Universität Essen

information systems and software engineering

department 5, information systems

Universitätsstraße 9

D-45141 Essen

Seminar work DII:

Design and implementation of a web-based calendar system

(WML - Version)

Submitted the department of economic science
of the University of Essen from

Ednan Masovic

Süllenstr. 78

40599 Düsseldorf

Matr-Nr. : 1080774

responsible person: Reinhold Klapsing

delivered at: 10. Dezember 2000 (WS 2000/01)

Table of contents

1. Introduction

If the Web information are made available not only in HTML [5] but also in WML [2] via a WAP-gateway [3] then you do not only achieve one of the classical Internet user, who uses the local terminals (PC, TV,...) for „Internet-Surfing“ but also the substantially more strongly growing number of persons, who possess a mobile device (e.g. WAP-device) and gots informations with it.

A combined use of local and mobile devices for "Internet Surfing" would be the alternative to achieve a larger number of Internet users and to broadcast the own Web information to a broader mass than up to now (only in HTML [5]!).

A critical aspect with the development of HTML and WML [5,2] information systems is the developping time and the associated costs, since one the information must be available always in two versions on the Web server. There are special WAP gateways, which tries to convert existing HTML information, so well goes, automatically in WML [2]. Usually such a automated transformation is not sufficient however, because the information for the small displays must be structured completely differently. That means each updating of the HTML files draws updating of the WML files after itself and the work expended would almost double itself.

A further important aspect are the relatively high costs, which come to the Internet user if he uses the mobile device, since "Internet-Surfing" is up-to-date with local terminals 20 times more favorable than with mobile devices.

Also the input comfort (mouse, PC keyboard (101/102 keys)) is locally by far better and the relatively large and multicolored display (PC monitor) increases the legibility of the data. So the user can dedicates themselves completely to its original intention, to the information query, without watching on the clock and trying with the 20 keys, which are usualy available on mobile devices, to manage the "Web jungle".

All these aspects play a central role during the system development and - implementation of the '**Web Calendar**", as hybrid web information system for local and mobile devices, i.e. the information are made available in HTML and WML. This document deals with the development and implementation of the WML version of the system available at:

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

The HTML version processes Thomas Jungmann in his work available at:

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

1.1 „Web-Calendar“

In this work both, the HTML and the WML contents, are not static files. They are generated automatically and dynamically from the information stored in the database “CalendarStore” by "Rexx" scripts. The updating of the HTML and WML files is limited to the updating of the used database "CalendarStore“. The time invested in the updating of the combined Web information remains equivalent large to single solution (only HTML!).

The costs for the user of the mobile Web supply can be influenced unfortunately only indirectly by trying to design the pages "as simple and practical as possible", i.e. not to offer multimedially complex solutions for the mobile devices, since these are connected with long loading times and cause therefore higher costs.

The relatively worse input modes of the mobile devices (only a digit block) are only used for navigation and the data inputs has to be reduced to the most necessary ones (username and password). In this work the appointments are maintained locally with the HTML Browser and only recalled mobilely with the WAP capable device. The actual appointment the user agent gets in 2 to 5 steps over a simple and self-describing navigation structure (see the chapter 7 - " Screenshots of the prototype ").

A prototype for mobile devices (*only retrieve of appointments*!) can be found at:

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

The HTML-prototype (*edit, retrieve,… of appointments*!) can be found at:

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

1.2 Milestones of the work

In the further process of the document first the system is independently of the used scripting language developed and afterwards in Rexx implemented. The description of the Rexx specific implementation can be found in chapters 4 and 5. The remaining sections deal with the documentation of the system independently of the used scripting language.

2. Requirements analysis

2.1 Retrieving of personal appointments by the Web

The task is to create a web-based application, which enables the user agent to retrieve existing entries from its personal appointment calendar by mobile devices (WAP-device).

The specifications for WML and WAP are to be found in the references [2,3]. At present the most of the WAP devices, which are used up-to-date by users, do not completely supports even the version 1.1 of WML (defined 1999). For this reason when testing the system an emulator of the first WAP capable device is used, the “Nokia 7110”. The emulator: “NOKIA WAP Toolkit version 2.0” offers Nokia to each WAP developer free of charge at the disposal: http://www.forum.nokia.com/ .

Application is implemented over WML-form (“GET”-method), which communicates over the pre-defined CGI [6] with Rexx-scripts.

The respective appointments are administered in a SQL database, which is addressed by appropriate Rexx procedures. The data as well as the procedures for data manipulation (insert, delete and edit) generates Thomas Jungmann in the context of his project.

Application must offer a authentification mechanism (e.g. password query), in order to check the identity of the user agent and to show the user specific appointments.

A following Session-Management [5] manages the problems of the used HTTP [7] with establishing a permanent connection, i.e. a authentification is necessary only at the first request. At no further request the session is to be terminated automatically after exceeding of a given time limit or by the user agent (logout).

Due to the relatively bad input possibilities (no mouse, compact keyboard...) of the used mobile devices the appointments are to be retrieved with few keyboard entries.

2.2 Emphasis of this work

- Apart from the user specific information query from a SQL database is afterwards the dynamically generated WML output for WAP Devices, which support WML1.1 [2] to be "optimal" arranged.

