zum „Internet-Surfen“ verwenden, sondern auch die wesentlich stärker wachsende Anzahl von Personen, die ein mobiles Endgerät (z.B. WAP-fähiges Handy) besitzen und darauf Informationen abfragen können.

Ein kombinierter Einsatz von lokalen und mobilen Endgeräten zum „Internet-Surfen“ wäre die Alternative, eine größere Anzahl von Internet-Usern zu erreichen und die eigenen Web-Informationen somit einer breiteren Masse als bis jetzt (nur in HTML [5]!) zur Verfügung zu stellen.

Ein kritischer Aspekt bei der Entwicklung von HTML und WML-Informationssystemen ist der Entwicklungsaufwand und die damit verbundenen Kosten, da man die Informationen immer in zwei Versionen auf dem Web-Server verfügbar halten muß. Es gibt spezielle WAP-Gateways, die versuchen, bestehende HTML-Informationen, so gut das geht, automatisch in WML [2] umzuwandeln. Meist genügt eine solche automatisierte Umwandlung aber nicht, weil die Informationen für die kleinen Displays völlig anders strukturiert werden müssen. Das bedeutet, jede Aktualisierung der HTML-Files zieht eine Aktualisierung der WML-Files nach sich. Der Arbeitsaufwand würde sich somit nahezu verdoppeln.

Ein weiterer wichtiger Aspekt sind die relativ hohen Kosten, die auf den Internet-User zukommen, wenn er das mobile Angebot nutzt, da „Internet-Surfen“ aktuell mit lokalen Endgeräten 20 mal günstiger ist als mit mobilen.

Auch der Eingabe-Komfort (Maus, PC-Tastatur (101/102 Tasten)) ist lokal bei weitem besser, und die relativ große und bunte Anzeige (PC-Monitor) erhöht die Lesbarkeit der Daten. Damit kann der User sich ganz seiner ursprünglichen Intention, der Informationsabfrage, widmen.

Alle diese Aspekte spielen eine zentrale Rolle bei der *Systementwicklung und –implementierung des* ***„Web-Calendars“,*** *als hybrides Web-Informationsangebot für lokale und mobile Endgeräte*, d.h. die Informationen werden in HTML und WML zur Verfügung gestellt. Dieses Dokument befasst sich mit der Entwicklung und Implemetierung der WML-Version des Systems wird in diesem Dokument bearbeitet:

http://swt.wi-inf.uni-essen.de/~emasovic/ .

Die HTML-Version berbeitet T. Jungmann: http://swt.wi-inf.uni-essen.de/~tjungman/ .

1.1 „Web-Calendar“

In dieser Arbeit sind weder das HTML-File noch das WML-File statische Dateien, sondern werden automatisch und dynamisch aus den in einer Datenbank gespeicherten Informationen mit Hilfe von „Rexx“-Skripten generiert. Somit beschränkt sich die Aktualisierung der HTML- und WML-Files auf die Aktualisierung der gemeinsam genutzten Datenbank „CalendarStore“. Der Arbeitsaufwand bei der Aktualisierung der kombinierten Web-Informationen bleibt also ungefähr gleich groß.

Die Kosten, die auf den User des mobilen Web-Angebots zukommen, können leider nur indirekt beeinflusst werden, indem versucht wird, die Seiten möglichst „einfach und praktisch“ zu gestallten, d.h. keine multimedial aufwendige Lösungen für die mobilen Endgeräte bereitstellen, da diese meistens mit langen Ladezeiten verbunden sind und somit höhere Kosten verursachen.

Die relativ schlechteren Eingabemöglichkeiten der mobilen Endgeräte werden lediglich zum Navigieren genutzt. Die Dateneingabe, die mit dem unpraktischen Ziffernblock (WAP-„Handy“) geschrieben werden muß, soll auf das Notwendigste (Username, Passwort) reduziert werden. In dieser Arbeit werden die Termine lokal mit dem HTML-Browser gepflegt und mobil mit dem WAP-Endgerät lediglich abgerufen. Zu dem eigentlichen Termineintrag kommt man in zwei bis fünf Schritten, über eine einfache und selbsterklärende Navigationsstruktur (siehe Kapitel 7 – „Screenshots des Prototypen“).

Ein WML-Prototyp für mobile Endgeräte (*nur Abrufen von Terminen*!) ist zu finden unter:

http://swt.wi-inf.uni-essen.de/~emasovic/index.wml

Der HTML-Prototyp (*Abrufen, Editieren,... von Terminen*!) ist zu finden unter:

http://swt.wi-inf.uni-essen.de/~tjungman/logon.html

1.2 Meilensteine der Arbeit

Im weiteren Verlauf des Dokumentes wird zunächst das System unabhängig von der verwendet Programmiersprache entwickelt und anschließend in Rexx implementiert. Die Beschreibung der Rexx-spezifischen Implementierung ist in den Kapitel 4 und 5 zu finden. Die übrigen Kapitel befassen sich mit der Dokumentation des Systems, unabhängig von der verwendeten Programmiersprache.

2. Anforderungsanalyse

2.1 Abrufen von persönlichen Terminen mittels Web

Aufgabe ist es, eine web-basierte Anwendung zu erstellen, die es einem Benutzer ermöglicht, bestehende Einträge aus seinem persönlichen Terminkalender mit Hilfe von mobilen Endgeräten (z.B. Wap-Handy...) abzurufen.

Die Spezifikationen für WML und WAP sind in den Referenzen [2,3] zu finden. Derzeit wird von den meisten WAP-Endgeräten, die aktuell von Benutzern verwendet werden, die Version 1.1 von WML (1999 definiert) einigermaßen vollständig unterstützt. Aus diesem Grund wird beim Testen des Systems ein Emulator des ersten WAP-fähigen Handys benutzt, des Nokia „7110“. Den Emulator „NOKIA WAP Toolkit Version 2.0“ stellt Nokia jedem WAP-Entwickler gratis zur Verfügung bei: http://www.forum.nokia.com/ .

Realisiert wird die Anwendung über WML-Forms ("GET"- und "POST"-Methode), die über die vordefinierte CGI-Schnittstelle [6] mit Rexx-Skripten kommunizieren.

Die jeweiligen Termine werden in einer SQL-Datenbank verwaltet, welche mittels entsprechender Rexx-Prozeduren angesprochen wird. Die Daten sowie die Prozeduren zur Datenmanipulation (Einfügen, Löschen und Editieren) generiert *Thomas Jungmann* im Rahmen seines Projektes.

Damit *benutzerorientierte* Daten abrufbar sein können, muß die Anwendung ein **Authentifizierungsmechanismus** (z.B. Passwortabfrage) bieten, um die *Identität des Benutzers* sicherzustellen.

Ein anschliessendes **Session-Management**[5] soll die Zustandslosigkeit des verwendeten HTTP[7] umgehen, d.h. eine Authentifizierung ist nur mit der *ersten* Anfrage notwendig. Bei keiner weiteren Anfrage soll die Session, nach Überschreiten eines vorgegebenen Zeitlimits, automatisch beendet werden.

Aufgrund der relativ schlechten Eingabemöglichkeiten (keine Maus, kompakte Tastatur,...) der verwendeten mobilen Endgeräte sollen die Termine mit wenigen Tastatureingaben abrufbar sein.

2.2 Schwerpunkt dieser Arbeit

- Neben der benutzerspezifischen Informationsabfrage aus einer SQL-Datenbank soll der anschliessend dynamisch generierte WML-Output für WAP-Handys, die WML1.1[2] unterstützen, „optimal“ gestaltet werden.

