Software Solutions for Recovery of the Lost Data from Mobile Phones, Tablets and Computers, Used in the Educational Process

ANTOANELA NAAJI
Department of Economical and Technical Sciences
"Vasile Goldis" Western University
Arad, ROMANIA

MARIUS POPESCU
Department of Economical and Technical Sciences
"Vasile Goldis" Western University
Arad, ROMANIA

MONICA CIOBANU
Department of Economical and Technical Sciences
"Vasile Goldis" Western University
Arad, ROMANIA

Abstract—It is already known that during the pandemic the educational process took place most of the time online or at most in a hybrid format. Within these activities, there are situations in which, in addition to the online teaching, it is necessary to conduct and store courses, homework or other teaching materials on various types of devices (phones, tablets or computers). When they are disconnected incorrectly or become corrupt, errors occur, and the information may no longer be accessed. In the case of the Windows operating system if the disconnection of the PCs is performed by the users in an erroneous way, after reconnecting the storage devices, they still have access to the data. In the case of the Linux operating system if the disconnection is not the professional one, most of the time there are chances that at reconnection, the data that was saved on that storage device cannot be any longer accessed. The current paper presents two methods for recovering the lost information: one for the educational data recovery, and one for recovering contacts, which are often important in the communication process. These methods involve techniques for saving educational information/contacts from destroyed, corrupt or inaccessible storage devices when they cannot be used normally. A first application, proposed for PCs, uses the Master File Table and tries to retrieve information, by using an algorithm implemented in the C++ language, with multiple advantages over the algorithms found in similar applications. The second software application is developed for the Android platform and it was created to make a backup to the contacts from the smartphone and save them in a text (.txt) file. The software applications presented in the paper are useful especially for those who own computers and phones from older generations.

Keywords—data recovery, exhaustive algorithms, user-friendly interface, connectivity, efficiency, flexibility

Received: April 10, 2021. Revised: May 10, 2021. Accepted: May 20, 2021. Published: May 26, 2021.

1. Introduction

Pupils, students (course attendees) and teachers can now communicate with anyone from anywhere and anytime, using free applications such as Skype, Zoom, Adservio, Google Classroom and others. Thus, conversations are held in the educational process, similar to face-to-face conversations. In the case of online classes, the only people who can get a link from the online class are the students who participate based on the invitation received from the teacher in private, in the messaging within the platform. The accounts of the students and teachers are secured by username, password, two-step authentication, data set by each school/faculty when creating the account. Moreover, understanding society and culture is important when designing and developing IT technologies, especially in the educational field [1, 2]. Therefore, IT technologies are becoming ubiquitous in the educational process, and can have a major impact on learning. Learning will be directed more and more towards the outside of the classroom, a case that can positively influence the way of collaboration of those involved in the educational process. M-learning (mobile learning) is defined as a dynamic learning environment which use mobile technologies, especially in the field of education [3, 4, 5, 6].

There are different programming environments for mobile technologies, which are chosen depending on the target customers, the platform and the used terminal, the utility, etc. Programming environments, constituted as platforms can be grouped into 4 major categories: databases (SQL Server, Oracle, Postgre SQL, SQL Server Compact, SQLite), programming languages (C #, C ++, Java, Android, Objective-C, iOS, T-SQL), development platforms (Visual Studio, Windows Azure, Eclipse, Xcode) and frameworks (.NET (ASP.NET, ASP.NET MVC, Win Forms, Compact Framework), Bootstrap/jQuery/Telerik, Android Framework, Cocoa Touch Frameworks).

The challenge of using mobile devices and technologies to turn online learning into an everyday educational process is not at all recognized as learning. However, in pandemic conditions this is necessary. When accidentally deleting a file or a folder, it is recommended not to copy or move files to that device until a data recovery program is used. Thus, the chances to recover the data increase. Some data can be retrieved by using free applications available *online*, but this does not apply to anything and under any conditions. The most common data recovery situations occur due to deletion when the operating system is destroyed. If files are lost due to an error, there are methods to recover data that was accidentally deleted or due to certain system errors. For example, this can be done with a free program that can recover data or files from a hard drive.

