Network Protocols and Algorithms
ISSN 1943-3581
2016, Vol. 8, No. 3

A New Z39.50 Protocol Client to Search in Libraries and Improve Research Collaboration

Albert Rego, Laura García, Miguel Llopis, Jaime Lloret

Integrated Management Coastal Research Institute, Universidad Politecnica de Valencia,

C/ Paranimf nº 1, Grao de Gandía – Gandía, Valencia (Spain)

E-mail: alremae@teleco.upv.es, laugarg2@teleco.upv.es, the_cuc@hotmail.com, jlloret@dcom.upv.es

Received: September 3, 2016 Accepted: October 17, 2016 Published: October 30, 2016

DOI: 10.5296/npa.v8i3.10147 URL: http://dx.doi.org/10.5296/npa.v8i3.10147

Abstract

The increasing number of information available at libraries and the necessity to find a mechanism to look for information at several libraries at the same time promoted to the creation of the Z39.50 protocol. Since it has been launched, many Z39.50 clients have been developed, but most of them only have access to a reduced number of libraries catalogues and are focused on the point of view of the library users. We present a Z39.50 client that adds new features focused on the point of view of researchers and publishing houses. With our proposal the user can add the catalogue of any library whenever he wants in order to facilitate the process of looking for the desired information. Our Z39.50 client shows as well the situation of the closest libraries where the results of the information searched by the user is located. Finally, we have compared our Z39.50 client with the existing ones.

Keywords: Z39.50, Integrated Library System, Bibliographic Reference Software, MARC, OPAC.

1. Introduction

Fremont Rider in 1944 estimated that the growth of information stored in American university libraries is multiplied by two every 16 years [1]. He estimated that Yale University Library would reach 200,000,000 volumes in 2040. It currently has more than 15 million copies [2]. It is not known if it will achieve the figure Rider estimated, but it is a fact that the information is growing more rapidly mostly due to the contents generated on the Internet [3]. This growth of information has led to the coining of the term Big Data, which is used to describe big quantities of information and is the subject of recent studies in information management.

Due to the increased information, the need to classify data arises in order to access it more easily. The development of technologies allowed the creation of various computer systems for classification of information such as MARC (Machine-Readable Cataloging) [4] which created a common framework for cataloging documents in libraries. Another one that uses these technologies is the OPAC (Online Public Access Catalogue) system. It allows a database to catalog the contents of a library electronically.

The telematic cataloging of the contents of the libraries was a great step in helping the users to find the documents they needed. But those systems only allowed the access to information stored in each library separately. That created the necessity of a unified catalogue to permit the search of the desired information to be done in all libraries at the same time. To fill that need the Z39.50 protocol was created.

Z39.50 [5] standard (Information Retrieval (Z39.50), Application Service Definition and Protocol Specification ANSI / NISO Z39.50-2003) began to take shape in 1970 in the United States by the Library of Congress, the Computerized Library center Online (OCLC) and Information Network for Research Libraries. Its purpose was to share cataloging in a standardized way. Since then, several versions have been approved, the first in 1988. In 1992 the second version of the protocol (ISO 10162/10163), and in 1997, the third (ISO 23950), which implemented its development over TCP/IP, allowing access through the Internet.

Z39.50 is a communication protocol between a client and a server. Sessions inside one connection between both nodes are known as Z39.50-association or Z-association. These sessions are initiated by the client. Since Z-association is open, both server and client can start any operation defined in Z39.50 protocol. In the same way, Z-association can be closed by either client or server, or implicitly terminated by loss of connection.

The main goal of Z39.50 is to provide a standard to search information into an external database whatever its data organization. Thus, Z39.50 is widely used in some of the biggest libraries. This goal is achieved because the communication between the client and the server is standard and independent to the database. Once the server receives the message with the client's request, it can easily converse this standard request into a specifically database query.

The term database, as used in Z39.50 standard [5], refers to a collection of records. These records are simple collections of information. When a client request some information located in the server's database, it obtains not a row of the database, but a record with the information.

Consequently, server's database structure is transparent to the client.

Meanwhile the Z-association is alive, the server never saves the client state. The communication between both nodes is simple and stateless. The server or the client can start any operation described in the standard, some of which can be composed by two or three services, regardless of the previous operations.

Obviously, some actions like close a non-existing Z-association or delete the result of a previous search operation without search are actions that the server cannot do. In this cases, the server response shows that an error has happened, by using one or many fields of the response message. This does not imply that the server remembers the state of the client. The protocol is stateless.

Many libraries worldwide use this protocol to allow its users the access to all the information they have available. In Spain several libraries use it to catalogue their material. One of them is the National Library of Spain [6]. It has over 30 million documents and was created in 1712. On the Internet, several lists of libraries that support Z39.50 protocol can be found. Some of them are lists of Spanish libraries and organizations, as the ones in [7] and [8]. Others are from other countries such as the list presented by the American Library of the Congress [9], where they provide access to libraries all over the world.

With the Z39.50 protocol the records of multiple libraries can be accessed, but with the applications available today the information of servers from other libraries that are not entered the application log cannot be added. With the new Z39.50 application that is presented in this paper, we intend to improve the search system of the Z39.50 protocol allowing adding new servers where searches of the desired information can be done. Also, this application can modify and remove the desired server. A new location functionality is added as well, that allows to detect the location of the libraries where the user can find the sought-after documents as well as their closeness to the user's location. At the same time the application shows on a map the location where the intended library is. The obtained results can be printed or saved on a file.

The rest of the article is organized as follows. Section 2 describes the related works. Section 3 describes the Z39.50 protocol operation and messages. Section 4 includes the Z39.50 protocol procedure, its algorithm and message flow. The developed Z39.50 client is shown in Section 5. Section 6 presents a comparative of the existing Z39.50 clients. Finally, Section 7 includes the conclusion and future work.

2. Related work

In this section, some of the studies that have been done on the use of the Z39.50 protocol since its launching are exposed.

In [10] M. R. Genesereth et al. present an information integration system that accesses a wide variety of databases such as Z39.50 and SQL databases, giving the user the perception that the system is centralized. They use different types of wrappers to access the information

of the different databases available and the information over the internet. It has several user web interfaces as well as a programmatic interface that allows to store the results on the computer of the end user.

In [11] R. R. Larson creates a method to build resource Discovery indexes using the Z39.50 protocol to avoid the creation of a new protocol and the interoperability problems associated with it. To do this, he utilizes the Explain facility and the Browse facility provided by the Z39.50 protocol. The Explain facility is used to obtain different types of information about the server such as the supported databases or error information. The Browse facility is used to access the listing of the information available at the database.

In [12] A. Corfield et al. present the JAFER toolkit. It is an API (Application Programming Interface) based on XML (eXtensible Markup Language) that employs the Z39.50 protocol. This API utilizes an abstraction layer that uses XML to simplify the complexity of the Z39.50 protocol. It has been developed using Java to allow the client to operate in multiple environments. To make the obtained results easier to read, the bean transforms the difficult XML result into an easier one such as the MODS schema.

In [13] D. Boberic and D. Surla propose an editor based on XML that permits searching the information in different databases. The application they have created use the Z39.50 protocol to access the desired databases. To make the application they have used the aforementioned JAFER toolkit and they have utilized Java to create the editor. This editor allows to use all the attributes available form bib-1 and is incorporated I the fourth version of the BISIS library information system.

Nowadays there are several open source Z39.50 clients to access the catalogues of a great number of libraries. A description of them is showed below.

BibData [14] is a Z39.50 software available both online and as an application. It belongs to the BidData.Com Company located in Ontario, Canada and founded in 2005. It's available for Windows XP, Vista, 7 and 8. It has several features such as MARC record rating and editor, to import and export to MARC and text and ISBN/Titles batch search.

