Organizational management system in an heterogeneous environment - A WWW case study

A. Silva, J. Borbinha, J. Delgado
Telematics Systems and Services Group
INESC - Instituto Engenharia de Sistemas e Computadores
Rua Alves Redol, 9, 1000 Lisboa, Portugal
Alberto.Silva@inesc.pt, Jose.Borbinha@inesc.pt, Jose.Delgado@inesc.pt

Abstract

W3 (World Wide Web) technology is being used over the world by a growth number of users. However that technology is mainly used to navigate or to retrieve hypertext information over the Internet. This paper describes some experiences currently tried at INESC, a Portuguese research institute. W3 and data bases technologies were used to develop, in a novel way, traditional information systems, that could be executed and accessed in a client-server model from a great variety of platforms and from any computer connected to the Internet.

Keywords

WWW, client-server computation, information systems, CGI gateways, databases, Internet

1 INTRODUCTION

Nowadays the spectacular Internet growth over the world is intrinsically due to the characteristics of the W3 model and its technology. the easy of use of its hypermedia navigation model, associated with its language (HyperText Markup Language) and protocol (HyperText Transfer Protocol), the availability of high quality W3 clients, like Mosaic or Netscape Navigator, hosted in a broad range platforms and environments, were decisive to the proliferation and the success of W3 and therefore to the Internet growth.

At INESC, a research institute in the areas of information technology, the Telematics Systems and Services Group has started in 1994 a few projects in the W3 area. One of those

projects is the InInWeb (INESC In The Web), an administrative and general information system presented in this paper.

It is not the objective of this paper to explain what is the W3 [Berners-Lee 94], what are relational data bases [Codd 70], SQL [CODASYL 78], or even analysis and design methodologies [Rumbaugh 91]. These technologies were used direct or indirectly in this project but are conveniently described in other documentation.

The document is organized in the following way:

- Section 2 shows the computational model subjacent to the W3 architecture;
- Section 3 presents the organization INESC, the services developed and some particular characteristics of the system;
- Section 4 presents the GENESIS library, a C++ library developed to help the developement of W3 services in an object oriented manner (and which is the base of the InInWeb sistem);
- Section 5 depictes the service developed and the perspective of integration with other similar administrative services;
- Section 6 points briefly related or similar projects that are being developed around the world;
- Finally, section 7 concludes the work done and makes some considerations to the future.

2 THE W3 COMPUTATIONAL MODEL

W3 follows basically the client-server model that uses a simple stateless protocol (HTTP) over the current TCP/IP stack. This protocol is very adequate for the retrieval of hypermedia information but presents some drawbacks when used to simulate interactive connections, or when the clients need to handle some event notification occurred at the server side.

In general, the W3 clients (also called “browsers”) have GUI capabilities, are multi-platform (X-Windows, MS-Windows, Macintosh, etc.), are multi-protocol (HTTP, FTP, GOPHER, NEWS, file, etc.), and parse and show conveniently HTML information. Recent clients have additional capabilities, for instance: built-in visualisation of different image formats (e.g. GIF, BMP), capability to invoke external multimedia players, availability of a CCI (Common Client Interface), or even secure transactions. Mosaic, from NCSA, was a great contribution to the W3 wave and nowadays the majority of the relevant clients are Mosaic-like.

From the other extreme there are the W3 servers (also called “HTTPd” or “HTTP daemons”). HTTP servers are in general multi-thread (it means they accept concurrently multiple requests), have configuration, logging and protection/authorization files, and support the CGI (Common Gateway Interface) protocol.

Servers answer client requests by returning HTML pages retrieved from the file system or by forwarding the data received from the CGI.

The CGI provides a mechanism to extend the basic functionality of an HTTP server by connecting it to other specific processes, called “gateways”, invoked at run time. It is important

to note that those gateways are completely independent processes that are launched by the HTTP server process. The way the data is communicated between the server and the gateway is conditioned by the CGI specification.

Figure 1 illustrates the general computation model above described.