- Furthermore this work demonstrates nearly the technical possibilities, which are given on the market by the different manufacturers of mobile devices, today.

- The question is to be clarified, in what respect the different standards become transferred by the manufacturers, e.g. WML 1.1[3], WML 1.2, WML 1.3.

- A further important aspect is the use of Object Rexx as Scripting engine and the associated difficulties with the implementation of the prototype.

2.3 Needed components

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**	

components to be developed

2.3.1 Session-Management

Transaction procedure with HTTP

- **Connection establishment** by the user agent
- **Request:** of the user agent to the WWW server a certain object to transmit
- **Response:** of the WWW-Servers
- **Connection closed**

After each request the Web server goes into its origin status back. It does not exist a possibility for the web server to bring two sequential requests in connection and so to start up a session. The session management eliminates this "weakness" of the HTTP [7] and enables a session between the client and the server. There are exactly three mechanisms to arrange a session with HTTP:

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

In this work the Post-Fields are used to insert a unique character string in each displayed page (session Identifier) for the identification of the users. This unique character string transmits the user agent with each request to the server.

e.g. („Index.wml“) for 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-Scripts

The application is implemented with Object Rexx - classes and - scripts. The scripts are needed for access to the database and to generate the dynamic WML-decks and - cards.

2.3.3 WML-Decks

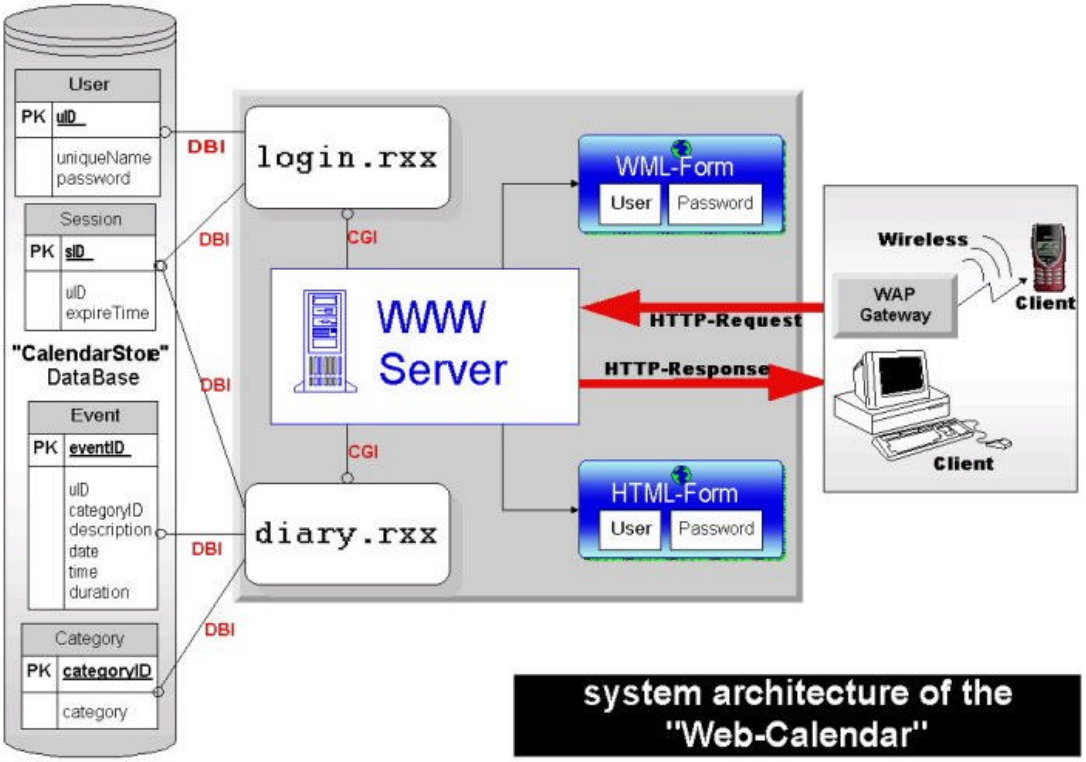
HTML pages cannot be displayed on a WAP device. This problem is solved by the WML-decks defined in WML1.1 [2]. WML is an application of XML [4]. In this system the WML-decks are generated dynamically containing information, which was taken from the database "CalendarStore".

2.3.4 SQL – Database

The database "CalendarStore" contains four relation tables, performing following features:

- administration of users and passwords of the system,
- storing information for the session management temporarly,
- categorizing the appointments,
- storing the journals of the individual events.

3. System architecture



System architecture of the web-based appointment calendar

3.1 Clients

The client can be a web browser of a PC or a WAP-device communicating via the Wireless Application Protocol (WAP [3]) with a WAP gateway.