Alejandría nBibliotecas [15] is an online client designed to help small libraries with the cataloging of their information. It supports OPAC and MARC as well as the Z39.50 protocol. They have over two hundred clients, many of them are libraries form universities and enterprises located in Spanish speaking countries. Their headquarters are located in Venezuela.

Evergreen [16] is an open software for libraries. The Evergreen Project started in 2006 in Georgia, USA, and is used in over 250 public libraries of the state of Georgia. It supports OPAC, MARC, OpenSearch, RDA, RSS, Z39.50 and many other standards ant it is available for Windows, Linux and Mac operative systems.

Greenstone [17] is a software that allows to organize the collection of a library. It was created by the New Zealand Digital Library project at the University of Waikato with the cooperation of the Human Info NGO and UNESCO. It allows the publishing of the

information on the internet and on DVD and USB drives.

JZKit3 [18] is an open software toolkit developed completely in Java. This toolkit is designed to help Z39.50 application developers. It is developed or Windows OS and supports MARC, XLS and Z39.50. It is a product of Knowledge Integration enterprise, founded in 1999 and their clients are mostly European and North American.

There are other Z39.50 open source clients with similar features as the aforementioned clients. Some of them are Koha [19], Mercury Z39.50 Client [20], Net-Z3950-Simple2ZOOM [21], OPACIAL [22], OPALS [23], Pazpar 2 [24], PMB [25], Potthakalaya [26], Senayan [27], SobekCM [28] and Virtual Library for Moodle [29]. Our proposal adds other functionalities such as adding library servers anytime the user wants and providing the location of the libraries with the result of the user's search. With this, our proposal becomes a more complete tool that improves the user experience with Z39.50 clients. This is a helpful tool for helping researchers and publishers locate where their work or products are, so they can use that information in their benefit. There are other papers that create a system for researchers coperation like the one in [30]. In it, Lloret J. Et al. propose a B2B (Bussiness to Bussiness) architecture and protocol that uses the ebXML protocol. Their proposal permits to establish a connection between research entities.

3. Z39.50 Protocol operation and Messages

This section describes the nine operations implemented in Z39.50 and the messages that are sent by the client or the server in these operations.

3.1 Initialization Operation

The Initialization operation is composed by one single service: Init Service. Using this service, the client can establish the Z-association with the server.

Init Service is a really simple service composed by only two messages. First, the client sends an Init Request message which contains, as Table1 shows, the options of the Z-association with other parameters. The server analyzes each one of the options proposed by the client and, if the proposed one is plausible, sends an Init Request with the parameter Result set to "accept" and the Z-association is established, else the server may change the values or may also respond negatively. If the values have been changed by the server and the client does not wish to accept them, it may terminate the Z-association, via the Close service described later. If the server has responded negatively, the client may attempt to initialize again.

Both messages, Init Request and Init Response are shown in Table 1 and Table 2. Table 3 shows the different options which server and client can negotiate. The paper is not focused on what these options individually do. However, they activate or not services or some options on those services, like segmentation.

Table 1. Init Request message structure.

Field	Description
Version	Version of the standard to use.
Id (optional)	Used by the server to authorize.
Options	Options that can be negotiated
Preferred-message-size	Preferred size of messages in bytes.
Exceptional-record-size	Preferred size of records during Present operation.
Implementation-id (optional)	Identifier for the client implementation.
Implementation-name (optional)	Descriptive name for the client implementation.
Implementation-version (optional)	Descriptive version for the client implementation.
User-information-field (optional)	May be used for additional no-standard information.
Other-information (optional)	May be used for additional no-standard information (Not recommended)
Reference-id	Used to link the request with response.

Table 2. Init Response message structure.

Field	Description
Version	Version of the standard to use.
Options	Parameters values during Z-association.
Preferred-message-size	Size of messages in bytes.
Exceptional-record-size	Size of records during Present operation.
Result	
Implementation-id (optional)	Identifier for the server implementation.
Implementation-name (optional)	Descriptive name for the server implementation.
Implementation-version (optional)	Descriptive version for the server implementation.
User-information-field (optional)	May be used for additional no-standard information.
Other-information (optional)	May be used for additional no-standard information (Not recommended)
Reference-id	Used to link the request with response.

Table 3. Options in Init Service.

Option Bit	Option
0	Search
1	Present
2	Delete Result Set
3	Resource Report
4	Trigger Resource Control
5	Resource Control
6	Access Control
7	Scan
8	Sort

9	Unused
10	Extended Services
11	Level 1 Segmentation
12	Level 2 Segmentation
13	Concurrent Operations
14	Named Result Sets
15	Encapsulation
16	resultCount parameter in Sort Response
17	Negotiation Model
18	Duplicate Detection
19	Query type 104
20	PeriodicQuery ES Correction
21	Use of string values for schema in compSpec

3.2 Search Operation

Search operation is probably the most important operation used by a client due to the fact that Z39.50 protocol is used by the clients to consult data. The search operation is composed by one single service: Search Service. Using this service, the client can establish and retrieve records of data by requesting that the server apply a query to a set of their databases.

When the service executes the query it creates a result set, which is the representation of the records identified by the query. The server keeps the result set in order to improve performance in case of the client requesting the same query.

The structure of the messages is described in Table 4 and Table 5.

Table 4. Search Request message structure.

Field	Description
Query-type	Version of the standard query to use.
Query	Query
Database-names	Databases where the query will be executed.
Result-set-name	Name that will be set to the result set.
Replace-indicator	Indicates if a result set with the same set will be override or not.
Small-set-element-set-names (optional)	Names for the sets if they are small-set.
Medium-set-element-set-names (optional)	Names for the sets if they are medium-set.
Preferred-record-syntax (optional)	Syntax that the client prefer for the records.
Small-set-upper-bound	Indicates the maximum size of a record to be considerate small.
Large-set-lower-bound	Indicates the minimum size of a record to be considerate large.
Medium-set-present-number	Maximum number of response records to be returned if the set is Medium-set.
Additional-search-information (optional)	May be used to indicate the preferred format or context of the information.
Other-information (optional)	May be used for additional no-standard information
Reference-id (optional)	Used to link the request with response.

Table 5. Search Response message structure.

Field	Description
Response-records	N response records.
Result-count	Number of database records identified by the result set.
Number-of-records-returned	Total number of records returned.
Next-result-set-position	Position of the next result set.
Search-status	Indicates success or failure.
Result-set-status	Indicates if the result set is partial.
Present-status	Indicates how partial the result set is.
Additional-search-information (optional)	May be used to indicate the preferred format or context of the information.
Other-information (optional)	May be used for additional no-standard information
Reference-id (optional)	Used to link the request with response.

3.3 Retrieval Operation

Retrieval operation allows the client to request response records corresponding to database records using Present Service. In addition, Segment Service provides to the server one way to send the records in several messages. This is only possible if segmentation is active and the result of Present Service does not fill within the Present response message.

In Table 6 and Table 7 the structure of messages that Present Service implements is shown. Then, Table 8 shows the fields of the Segment Service's message.

Table 6. Present Request message structure.

Field	Description
Number-of-records-requested	Number of records to retrieve.
Result-set-start-position	Which of the N result sets will be the first to retrieve.
Additional-ranges (optional)	Indicates advanced mode to select the result sets.
Result-set-id	Indicates the temporal name of the transient result set created during the Z-association.
Element-set-names (optional)	The client may indicate the composition of the result sets.
Preferred-record-syntax (optional)	The client may specify a preferred record syntax for retrieval record
Comp-spec (optional)	An alternative field to indicate the composition of the result sets.
Max-segment-count (optional)	Maximum number of segments.
Max-segment-size (optional)	Size of the largest segments.
Max-record-size (optional)	Size of the largest record.
Other-information (optional)	May be used for additional no-standard information
Reference-id (optional)	Used to link the request with response.