- Desweiteren werden in dieser Arbeit die technischen Möglichkeiten, die heute auf dem Markt durch die unterschiedlichen Hersteller von mobilen Endgeräten gegeben sind, mit dieser Arbeit näher demonstrieren.

- Die Frage inwiefern, die unterschiedlichen Standards wie z.B. WML 1.1[3], WML 1.2 oder WML 1.3 von den Herstellern umgesetzt werden, soll geklärt werden.

- Ein weiterer wichtiger Aspekt ist der Einsatz von Object-Rexx als Scripting-Engine und die damit verbundenen Schwierigkeiten bei der Implementierung des Prototypen.

2.3 Benötigte Komponenten

benötigte Komponenten:	definierte Schnittstellen:
Session-Management	HTTP bzw. WAP
Rexx-Scripte	CGI
WML-Decks	DBI
SQL-Datenbanken	
verwendeter Server: http://**swt.wi-inf.uni-essen.de**	

zu entwickelnde Komponenten

2.3.1 Session-Management

Transaktionsablauf mittels HTTP

- **Verbindungsaufbau** durch den User
- **Request**: Anfrage des Users an den WWW-Server, ein bestimmtes Objekt zu senden
- **Response**: Antwort des WWW-Servers
- **Verbindungsabbau**

Nach jeder Anfrage geht der Webserver in seinen Ursprungszustand zurück. Es besteht somit für den Webserver keine Möglichkeit, zwei aufeinander folgende Anfragen in Verbindung zu bringen und somit eine Session zu initiieren. Das Session-Management umgeht diese „Schwäche“ des HTTP[7] und ermöglicht eine Session zwischen dem Client und dem Server. Es gibt unteranderem folgende Mechanismen, um eine Session mit HTTP zu gestalten:

- **Hidden-Form-Fields** nach HTML 4.01[5]
- **Post-Fields** nach WML 1.1.[2]
- **Cookies** nach [RFC2109]
- **PATH-Info** nach CGI [6]

In dieser Arbeit werden die Post-Fields verwendet, um eine eindeutige Zeichenkette (Session-Identifier), zur Identifikation des Users, einzubauen. Diese eindeutige Zeichenkette sendet der User Agent mit jeder Anfrage an den Server mit.

Beispiel („Index.wml“) für Postfields:

```
<do type="accept" label="login">
 <go href="cgi-bin/login.cgi" method="get">
  <postfield name="user" value="$(user)" />
  <postfield name="pw" value="$(pw)" />
 </go>
</do>
```

2.3.2 Rexx-Skripte

Die Programm-Logik wird mit Hilfe von Object Rexx – Klassen und – Skripten implementiert. Die Skripte werden zum Zugriff auf die Datenbank und zum Generieren der dynamischen WML-decks und –cards benötigt.

2.3.3 WML-Decks

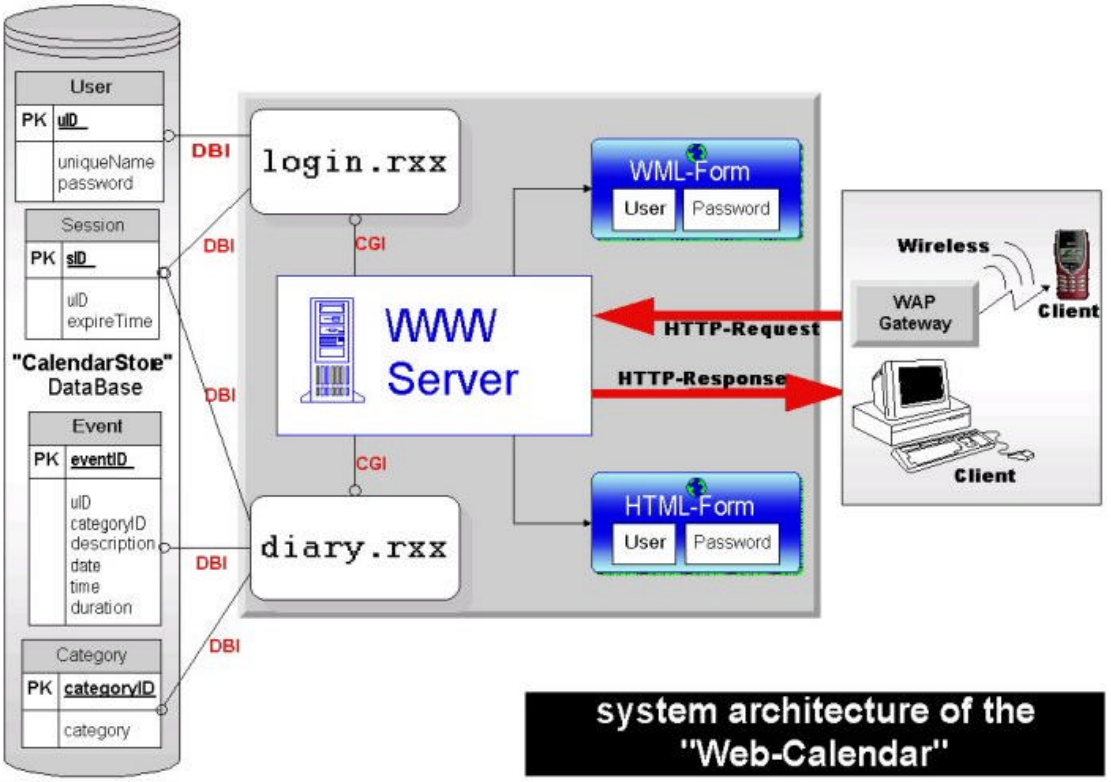
HTML-Seiten können nicht auf einem WAP-Endgerät angezeigt werden. Dazu dienen die WML-Decks die in WML1.1 [2] definiert sind. WML ist eine Applikation von XML [4]. Die WML-Decks werden in diesem System dynamisch generiert mit Hilfe von Rexx-Skripten und enthalten Informationen, die der Datenbank „Calendar-Store“ entnommen wurden.

2.3.4 SQL – Datenbank

Die Datenbank „Calendar-Store“ beinhaltet vier Relationstabellen, die folgendes leisten:

- Benutzer und Passwörter des Systems verwalten,
- Informationen für das Session-Management temporär speichern,
- die Termine kategorisieren,
- und die Journal-Einträge der einzelnen Events speichern.

3. Systemarchitektur



Systemarchitektur des web-basierten Terminkalenders

3.1 Clients

Der Client kann der Web-Browser eines PCs oder ein WAP-fähiges Endgerät (WAP-„Handy“) sein, welches über das *Wireless Application Protocol* (kurz WAP [3]) mit einem WAP-Gateway kommuniziert.

3.2 HTTP und WAP

Das WAP-Gateway kommuniziert mit dem Web-Server wiederrum über das Internet mit dem HyperText Transfer Protokol (kurz HTTP [7]). „Das WAP ist das Ergebnis der Bemühungen des WAP-Forums, eine industrieweite Spezifikation einer Technologie zu fördern, die es ermöglicht, Anwendungen zu erstellen, die über drahtlose Netze laufen“[1].