Such software are: EasyRecovery, EASEUS *Data Recovery*/*EASEUS Partition Recovery*, *Ontrack EasyRecovery Professional*, *GetDataBack* and *Recuva*, programs that can help us solve these problems [7].

The aforementioned programs are recommended, but each of them is recommended for certain types of files. Data recovery was done until recently only by software specialists, but with *Recuva free* or other programs, data recovery can be done also by less experienced users. Data recovery is not done as a *copy/paste* from one partition to another but takes more time and depends on the amount of information that need to be recovered.

Also, transfer rates (read/write) cannot be compared between an HDD on SATA and an external one on USB. Using such programs, we can recover data that was deleted from a partition of an HDD, if the HDD has not been formatted and no new information has been overwritten over the original ones.

How data is stored on the storage device differs from one operating system to another. For example, the Apple systems use their own file system. In case of losing the entire database with phone numbers, there are 3 ways to back up the phonebook: paid back-up (at a specialized store), free back-up (on PCs, by synchronization) and internet backup. For backup on the PC there are specific modes for each phone, through their software [8, 9]. Usually, the smartphone sync software is on a CD, if not on the manufacturer's website. Internet backup is one of the least used, although it is among the safest. If the laptop is backed up, there is a possibility of losing the data if: both the smartphone and the laptop are stolen, or both devices are damaged, because of an accident, calamities, etc. There are services that work if the internet is active and offer free backup to the phonebook, which lasts very little. The service only works if the internet is active on the mobile. Another online service offers a social media system in addition to the synchronization system, respectively [10], a place to synchronize calendars, contacts, microblog archive, RSS (Received Signal Strength) and other communication systems, more or less important. It is recommended to use all these options to save the phone contacts, by synchronizing with an online database, with personal computer, or at a specialized store. This will keep your information safe, even if the phone is lost, infected, or contacts are accidentally deleted. If all the variants described above are backed up, then the chances of losing something from the agenda tend to zero. All the backup methods and applications described above do not need *root* to apply them [11].

During the COVID-19 pandemic, the educational process (in Romania) often took place online. In this activity, in addition to online teaching, it is necessary to carry out and store courses, homework or other teaching materials. However, not all attendees (pupils and students) have high-performance devices (phones, tablets or computers) equipped with updated software, which allow them to recover data and contacts in the case that the operating system is destroyed, and the data/contacts are permanently deleted. In this context, the following sections present two software applications, developed for the devices which run older versions of operating systems, to make it possible to recover lost data and contacts when the (older manufacturing) devices have been corrupted or when the user accidentally deleted files/contacts.

2. Platforms, Applications and Backup Techniques

For the *Unix* platform the most used application is *extundelete* [12], which recovers the data stored on the device (declared as ext3 or ext4 format) corrupted (the operating system does not boot) or when the user accidentally deleted critical files. The types of *ext3* and *ext4* partitions are the most common on such an operating system.

Third extended filesystem (ext3) is a "*journaled*" type of file. The limit of an *ext3* partition is 32TB when the size of a block is 8 KB. There are 3 levels of journaling implemented in *ext3* (*journal*, *ordered* and *writeback*).

Fourth extended filesystem (*ext4*) was developed to make improvements to the *ext3* file type, while maintaining compatibility with its predecessor. The limit of an *ext4* partition has been extended to 1 EB, and the size of a file is 16 TB. *Extundelete* can recover the content of a node (uses concepts from the *ext3grep* program, which proved to be a successful data recovery code for *ext3* and *ext4* partition types) by searching the logged system file for an old save of that node.

Using this information, it determines the location of the file within the file system and saves the data corresponding to that node in the recovery file. The recovery is possible by the fact that the application associates the node number with the name of the deleted file in a directory, given that this information is left behind after deleting the file. This rule no longer applies when a storage device is formatted and each byte is overwritten with zeros, but such formatting is done for high-security files [13].

If the formatting was not overwritten, then *extundelete* will try to associate for a match of node number and file name in older copies that are still saved in the log. The *extundelete* interface is made using the Qt for Linux framework [14]. In order to have access to modify partitions, the run must be done under full rights - administrator (*root*).