Table 7. Present Response message structure.

Field	Description
Response-records	Records to be send.
Number-of-records-returned	Number of database records in the response.
Next-result-set-position	Next result set not included in the response.
Present-status	Indicates how partial the result set is.
Other-information (optional)	May be used for additional no-standard information
Reference-id (optional)	Used to link the request with response.

Table 8. Segment Request message structure.

Field	Description
Segment-records	Segments to be send.
Number-of-records-returned	Number of database records in the response.
Present-status	Indicates how partial the result set is.
Other-information (optional)	May be used for additional no-standard information
Reference-id (optional)	Used to link the request with response.

3.4 Delete Operation

Delete Operation, composed by only one service (Delete Service), is used by the client in order to eliminate specified results sets, or all result sets created during the Z-association.

Although the results sets are deleted after the Z-association is closed, the client can delete them during the Z-association in order not to violate the result sets limit that the server might have. In order to delete the results sets, the client and the server sends messages structured like Table 9 and Table 10.

Table 9. Delete Request message structure.

Field	Description
Delete-function	Indicates if delete all sets or only a specified result sets.
Result-set-list (optional)	Indicates what specified result sets are going to be deleted.
Other-information (optional)	May be used for additional no-standard information
Reference-id (optional)	Used to link the request with response.

Table 10. Delete Response message structure.

Field	Description
Delete-operation-status	Indicates “success” or “list”.
Delete-list-status (optional)	List the failures in the delete operation.
Number-not deleted (optional)	Indicates how many result sets were not deleted
Bunk-statuses (optional)	Indicates the statuses of the sets that were not deleted.
Delete-msg (optional)	Contains an optional text message.
Other-information (optional)	May be used for additional no-standard information
Reference-id (optional)	Used to link the request with response.

3.5 Access-control Operation

The server can, after an initiating request is received, challenge a client using the Access-control service. The Access-control can be used in order to challenge a client during a specific operation or to the Z-association. The server sends an Access-control request. If the client response is acceptable, the operation can be continued or the Z-association can be established. However, if the client fails to respond correctly the server ends the operation or does not open the Z-association.

The available challenges for access-control are password challenges, algorithmic authentication and public key cryptosystems.

Table 11 and Table 12 show the messages that implement Access-control service. In Fig. 1 one possible use of Access-control is shown. In this case, the access control is executed immediately after the Init Request, so it affects to the entire Z-association. If the client fails, the Z-association will be closed.

Table 11. Access-control Request message structure.

Field	Description
Security-challenge	Definitions for format and content of the challenge.
Other-information (optional)	May be used for additional no-standard information
Reference-id (optional)	Used to link the request with response.

Table 12. Access-control Response message structure.

Field	Description
Security-challenge-response	Content of the challenge response.
Other-information (optional)	May be used for additional no-standard information
Reference-id (optional)	Used to link the request with response.

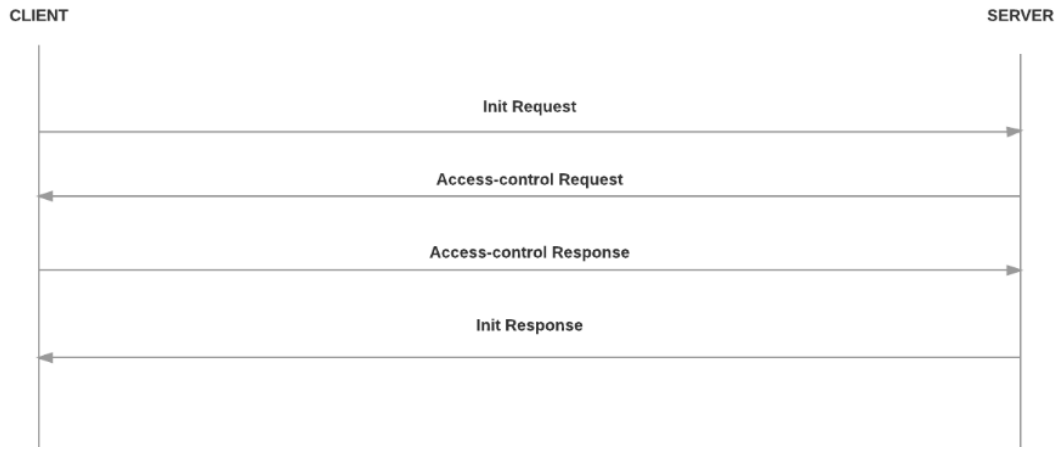

Figure. 1. Access-control example communication diagram.

3.6 Accounting/Resource Control Operation

Resource Control Operation is the mechanism that allows the server to manage the

resources consumed by the clients in order not to overpass the available resources. To achieve this, Resource Control Operation is composed by three services.

Resource-control service allows server to communicate with the client to notify that either actual or predicted resource consumption will exceed agreed upon limits, and the client has to accept to continue an operation.

For example, during a Search operation, this service can be used to inform the client about the status of a result set that is being generated and to indicate the progress of the operation.

Trigger-resource-control service permits the client to request the server to execute the Request-control service by sending a Request-control request. Resource-report service is a mechanism that allows the client to request a report from the server about a completed operation or about the Z-association. Table 13 to Table 17 shows all the different messages that implements Resource-control Operation.

Table 13. Resource-control Request message structure.

Field	Description
Resource-report (optional)	Indicates the current and estimated resource consumption.
Partial-results-available (optional)	As a part of a search operation, it indicates the status of the result set.
Suspended-flag (optional)	When the request pertains to an operation, this flag indicates if the operation has been suspended until the response arrives.
Response-required	The server indicates if a response is required.
Triggered-request-flag (optional)	When the request pertains to an operation, this flag indicates if the request resulted from a Trigger-resource-control request.
Other-information (optional)	May be used for additional no-standard information
Reference-id (optional)	Used to link the request with response.

Table 14. Resource-control Response message structure.

Field	Description
Continue-flag	When the request pertains to an operation, this option indicates if the server has to continue processing the operation.
Result-set-wanted (optional)	Allows the server to maintain partial result sets for Present operations.
Other-information (optional)	May be used for additional no-standard information
Reference-id (optional)	Used to link the request with response.

Table 15. Trigger-resource-control Request message structure.

Field	Description
Requested-action	Indicates if the client requests a resource-report request or a resource-control request.
Preferred-resource-report-format (optional)	The client may indicate a preferred resource report format
Result-set-wanted (optional)	Allows the server to maintain partial result sets for Present operations.
Other-information (optional)	May be used for additional no-standard information
Reference-id (optional)	Used to link the request with response.

Table 16. Resource-report Request message structure.

Field	Description
Preferred-Resource-report-format (optional)	The client may indicate a preferred resource report format
Op-id (optional)	It may refer to the operation which the client wants to be reported about.
Other-information (optional)	May be used for additional no-standard information
Reference-id (optional)	Used to link the request with response.

Table 17. Resource-report Response message structure.

Field	Description
Resource-report-status	Indicates if the resource report has been successful or the solution given is partial.
Resource-report (optional)	Indicates the current and estimated resource consumption.
Other-information (optional)	May be used for additional no-standard information
Reference-id (optional)	Used to link the request with response.

3.7 Sort Operation

Sort Operation is divided into two services: Sort Service, which allows a client to request the server to sort the result set obtained from a Search Operation, and Duplicate-detection Service, which permits the client to request the server to delete possible duplicated result sets.

Table 18 and Table 19 show Sort Service message structure. On the other hand, Table 20 and Table 21 show Duplicated-detection Service message structure. These last messages have fields that can be included more than once. These fields are marked in the tables with a “+” symbol.