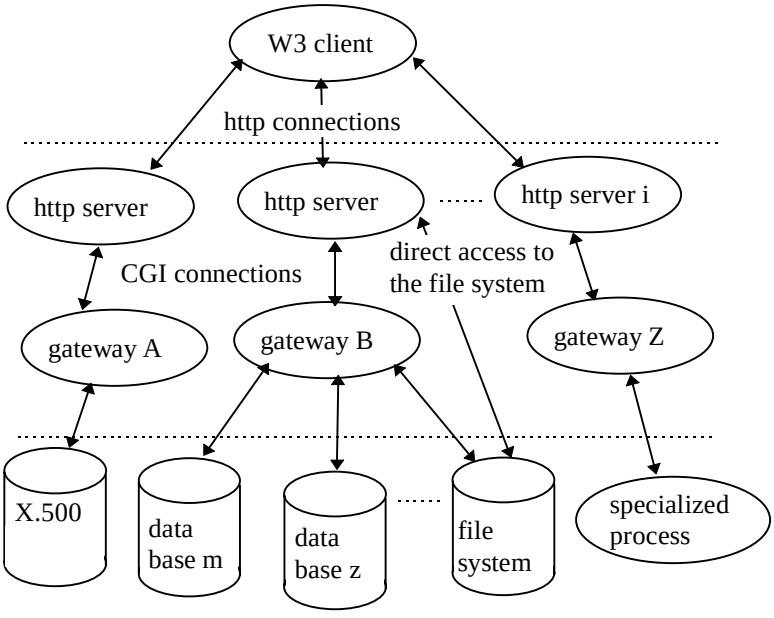

Figure 1 The W3 computational model.

Gateways are specific applications that can interact with several data repositories (e.g. relational data bases systems, file systems, X.500 distributed directories, etc.) or even with other specialised processes. However all the CGI gateways are "short life" processes, this means that they are not interactive processes. Any CGI gateway presents the following basic functionality:

1. It is created by the HTTP server;
2. It receives data via the CGI channel, by its standard input and by environment variables;
3. It parses the received data and checks its validity;
4. It executes some queries or calculations;
5. It produces some results (usually information in the HTML format) that will be read by the HTTP server which will redirect it to the client;
6. It ends itself.

3 THE INESC CASE STUDY

3.1 The INESC organization

INESC is a relatively complex and particular organization of the Portuguese academic and entrepreneurial society with some similar aspects to a services' organization. However, INESC

is a not-for-profit R&D institute, having public universities and telecommunication operators as associates. It is present in such geographical areas as Lisbon, Porto, Aveiro, Coimbra and Braga.

The organizational units at INESC are groups, centers and departments. Groups and centers are the operational units, while the departments are the supporting ones. Groups develop, as their main goal, R&D projects, while the main function of centers is to execute technology transfer. Groups and centers are distributed according strategy vectors. INESC has also developed, indirectly through its participated organizations, other complementary activities, such as professional training and company incubation.

The information system of INESC is very complex. It has very heterogeneous environments, either at the software or hardware level. It has machines from different manufacturers with different operating systems. Happily, the majority of its machines are connected to Internet (TCP/IP) and, consequently, they can share information and other resources using the W3 technology.

For a deep analysis of the INESC organization the reader can consult the specific documentation provided by its Communication Department [INESC 94].

3.2 The "INESC In the Web" service

The "INESC In the Web" ("InInWeb") service is an information and administrative support service to be used by people from and outside the organization. The InInWeb service has currently the following functionalities:

- A generic information service about INESC, that gives access to a well structured set of public information about INESC in an easy way (from or outside the organization), its collaborators, its functional units (groups, centers and departments) or its activities. This service was based in the concept of "Organizational Folder" (OF) that was firstly implemented in a paper support, where all the relevant entities had textual or image information, such as historical view, curriculum, objectives and activities.
- An administrative service called "Timecards", which is a service to manage and control the labour distribution, by each person, for activities and for work units. Because it handles information about people, work units and groups/centers, it makes sense to integrate it with the previous one.
- An administrative service called "Project Management", which intends to give support to the management and control of the projects developed by different groups. This service handles basic information, such as European programs and projects, consortiua, partners, etc., but also handles other resources such as labour, timecards, and financial (budgets and cost statements) information. One special emphasis was given to the management of European projects (such as Esprit or Race projects) because they are an important area of research and labour at INESC. Because this service also handles information about people, work units and groups/centers, it also makes sense to integrate it with the previous two mentioned.

The home page of the InInWeb is ilustrated in the figure 2.