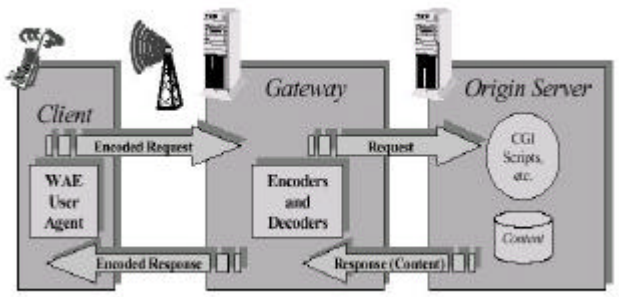


Figure 2. WAP Programming Model

[1] „WAP Architecture Specification“ - Version 30-Apr-1998 - URL http://www.wapforum.org/

3.3 WWW-Server

Der WWW-Server (Apache [10]) nimmt Anfragen als HTTP-Daemon entgegen und schickt Ergebnisse zurück. Bei der ersten Anfrage schickt der Server das „index.wml“-File, ein statisches vom System bereitgelegtes WML- oder HTML-Formular [2,5], als Antwort bzw. Ergebnis.

3.4 HTML vs. WML

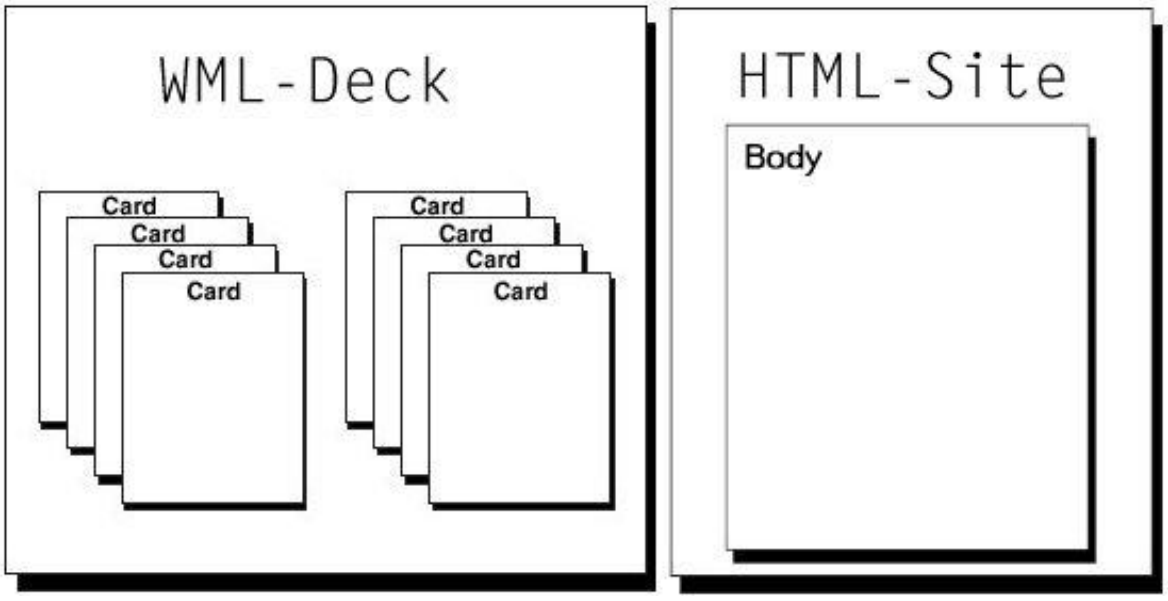
„WML ist eine Auszeichnungssprache, die auf XML [4] basiert und zum Einsatz kommt in der Spezifizierung der Schnittstelle zwischen dem Inhalt und dem Benutzer eines Web-Informationssystems für schmalbandige mobile Endgeräte“ [2], wie z.B:

- Mobil-Telefone (Handys, GSM)
- PDA’s (Personal Digital Assistants, Organizer)
- Palmtop-Computer
- Auto-Navigationsgeräte

WML ist eine Applikation von XML [4], demnach muß auch jedes WML-Deck [2] mit folgenden Direktiven an den XML-Parser beginnen:

<?xml version="1.0"?>
✍ Die zugrundegelegte Version als Direktive an den XML-Parser

<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN" „http://www.wapforum.org/DTD/wml_1.1.xml">
✍ Die URL der DTD, um das Dokument auf „Gültigkeit“ prüfen zu können.



WML-Deck und HTML-Seite

3.5 Initiierung einer Session - login.rxx & CGI

Durch ein mobiles Endgerät wird zu Beginn der Sitzung das Login-Formular (WML-Deck) vom Server angefordert. Die eingegebenen Daten (Name, Paßwort) werden im Query-String eines URL's [11] an den Web-Server übermittelt, welcher diese mittels CGI [6] an das **login.rxx**-Script weiterleitet. Mittels Rexx/SQL [12] werden die Eingaben mit den Einträgen in der User-Datenbank (My-SQL[2]) validiert. Rexx/SQL bietet eine Schnittstelle zu SQL-Datenbanken. Bei korrekter Eingabe wird eine SessionID generiert, dem Benutzer zugeordnet und in der Session-Datenbank temporär gespeichert. Vollständigkeitshalber soll natürlich auch das Ausloggen ermöglicht werden. Hierzu wird ebenfalls das **login.rxx**-Script dienen.

Bemerkung: Die SessionID wird nach erstmaliger Ermittlung durch Post-Fields mit jeder weiteren Anfrage des Client an den Server übertragen. Nach jeder Übertragung der SessionID wird das Zeitlimit der aktuellen Session in der Session-Tabelle der „CalendarStore"-Datenbank neu gesetzt. Wenn in einem bestimmten Zeitabschnitt keine weitere Anfrage erfolgt wird die Session ID automatisch in der Datenbank gelöscht und somit die Session beendet. Die Session kann auch vom User Agent beendet werden mit der Option „logout".

3.6 Ansichtsauswahl - diary.rxx & CGI

Nachdem der Login-Vorgang erfolgreich abgeschlossen wurde, generiert die Anwendung eine Dynamische WML Seite, die den Benutzer begrüßt und verschiedene Ansichtsoptionen (Year-, Month-, Week-, Dayview) und Ansichtskategorien (TO-DO und EVENT nach ICalendar [1]) anbietet. Die ausgewählten Ansichtsoptionen werden als Steuerinformationen im QUERY_STRING mit der „GET"-Methode über die CGI-Schnittstelle an das Skript weitergegeben. Nach Auswahl der Ansichtsoption wird eine weitere Dynamische WML Seite mit benutzerspezifischen Termindaten erstellt. Die Termindaten des Benutzers werden durch einen DBI-Zugriff auf die Diary-Datenbank ermittelt. Hierzu wird das **diary.rxx**-Script dienen.

Bemerkung: Die unterschiedlichen Zeitzonen werden von der Anwendung nicht beachtet. Die Zeitzone entspricht der geografischen Position des Web-Servers. Als Kalenderscala

[2] current Version My-SQL 2.4 - URL: http://www.tcx.se/

wird sowohl der julianische als auch der gregorianische Kalender unterstützt. Im Jahre 1582 führte Papst Gregor XIII den gregorianischen Kalendar und korrigierte den Fehler von 10 Tagen, den der julianische verursachte, indem er 10 Tage im Jahre 1582 (4. Oktober folgte dem 15.) gestrichen hatte. Der Gregorianische Kalendar sieht neben den üblichen Schaltjahren alle 4 Jahre noch ein weiteres Schaltjahr alle 400 Jahre. Das bedeutet, die eigentlichen Schaltjahre 1700, 1800, 1900 sind keine Schaltjahre, da sie nicht durch 400 teilbar sind aber die Jahre 1600, 2000, 2400,... werden wie gehabt als Schaltjahre betrachtet. Das ist darauf zurückzuführen, dass die Erde ca. 364 Tage 5 Stunden und 48 Minuten benötigt um die Sonne einmal zu umkreisen.