Table 18. Sort Request message structure.

Field	Description
Input-result-sets	Name of the result set to be sorted.
Sorted-result-set	The name of the sorted set.
Sort-sequence	Elements that are to be used for sorting.
Other-information (optional)	May be used for additional no-standard information
Reference-id (optional)	Used to link the request with response.

Table 19. Sort Response message structure.

Field	Description
Sort-status	Indicates if the operation was performed successfully or not.
Result-set-status	If Sort-status is failure, it indicates the contents of the sorted set.
Diagnostics (optional)	If Sort-status is failure, it indicates one or more diagnostics records.
Result-count	Shows the size of the sorted result set.
Other-information (optional)	May be used for additional no-standard information
Reference-id (optional)	Used to link the request with response.

Table 20. Duplicate-detection Request message structure.

Field	Description
Input Result Set Id +	Id of the transient result sets to merge or delete.
Output Result Set Name	Name of the result set.
Applicable Portion of Record (optional)	Indicates what fields are used to find the matches.
Duplicate-detection Criterion (optional) +	Indicates what criterion is used to say two fields match.
Clustering (optional)	Allows clustering the result sets.
Retention Criterion +	Specifies criteria used to include or exclude records into equivalence class.
Sort Criterion (optional) +	Used to provide criteria for ordering records within an equivalence class.
Other-information (optional)	May be used for additional no-standard information
Reference-id (optional)	Used to link the request with response.

Table 21. Duplicate-detection Response message structure.

Field	Description
Status	Indicates if the operation has been successfully or not.
Result count (optional)	Size of the output result.
Diagnostic (optional) +	The server may include diagnostics in the response.
Other-information (optional)	May be used for additional no-standard information
Reference-id (optional)	Used to link the request with response.

3.8 Browse Operation

Browse Operation consists of a single service, Scan. This service is used to scan an ordered term list. The client specifies which term list wants to scan and the starting term.

The messages are structured like in Table 22 and Table 23.

Table 22. Scan Request message structure.

Field	Description
Database Names	Set of database to which the list pertains.
Term-list-and-start-point	Term list and start point to scan.
Step-size (optional)	Number of records to skip between two result records.
Number-of-entries	Proposed number of entries to list.
Position-in-response	First term of the response relative to start point.
Other-information (optional)	May be used for additional no-standard information
Reference-id (optional)	Used to link the request with response.

Table 23. Scan Response message structure

Field	Description
Step-size (optional)	Number of records skipped between two result records.
Number-of-entries	Actual number of listed entries.
Position-in-response (optional)	Actual first term of the response relative to start point.
Scan-status	Status of the scan (success, partial, etc.)
Entries (optional)	Entries listed.
Other-information (optional)	May be used for additional no-standard information
Reference-id (optional)	Used to link the request with response.

3.9 Termination Operation

Termination Operation consists of a single service, Close. Close service is provided to allow the client or the server to end a Z-association. When one of them sends the Close Request, it has to wait until a Close response arrives and consider the Z-association closed. The structure of the Close messages is described in Table 24 and Table 25.

Table 24. Close Request message structure.

Field	Description
Close-reason	Describes the reason of closing the Z-association.
Diagnostic-information (optional)	Server-only. Optional text message about diagnostic information.
Resource-report-format (optional)	Client-only- Request a report.
Resource-report (optional)	Server-only. May include a report.
Other-information (optional)	May be used for additional no-standard information
Reference-id (optional)	Used to link the request with response.

Table 25. Close Response message structure.

Field	Description
Close-reason	Describes the reason of closing the Z-association.
Diagnostic-information (optional)	Server-only. Optional text message about diagnostic information.
Resource-report (optional)	Server-only. The report requested by the client.
Other-information (optional)	May be used for additional no-standard information
Reference-id (optional)	Used to link the request with response.

4. Z39.50 protocol procedure

In this section, we are going to show an example of a common use of Z39.50 protocol.

The example purpose consists of a set of steps from establishing the Z-association to closing it. During this Z-association the client requests the server to search some information and then to present this information to the client. This use case is the most common use of protocol Z39.50, consult some information.

Fig. 2 shows the communication between client and server and the messages they send in this example. In addition, Fig. 3 shows the flowchart of the server for this example.

The messages shown in Fig. 2 are the described, for each service, in the previous section.

The server flowchart shown in Fig. 3 contains the logic necessary to provide the services used in this example: Init Service, Search Service, Present and Segment Services and Close Service. When a request arrives, the server analyzes it and sends the proper request to the client, by doing the operation requested, like searching in the database or negotiating the Z-association parameters.

Figure 2. Protocol Z39.50 communication diagram.

5. Z39.50 client implementation

The functions that any Z39.50 client has to perform are the ones described below. It has to be able to stablish a connection with other Z39.50 servers. To do that, the client has to make a connection request, stablish it, maintain it, hold it for the required time and then, it has to end the connection according to the protocol that has been utilized. During the connection, the client has to be able to send a search request. This request has to use a language that can be understood by the server. The client has to understand the data received by the server and display it in a simple and intelligible way.

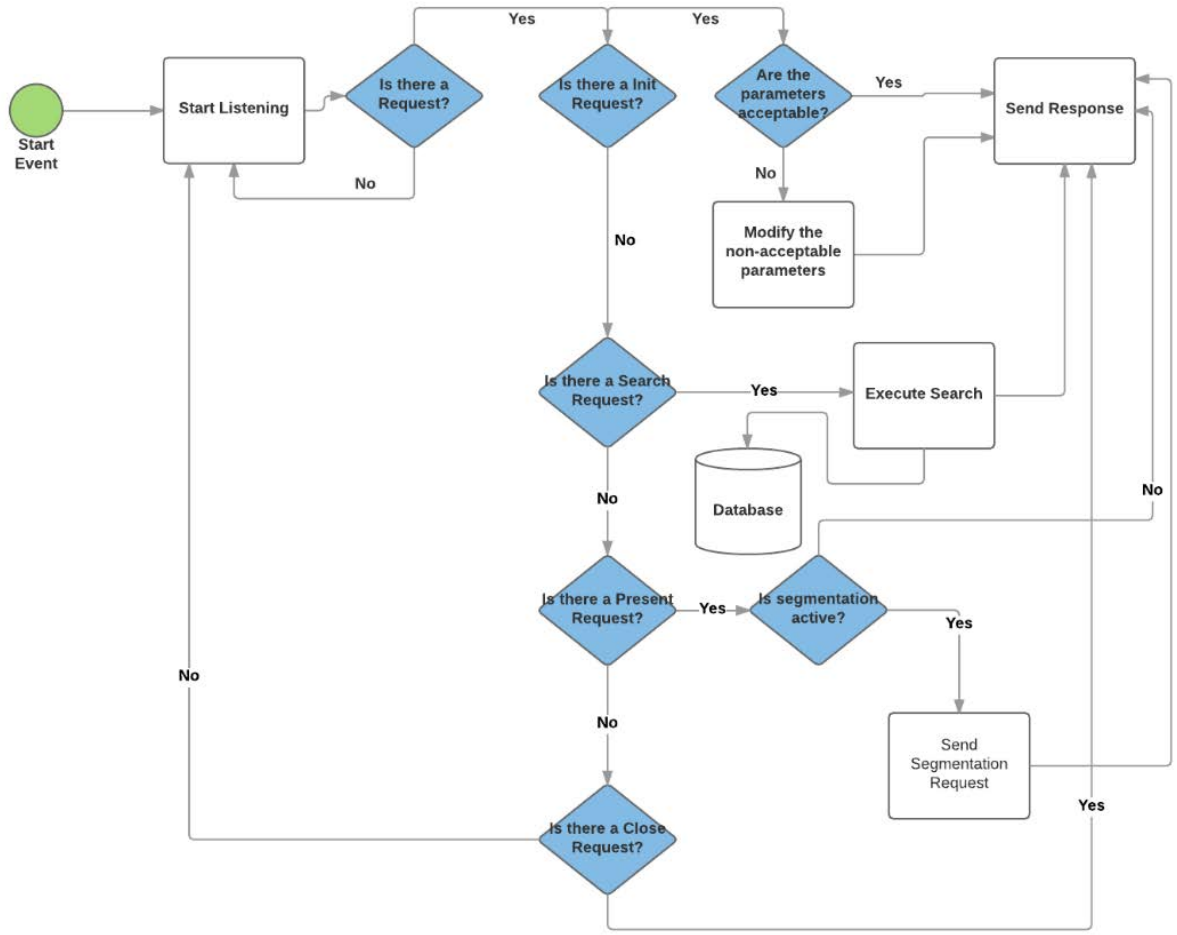

Figure 3. Server operation flowchart.