3.2 HTTP and WAP

WAP gateway communicates with the Web server via Internet with the Hypertext Transfer Protokol (HTTP [7]). "The Wireless Application Protocol (WAP [3]) is a result of the WAP Forum's efforts to promote industry-wide specifications for technology useful in developing applications and services that operate over wireless communication networks".[1]

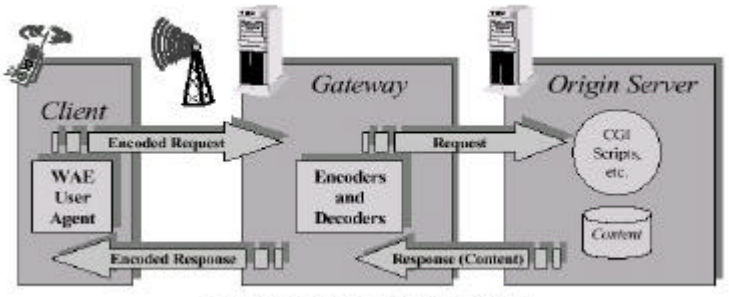


Figure 2. WAP Programming Model

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

3.3 WWW-Server

The used WWW-server is an Apache [10]. It receives requests as HTTP Daemon and sends the results back. With the first request the server sends the "index.wml" file, a static WML or HTML-Form [5] generated by the system, as response or result.

3.4 HTML vs. WML

“WML is a markup language based on XML [4] and is intended for use in specifying content and user interface for narrowband devices, including cellular phones and pagers.”[2],

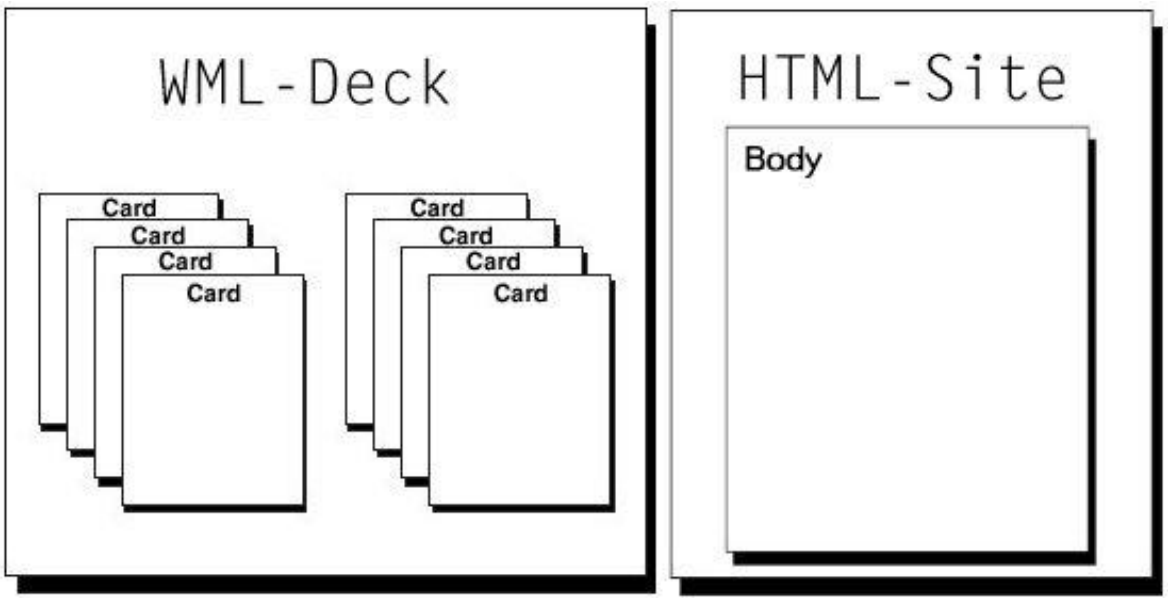
for e.g.:

- Mobil-Phones (Devices, GSM)
- PDA’s (Personal Digital Assistants, Organizer)
- Palmtop-Computer
- Auto-navigation devices

WML was developed with XML [4], therefore each WML-Deck [2] has to start with the following directives to the XML-Parser:

```
<?xml version="1.0"?>
```
✍ directive for the XML Parser

```
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN" „http://www.wapforum.org/DTD/wml_1.1.xml">
```
✍ The URL of the DTD, to check if document is"valid"



WML-Decks und HTML-Sites

[2] „WAP WML“ – WAP-191-WML - 19 Februar 2000 –URL: http://www.wapforum.org

3.5 Starting up a session - login.rxx & cgi

At the beginning of the session the login-form (WML deck) of the server is requested by a mobile device. The input data (name, password) are transmitted in the QUERY_STRING of the URL [11] to the Web server, which this passes to the **login.rxx**-script via CGI [6]. The inputs are validated with the entries in the user Datenbank (My SQL[3]) via Rexx/SQL [12]. Rexx/SQL provides a powerful interface to SQL databases. In the case of succesfull validaton a session ID is generated, assigned to the user and stored in the session database temporarly. The script suplies also the logout function.

Remark: After first determination of the session ID it will be transferred with each further request by post-fields to the client. After each transfer of the session ID the time limit of the current session is set to maximum in the session table of the "CalendarStore" database. If in a certain time period no further requests take place then the session ID is deleted automatically in the data base and so the session is terminated. The termination can also be done by the user agent (logout).