4. Klassensystem

Im weiteren Abschnitt werden die einzelnen Klassen und deren Beziehungen näher erläutert. Systemtechnisch ist jede Klasse für sich in einer separaten Datei abgelegt. Alle Dateinamen (fast alle) entsprechen dem Klassennamen mit dem Zusatz "...lib.rxx", d.h. die Datei **"sessionlib.rxx"** enthält die Klasse **session**.

Bsp.:

Datei: "cgilib.rxx" beinhaltet nur eine öffentliche Klasse

- `::CLASS cgi PUBLIC`

Datei: "sessionlib.rxx" beinhaltet nur eine öffentliche Klasse

- `::CLASS session PUBLIC`

und so weiter

4.1 UseCase-Notation

Die Abbildung zeigt die Klassenhierarchie, sowie alle Methoden und Attribute, die eine Klasse bereitstellt.

Die öffentlichen Klassen (in der Abbildung: *Viereck*)

- **`::CLASS`** `[name_of_the_classlib]` **`PUBLIC`**

  - die Attribute der Klasse

  (in der Abbildung*: mittlerer Abschnitt des Vierecks*)

  `::METHOD` `[name_of_the_method]` **`ATTRIBUTE`**

  - die dadurch bereitgestellten Methoden

  (in der Abbildung:*unterer Abschnitt des Vierecks mit "+" gekennzeichnet*)

  `::METHOD` `[name_of_the_method]` **`PUBLIC`**

  - die Klassenhierarchie

  (in der Abbildung: *mit Pfeil und verwendet gekennzeichnet*)

  `::REQUIRES` `"[name_of_the_classlib]"`

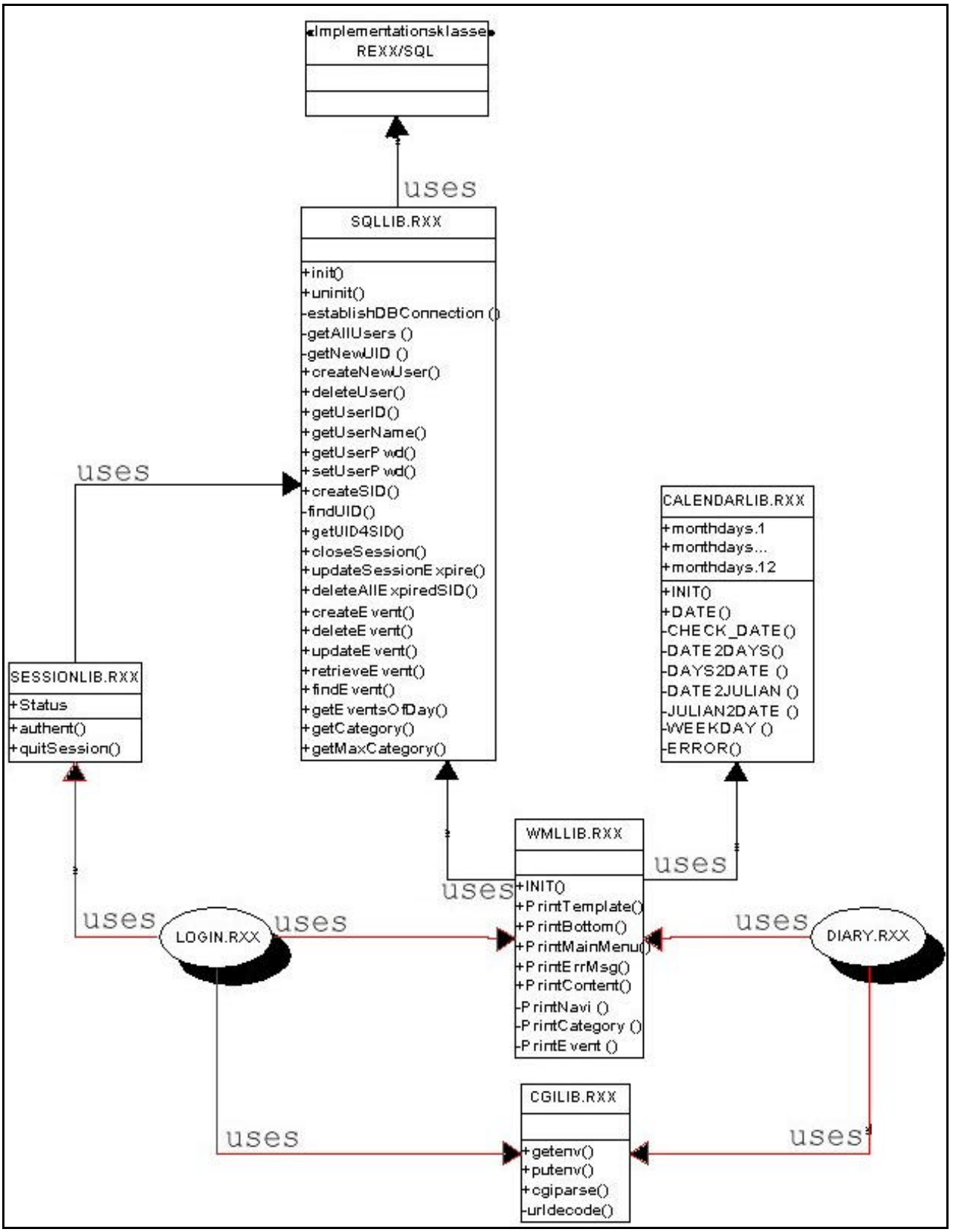


Abbildung des Klassensystems

4.2 Klassendokumentation

Die Klassen werden einzeln dokumentiert, d.h.:

- Die einzelnen Methodenaufrufe,

`[a_instance_of_the_class]~[method]()`

- die erwarteten Argumente

`[a_instance_of_the_class]~[method]`**`(ARG(1),...ARG(n))`**

- und die Rückgabewerte

`[a_instance_of_the_class]~[method]()`

`RESULT`

, die sie liefern.

4.2.1 Eigene Klassen:

4.2.1.1 CGILIB

Beschreibung:

Die Klasse CGI soll den Zugriff auf Umgebungsvariablen vereinfachen. Die Klasse erlaubt es ferner, eine Instanz zu erzeugen, also ein Objekt zu bilden und dann die Methoden aufzurufen. Die Methoden "getenv" und "putenv" vereinfachen das Auslesen und Setzen von Umgebungsvariablen. Im Falle, dass die gewünschte Umgebungvariable der QUERY_STRING ist, kann sie mit der Methode "cgiparse" url-decodiert, geparst und anschließend in dem einzigen Attribut der Klasse "cgi.", einer Stemvariable, gespeichert werden.

(siehe auch die Dokumentation im Quell-Code:
„http://swt.wi-inf.uni-essen.de/~emasovic/src/sqllib.rxx“).