Up to this point, the basic functions of a Z39.50 client have been described. The advantage of this client over the existing ones is that, while most of them have been developed to use a single server or a closed group of them, the one implemented in this study allows to make a query to several servers simultaneously. To do that, there is a list of the servers that can be activated or deactivated for each query. This permits to search on a single server, on some of them or on all of them at the same time. Also, the list can be edited to add the desired servers. This increases the ability to search of the client. If there is a modification in any of the listed servers, the client adds the possibility to change it. In the same way, if any server ceases to be operational, it can be removed from the list.

To make a query the user can enter the Author of the book, the type and medium of the document, the Title or its ISBN. The type of search has been divided into the four aforementioned categories in order to display a simpler graphic interface and also, because those are the most common categories used in making a query. If the need to increase the number of said categories arises, it can be programed easily so as to search by every available type.

The reason for classifying the books and documents of a library is to find them easily. Hence, if the search is done in several servers at the same time, it is a good idea to determine which of the libraries with a positive result is located the closest to the user or to the desired collection point. This is the other advantage our client offers over the rest of the existing clients. Once the user makes a query, the application shows the library that is closest to the indicated point. To do that, the user has to enter an address. When the results are received, a web browser will open marking the location of the library on a Google Maps map. Wherewith, it is possible to obtain the fastest route to arrive to the destination.

Apart from what it is explained above, it is possible as well to select a maximum number of results for each server. Once those results are obtained, it is possible to print them or save them in an Excel worksheet. Finally, it is also possible to change the language of the application and access a help document for the better understanding of the program.

After presenting the different functions that the client is available to perform, we are going to explain how the application implements them. To do so, each one of the parts of the application will be shown. Firstly, Fig 4 shows a general view of the graphic interface.

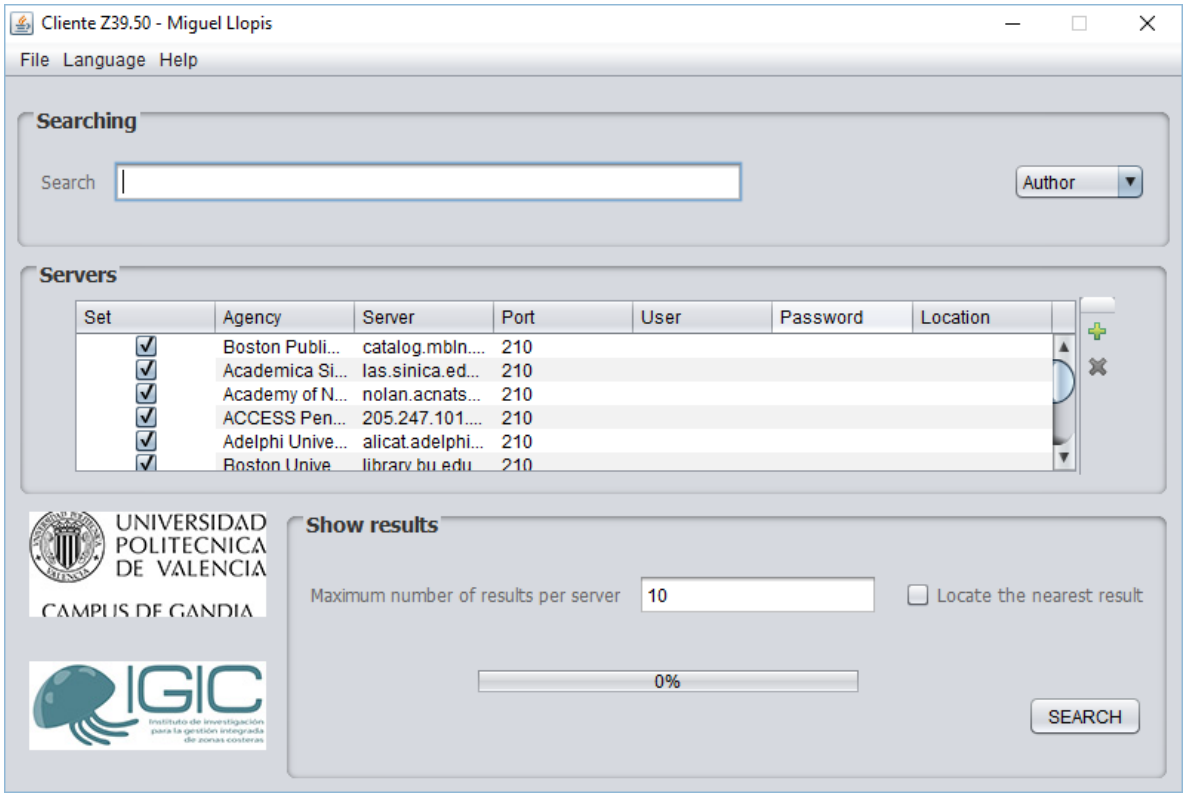

Figure 4. Developed Z39.50 client interface.

As can be seen, the application consists of four distinct sections: the menu, the searching section, the servers and the results. Each of the sections will be explained in detail. Fig. 5 shows the menu bar available at the application. It is a simple menu bar. It is divided into three distinct menus. The “File” menu is used to clean the query and to start a new one or to

exit the application. The “Language” menu allows to change the language of the application and at the “Help” menu it is possible to obtain information about the operation of the program and the people responsible for it.

Possibly, the most important section is the “Searching” one. The text that the user wants to search has to be written there. It must be indicated if the search is performed by Author, Title or ISBN code. Fig. 6 shows this section in detail.

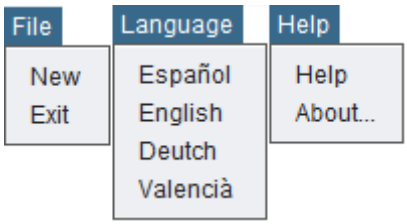

Figure. 5. Menu bar options of the developed Z39.50 client.

Figure 6. Search box of the developed Z39.50 client.

As it is shown in Fig. 7, there is a list of the servers in a table. These servers can be selected or deselected depending on whether the user wants to make a query on them or not. Also, at the right of the list there are two buttons. The green plus symbol allows to add a new server and the cross allows to delete the selected server. Selecting a specific server opens a window that allows to modify an existing server. The characteristics of the server than can be modified are the name of the Entity, its Server, the Port it utilizes, the Password and User name in case that it is necessary for that server and the location of the library.

The last part of the interface, which is shown in Fig. 8, is the section where the results of the search are displayed. The “Search” button sends the query. The box is used to limit the maximum number of results shown for each server, which is 10 by default. Finally, the checkbox is used to specify to the program that of all the available results, it has to select which one is located at the closest library from the indicated place.