3.6 View selection - diary.rxx & cgi

After the log in process was successfully concluded generates the application a dynamic WML page, welcoming the user and offering various view options (Year-, Month-, Week-, Dayview) and category options (TO-DO and EVENT specified in ICalendar [1]). The selected view and category options are passed as control information in the QUERY_STRING ("GET"-method) via GCI [6] to the script. After selection of the options a further dynamic WML page with user specific appointments is created. The appointments of the user are determined via DBI accessing the diary-database. For this the **diary.rxx**-script will serve.

Remark: The different timezones are not considered by the application. The timezone corresponds to the geographical position of the web-server. Both, the Julian and the Gregorian calendar are supported as calenderscala. In 1582 the error in the Julian calendar accounted already for 10 whole days. The roman catholic pope Gregor XIII therefore had the Julian calendar corrected by skipping 10 days in 1582 (4. October was followed by the 15.). The Gregorian calendar considers apart from the usual leap years every 4 years still

[3] AktuelleVersion My-SQL 2.4 - URL: http://www.tcx.se/

another further leap year every 400 years. That means, the actual leap years 1700, 1800, 1900 are not leap years, because they are not dividable by 400. However the years 1600, 2000, 2400... are regarded as leap years. That is to be attributed to the fact that the earth needs approx. 364 days 5 hours and 48 minutes to circle around the sun once.

4. Class system

In further paragraph are nearly described the individual classes and their relations. Generaly, each class for itself is stored in a separate file. All file names (almost all,;)) correspond to the class name with the addition "... lib.rxx ", i.e. the file "**sessionlib.rxx**" contains the class **session**.

e.g.:

File: "**cgilib.rxx**" containes only a public class cgi

- `::CLASS cgi PUBLIC`

File: "**sessionlib.rxx**" containes only a public class session

- `::CLASS session PUBLIC`

and so on

4.1 UseCase-Notation

The figure shows the class hierarchy, as well as all methods and attributes, the one class supplies.

The public classes (in the figure “class system”: *Square*)

- `::CLASS [name_of_the_classlib] PUBLIC`

  - the attributes of the class

  (in the figure: *middle paragraph of the square*)

  `::METHOD [name_of_the_method] ATTRIBUTE`

  - the suplied methods

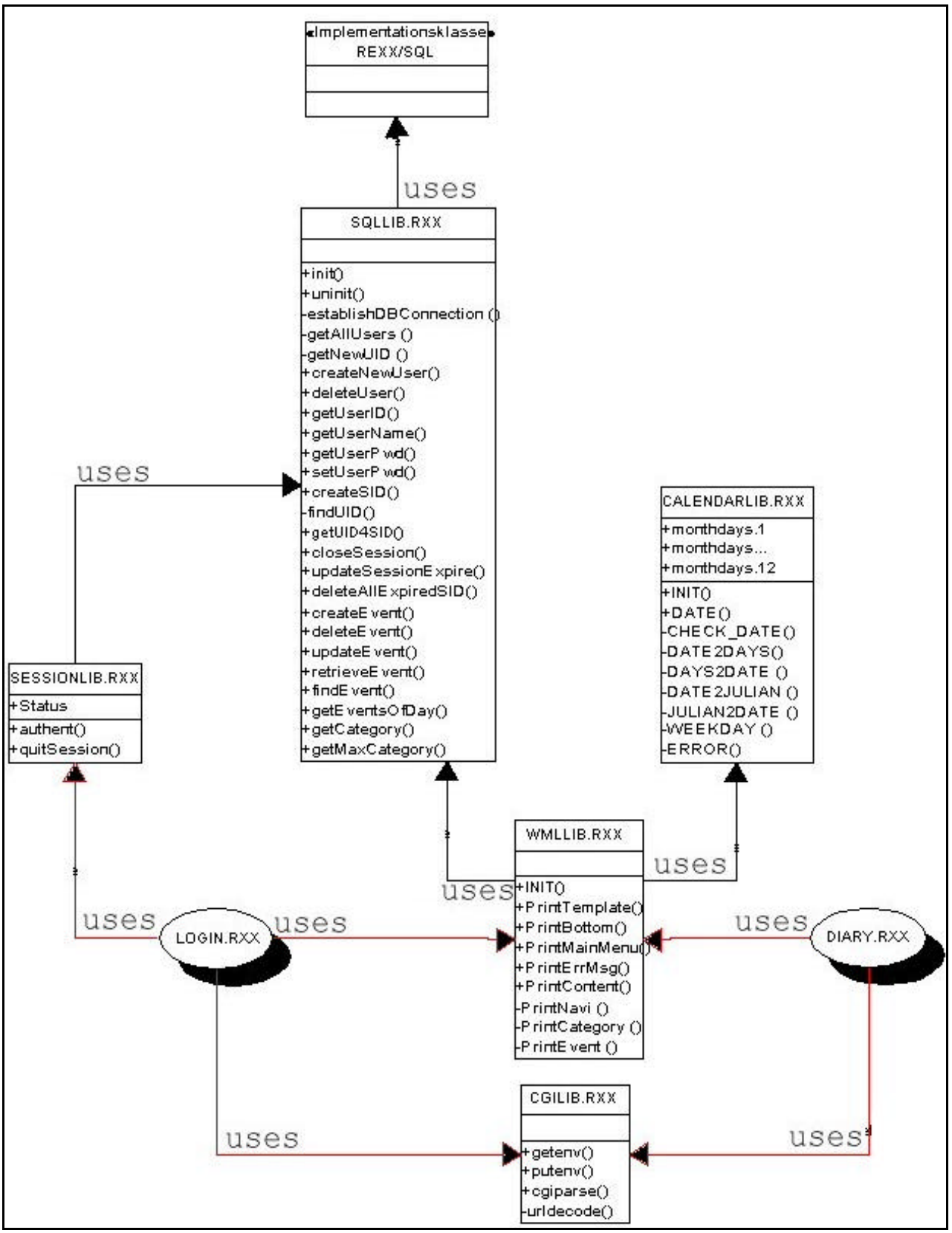
  (in the figure: *lower paragraph of the square signed with "+”*)