Als Ausgangspunkt für diese Klasse diente die in classic-Rexx geschriebene Prozedur: CgiParse() von Sascha Prins

„CGILIB.rxx“

```
CLASS cgi PUBLIC
   ATTRIBUTE
            cgi.
   METHOD
               INIT()
                 RETURN:
                 CGI-OBJECT: An instance of cgi

               cgiparse()
                 RETURN:
                 CGI.: Stem-variable containig the parsed values
                 QueryString - e.g. user is stored in cgi.user
                 EXPOSE:
                 CGI.

               URLDecode(ARG1)
                 ARGUMENTS:
                 ARG1: An url-encoded string
                 RETURN:
                 decoded ARG1: The url-decoded string

               getEnv(ARG1)
                 ARGUMENTS:
                 ARG1: Name of the environment variable
                 RETURN:
                 value of ARG1: Value of the environment variable

               putEnv(ARG1, ARG2)
                 ARGUMENTS:
                 ARG1: the name of the environment variable
                 ARG2: the value of the environment variable
```

4.2.1.2 SQLLIB

Beschreibung:

Die Klasse **SQLInterface** soll den Zugriff auf die Datenbank regeln. Neben dem Zugriff werden auch Methoden zur Manipulation von Datenbankinhalten, der Datenbank "Calendar-Store", bereitgestellt. In dieser Druckversion werden nur die wichtigsten Methoden vorgestellt. Alle Methoden der SQLLIB sind Abrufbar unter:

„http: //swt.wi-inf.uni-essen.de/~emasovic/klassensystem.htm“.

(siehe auch die Dokumentation im Quell-Code:
„http://swt.wi-inf.uni-essen.de/~emasovic/src/sqllib.rxx“)

> Diese Klasse benutzt das Rexx/SQL - Interface von Mark Hessling
> http://www.lightlink.com/hessling/rexxsql/

„SQLLIB.rxx”

```
CLASS SQLInterface PUBLIC
   REQUIRES
   ATTRIBUTE
   METHOD
                INIT()
                  RETURN:
                  SQL-OBJECT establishing the DB-Connection
                UNINIT()
                  RETURN:
                  closing the DB-Connection
                ...
```

4.2.1.3 WMLLIB

Beschreibung:

Die Klasse **wmldeck** soll den dynamischen Aufbau der Seiten vereinfachen. Häufig benötigte Komponenten werden in dieser Klasse zentral verwaltet, wie z.B. die Processing Instructions "<?xml version="1.0"?>", die mit jedem weiteren Dokument generiert werden müssen . . In dieser Druckversion werden nur die wichtigsten Methoden vorgestellt. Alle Methoden der WMLLIB sind Abrufbar unter:

„http: //swt.wi-inf.uni-essen.de/~emasovic/klassensystem.htm“.

(siehe auch die Dokumentation im Quell-Code:
„http://swt.wi-inf.uni-essen.de/~emasovic/src/wmllib.rxx“)

„WMLLIB.rxx”

```
CLASS wmldeck PUBLIC
 REQUIRES
        "sqllib.rxx"
        "calendarlib.rxx"
  ATTRIBUTE
  METHOD
         INIT()
          RETURN: wmldeck-OBJECT containing following header-elements:
            1. content-type (e.g. 'content-type:text/vnd.wap.wml')
            2. Processing Instructions for the xml-parser
            3. related DTD

         PrintTemplate()
          RETURN: STDOUT: - Creates a back-button

         PrintBottom()
          RETURN: STDOUT: </WML> - Closes the wml-dokument

         ...
```

4.2.1.4 SESSIONLIB

Beschreibung:

Die Klasse **session** ist dient der Authentifizierung von Benutzern. Desweiteren wird nach einer erfolgreichen Authentifizierung eine SessionID generiert und im Attribut "status" gespeichert. Das Auslogen und das anschliessende Löschen der SessionID aus der Datenbank werden ebenfalls von der Klasse bereitgestellt. Die Klasse wird demnach für das Initiieren sowie Beenden einer Session benötigt.

(siehe auch die Dokumentation im Quell-Code:
„http://swt.wi-inf.uni-essen.de/~emasovic/src/sessionlib.rxx“)

„SESSIONLIB.rxx”

```
CLASS session PUBLIC
  REQUIRES
        "sqllib.rxx"
  ATTRIBUTE
         status
  METHOD
         authent(ARG1, ARG2)
          ARGUMENTS:
            ARG1: Login-Name of the current user!
            ARG2: Password of the current user!
          RETURN:
            0 - User not found!
            1 - Password failure!
            SessionID - OK!
          EXPOSE:
            Status

         quitSession(ARG1)
          ARGUMENTS:
            ARG1: The session Identifier
          RETURN:
            0 - error
            2 - Login out!
           EXPOSE:
            status
```

4.2.1.5 CALENDARLIB

Beschreibung:

Die Klasse **`calendar`** stellt Methoden zur Zeitumrechnung bereit. Die Klasse mit ihrer einzigen öffentlichen Methode DATE(ARG(1),ARG(2)) liefert ausgehend vom übergebenen Datum in ARG(2) und der entsprechenden Operation, übergeben in ARG(1), als Ergebnis ebenfalls ein Datum oder eine Datumskomponente (Wochennummer, Monatsname,...). Beachtet werden sowohl der julianische als auch der gregorianische Kalender.

(siehe auch die Dokumentation im Quell-Code:
„http://swt.wi-inf.uni-essen.de/~emasovic/src/wmllib.rxx“)

Als Ausgangspunkt für diese Klasse diente die in classic-Rexx prozedural geschriebene:
datergf.cmd (version: 1.6 - 1996-04-30)
von
Rony G. Flatscher, Rony.Flatscher@wu-wien.ac.at

„CALENDARLIB.rxx”

```
CLASS calendar PUBLIC
  REQUIRES
  ATTRIBUTE
    monthdays.1
    ...
    monthdays.12
  METHOD
    INIT()
     RETURN:
       calendar-OBJECT containing the monthdays array.
     EXPOSE:
       monthdays.
```

DATE(ARG1, ARG2)

ARGUMENTS:

ARG1:

EXAMPLE of the accepted parameters:

IF ARG(2) is the 1./March/2001 then is required value of ARG(2)=20010301 and the RESULT accords to the following flags as ARG(1):

yb - the year begins on **20010101**

ye - the year ends on **20011231**

y - yearnumber: **2001**

ny - next year: **20020101**

py - previous year: **20000101**

mb - the month begins on **20010301**

me - the month ends on **20010331**

m - monthnumber: **3**

nm - next monthnumber: **4**

pm - previous monthnumber: **2**

mn - monthname is **March**

wb - the week begins on **20010226**

pwb - he week begins on **20010219**

nwb - the week begins on **20010305**

we - the week ends on **20010304**

w - weeknumber: **9**

pw - previous weeknumber: **8**

nw - next weeknumber: **10**

d - daynumber in month: **1**

nd - next daynumber in month: **20010302**

pd - previous daynumber in month: **20010228**

dn - dayname is **Thursday**

di - daynumber in week: **4**

ARG2: a date [YYYYMMDD]

RETURN:

[**YYYYMMDD, YYYY, MM, DD, WORD, DIGIT**,...]

the type of the returned value is according to ARG(1)

4.2.2 Vorgegebene Klassen

4.2.2.1 Rexx/SQL - Version 2.3

Die REXX/SQL – Funktionsbibliothek (Version 2.3 - 26 June 1998) [12] wird von der „sqllib.rxx“ verwendet, um mit der SQL-Datenbank „Calendar-Store“ zu kommunizieren.

REXX/SQL [12] besteht aus einer Reihe von externen Funktionen, die die wichtigsten Möglichkeiten anbietet, eine Verbindung herzustellen, auf Daten zuzugreifen und diese in jeder SQL-Datenbank und für jedes SQL-basierende Datenbank-System, welches eine entsprechende 3GL API anbietet zu manipulieren. Mit Hilfe von REXX/SQL ist es möglich, mit der SQL-Datenbank zu kommunizieren.

