Project Documentation - AppRepositoryS

Cocei Tiberiu, Group B4

December 12, 2018

1 Introduction

Stocking information and allowing clients to search for it using different criteria is an important aspect of many applications and most websites.

The project name AppRepositoryS describes a server used for storing software and hardware requirements of a multitude of applications. It also allows any client connected to it to add a new application or to search for any of its stored applications using criteria such as name, requirements, producer, status (open source, freeware, shareware) etc.

This can be done through a variety of ways, such as using different text files to stock the information or using a database. I chose the latter, since it is easier and more efficient to add a new line to the database instead of creating an entire new file when adding a new application or to search every file when the client makes a search, compared to searching every line in the database.

2 Utilised technologies

For communication the TCP/IP protocol is used as it provides reliable, ordered and error-checked delivery of a stream of bytes between client and server as well as vice versa. While the UDP protocol would be overall faster, losing information would lead to possible failures when searching for applications or incomplete data entry when adding a new application.

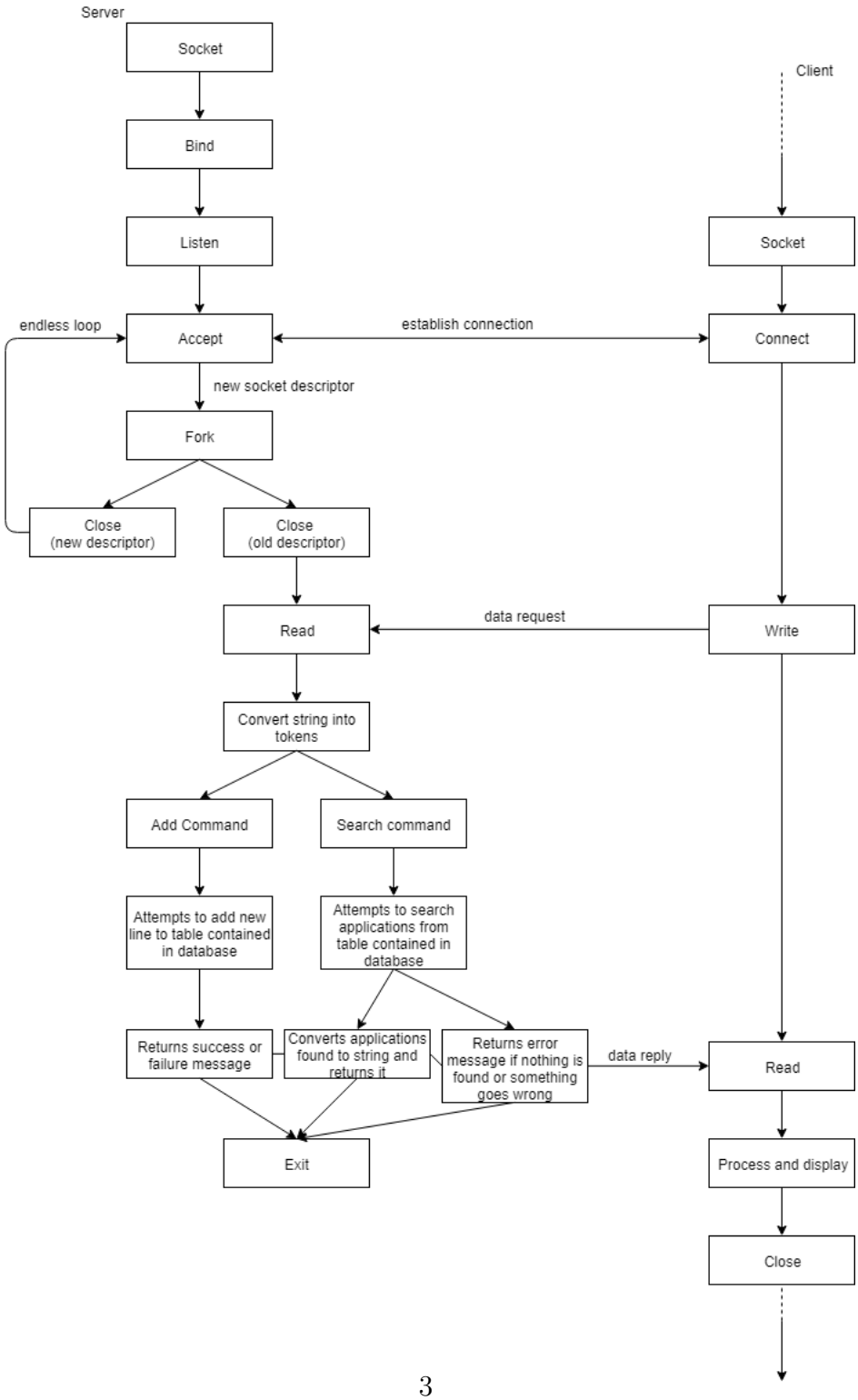
SQLite is a relational database management system contained in a C programming library. It allows the server to work with a database instead of a multitude of text files.

3 Application architecture

The AppRepository server handles two different commands: add and search. If the client wishes to add a new application to the database he/she may use the first command. Upon selecting it, the client will need to fill in information regarding it. Fields such as name, producer, status and operating system must be filled in while other fields such as RAM memory and additional libaries may be omitted. If the client wishes to search for an existing application in the database, he/she may use the second command. Upon doing so a similar process takes place, however this time all fields are optional.

Assuming that the server receives the information in a string, it will use the 'strtok' command to divide it. The first word in the string would indicate whether or not the client wishes to add or search an application. Each even numbered word would indicate a field name, with the odd numbered word following it indicating the information to add or search in said field. Afterwards, using commands provided by the SQLite library, a new line will be added to the database if a new application is being added or every line in the database will be checked for a possible match if not.