  `::METHOD [name_of_the_method] PUBLIC`

  - the class hierarchy

  (in the figure: *with arrow characterized*)

  `::REQUIRES "[name_of_the_classlib]"`



Class system

4.2 Class documentation

The classes are documented separately, i.e.:

- Individual method calls,

`[a_instance_of_the_class]~[method]()`

- the expected arguments

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

- and the return values (RESULT in Rexx)

`[a_instance_of_the_class]~[method]()`

`RESULT`

, they supply.

4.2.1 Own classes

4.2.1.1 CGILIB

Description:

The class cgi is to simplify the access to environment variables. The methods "**getenv**" and "**putenv**" simplify the selection and the setting of environment variables.

In the case that the value of the QUERY_STRING is desired the class suplies a method "cgiparse", which url-decodes, parses and finaly stores the values in the only attribute of the class "cgi.", a Stemvariable.

(see also the documentation in the source code:

„http://swt.wi-inf.uni-essen.de/~emasovic/src/sqllib.rxx“).

As starting point for this class served the procedure wroted in classic Rexx:
CgiParse()
from
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

Description:

The class **SQLInterface** is to regulate the access to the database. Apart from the access also methods are supplied for the manipulation of database contents of the database " CalendarStore". In this printing version not all methods, only the most important are presented. All methods of the SQLLIB. There are here:

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

(see also the documentation in the source code:
„http://swt.wi-inf.uni-essen.de/~emasovic/src/sqllib.rxx“)

This class uses the Rexx/SQL - INTERFACES of 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

Description:

The class **wmldeck** is to simplify the dynamic generating of the WML-decks. Frequently necessary components are central administered in this class, e.g. the processing instructions "<?xml version="1.0"?>". With each further document the processing instructions must be generated. In this printing version not all methods, only the most important are presented. All methods of the WMLLIB. There are here:

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

(see also the documentation in the source 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

Description:

The class **session** supplies a method needed for authentification of users. Furthermore after a successful authentification a session ID is generated and stored in the attribute "status". The outlog-function with the following deletion of the session ID from the database "CalendarStore" is also supplied by the class. The class is needed to start up as well as terminate a session.

(see also the documentation in the source 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

Description:

The class **`calendar`** suplies methods that symplify operations with dates , i.e.. the class with its only public method **DATE(ARG(1), ARG(2))** supplies according to the transferred date in ARG(2) and the appropriate operation, transferred into ARG(1), as result a date or a date component (week number, name of the month...). Both are considered the Julian one and the Gregorian calendar.

(see also the documentation in the source code:
„http://swt.wi-inf.uni-essen.de/~emasovic/src/wmllib.rxx“)

As starting point for this class served the datergf.cmd wroted in classic Rexx: datergf.cmd (version: 1.6 - 1996-04-30) from 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 Used classes

4.2.2.1 Rexx/SQL - Version 2.3

The REXX/SQL [12] - function library (version 2.3 - 26 June 1998) is used by "sqllib.rxx" to communicate with the SQL database "CalendarStore".

“Rexx/SQL [12] consists of a number of external Rexx functions which provide the necessary capabilities to connect to, query and manipulate data in any SQL database”[4]. With REXX/SQL it is possible to communicate with the SQL database.

(see also http://www.lightlink.com/hessling/rexxsql/)

[4] Rexx interface to SQL databases - Version 2.3 - 26 June 1998 – © **Mark Hessling**
URL: http://www.lightlink.com/hessling/rexxsql/

5. Scripts

5.1 login.cgi

The "**login.cgi**" script is a shell script . It sets the necessary environment variables before it executes the “login.rxx" script with the appropriate Rexx interpreter. The environment variables may not be set at run-time of the actual Rexx-scipts because the Rexx interpreter is obviously not able to manage it. This methodology was also a first step to store the actual implementation-dependent and the configuration-technical information externally into a separate file. Beside the paths of the used libraries also the path of the Rexx interpreter can be simply indicated/modified here (thanks at Thomas Jungmann, Martin Rueschhoff and Carsten Mjartan for the friendly assistance!).

(see also the documentation in the source code:
„http://swt.wi-inf.uni-essen.de/~emasovic/src/login.txt“)

5.2 login.rxx

The "**login.rxx**" script is a Rexx script. Logging in and the logging out are processed by this script. After the succesfull authentification a session identifier is generated in the random procedure. With the help of the session identifiers (a twelve-digit number) the user lets itself be identified with each further request, so that a further authentication is no longer necessary. After logging out the session Identifier is deleted from the session table of the “CalendarStore” database and the users must log in again with a further request.