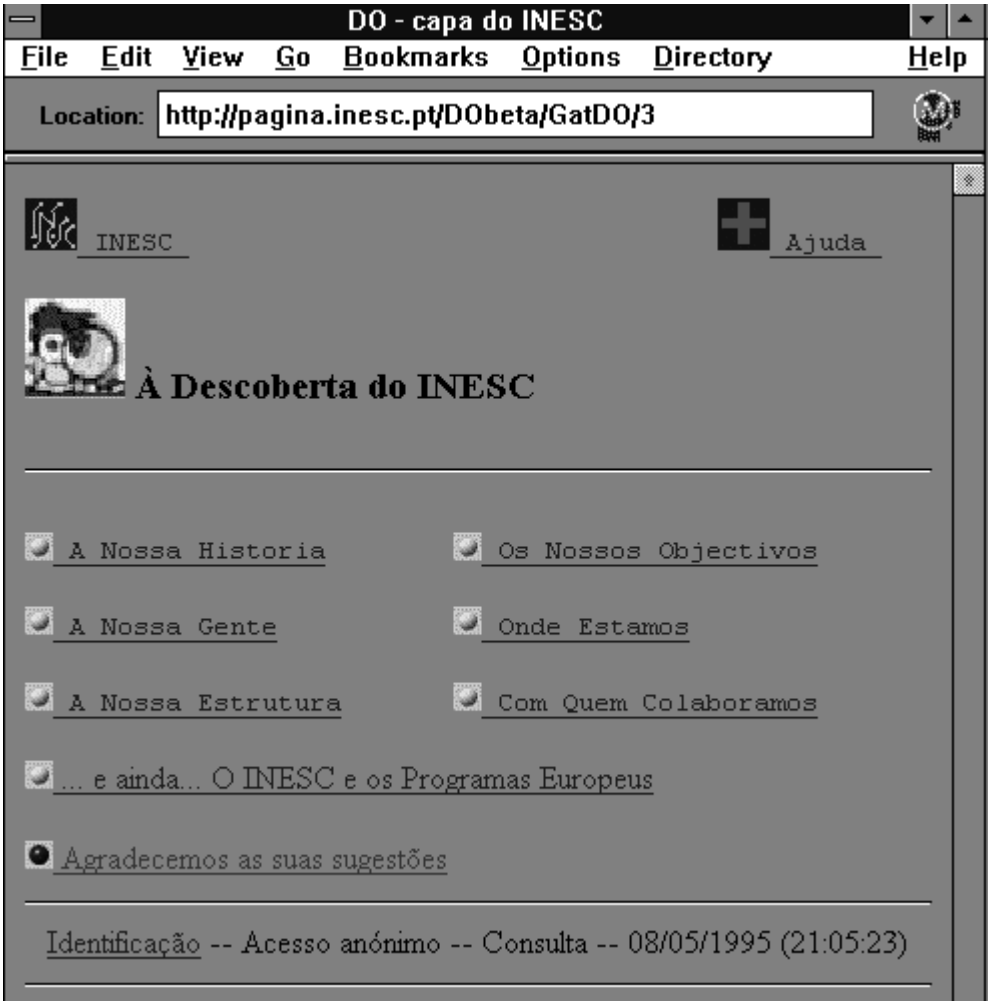

Figure 2 The InInWeb service home page.

The information handled by the InInWeb service is stored in a file system and in a relational data base system (SQL Server from Sybase [Sybase 94]). The information saved in the file system is mainly text and image information, e.g. histories, objectives, photos or other pictures from people. All the other information is organized in a relational way.

4 THE GENESIS LIBRARY

4.1 Introduction

GENESIS is a C++ library developed to help the authoring of W3 services. It brings the details of the construction of W3 gateways and HTML pages to a higher level (C++ classes), so that the programmer doesn't have to be a "W3 expert".

Distributed services (gateways on different machines) using heterogeneous repositories of information (such as Relational Data Bases and HTML or even non-HTML files) are supported.

Using GENESIS, a significant number of details in the construction of W3 gateways and HTML pages are hidden from the author, who can concentrate in the authoring work.

The actual version of the GENESIS library intends to be used directly by programmers, but future versions will be targeted to be integrated in authoring environments.

4.2 The Basic GENESIS Components

The current version of GENESIS library consists of a group of C++ classes that can be inherited to construct W3 services. Each class supports a concept, offering a group of basic functions related to it.

A service that intends to use GENESIS must be conceived from two perspectives:

- Actions: a service is structured as a web of actions (or sub-services);
- State: the state of the service is represented by the state of its resources.

A request to the service is an action to change and/or access resources.

Actions are performed by a special kind of objects, named agents. The main function of a gateway receives a request, parses the CGI parameters, discovers the classe of the requested agent and creates it. The agent gets the remaining parameters from the CGI and performs the request accessing to the resources, which are encapsulated in a special kind of classes named entities. In the end it is generated an HTML page as answer.