In Romania, the *Windows* platform is more common on personal computers (PCs) than on the *Unix* platform. New Technology Fille System (NTFS) is a file system for the *Windows* operating system with many improvements over the previous FAT (F*ile* A*llocation* T*able*) file systems. The features of the NTFS are:

- *NTFS Log* is a log file system that, using a log file (*$LogFile*), keeps information about changes, as in a log, before making final changes to the operating system.

Thus, the file system will not be corrupted in case of power failure or operating system failure; the internal structure is not altered by changes such as defragmentation or changes in the structure of the indexes in the file table and a consistent structure is maintained for possible *rollbacks* in cases of corruption.

- *Update sequence journal* (USN) constantly maintains information about changes to files or folders, as well as their attributes.

- E*ncrypting* F*ile* S*ystem* (EFS) allows data to be encrypted transparently for the user.

- V*olume* S*hadow* C*opy* (VSC) keeps historical file data on the NTFS partition, by copying old files and files that have recently been replaced into a *shadow* copy; thus, it is allowed to restore old files in case of *user* request, as well as to create *backups* of applications.

- M*aster* F*ile* T*able* (MFT) contains metadata about all files, folders and metafiles from *disk.MFT* (contains data about file name, location, size and permissions on the file).

The files are compressed into pieces of 16 clusters with a size of 4 kB. The advantage of compressing files is in the case of duplicate data.

The most common HDD defects are:

- The logical fault is characterized by partition failures, file system damage, accidental formatting or deletion. As logical defects that require data recovery from the HDD we can mention: accidental deletion of a file, folder or an entire partition, corruption of the file system due to a virus, accidental formatting of a partition, bad sectors that cause the operating system to crash.

- The electrical defect is characterized by a failure in the electrical board of the hard drive and occurs due to current fluctuations or improper disconnection.

- Firmware defect, when the firmware area of the hard disk represents information that allows the initialization and access of data on the surface of the hard drive. This information is on the surface of platters.

- Mechanical defect due to physical shocks, voltage fluctuations, prolonged use and manufacturing problems. When an HDD makes repetitive sounds, there is a high chance that it will have a mechanical problem.

Mobile technologies (mobile phones/smartphones) are defined as those portable and lightweight technological devices that can connect to the internet either through data cables or wireless connections [15].

Nowadays, more and more pupils and students have at least one smartphone of newer or older generation, being aware of the opportunities and facilities it offers in education. However, there are many people who do not know how to save contacts from the phonebook to Google, how to save applications, photos, etc. Smartphones are like PCs, endowed with an operating system, and therefore various operating errors can occur. There are situations when the smartphone is changed, lost or stolen, in which case everything needs to be saved somewhere or a backup made on the card, in the PC or in the cloud. If the backup is already done or is done before resetting the smartphone to factory defaults, it is very easy to restore the backups after the reset.

The Android platform [16] is an *open-source* platform for developing and running mobile applications, developed by Google and the Open Handset Alliance. Originally created for mobile phones, Android has become a major application platform for a wide range of mobile devices. It is a Linux-based platform, portable on a wide range of mobile platforms, optimized for devices with low power consumption and low virtual memory, it supports Java applications run by a Dalvik virtual machine, has multimedia support for vector graphics, high level of application security, component-based application architecture, SQLite-based data storage solutions, etc. Android has become one of the major mobile platforms, along with Windows Mobile, Symbian, iPhone and J2ME (J*ava* M*obile* E*dition*). To create software applications, platforms and applications are required [17], such as:

- J*ava* D*evelopment* K*it*, version depending on the age of the smartphone. Any version starting with JDK 6.0 is recommended, as Eclipse and the Android SDK need development tools included in the JDK (J*ava* D*evelopment* K*it*).

- *Eclipse IDE,* the use of an IDE-type visual development environment is not required, but it is recommended.

- *Android SDK Starter Package* are two distributions for the Starter package. If the tablet/PC operating system is Windows 7 and the installer distribution has been chosen, an error may occur on Windows because the installer does not detect the Java JDK.

- *Android SDK Manager*: Android SDK Components must also be installed, which include various development tools, documentation, Android platforms, external libraries, USB driver for Windows and application examples.