(see also the documentation in the source code:
„http://swt.wi-inf.uni-essen.de/~emasovic/src/login.rxx“)

5.3 diary.cgi

The "**diary.cgi**" script is a shell script.It takes over the same functions as the "login.cgi". The two scripts differ only in the Rexx scripts called afterwards, i.e. the "login.cgi" starts afterwards with the appropriate Rexx interpreter "login.rxx" and "diary.cgi" starts "diary.rxx ".

(see also the documentation in the source code:
„http://swt.wi-inf.uni-essen.de/~emasovic/src/diary.txt“)

5.4 diary.rxx

The "**diary.rxx**" script is a Rexx script. All requests of the users are intercepted by this script and appropriate steps are initiated. With the request transmitted informations are processed without being checked. Only the session ID is validated and afterwards the desired information (suitable to the session!) displayed. The user will become determined by the session ID and his appointments will be displayed according to the desired display options on his mobile device. The modification of the appointments or the creation of new ones is not offered, due to the relatively bad input modes and the time delay resulted from it, as well as the higher "surf"-costs of most mobile devices.

(see also the documentation in the source code:
„http://swt.wi-inf.uni-essen.de/~emasovic/src/diary.rxx“)

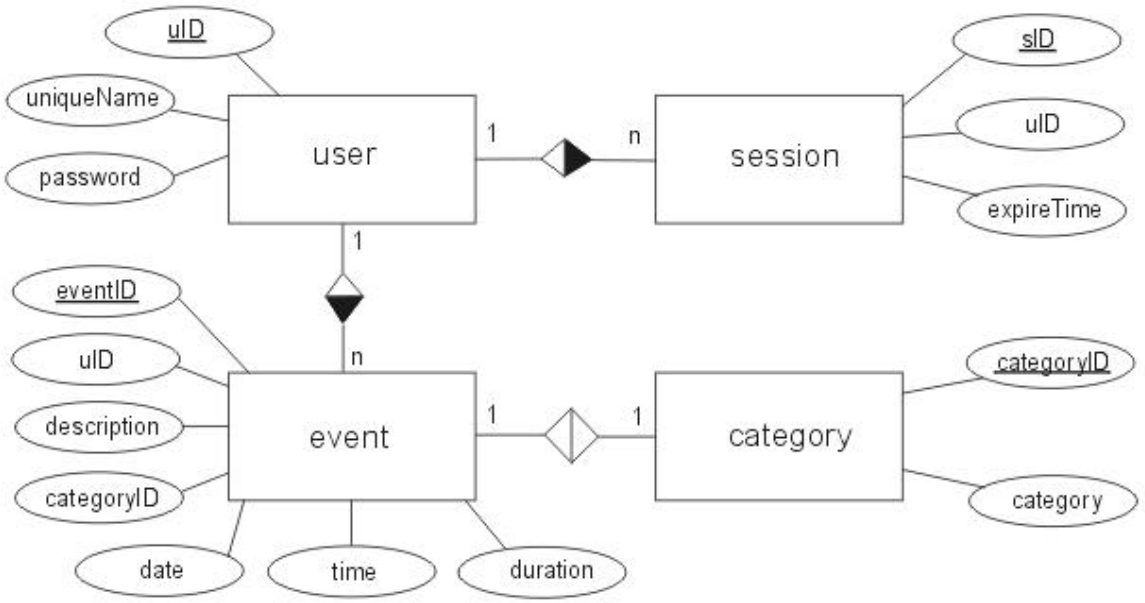
6. Database conception

6.1 „CalendarStore“

Description: As specified in ICalendar [1] an appointment calendar should consists of the following components: `TO-DO, EVENT, JOURNAL`. For this work this means that there must be at least two categories (`TO-DO, EVENT`) of appintments with in each case a pertinent description of it (`JOURNAL`). This methodology causes implicitly two tables in the databases (**Category** and **Event**)!

The tables (**User** and **Session**), which are needed for the session management, come still in addition to the first two. Such a methodology is mandatory, due to the fact that Cookies on mobile devices cannot be stored at present.

See in addition also the database conception:
http://swt.wi-inf.uni-essen.de/~tjungman/datenbank.html
from Thomas Jungmann!



ER-Modell of the database „CalendarStore“

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

7. Screenshots of the prototype

7.1 Login-Menu

The login-page ("index.wml") is the entrance page and the only static page in the whole system. Exactly, the login-page is a WML-form, which expects the user name and the user password as input. The input data are passed with the “GET”-method in the QUERY_STRING [11] to the "login.rxx"-script. The script verifies the inputs with the entries in the user – table of the database "CalendarStore" and shows at success the Main-Menu with the user-specific view-options. If verifing should fail, then is an appropriate error message temporarly generated by the "login.rxx" script. Temporarly means that a timer is set to limit the displaying of a WML page temporally.