In a more detailed way GENESIS offers the following basic concepts, described below.

4.2.1 Gateways

A service is a web accessible through one or more gateways. A gateway gives access to a group of actions, which are performed by agents.

Gateways can be located in different machines, but this only has to be specified once by the programmer, in a configuration phase. Apart from that, the author can program all the service as if were in the same homogeneous environment, using logical references (the generation of the real gateway addresses will be automatic and transparent to the programmer).

The passing of parameters from one access to the next one (such as the user identification, for example), is completely managed by the gateways, so the programmer only has to deal with objects (e.g., the parameters passed from one action to the next one are, at the programming level, objects).

4.2.2 Agents

An agent is a specialized class, responsible for a specific part of the service (a specific group of actions). An agent can be accessible from one or more gateways, which can be in different machines. This is possible because agents represent parts of the service (functions), and are not directly related with the state of the service. An agent has two basic kinds of methods:

- methods to help other agents in the processing of their requests.
- methods to answer user's requests;

The first kind of methods represent the contribution of the agent for the service (the requests it knows how to answer), while the second ones are the answers to those functions.

To answer an user's request, an agent performs three steps:

- interprets and validates the request, checking the level of the user and what kind of answer he/she will have access;
- processes the request (changing the data repositories, if it is the case);
- builds the HTML answers page.

In the third step it can be requested the help of other agents, in order to get information to build anchors to give access to them.

4.2.3 Pages

The answer to an user's request is always a HTML page. Each agent has a group of pages to answer to all of the requests it can perform. Pages have a structure with three parts: a header, a body and a footer. Each of these three parts consists of a list of elements.

Classes of pages can share the same header and/or footer in order to uniform all the service or only some parts of it (for example, all the pages of the service or of a specific agent can have the same header and footer).

4.2.4 Elements

Pages are composed of lists of elements. There are two basic kinds of elements, HTML elements and high level elements.

HTML elements are classes that represent all the HTML types. If the programmer uses these classes he/she doesn't have to worry about the HTML details.

On the top of the basic HTML elements were also built a group of more complex elements. These elements were found to be very common in the applications where GENESIS has been used, and they include examples such as tables and composed forms (forms mixing radio buttons and boxes). Elements are very useful because they permit to manipulate HTML and complex HTML structures at a C++ level.

4.2.5 Entities

Entities are abstractions on the top of the data repositories, offering an object oriented layer of them. Agents can change the state of the entities, which means they can change the state of the service. Pages can read the state of the entities in order to present that information to the user.

Each entity class has two kinds of methods: methods independent of the repositories and methods dependent of them. Methods dependent on the repositories have to be all written by the programmer, but an entity class can use more than one repository (a data base and a file system, for example).

4.2.6 Users and Security

An user is a special kind of entity. Users are identified by a name and a password. Users can have capabilities, which can be checked by agents, for requests processing, or by pages, for answers to requests.

The Genesis library offers a prototype of an user's class as also an authentication agent, which offers a group of pages to support the process of user authentication. Once an user is identified, an encrypted timestamp key is passed from page to page, simulating a connection oriented session.

The CGI mechanism recognizes the user entity, giving a special support to this feature (the author only has to program the repository dependencies for the user element).

5 THE ININWEB SERVICE ARCHITECTURE

5.1 The InInWeb computational model

Currently the InInWeb service corresponds to just one CGI gateway that handles all the requests about the three integrated functionalities described in section 3. The figure below depictes its computational model.