- A*ndroid* D*evelopment* T*oolkit (ADT) Plugin*, for Eclipse.

The simplest technique to make a *backup* in Android, which requires *root* rights on your smartphone, consists in using the *Titanium Backup* application on Google Play. Because not everyone wants to *root* the Android, they resort to other more sophisticated variants, which require the Android SDK, respectively, the *Android Debug Bridge* command. Backing up everything on the smartphone, after it has been confirmed in advance, involves using the *adb backup-apk-shared-all-f C:\User\NAME\backup.ab* command, in which case all the apps installed on the smartphone, all the settings and all the contents of the SD (S*ecure* D*igital*) card will be saved in the destination provided by the command. The created safety backup can be later restored on any Android using the *adb restore C:\User\NAME\backup.ab* command. This *backup* technique is not easy at all and there is no application that works for smartphones from different manufacturers. Because of this, in 90% of cases, the *backup* is created by Google in the cloud.

In iOS, we can perform an alternative *backup*: either in iCloud or using a laptop/computer. Thus, the steps followed in iCloud are: store backups (usually in up to 2 TB of storage space), encrypt, then create and use the *backups* in any location via Wi-Fi connection. When using computers: *backups* are stored on your Mac/PC (the storage capacity depends on the available space on your computer), encryption (by default, the encryption is disabled), and *backups* are created and used on your Mac or PC.

3. Development of Algorithms

3.1 Application for PCs and tablets

Before any recovery, two completely different situations are considered, in which there is or may not be a chance to recover files from a hard drive. When that storage environment no longer makes a sound when connected to power, the ability to ever access information from it tends to zero. Another case is when the PC no longer starts, the operating system no longer loads or the partitions on a hard disk are so defective that it requires formatting. The idea is that if some files no longer exist or can no longer be copied and the operating system displays surprising errors, we need an external hard drive or a USB stick to recover them.

We must take into account that we can never recover data from a defective hard drive on the same defective hard drive. Also, the entire operation becomes more complicated if the respective information has been overwritten with new ones. There is an amalgam of utilities on Windows that can be used for the purpose of recovering files. In cases of major emergency, or when there is a risk of overwriting the searched files, a *Data Recovery* program is installed, and with a USB stick *with Ubuntu Live* the data can be recovered.

Utilities like TestDisk, PhotoRec or Foremost are only found on Linux, and their mode of operation is so efficient that if it does not recover what it is looking for, it is almost certain that there is no other chance.

Even if the device is old, there is an increased level of compatibility with a wide variety of operating systems and devices. If we are using a computer that is a few years old, then we are talking about file systems such as NTFS, FAT32 or exFAT. NTFS has a major compatibility issue with something other than Windows. A log of changes to the contents of an NTFS storage medium helps to recover files in the event of a power failure or some stability issues with Windows. FAT32 is perfect for ensuring compatibility, as long as there are slim chances for files larger than 4GB. exFAT exceeds the 4GB limit but will not be compatible with anything other than a PC. NTFS is the best, the most modern and with the widest variety of options for ensuring security, unfortunately it is not suitable anytime and anywhere, but especially in the case of hardware.