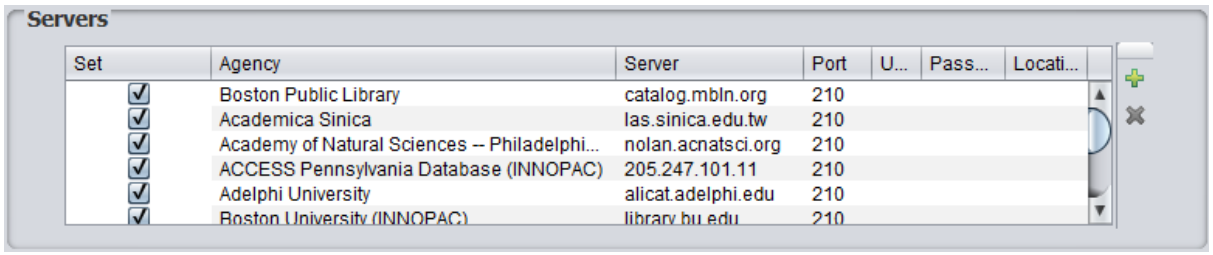

Figure 7. Servers box of the developed Z39.50 client.

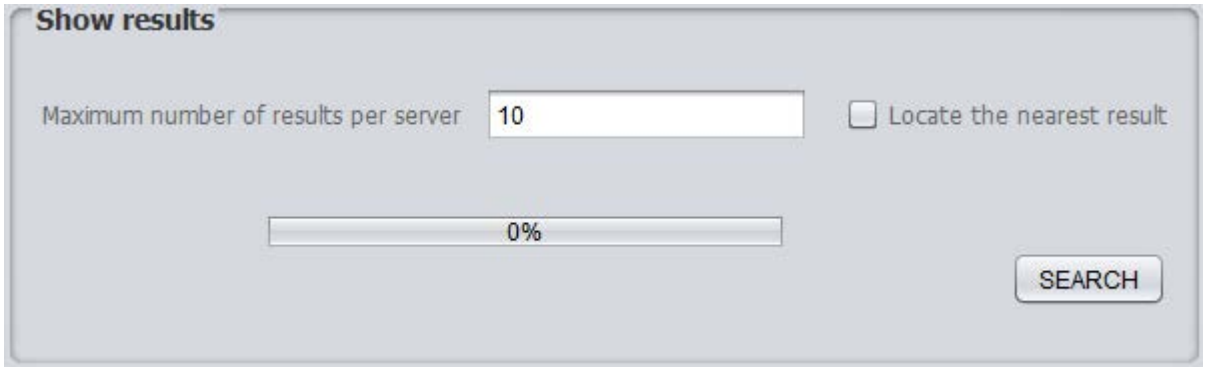

Figure 8. Results configuration box of the developed Z39.50 client.

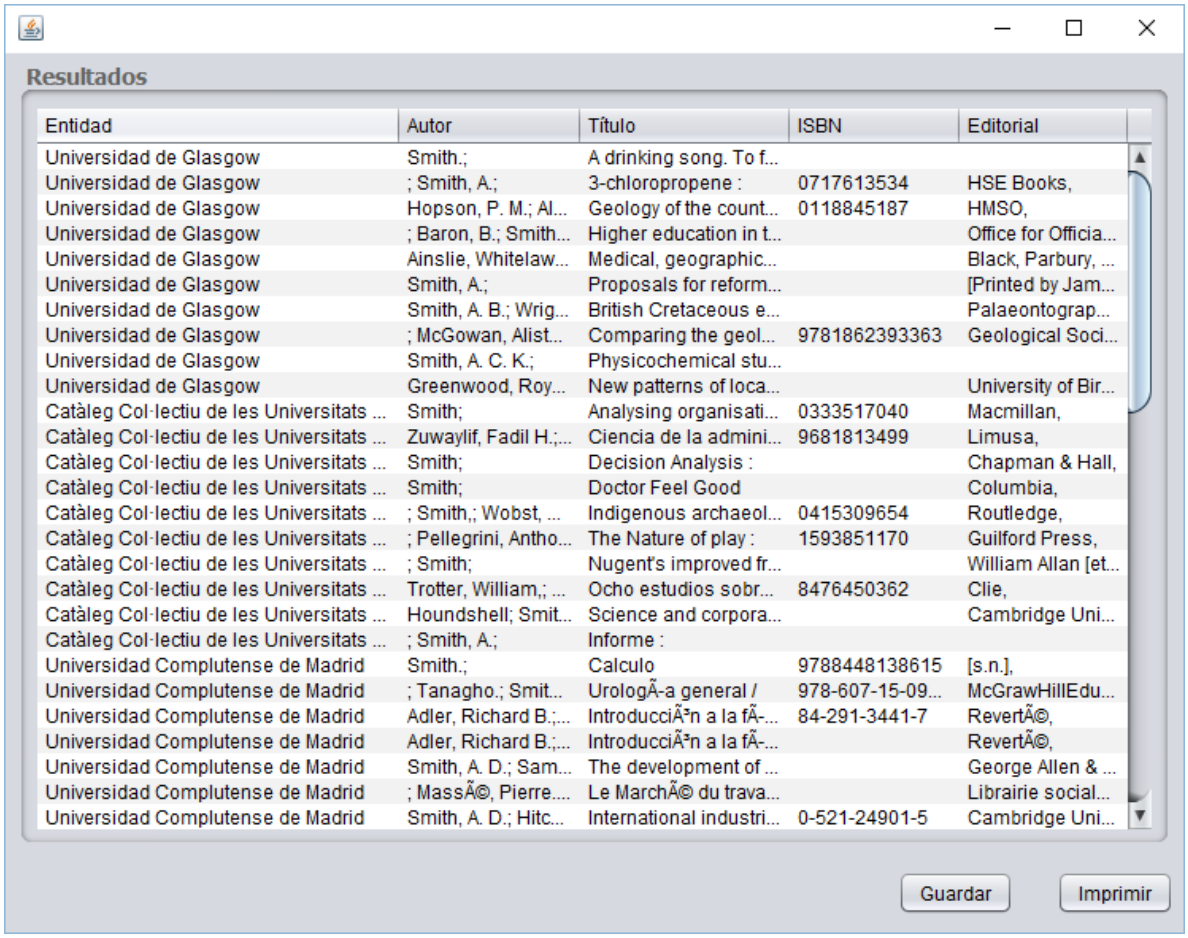

Figure 9. Results after a search.

If the location checkbox is not selected, a new window will open with the obtained results. That window is shown in Fig. 9. The results are presented in a table classified by Entity, Author, Title, ISBN and Publishing House. The “Print” button opens the printing properties window, and the “Save” button opens a dialog box that allows to select the location and the name of the file. This file will be saved with the “.cvs” format in order to ease opening it as an Excel table.

If the “Locate the closest result” checkbox is activated a window will open when the user makes a query. The address from which the closest server is located must be entered in that window. That address will be translated into utm coordinates and, utilizing an algorithm to calculate distances, the application will obtain the closest library of all the positive results.

Once the address is entered, the results will be displayed and a web browser will open with a Google Maps map indicating where the closest library is placed. Also, utilizing the functionalities of Google Maps, it will be able to extend or reduce the field of vision, as well as finding the fastest route to the desired library. An example can be seen on Fig. 10.