(siehe hierzu auch http://www.lightlink.com/hessling/rexxsql/)

5. Skripte

5.1 login.cgi

Das „login.cgi“-Skript setzt die benötigten Umgebungsvariablen, bevor es das „login.rxx“-Skript mit dem entsprechenden Rexx-Interpreter ausführt. Die Umgebungsvariablen dürfen nicht zur Laufzeit der eigentlichen „---.rxx“-Skipte gesetzt werden, da offenbar der Rexx-Interpreter zur Laufzeit die shell-Umgebung nicht beeinflussen kann. Diese Vorgehensweise ist auch ein erster Schritt gewesen, die eigentlichen implementationsabhängigen und konfigurationstechnischen Informationen, in eine separate Datei auszulagern. Neben den Pfaden der benutzten Bibliotheken kann auch der Pfad des Rexx-Interpreters hier einmalig angegeben/geändert werden (Dank an Thomas Jungmann, Martin Rüschhoff und Carsten Mjartan für die freundliche Hilfe!).

(siehe auch die Dokumentation im Quell-Code:
„http://swt.wi-inf.uni-essen.de/~emasovic/src/login.txt“)

5.2 login.rxx

Das Einloggen und das Ausloggen wird von „login.rxx“-Skript abgearbeitet. Beim Einloggen wird der User authentifiziert und bei Erfolg eine Session-Identifier nach dem Random-Verfahren generiert. Mit Hilfe des Session-Identifiers (eine zwölfstellige Zahl) lässt sich der User bei jeder weiteren Anfrage eindeutig identifizieren, so dass eine weitere Authentifizierung nicht mehr notwendig ist. Beim Ausloggen wird der Session-Identifier aus der Session-DB unwiderruflich gelöscht. Der Benutzer muss sich bei einer weiteren Anfrage erneut einloggen.

(siehe auch die Dokumentation im Quell-Code:
„http://swt.wi-inf.uni-essen.de/~emasovic/src/login.rxx“)

5.3 diary.cgi

Übernimmt dieselben Funktionen wie das „login.cgi“. Beide Skripte unterscheiden sich lediglich in dem anschließend aufgerufenen Skript, d.h. das „login.cgi“ startet anschließend mit dem entsprechenden Rexx-Interpreter das „login.rxx“. Das „diary.cgi“ startet „diary.rxx“.

(siehe auch die Dokumentation im Quell-Code:
„http://swt.wi-inf.uni-essen.de/~emasovic/src/diary.txt“)

5.4 diary.rxx

Alle Anfragen der Benutzer werden von diesem Skript entgegengenommen und entsprechende Schritte eingeleitet. Die mit der Anfrage gesendeten Informationen werden ohne Überprüfung abgearbeitet. Lediglich die Session-ID wird validiert, anschließend werden die gewünschten Informationen (passend zur Session!) angezeigt. Anhand der SessionID wird der User ermittelt und dessen Daten werden dann entsprechend seinen gewünschten Anzeigeoptionen auf dem mobilen Endgerät angezeigt. Die Modifikation der Daten bzw. das Anlegen von neuen Terminen wird nicht angeboten, aufgrund der relativ schlechten Eingabemöglichkeiten und des dadurch entstandenen Zeitverlustes, sowie der höheren „Surf“-Kosten durch lange Verweildauer im Web.

(siehe auch die Dokumentation im Quell-Code:
„http://swt.wi-inf.uni-essen.de/~emasovic/src/diary.rxx“)

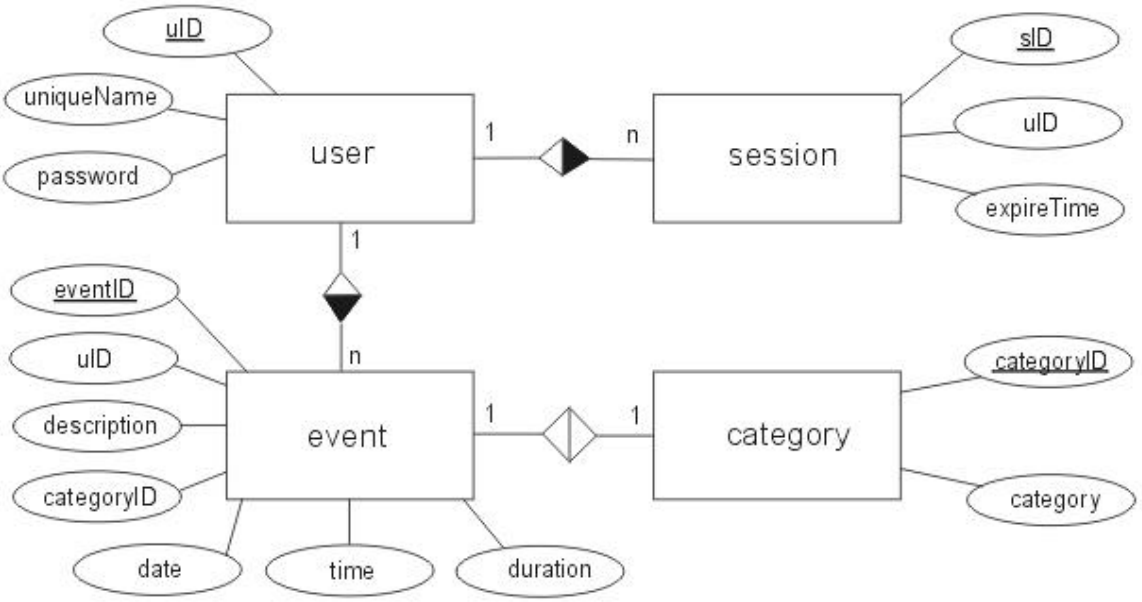
6. Datenbankkonzeption

6.1 „CalendarStore“

Bemerkung: Nach ICalendar [1] sollte ein Terminkalender unter anderem aus folgenden Komponenten bestehen: `TO-DO, EVENT, JOURNAL`. Für diese Seminararbeit bedeutet dies, dass es mindestens zwei Kategorien (`TO-DO, EVENT`) an Terminen geben muss, mit jeweils einer dazugehörigen Terminbeschreibung (`JOURNAL`). Diese Vorgehensweise bedingt implizit zwei relationale Tabellen (**Category** und **Event**)!

Die relationalen Tabellen (**User** und **Session**), die für das Sessionmanagement benötigt werden, kommen noch hinzu. Aufgrund der Tatsache, dass auf mobilen Endgeräten derzeit keine Cookies gespeichert werden können ist eine solche Vorgehensweise zwingend.

Siehe Dazu auch die Datenbankkonzeption:
http://swt.wi-inf.uni-essen.de/~tjungman/datenbank.html
von Thomas Jungmann!