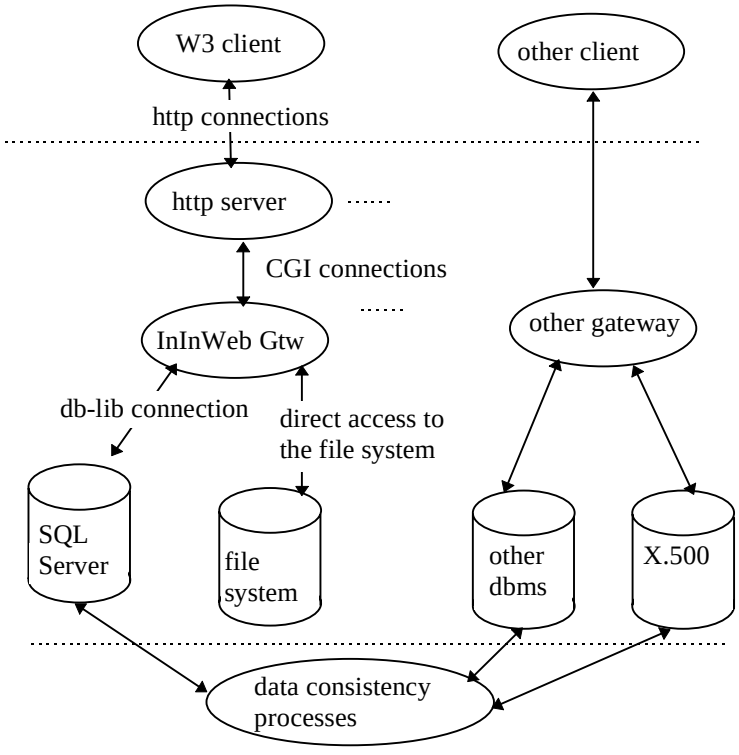

Figure 2 The InInWeb service computational model.

All the information of the service is stored in a data base of the SQL Server, from Sybase, and in a hierarchical structure of the file system. The data and control communication between the gateway and the SQL Server is supported over a DB-Library connection. That kind of connection was chosen because both processes are running in the same computer and it was the easiest and fastest connection found. However, future experiences will take into account the ODBC connection standard.

The previous model raises some questions that were taken during the analysis phase. One of them was how to maintain the consistency of the information managed in the InInWeb data base with equivalent information (such as the telephone or room number) kept by other services in different computers and different data repositories (such as X.500 or other DBMS). The first idea was to integrate a transactional capability into the InInWeb gateway. However this solution became impractical with the increasing number of potential interdependencies between services, and the need for efficient and simple CGI gateways. So, it was decided that the data consistency problem would not be solved at the gateway level, and that it will be handled by another independent and specialized processes.

Other important question was about the concept of a cooperative gateways pool. If the InInWeb gateway presents three integrated functionalities would it be divided in three cooperative gateways? One specialized on the Organizational Folder service, other in the Timecards and the third on the Project Management service? This solution apparently has the advantage to be lighter (i.e. each separated process seems to be lighter and more efficient than the sum of all of them), but presents the drawback of the mutual interface knowledge between

all the cooperative gateways. This question needs to be better analysed and detailed in future work.

5.2 Some architectural aspects

This sub-section presents some important or interesting aspects that were realized during the InInWeb service realization.

5.2.1 HTTP - A stateless protocol

As it is mentioned at section 2, the HTTP is a stateless protocol very adequate to the retrieval of hypermedia information. However the class of applications proposed initially is mainly interactive. So, the initial problem was: how to establish and to maintain an interactive session over a stateless protocol? This question can be asked from another, more technical, point of view: how can the gateway know what will be the HTML document to be generated?

The solution is the use of state variables, this means variables that are created and passed in the URL (Uniform Resource Locator), in the form name/value pair, whose values indicate the actual state. So, the gateway behaves itself as a state transition machine. Depending on the current state (passed on the current URL), the gateway analyse the other parameters and its standard input and based on that it will generate a HTML page whose links will correspond to URLs with future state values.

This mechanism of passing parameters in URLsis used in several situations, whenever necessary, such is the case of the authentication information. However all the parameters are conveniently coded with a way that nobody, even who sees the URL, can understand its meaning. The next figure illustrates an URL with encoded parameters.

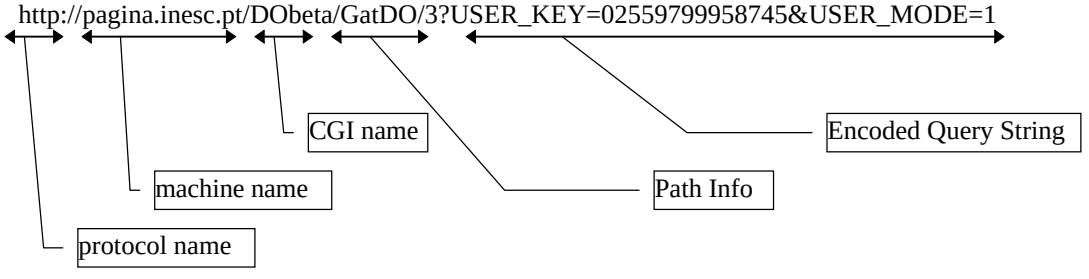

Figure 4 Example of an URL with some encoded parameters.