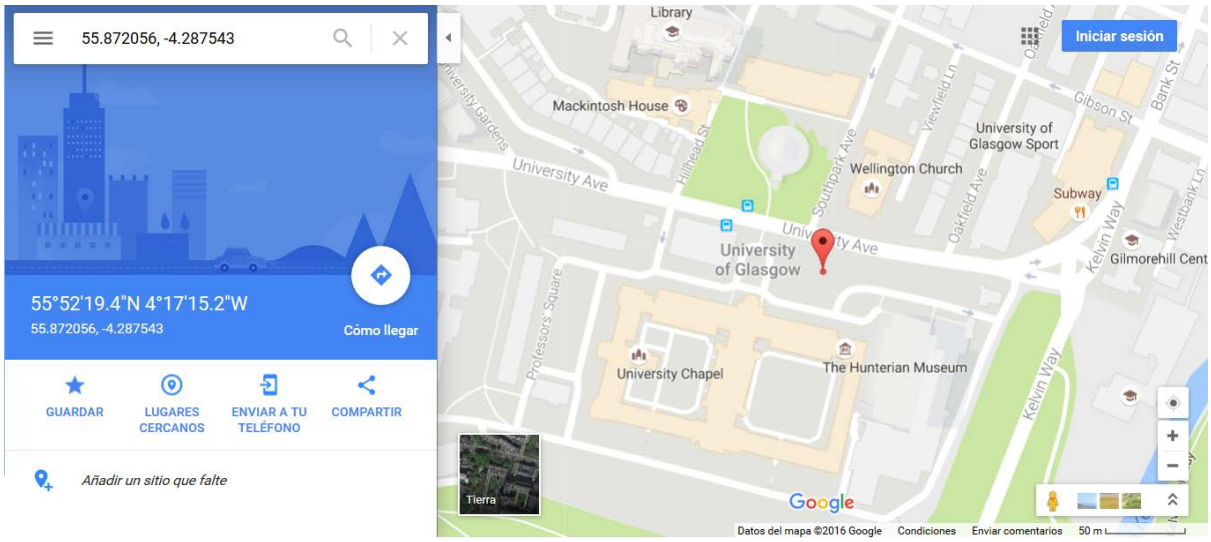

Figure 10. Location of the closest Library.

6. Comparative with other Z39.50 clients

In order to implement our propose we had to choose one of the open source clients available on the web to create a new client with the new features proposed. To do so a big number of these clients have been analyzed attending to a great number of features, so we could choose the optimal tool.

After analyzing the clients shown in Table 26 and Table 27 it concluded that every one of them is oriented to the point of view of the library user. Many of them are able to search the results in different formats, like Evergreen or Greenstone, and only some of them allow the user to visualize the desired content, such as Greenstone and SobekCM. Author and availability are features supported by most of the clients.

The access to multiple databases is only featured in Alejandría and JzKit. It shows that it is an aspect in which most of the available clients are lacking. Most of the clients are created to manage just one library. Only a few of them can search multiple databases and those who only can access those located in their servers such as BibData.com or Evergreen.

Some of the servers can locate in which library the desired document is located, such as Koha and OPACIAL. But after analyzing all of the clients, if it is possible to add a new server, none of them can locate the document taking into account the new server. Also, none of the clients can detect the closest library with the result of the search.

All the clients listed below are available to any user. Only some of them are multiplatform like Evergreen or Greenstone. The other ones are mostly either for Windows or Linux OS. All of them except BibData.com and OPACIAL support more than one standard. All of them support Z39.50 and most of them support MARC as well. The programming languages employed to implement the clients are mostly Java and Perl.

From the results obtained we have decided to use the JZKit tool for the reasons listed below. This tool uses MARC21, XMS and the Z39.50 protocol. With them searches and classifications about the desired information can be easily perform. JZKit provides a wide toolkit among which stand the Z39.50 Java libraries which are useful to our purposes. It is an open source tool, so everybody is able to modify the software to add the modifications and the suitable improvements. The last reason why we used JZkit to develop our solution is because of the programming language. It is entirely done in Java so the time it requires to alternate between different programming languages was avoided.

Table 26. Features comparison of Z39.50 clients. Part 1.

Name	Software Type	License	Development state	OS	Supported Technologies	Programming language
BibData.com [14]	Digital repository and online	GPL	Production/Stable	Windows	Z39.50	Java, Perl
Alejandría WorldLibrary [15]	N/C	N/C	Production/Stable	Windows	MARC21, Dublin Core, OAI, Z39.50	Several
Evergreen [16]	Integrated library system	GPLv2	Production/Stable	Multiplatform	MARC21, MARCXML, MODS, OpenURL, RSS, SIP2, XSLT, Z39.50	Perl, JavaScript
Greenstone [17]	Digital repository	GPLv2	Production/Stable	Multiplatform	Dublin Core, ePUB, MARC21, MARCXML, MODS, OAI, TEI, XLS, Z39.50	C++, Java, JavaScript, Perl

Jzkit [18]	N/C	GPLv2	Liberated	N/C	N/C	Java
Koha [19]	Integrated library system	GPLv3	Production/Stable	Multiplatform	LDAP, MARC21, MARCXML, OAI, RSS, SIP2, XSLT, Z39.50	Perl
Mercury Z39.50 Client [20]	Digital repository	N/C	N/C	Windows, Linux	MARC21, MARCXML, XSLT, Z39.50	N/C
Net-Z3950-Simple2ZOOM [21]	Interface Discovery	N/C	Production/Stable	Linux	SRU, Z239.50	Perl
OPACIAL [22]	Access Catalogue	Academic license	Historical/Abandoned	Search engine	Z39.50	PHP
OPALS [23]	Integrated library system	GNU General Public License	Production/Stable	Linux	MARC21, Z39.50	Perl
Pazpar2 [24]	Interface Discovery	GNU General Public License	Production/Stable	Linux	SRU, XSLT, Z39.50	Perl
PMB [25]	Integrated library system	GPL	N/C	Windows, Mac, Linux	OAI, RSS, Z39.50	PHP
Potthakalaya [26]	Compilation	Several	Production/Stable	Linux	BibTeX, Dunlin Core, epub, MARC21, Z39.50	Java, JavaScript, Perl, PHP, Python
Senayan [27]	Integrated library system	GPLv3	Production/Stable	Search engine	Dublin Core, OAI, MARC21, Z39.50	PHP
SobekCM Digital Repository Software [28]	Digital Repository and Metadata manipulation	GPLv3	Production/Stable	Windows	Dublin Core, EAD, MARC21, MARCXML, MODS, OAI, OPEN URL, RSS, SOLR, Z39.50	C #
Virtual Library for Moodle [29]	Interface Discovery	Public license	Development	Search engine	MARC21, Z39.50	PHP
Our Z39.50 Client	Interface Discovery	GNU General Public License	Development	Multiplatform	Z39.50	Java

Table 27. Features comparison of Z39.50 clients. Part 2.

Name	Databases	Multiple search	Location	Preview	Availability	Author	Publishing house	Multi-format search
BibData.com [14]	BibTex	In its servers	Partial	No	No	Yes	Yes	N/C
Alejandría World Library [15]	Several	No	N/C	N/C	Yes	Yes	Yes	Yes
Evergreen [16]	PostgreSQL	In its servers	Partial	N/C	Yes	Yes	Yes	Yes
Greenstone [17]	GDBM	No	No	Yes	N/C	Yes	Yes	Yes
Jzkit [18]	SRW and related databases	N/C	No	No	N/C	Yes	Yes	N/C
Koha [19]	MySQL	No	In library	N/C	Yes	Yes	Yes	No
Mercury Z39,50 Client [20]	OLE DB	In its servers	N/C	No	No	Yes	Yes	N/C
Net-Z3950-Simple2ZOOM [21]	N/C	N/C	N/C	N/C	N/C	N/C	N/C	N/C
OPACIAL [22]	My SQL	No	In library	No	Yes	Yes	Yes	Yes
OPALS [23]	Ms SQL	In its servers	Partial	N/C	N/C	Yes	Yes	Yes
Pazpar2 [24]	Ms SQL	In its servers	Partial	N/C	N/C	Yes	Yes	Yes
PMB [25]	My SQL	No	No	No	Yes	Yes	Yes	N/C
Potthakalaya [26]	My SQL	No	No	Yes	Yes	Yes	No	Yes
Senayan [27]	My SQL	N/C	N/C	No	Yes	Yes	Yes	N/C
SobekCM Digital Repository Software [28]	Ms SQL	In its servers	Partial	Yes	N/C	Yes	Yes	Yes
Virtual Library for Moodle [29]	My SQL	No	No	Yes	Yes	Yes	Yes	Yes
Our Z39.50 Client	Txt, cvs	Yes	Yes	No	No	Yes	Yes	Yes

7. Conclusion

With the proposed Z39.50 client we have been able to perform searches on several libraries at the same time, indicating as well where the closest result is located. We have demonstrated it with an example of its operation. Also, it is able to add, modify and remove any server.