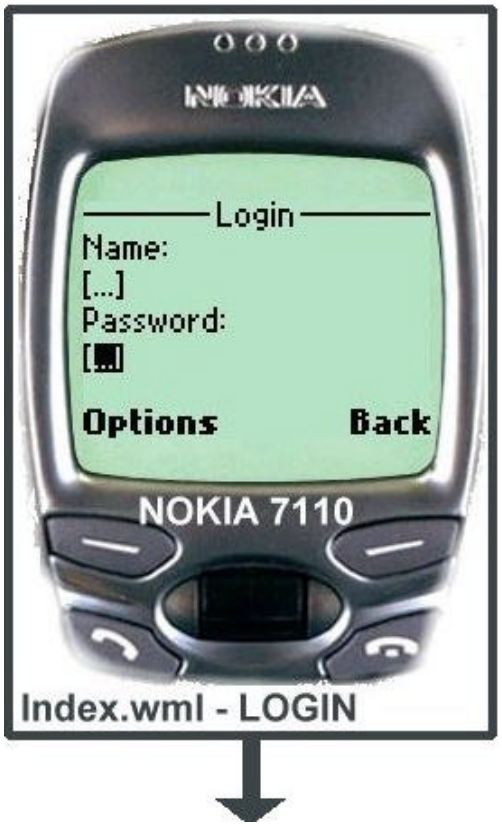
ER-Modell von der Datenbank „CalendarStore“

- **user** (uID,uniqueName,password)
- **event** (eventID,uID,categoryID,description,date,time,duration)
- **session** (sID;uID,expireTime)
- **category** (categoryID,category)

7. Screenshots des Prototypen

7.1 Login-Menu

Die Login-Seite („index.wml“) ist die Einstiegsseite und die einzige statische Seite im ganzen System. Genauer gesagt ist die Login-Seite ein WML-Formular, welches als Eingabe den Benutzer-Namen und das Benutzer-Passwort erwartet. Die vom Benutzer eingegebenen Daten werden mittels der GET-Methode im QUERY-STRING [11], an das „login.rxx“-Skript weitergeleitet. Das Skript verifiziert die Eingaben mit den Einträgen in der relationalen Tabelle **User** in der Datenbank **„Calendar-Store“** und zeigt bei Erfolg das Main-Menu, also die benutzerspezifische Ansichtsoptionen-Auswahl-Maske (Main-Menu). Sollte die Verifizierung fehlschlagen, wird vom „login.rxx“-Skript eine entsprechende Fehlermeldung temporär generiert. Temporär bedeutet, dass ein Timer gesetz wird, der eine WML-Seite zeitlich begrenzt anzeigt und dann wieder auf eine andere Seite (oder einfach wieder zurück!) wechselt.



„Index.wml“

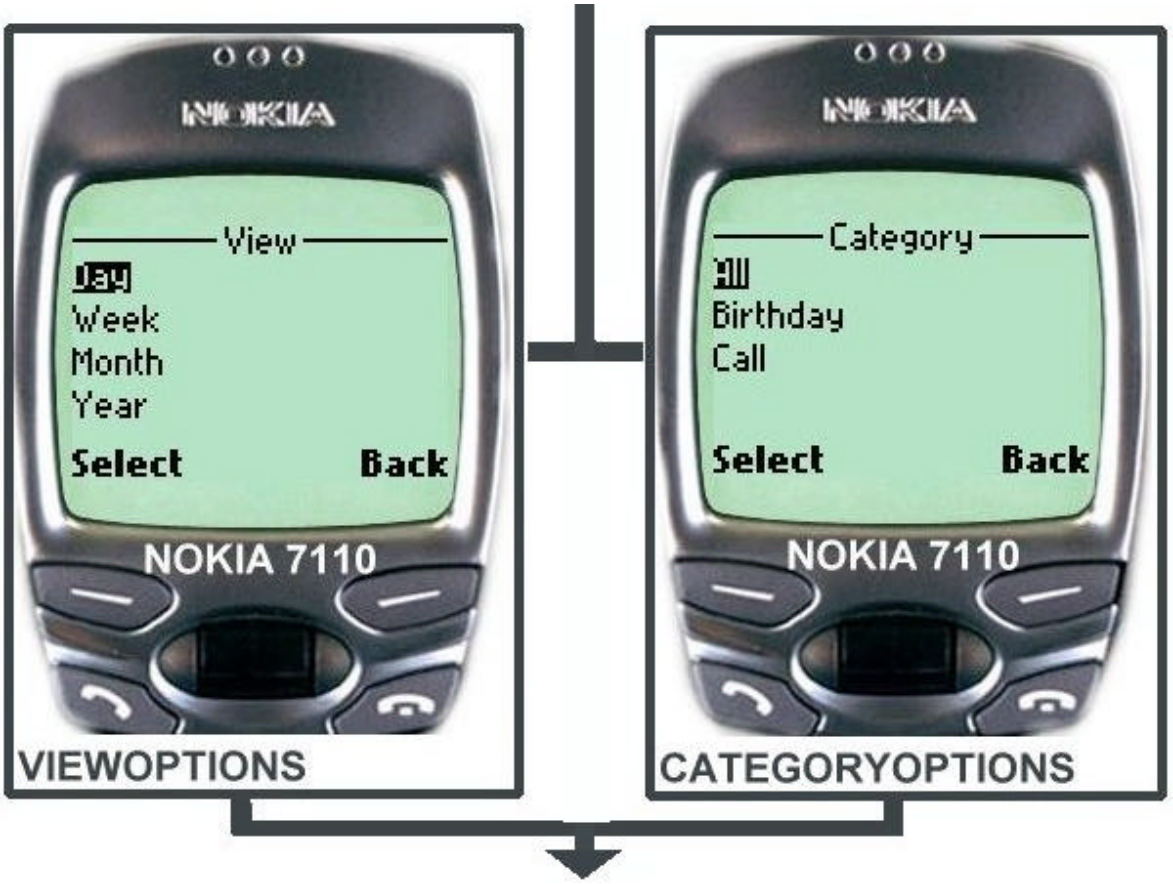
```
<?xml version="1.0"?>

<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD
WML 1.1//EN"
 "http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
  <template>
    <do type="prev" label="Back" optional="false">
      <prev/>
    </do>
  </template>
<card id="login" title="Login" newcontext="false">
<p align="center">
Name: <input name="user" value=""/>
Password:  <input name="pw" value=""
type="password"/>
<do type="accept" label="login">
 <go href="cgi-bin/login.cgi" method="get"
sendreferer="false">
  <postfield name="type" value="wml" />
  <postfield name="action" value="login" />
  <postfield name="user" value="$(user)" />
  <postfield name="pw" value="$(pw)" />
  <postfield name="sid" value="" />
 </go></do></p>
</card></wml>
```

7.2 Main-Menu

Nach erfolgreichem Einlogen wird das Main-Menu angezeigt und der Benutzer hat die Möglichkeit, die gewünschte Anzeigeoption (**Year, Month, Week, Day**) auszusuchen („**View**“-Feld). Neben der Anzeigeoption kann der Benutzer auch die gewünschte Terminkategorie (**Call, Birthday, All,...**) aussuchen und sich nur eine Bestimmte Art von Kategorien anzeigen zu lassen („**Category**“-Feld).

Die Eingabe-Felder

Über das Main-Menu wird das „Diary.rxx“-Skript ausgeführt und alle weiteren Anfragen werden ab nun nur noch von diesem Skript ausgeführt. Die **logout**-Funktion wird im **options**-menu bereitgestellt.