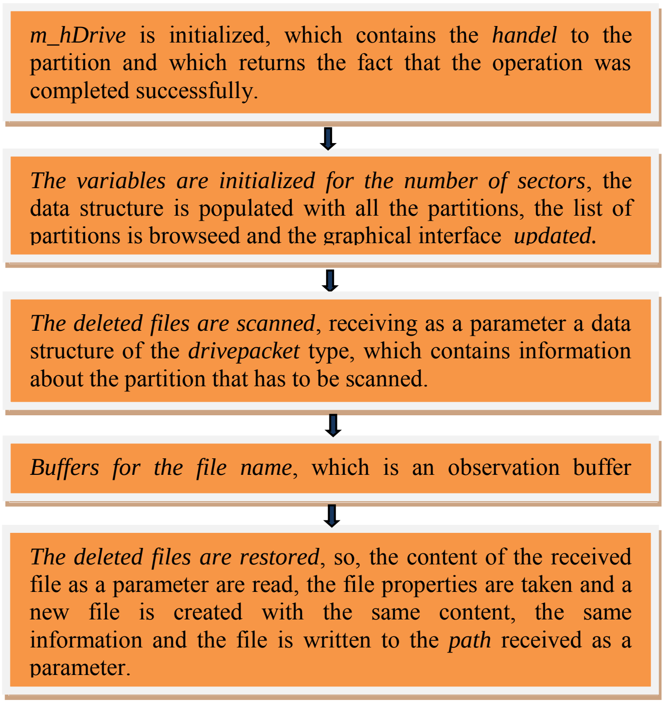
The file system on the computer has a volume table, which contains, among other things, an entry for each file on the hard disk, with the address of the location where the file is stored. When the file is deleted, some information in the table is removed to make way for free space. The physical data remains stored on the hard drive until it is rewritten by adding new files. Typically, specialized data recovery applications use this table for data recovery. If the data about the deleted file on the volume table is intact, then it will contain a pointer to the deleted file. A read of the table followed by a search of the address in the table can lead to the recovery of the file when it has not been rewritten with other information. If the file information was rewritten with information from another file, then the application scans all the blocks on the hard drive and assigns the information in the table to the found files.

If such a file is found it means that the file has not yet been deleted. If it does not find any file with correspondence in the table, then the file is marked as deleted, finally it can be recovered. The most important step in data recovery is not to use the storage device after losing the file, before using the data recovery program. Simply copying a file can rewrite data to the location of the old file and lose the file. Using these principles, the developed application uses the MFT table of the files and tries to recover them. All the *api* functions, as well as the interface that exports it, are written in C++. The functions used are compiled and exported to a file with the *dll* extension.

1. Application management interface

The link file, present in both the graphical interface and in the *dll*, contains the functions called for the application and is *IRecuperareDateManager.h*. It contains a data structure with information about each partition, *drivepacket* and a class of virtual methods, to *update* the graphical interface. When a new element is found that changes the graphical interface, it is transmitted by calling a function in the interface. Information about the found partitions is sent using a *drivepacket* structure, which is stored in the graphical interface for later use, and the partition name. It also sends information about the files found on the partition, such as: file index (file number used for display only), file name, file properties, whether or not it has been deleted, and the number of occupied clusters.

The basic interface is designed on purely virtual methods from *IrecuperareDateManager*. The flowchart containing the steps for developing the algorithm is presented in Fig. 1.


m_hDrive is initialized, which contains the *handel* to the partition and which returns the fact that the operation was completed successfully.

The variables are initialized for the number of sectors, the data structure is populated with all the partitions, the list of partitions is browseed and the graphical interface *updated.*

The deleted files are scanned, receiving as a parameter a data structure of the *drivepacket* type, which contains information about the partition that has to be scanned.

Buffers for the file name, which is an observation buffer

The deleted files are restored, so, the content of the received file as a parameter are read, the file properties are taken and a new file is created with the same content, the same information and the file is written to the *path* received as a parameter.

Fig.1 Basic interface development algorithm.

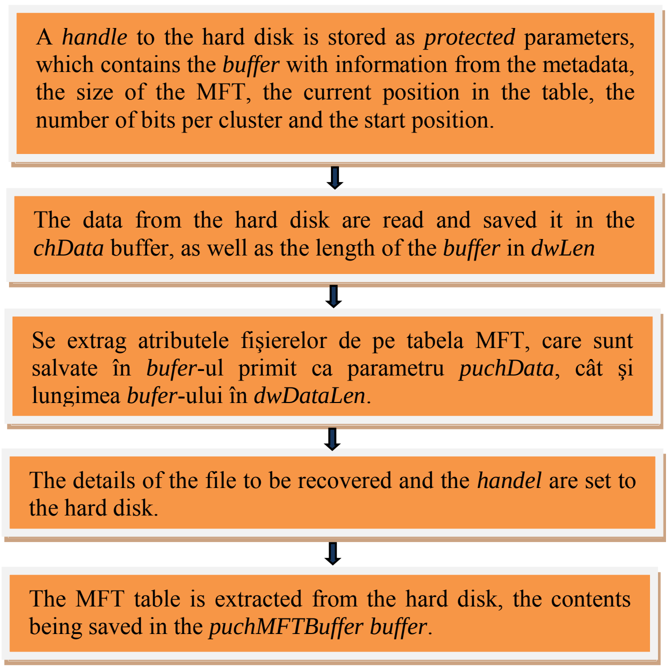
Step I. An object of *IcalbackInterface* type is registered, which is kept as a private variable, so that it can be used to *update* the graphical interface. The received parameter, being of the *pointer* type, is checked if it is valid and the error code is returned, for a possible message from the graphical interface. The hard disk is scanned for partitions, and then the file that contains information about the partitions on the hard disk is accessed, and the *handel* of the file is kept in *hDrive*. When it encounters a problem, the error code is returned. Information is read from the partition file and the information in it is stored.