Login - Page

„Index.wml“

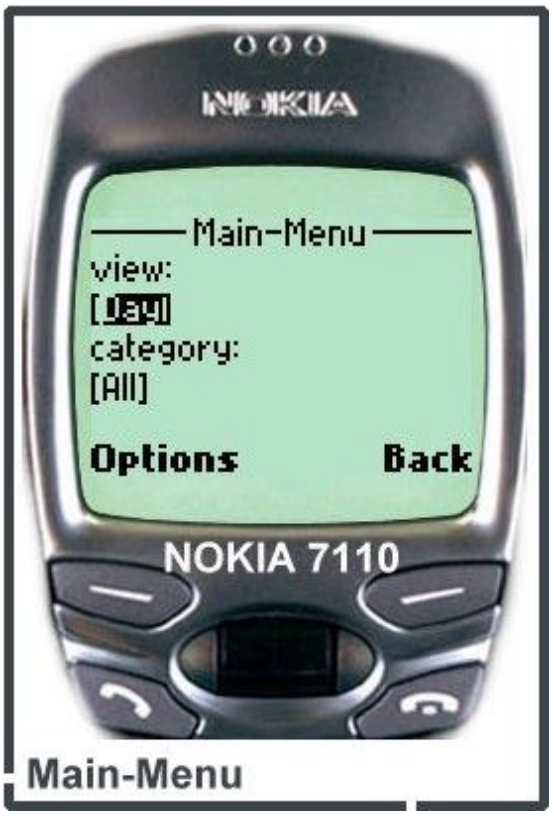
```
<?xml version="1.0"?>

<!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>
```

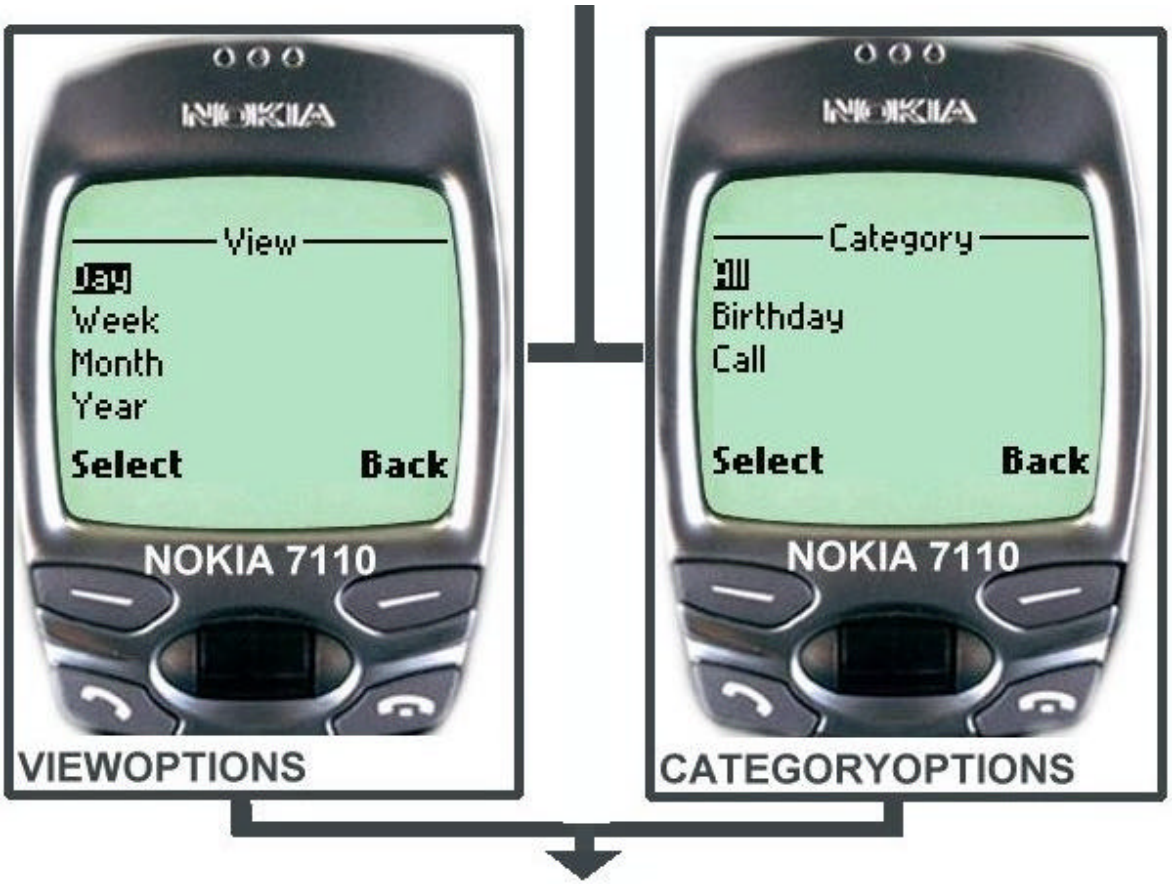
7.2 Main-Menu

After successful authentfication the Main-Menu is displayed and the users has the possibility to select the desired view type (**Year, Month, Week or Day** in the "**View**" field). Apart from the display option the user can select also the desired appointment category (**call, birthday, ALL**...) and only a certain type of categories will be displayed ("**Category**" field).