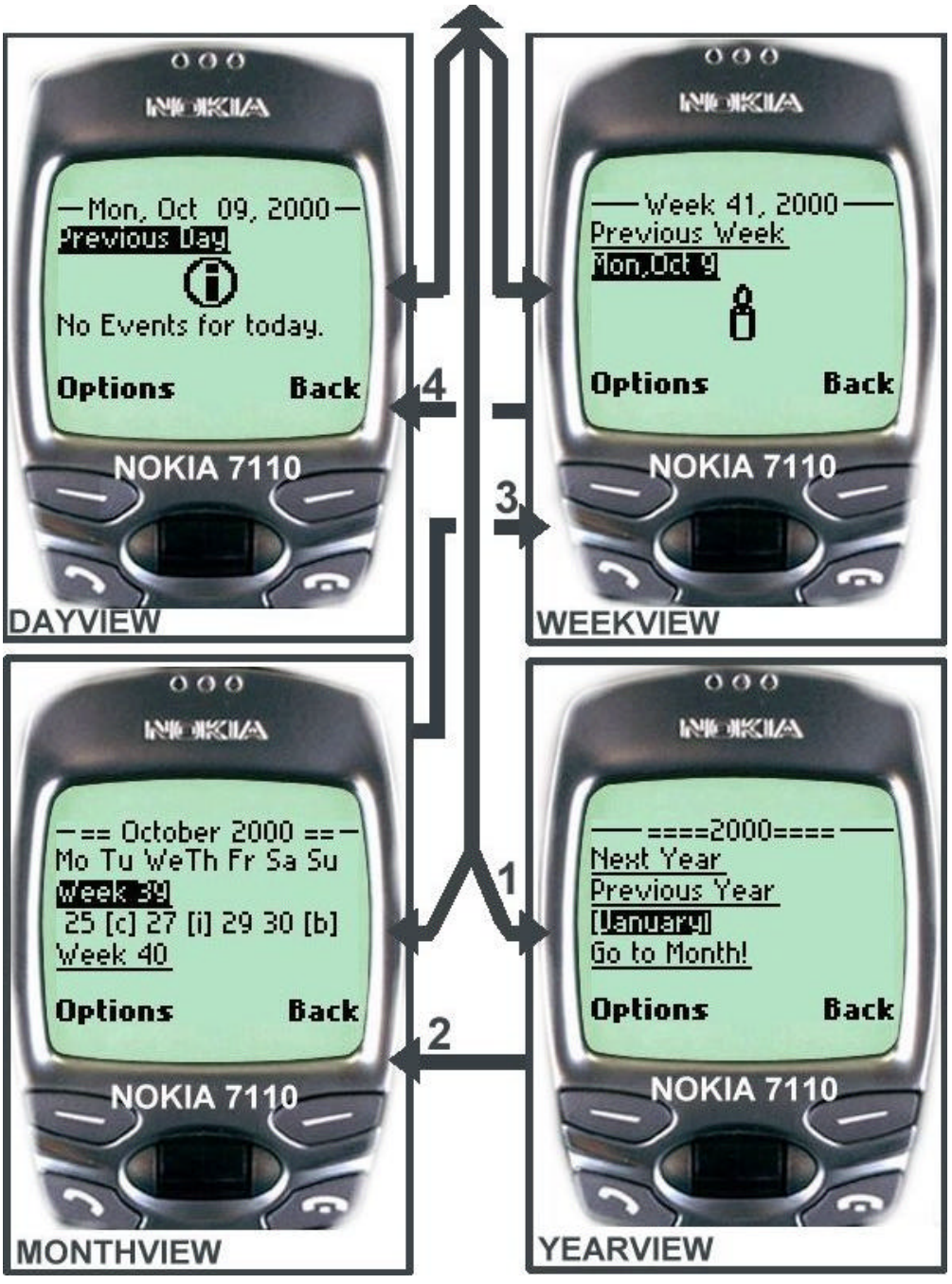
Auswahloptionen der einzelnen Eingabe-Felder

7.3 View-Types

Grundsätzlich ist es möglich vom Main-Menu aus direkt jede Ansichtsoption zu wählen. Die Abbildung zeigt ein konkretes Beispiel für die Navigation. Die Ziffern auf den Pfeilen verweisen auf die Navigationsmöglichkeiten, von Ansichtstyp zu Ansichtstyp, d.h. wenn man im Main-Menu „Year-View“ gewählt hat, muss man vier Schritte (main-menu -> year -> month -> week -> day) machen bis zum gewünschten Termineintrag.

Es ist nicht möglich, von der Ansichtsoption „Year-View“ direkt nach „Day-View“ oder von „Month-View“ eine Ebene rauf nach „Year-View“ zu wechseln, usw...

Ansichtstypen und die Navigationsmöglichkeiten



8. Ausblick

Das System bietet noch keine Erinnerungsfunktion per SMS oder ähnlichem. Eine Erinnerungsfunktion würde das System um einen notwendigen Bestandteil ergänzen und als Produkt auf dem Markt konkurenzfähiger machen.

Die Datenübertragung vom WAP-Gateway zum Client könnte man mit WTSL [3] absichern und somit das System und die Daten des Users gegen Angriffe von Außen besser schützen.

9. Rechercheergebnisse

Reference

[1] - [RFC2445] – "**Internet Calendaring and Scheduling Core Object Specification**", F. Dawson, D. Stenerson - November 1998 - URL: http://www.ietf.org/rfc/rfc2445.txt

[2] - „**WAP WML**" – WAP-191-WML – WAPFORUM - 19 Februar 2000 – URL: http://www.wapforum.org/what/technical.htm

[3] - „**WAP Architecture Specification**" – WAPFORUM - 30-Apr-1998 – URL: http://www.wapforum.org/what/technical.htm

[4] - "**Extensible Markup Language (XML) 1.0 (Second Edition)",** Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler - W3C Recommendation 6 October 2000 – URL: http://www.w3.org/TR/REC-xml

[5] - "**HTML 4.01 Specification"** – Dave Ragget, Arnaud Le Hors, Ian Jacobs - W3C Recommendation 24 December 1999 – URL: http://www.w3.org/TR/html401/

[6] - **"The WWW Common Gateway Interface Version 1.1"**, Ken A L Coar, D.R.T. Robinson, INTERNET-DRAFT 25 June 1999 – URL: http://CGI-Spec.Golux.Com/draft-coar-cgi-v11-03.txt

[7] – [RFC 2616] – "**HTTP 1.1**" - T.Berners-Lee, P. Leach, L. Masinter, H. Frystyk, J. Mogul, J. Gettys - June 1999 - URL: http://www.w3.org/Protocols/rfc2616/rfc2616.html

[8] - IETF working group - URL: http://www.ietf.org/

[9] - W3C Technical Reports and Publications – URL: http://www.w3.org/TR/

[10] - Apache HTTP Server Project – URL: http://httpd.apache.org

[11] – [RFC2396] – „**Uniform Resource Identifiers URI**" - T. Berners-Lee, R. Fielding, U.C. Irvine, L. Masinter - August 1998 – URL: http://www.ietf.org/rfc/rfc2396.txt

[12] – "**Rexx/SQL"** - powerful interface to SQL databases – M. Hessling - 12 September 2000 – URL: http://www.lightlink.com/hessling/rexxsql/

Software / Literatur

- REXX-libraries
Rexx/SQL
URL: http://www.lightlink.com/hessling/rexxsql/
datergf.cmd
URL: http://swt.wi-inf.uni-essen.de/~emasovic/seminar/src/datergf.txt
CGIParse()
URL: http://swt.wi-inf.uni-essen.de/~emasovic/seminar/src/cgiparse.rxx
CGI-lib
URL: http://www.slac.stanford.edu/slac/www/resource/how-to-use/cgi-rexx/
...more libraries
URL: http://wuarchive.wustl.edu/systems/os2/dev/rexx/

- SQL-Datenbank
My-SQL 2.4 - URL: http://www.tcx.se/

- Server
Aktuelle Apache-Version : http://httpd.apache.org/dist/apache_1.3.14.tar.gz

- REXX-Dokumentation
URL: http://users.comlab.ox.ac.uk/ian.collier/RexxDocs/index.html

- WML-Dokumentation
Nokia-Wapkurs - URL: http://7110.nokia.de/wapkurs/wapkurs_set.html
Einführung in WML - URL: http://www.boku.ac.at/htmleinf/

- Vorhandenes Angebot
WAP@Büro
URL: http://www.wap-consulting.de/Produkte/WAPatBuro/wapatburo.html
Waphome - URL: http://waphome.ch/