Step II. The graphical interface is populated with information about the found partition, by using the *callback* function, with the partition name and the data structure, with information about the partition.

Step III. If the *handle* in the MFT table is invalid, the information file is read and the *handel* is retained, and if the file failed to open, an error code is returned. The *handel* is set to the MFT table and the start sector is set to scan the files. An object of type CNTFSDrive is initialized, and if the initialization failed, an error is returned. The list of possible file addresses is browsed.

Step IV. From the NTFS object the information needed to populate the file name are read, by filling in the *CNTFSDrive::ST_FILEINFO stFInfo* structure, which contains information about the files. If the operation failed, the error code is returned, the file properties *buffer* is initialized, and the properties are added to the *string*. It is verified if the file has been deleted or not and the information is kept in the property *buffer*. The graphical interface is *updated* with the new information and the used buffers are cleaned. This file scanning process can be time consuming, and it may also consume a lot of resources.

Step V. The file is restored to the desired location. Memory is freed by deleting buffers and closing *handels* to the file. The *dll* export functions create and destroy "n" objects of *IrecuperareDate* type. Both the class that manages the metadata about each file on the hard disk (Fig.2) and the NTFS management (Fig.3), is performed by respecting certain algorithmic steps.



A *handle* to the hard disk is stored as *protected* parameters, which contains the *buffer* with information from the metadata, the size of the MFT, the current position in the table, the number of bits per cluster and the start position.

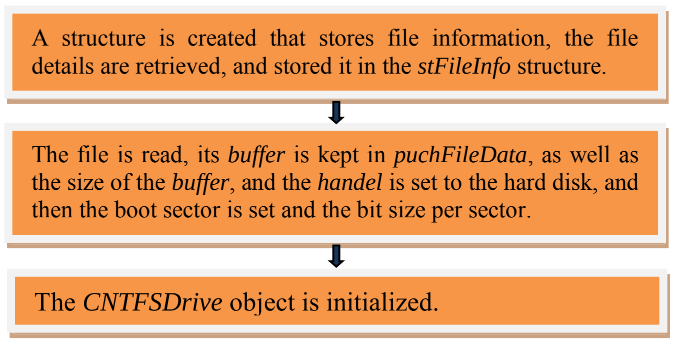
The data from the hard disk are read and saved it in the *chData* buffer, as well as the length of the *buffer* in *dwLen*

Se extrag atributele fișierelor de pe tabela MFT, care sunt salvate în *bufer*-ul primit ca parametru *puchData*, cât și lungimea *bufer*-ului în *dwDataLen*.

The details of the file to be recovered and the *handel* are set to the hard disk.

The MFT table is extracted from the hard disk, the contents being saved in the *puchMFTBuffer buffer*.

Fig. 2 MFT Management.



A structure is created that stores file information, the file details are retrieved, and stored it in the *stFileInfo* structure.

The file is read, its *buffer* is kept in *puchFileData*, as well as the size of the *buffer*, and the *handel* is set to the hard disk, and then the boot sector is set and the bit size per sector.

The *CNTFSDrive* object is initialized.

Fig. 3 NTFS Management.