After a new application is added correctly a success message will be returned to the client. If the add command fails (incomplete mandatory field or other types of errors) an error message will be returned instead, which will specify what went wrong. If the search command finds one or more applications, they will have their information converted into a string which will be sent back to the client. The client will afterwards modify it in a presentable way and display it. If the command fails (no applications found or other types of errors) a message will be returned describing what went wrong. If all fields are empty, the search command will display all the applications stored in the database.


Server
Socket
Bind
Listen
endless loop
Accept
new socket descriptor
Fork
Close
(new descriptor)
Close
(old descriptor)
Read
Convert string into
tokens
Add Command
Search command
Attempts to add new
line to table contained
in database
Attempts to search
applications from
table contained in
database
Returns success or
failure message
Converts applications
found to string and
returns it
Returns error
message if nothing is
found or something
goes wrong
Exit
Client
Socket
establish connection
Connect
data request
Write
data reply
Read
Process and display
Close

4 Implementation details

Algorithm 1 This section of code will check whether or not the database can be opened.

```
1: sqlite3 *bd;
2: int bdaux = sqlite3_open("repository.db", &bd);
3: if (bdaux != SQLITE_OK) then
4:     fprintf(stderr, "Database cannot be opened: %s\n",
   sqlite3_errmsg(bd));
5:     sqlite3_close(bd);
6:     return 1;
7: end if
```

Algorithm 2 This section of code will check if the tables exist. If not, they will be created in the next section.

```
1: Cursor c = bd.rawQuery("SELECT COUNT(*) FROM sqlite_master
   WHERE type= ? AND name= ?", new String[] {"table", BaseInfo});
2: if (!c.moveToFirst()) then
3:     //do next section of code
4: end if
5: c.close();
```

Algorithm 3 This section of code will create the tables if they did not exist.

```
1: char *err_msg = 0;
2: char *tables = "CREATE TABLE BaseInfo(Id INT NOT NULL, Name
   TEXT NOT NULL, Producer TEXT NOT NULL, Status TEXT NOT
   NULL, Category TEXT, Translations TEXT);"
3: "CREATE TABLE Software(Id INT NOT NULL, OS TEXT NOT
   NULL, Programming_Language TEXT, UI TEXT, Size TEXT, ExtraL-
   ibrary1 TEXT, ExtraLibrary2 TEXT, ExtraLibrary3 TEXT);"
4: "CREATE TABLE Hardware(Id INT NOT NULL, CPU TEXT, GPU
   TEXT, RAM TEXT);";
5: bdaux = sqlite3_exec(bd, tables, 0, 0, &err_msg);
```

Algorithm 4 Code section for adding an application to the database.

```
1: char *tables = "INSERT INTO BaseInfo(nrId, Name, Producer, Status,
   Category, Translations);"
2: "INSERT INTO Software(nrId, OS, ProgrammingLanguage, UI, Size,
   ExtraL1, ExtraL2, ExtraL3);"
3: "INSERT INTO Hardware(nrId++, CPU, GPU, RAM);";
4: bdaux = sqlite3_exec(bd, tables, 0, 0, &err_msg);
5: //success message or error handling
```

Algorithm 5 Code section for searching applications from the database.

```
1: Cursor c = bd.rawQuery("SELECT a.Name, a.Producer, a. Status,
   a.Category, a.Translations, b.OS, b.Programming_Language, b.UI,
   b.Text, b.ExtraLibrary1, b.ExtraLibrary2, b.ExtraLibrary3, c.CPU,
   c.GPU, c.RAM FROM BaseInfo a natural join Software b natural
   join Hardware c WHERE a.Name = ? AND a.Producer = ? AND
   a.Status = ? AND (a.Category = ? OR a.Category IS NULL) AND
   (a.Translations = ? OR a.Translations IS NULL) AND b.OS = ?
   AND (b.Programming_Language = ? OR b.Programming_ IS NULL)
   AND (b.UI = ? OR b.UI IS NULL) AND (b.Text = ? OR b.Text
   IS NULL) AND (b.ExtraLibrary1 = ? OR b.ExtraLibary1 IS NULL)
   AND (b.ExtraLibrary2 = ? OR b.ExtraLibrary2 IS NULL) AND
   (b.ExtraLibrary3 = ? OR b.ExtraLibrary3 IS NULL) AND (c.CPU
   = ? OR c.CPU IS NULL) AND (c.GPU = ? OR c.GPU IS NULL)
   AND (c.RAM = ? OR c.RAM IS NULL)", new String[] { Name, Pro-
   ducer, Status, Category, Translations, OS, ProgrammingLanguage, UI,
   Size, ExtraL1, ExtraL2, ExtraL3, CPU, GPU, RAM});
2: //note: query will fail to find applications which have a requirement and
   given string does not at the moment
3: if (c.getCount() > 0) then
4:     c.moveToFirst();
5:     do
6:     //add to return string
7:     while (c.moveToNext());
8: else//error, nothing has been found
9: end if
```

5 Conclusion

AppRepositoryS is a server that enables the client to search or add applications to its database. It easily does this using SQL Queries with the help of SQLite library. It can also handle any amount of clients due to its concurrent structure. The server is a good choice for working with limited information of a variable amount of applications of any type.

6 Bibliography

https://en.wikipedia.org/wiki/SQLite

https://en.wikipedia.org/wiki/Transmission_Control_Protocol

https://profs.info.uaic.ro/~gcalancea/Laboratorul_7.pdf

http://zetcode.com/db/sqlitec/

https://stackoverflow.com/questions/1601151/how-do-i-check-in-sqlite-whether-a-table-exists