5.2.1 Security considerations

Due to the InInWeb service characteristics, some information is public (mainly the institutional INESC's information) but other (the administrative information such as some generic reports, or private information about people, units, or work units) must be protected and have access

conditioned. Consequently it was conceived a generic mechanism able to handle the user level in a protected way.

With the INESC reality in attention it was defined the following users levels:

- *SuperUser*, corresponds to the service manager, the user level that owns all the potential capabilities.
- *Director*, corresponds to a member of the board of directors.
- *Administrative*, corresponds to the collaborator responsible for doing the administrative tasks.
- *Unit Leader*, group, center or department leader.
- *Unit Secretary*, group, center or department secretary.
- *Work Unit Leader*, corresponds to the work units leader.
- *Collaborator*, corresponds to a regular collaborator.
- *Anonymous User*, corresponds to a non identified user.

These levels are not hierarchically organized in any capability order. Instead they would be better understood as several levels to control access, in either consulting or change mode, to different information classes depending on the person and on its own user level.

Any access to the InInWeb service will start, by default, in the *Anonymous User* level. Therefore the user will be able to only consult the institutional, or public, information. To change to an authenticated level and so to have access to different service capabilities, the user must select the Identification link that is present at all services pages' footer. The next figure shows the generic identification page.

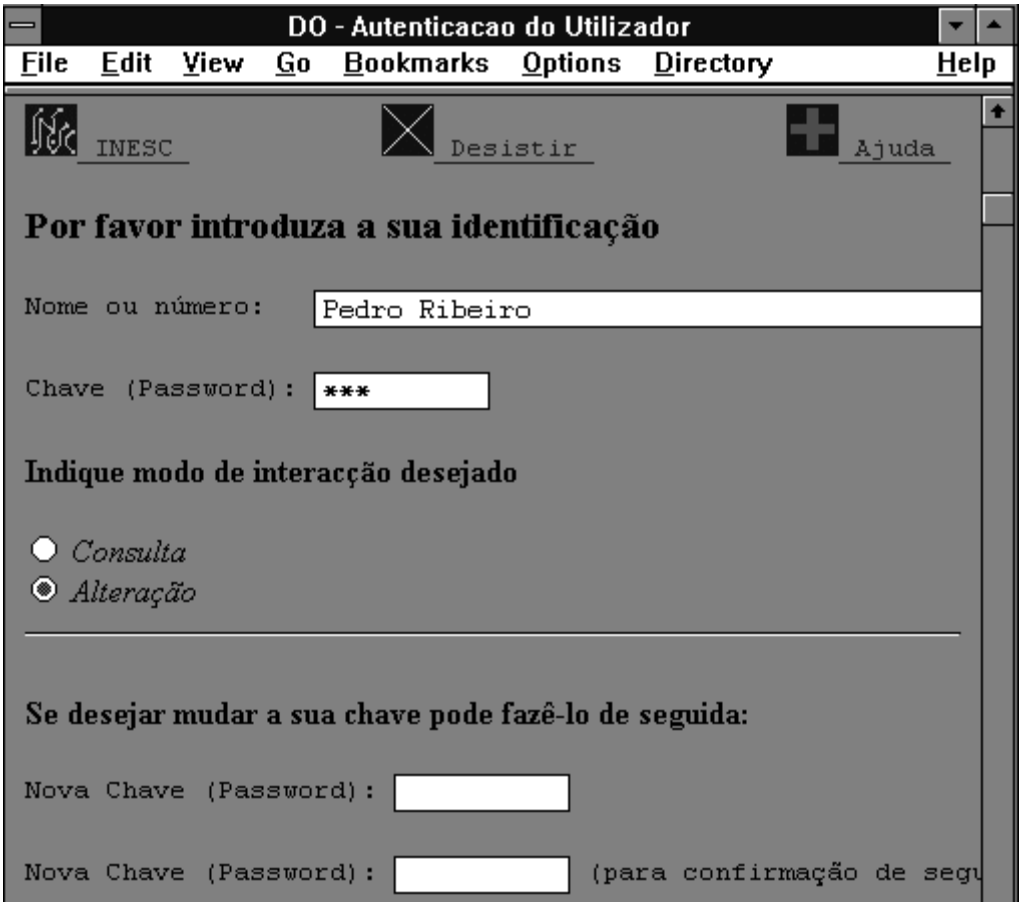

Figure 5 The generic identification HTML page.

After the identification link selection, the user must introduce his name, his password, and the desired operation mode (consulting or change). In this page he also could change his password to a newer one.