The graphical interface is made using the Qt *framework*, which is a *friendly*, multi-platform, intuitive and easy to use for *users*. Creating objects is simple, by *drag and drop*, thus increasing the efficiency in obtaining complex graphical interfaces. For the implementation of the *callback* interface, necessary for *updating* the graphical interface, Qt is used. When a graphical interface object changes its status, another graphical object is announced.

The graphical object emits a signal, it is connected in another graphical object to a slot, and when it receives the signal, it executes the slot. Any graphical object must be inherited from *Qobject*. The implemented class defines 2 methods of *updating* the graphical interface, which are called from the *dll* that contains the *api*. Thus, *typedefIrecuperareDate*(*CreateRecuperareDate)(void)*, defines a function pointer that returns a pointer to the class of *IRecuperareDate* and it does not receive parameters, having only the role of exporting an object of type *IRecuperareDate* from the *dll*. Also, *typedef void (*DestroyRecuperareDate)(IRecuperareDate*mydllObject)*, defines a function pointer that does not return any value and receives as a parameter a pointer to the *IrecuperareDate* class, which has the role of destroying an object loaded from the *dll*. Since the signals and slots receive a complex data type as a parameter, it must be defined. The data type sent from the *dll* by *api* is a defined data structure and must be recorded so that the object knows the type of variable sent by the signal and captured by the slot. When one of the two methods is called, a signal is emitted containing the information that must be updated in the graphical interface. The signals are defined as calls of *signals* type.

The graphical interface is created using the *Qt Creator framework*, which is a multiplatform *tool* (can be used for development on any operating system and uses C++). The main interface is a *QmainWindow* object and is the main window of the application [18]. The graphical interface is created by defining the methods, the *view* objects used, the variables kept in the objects and the slots.

A *Qmap* is used to retain the partition data structure, which maps a certain value after a certain index. In this case, a pointer to the *drivepacket* structure is kept for the partition name. The connection of the signals and slots is done in the constructor of the *MainWindowView* object.

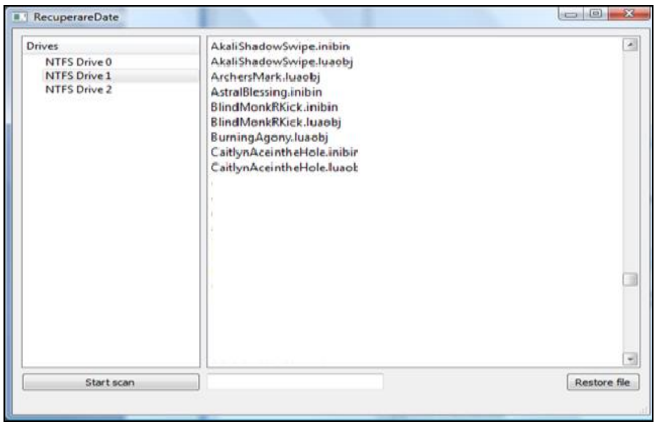
The signals emitted by the m*_pCallback* object to the *view* object, for the *update* of the *view* with new information. The signal is emitted when the scan button is *disabled* so that the *user* cannot *click*. The initialization of the graphical interface involves loading the *dll* with the functionality part, after which the functions of creating and destroying the *IRecuperareDate* object type, are loaded. The object is initialized, the *callback* is registered, and the partitions get scanned. *String DriveName* information and also a *drivepacket* data structure containing partition information, are received. The data structure is mapped to the partition name index for later use. A field is created in the *TreeView* from the graphical interface, that contains the text of the partition name and the graphical interface is updated with the new value. Information is received of type *int Pos*, position in the list, thing that is visible only in the list of items: *string* containing the file name found as deleted, *string* containing the list of file properties, *string* indicating whether the file was deleted and an *int* containing the number of occupied clusters.

Beforehand, the list is cleared of previous *items*, so as not to duplicate the information.

Thus, the graphical interface is updated with the new values. When the *Restore file* button is pressed, a *file browser* window opens, from which you choose the location where you want to *restore* the deleted file, as well as the possibility to save the file under another name. When changing a character in *search*, the list of files is cleaned and *updated* with the values that contain the text in the *search*. Scan functions can run for a long time, depending on the size of the partition. Therefore, they are run on a different *thread*. For the *default* constructor of the object, the *thread* priority is set to *idle*. To be able to capture signals in the *view* object type, the pointer is moved to *IrecuperareDate* on the execution *thread* (the pointer to the *callback* interface and the partition table). To execute the *thread,* the function of scanning the deleted files is launched.