The input fields

The Main-Menu executes the "diary.rxx"- script and all further requests off now are executed only by this script. The **logout** function is supplied in the **option** menu.

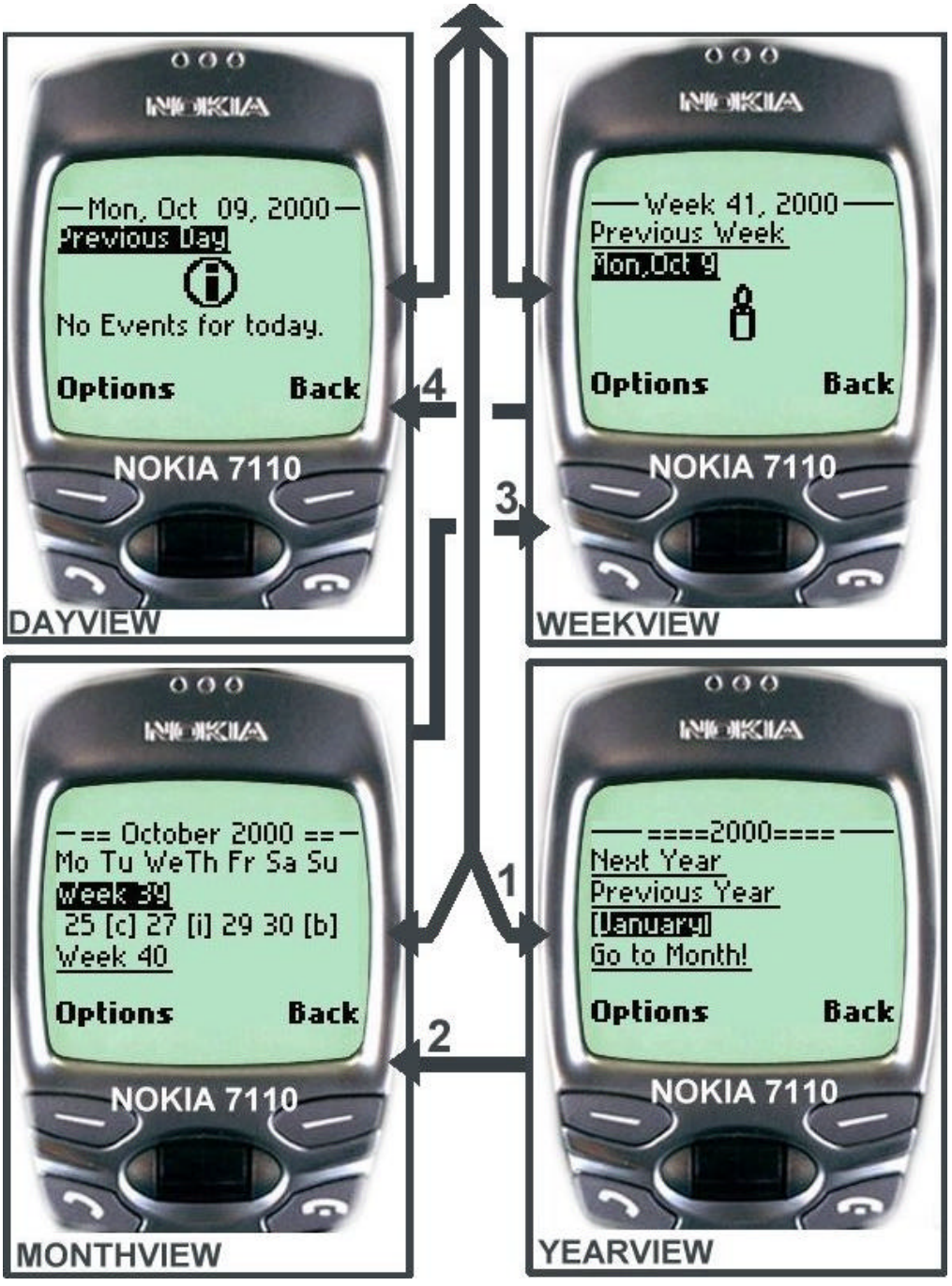


options of the input fields

7.3 View-Types

Basicly it is possible to select each view option from the main menu directly. The image shows a concrete example of navigation. The digits on the arrows refer to the navigation possibilities between the different view-types, i.e. if in the Main-Menu "Year-View" is once selected four steps (main-menu -> year -> month -> week -> day) has to be made up to the desired appointment entry.

It is not possible to change from the view-option " Year-View" to "Day-View" directly or from "Month-View" one level up to "Year-View", etc....



View-types and the navigation possibilities

8. Future Work

The system still offers no memory function by SMS or something similar. A memory function would complete the system as a commercial product and make it more competitive on the market.

The data communication of the server to the Client could be made more secure with WTLS[5] and protect by the way the system and the data users against attacks from the outside.

[5]The WTLS Protocol – Wireless Transport Layer Security based on TLS 1.0

9. Search results

References

[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**" - June 1999 - T.Berners-Lee, P. Leach, L. Masinter, H. Frystyk, J. Mogul, J. Gettys - 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 / Literature

- 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-database

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

- Server

Recent 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
Introduction to WML - URL: http://www.boku.ac.at/htmleinf/

- Available services

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