All the information concerning the authentication mechanisms: user, user levels and passwords are stored in the SQL Server.

In the meantime, how can we ensure that in successive accesses to the InInWeb service the user remains properly identified and accesses only the information he should? One solution is to request the user to systematically supply the password in each access to the server. This would be a very cumbersome solution. Another solution is to pass the password, eventually encrypted, in the URL as a name/value pair. This is a more attractive on the point of view of the user but is not as safe, since any user that somehow got access to the URL of these documents could subsequently access them directly.

Our proposal is based on the latter solution, but introduces a new variable in the model: time. When the URL to access a given restricted document is created, a password (pw') is generated as a result of an encrypting function (cript), whose parameters are the user's original password (pass) and a timestamp (ts). Therefore, pw’ = crip(pw, ts).

Whenever a request for a new document reaches the gateway, the timestamp stored in that file is checked to ascertain whether the time during which the password would be valid has already expired (5 minutes, for example). The re-encrypted password is then compared with another one generated locally from the password and the timestamp stored in the password file and, if valid, a new timestamp is obtained and a new password is coded. This password is then used to travel across the network. This process is repeated indefinitely.

The effectiveness of this scheme is guaranteed by:

- The user's password is not known;
- Even if someone has access to the password file (that is stored at the SQL Server), will only find encrypted passwords (with UNIX's crypt());
- If someone has access to the encrypting algorithm with the timestamp, won't be able to reproduce the URLs because he doesn't know which is the timestamp stored in the password file, even if he knows which is the encrypted password;
- If someone has access to a re-encrypted password, this one will only be valid until the authorized user makes another access, or at most until a given period (say, 5 minutes) is passed. This is due to the update of the timestamp in each access and to the time limit of its validity.

5.2.3 Integration services considerations

In the INESC context it will be developed other administratives services, such as “Human Resources Management” or “Library Management”. The implementation of these services will be based on the W3 technology. Therefore one relevant aspect to be handle is how to design and how to build a good integration of the services.

The integration of the various services (gateways) is based on their mutual knowledge, that is, on the knowledge of the URL's each one knows how to deal with. It should be noted that each service must know the address and interfaces that the other services export. But then what happens if one of the gateway is moved to another computer? Or if the interface exported by a service is changed? These aspects are relatively important in the development of distributed applications.

There are several technological solutions (DCE [DCE 92], CORBA [CORBA 91], COM [Udell 94]), each with its own specificities. The solution we used, W3, presents some of the problems referred above and which other technologies try to solve, namely the problem of name identification in distributed environments. Ideally, documents should be accessible independently of their physical location. This requirement implies a name server, which is still a research issue in the W3 context.

6 RELATED WORK

The W3 services construction, i.e. the development of CGI gateways that could dynamically generate HTML pages based on heterogeneous data repositories, is currently one important research and industrial area of interest. There are around the world others projects more or less similar to our, which are relevant to be referred:

- OMNIWeb [Matheus 95], from the National Space Science Data Center (NSSDC), USA. OMNIWeb provides access to a wide variety of data from NASA missions, allowing users to produce plots and retrieve data.
- Hyper-G [Andrews 95], from the Institute for Information Processing and Computer Supported New Media (IICM), USA. Hyper-G is a large-scale, multi-protocol, distributed, hypermedia information system that uses an object oriented database layer to provide information structuring and link maintenance facilities.
- OMNIS-WWW server [Clausnitzer 95], from the Technical University of Munich, Germany. OMNIS is a multimedia information retrieval system for the administration of documents in libraries and offices, and OMNIS-WWW server is a gateway between the OMNIS server and any W3 client.
- O2Web [O2Web 95], from the O2 Technology - SA, France. O2Web is said to integrate all the benefits of database technology with a complete set of tools to develop and deploy a W3 server based on the proprietary O2 system.

7 CONCLUSIONS AND FUTURE WORK

This paper has tackled the problem of management of institutional and administrative information in medium to large organizations, in which the information is distributed in various formats and data repositories. An adequate solution, which uses different technologies of informatics and telecommunications, such as W3 and relational databases, has been presented. The feasibility of the integration of different services and applications, namely the "Organization Folder", the "Timecards" and the "Project Management" executed in heterogeneous and distributed environments, supported by the technologies referred to above, has been demonstrated.