Unlike most of the existent clients, our application is able to connect and search on several servers simultaneously. It also allows printing or saving the results.

With our Z39.50 client the user avoids the utilization of different clients to search on different servers as it is the norm. Likewise, in case of obtaining various locations when searching for a book, it is possible not only to locate the closest library, but to obtain the fastest route with the help of the functionalities of Google Maps.

There are different aspects where our application could be further developed. One of the most immediate ones is the adaptation of our client to allow it to operate in mobile devices. Also, with the functionalities available in current devices the location process could be adapted to use the GPS of the device and to show on real time the route towards the desired library. It could also be adapted to be able to use it via web.

Another aspect could be expanding the application to be able to make queries taking into account more parameters, although the most common ones are already available. Likewise, it may not be necessary, but an advanced search could be created allowing performing cross searches.

With the collaboration of the libraries that wanted to join, a book reservation system could be developed. If they are available in digital format, they could be displayed through the device. We want to create a system that allows the interconnection between different libraries to be able to share information such as the one presented in [31].

Lastly, the application could support commercial functionalities in order to search for books in specialized shops and book stores. It also could enable to buy them without being in the book store. To do so, the willing book stores should have a catalog adapted to the Z39.50 protocol and grant its access to the application.

References

[1] Forbes.com. Gil Press, “A very short History of Big Data”. Available at http://www.forbes.com/sites/gilpress/2013/05/09/a-very-short-history-of-big-data/#52702dbe55da (Last accessed October 20, 2016)
[2] Yale Univerity Library. Available at http://ask.library.yale.edu/a.php?qid=1023126 (Last accessed October 20, 2016)
[3] julianmarquina.es. Julián Marquina, “Tendencias en torno al mundo de la información y de las bibliotecas”. Available at http://www.julianmarquina.es/tendencias-entorno-al-mundo-de-la-informacion-y-de-las-bibli

otecas-iflatrends/ (Last accessed October 20, 2016)
[4] Arriola O. and Butrón K., “Sistemas integrales para la automatización de bibliotecas basados en software libre”, Acimed, 2008, vol. 18, no 6, pp. 0-0. Available at http://bvs.sld.cu/revistas/aci/vol18_6_08/aci091208.htm (Last accessed October 20, 2016)
[5] Information Retrieval (Z39.50): Application Service Definition and Protocol Specification. Available at http://www.loc.gov/z3950/agency/Z39-50-2003.pdf (Last accessed October 20, 2016)
[6] National Library of Spain. Available at http://www.bne.es/es/Inicio/Perfiles/Bibliotecarios/SuministroRegistro/DescargaZ3950/InformacionTecnica/ (Last accessed October 20, 2016)
[7] Kronosdoc, list 1. Available at http://old.kronosdoc.com/gtbib_z3950.php?texto=&ordenar=TIPO&orden=ASC (Last accessed October 20, 2016)
[8] Kronosdoc. list 2. Available at http://old.kronosdoc.com/gtbib_z3950.php?campo=PAIS&texto=ES (Last accessed October 20, 2016)
[9] Library of the Congress. Available at http://www.loc.gov/z3950/gateway.html (Last accessed October 20, 2016)
[10] Genesereth M., Keller A., Duschka, O., “Infomaster: An information integration system”, Proceedings of the 1997 ACM SIGMOD international conference on Management of data, pp. 539-542, Tucson, AZ, USA, May 11-15, 1997. http://dx.doi.org/10.1145/253260.253400
[11] Larson R., “Distributed resource discovery: Using Z39. 50 to build cross-domain information servers”, Proceedings of the 1st ACM/IEEE-CS joint conference on Digital libraries, pp. 52-53, Roanoke, VA, USA, June 24-28 2001. http://dx.doi.org/10.1145/379437.379448
[12] Corfield A., Dovey M., Mawby R. and Tatham C., “JAFER ToolKit project: interfacing Z39. 50 and XML”, Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries, pp. 289-290, Portland, USA, 13-17 July 2002. http://dx.doi.org/10.1145/544220.544287
[13] Danijela B. and Dušan S., “XML editor for search and retrieval of bibliographic records in the Z39. 50 standard”, The Electronic Library, vol. 27, no 3, pp. 474-495, 2009. http://dx.doi.org/10.1108/02640470910966916
[14] BibData webpage. Available at http://bibdata.com/Default.aspx. (Last accessed October 20, 2016)
[15] Alejandría nBibliotecas webpage. Available at http://www.alejandria.biz/Productos/nBibliotecas/ (Last accessed October 20, 2016)
[16] Evergreen. Available at https://evergreen-ils.org/ (Last accessed October 20, 2016)
[17] Greenstone. Available at http://www.greenstone.org/index_es (Last accessed October 20, 2016)
[18] JZKit3. Available at http://www.k-int.com/products/jzkit (Last accessed October 20, 2016)
[19] Koha. Available at https://koha-community.org/ (Last accessed October 20, 2016)

[20] Mercury Z39.50 Client. Available at http://www.basedowtech.com/projects/mzc (Last access October 20, 2016)
[21] Net-Z3950-Simple2ZOOM. Available at http://search.cpan.org/~mirk/Net-Z3950-Simple2ZOOM/ (Last accessed October 20, 2016)
[22] OPACIAL. Available at https://foss4lib.org/package/opacial (Last accessed October 20, 2016)
[23] OPALS. Available at http://wordpress.hyperion.scoolaid.net/ (Last accessed October 20, 2016)
[24] Pazpar 2. Available at http://www.indexdata.com/pazpar2 (Last accessed October 20, 2016)
[25] PMB. Available at http://www.sigb.net/pmb/ (Last accessed October 20, 2016)
[26] Potthakalaya. Available at https://foss4lib.org/package/potthakalaya/releases (Last accessed October 20, 2016)
[27] Senayan. Available at https://sourceforge.net/projects/senayanlib/ (Last accessed October 20, 2016)
[28] SobekCM. Available at http://sobekrepository.org/ (Last accessed October 20, 2016)
[29] Virtual Library for Moodle. Available at http://vl4moodle.sourceforge.net/ (Last accessed October 20, 2016)
[30] Lloret J., Tomas J., Garcia M. and Lacuesta R., “A B2B Architecture and Protocol for Researchers Cooperation”, International Journal of Cooperative Information Systems, Vol. 22, Issue 2. Pp. 1350010-1 - 1350010-27, 2013. http://dx.doi.org/10.1142/S021884301350010X
[31] J. Lloret, F. Boronat, C. Palau and M. Esteve, “Two levels SPF-based system to interconnect partially decentralized P2P file sharing networks”, Joint International Conference on Autonomic and Autonomous Systems and International Conference on Networking and Services-(icas-isns' 05). Papeete, Tahiti, October 23-28, 2005. http://dx.doi.org/10.1109/ICAS-ICNS.2005.96