The most relevant advantages of the services developed and the technology used are:

- Computation based on the client-server model over an heterogeneous and distributed environment.
- Information locality transparent to the final user. Some information is kept in data bases in one, or more, machines. Other part of the information (mainly text or image information) could be kept locally at the chosen machine of each entity.
- Attractive user access and interface. Possibility of using graphical interfaces (GUI) with all of their inherent characteristics.

- Multi-platform clients, which means that the services developed could be accessed and executed from multiple environments, namely MS-Windows, Macintosh, X-Windows or VT-terminals.
- Possibility to mix traditional information with multimedia such as text, images, graphics, sound and video.
- Possibility to access different and heterogeneous data repositories.
- Guarantee of privacy and security in the information access and its management.

Although some limitations of the current W3 technology became apparent. There are important restrictions in the development of services whose aim is to be more than simple hypertext search and retrieval systems. The main limitations for our purposes have been the following:

- The W3 tools (clients, servers, editors, application development) are still in an initial stage of development (and not a mature technology)
- The hypertext paradigm underlying the W3 is not always the most adequate, specially in the tasks to manage database information (creation, modification, deletion)
- Definition of forms with several management, control and visualization limitations
- The HTTP protocol does not support secure (privacy and authenticated) transactions, namely:
  - the confidentiality of the information in the transaction is not guaranteed;
  - there is no guarantee of the identification of the entities involved in the transaction.

  The evolution of the protocol to support security is now under discussion [S-HTTP94].
- HTTP is a stateless protocol not very adequate to the traditional informatic services.

The work described in this paper, although interesting to the operation of the organization where it was developed, was mainly a case study to try out the emerging W3 technology. Future developments include:

- New administrative services.
- Continue with the design and implementation of new GENESIS library functionalities.
- Authentication and security mechanisms to be analyzed and thought out at the level of the definition of user profiles and concession of associated privileges, in a coherent and normalized way for an increasing number of services.
- Start the design of an authoring environment to help the W3 services development.
- Follow W3 and telematics technology tendencies, namely the Http-New Generation protocol and its specifications in the field of security, distribution and client-server interaction.

8 REFERENCES

Andrews, K. et al. (1995) Serving information to the Web with Hyper-G. *Computer Network and ISDN Systems*, vol.27 nº6, 819-26, April 1995.

Berners-Lee, T.; Cailliau, R.; Luotonen, A.; Nielsen, H.F. and Secret, A.. (1994) The World-Wide Web. Communications of the ACM., 76-82, August 1994.

Clausnitzer, A. et al.(1995) A WWW interface to the OMNIS/Myriad literature retrieval engine. *Computer Network and ISDN Systems*, vol.27 nº6, 1017-26, April 1995.

CODASYL (1978) Data Description Language Committee Report. Information Systems. vol. 3 nº4, 247-320.

Codd, E.F. (1970) A Relational Model for Large Shared Data Banks. *Communications of the ACM*, vol.13 nº6, 377-87, June 1970.

CORBA (1991) OMG. The Common Object Request Broker: Architecture and Specification.

DCE (1992) OSF. Introduction to DCE. Prentice-Hall.

INESC (1994) INESC's Institutional Information.

Matheus, G. J. and Towherd, S. (1995) NSSDC OMNIWeb: The first space physics WWW-based data browsing and retrieval system. *Computer Network and ISDN Systems*, vol.27 nº6, 801-8, April 1995.

O2Web (1995) O2 Technology. O2Web - Using object technology to build an Enterprise WWW Server. Commercial information. See also: HTTP://www.o2tech.fr.

Rumbaugh, J. et al. (1991) *Object-Oriented Modelling and Design*. Prentice Hall, New Jersey.

Sybase (1994) Sybase, Inc. Open Client DB-Library/C Reference Manual Release 10.0..

S-HTTP (1994) Draft of the Secure HTTP (S-HTTP) Protocol.
ftp://ftp.commerce.net/information/standards/drafts/sHTTP.txt

Udell, J. (1994) Componentware. *Byte*, May 1994.

9 BIOGRAPHY

Silva, A. is a researcher at INESC and an assistant lecturer at the IST/UTL (Technical University of Lisbon). His main interests are distributed computation, authoring tools, data bases and telematic applications development.

Borbinha, J. is a researcher at INESC and an assistant lecturer at the IST/UTL (Technical University of Lisbon). His main interests are in the telematic systems, in the organizational engineer and interfaces.

Delgado, J. is a researcher at INESC and a teacher at the IST/UTL (Technical University of Lisbon). His main interests are in the areas of domotic systems and telematic systems for the end users.