2. Results of the application

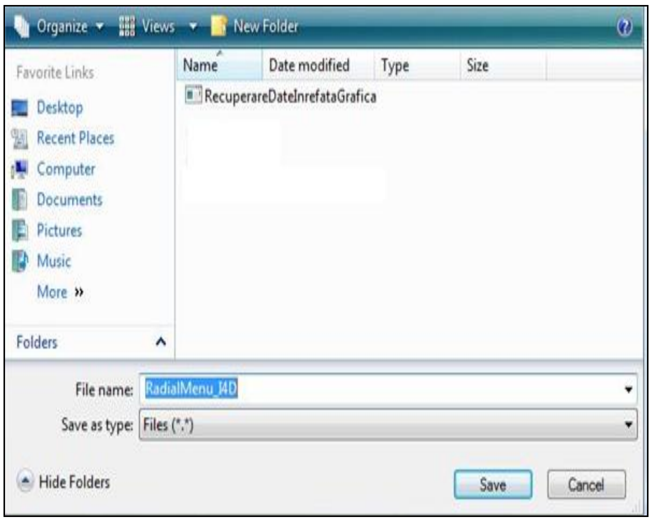
The application must be started with high rights by the administrator, to have access to the data that is necessary to run. When the application is started, the graphical interface loads the *dll* that contains the data recovery *api*. The partition list is queried and displayed (Fig. 4). On the left side of the interface is a *TreeView* object, which contains the list of partitions, and on the right, a *ListView* object, which contains the list of files on the selected partition. Below the partition list there is a start scan button for the partition (*start scan*), and below the file list there is a button to recover the selected file. Pressing the *start scan* button starts the scanning function and the list of deleted files is *updated*. At this point, a file will be selected from the list, and it can be *restored* to wherever it is to be added to the hard disk (Fig. 5). The *search* bar can be used for the easy access to the searched file.



NTFS Drive 0
NTFS Drive 1
NTFS Drive 2

AkaliShadowSwipe.inibin
AkaliShadowSwipe.luaobj
ArchersMark.luaobj
AstralBlessing.inibin
BlindMonkRKick.inibin
BlindMonkRKick.luaobj
BurningAgony.luaobj
CaitlynAceintheHole.inibir
CaitlynAceintheHole.luaok

Fig. 4 Main interface: file selection



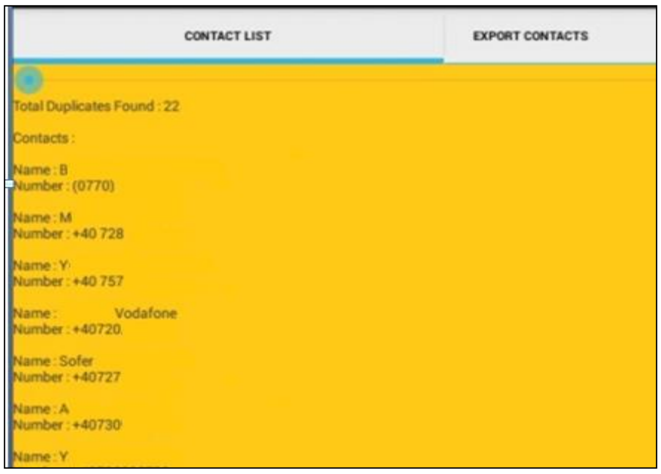
RecuperareDateInrefataGrafica

File name: RadialMenu_J4D
Save as type: Files (*.*)

Fig. 5 Interface for file restoring.

3.2 Application for the mobile phones

Depending on the degree of damage, the recovery of the contacts can be done even entirely. The degree of physical destruction of the card/phone storage will also dictate the difficulty of recovering the contacts contained. Also, there are cases where recovery can be done safely from home, and



Total Duplicates Found : 22
Contacts :
Name : B
Number : (0770)
Name : M
Number : +40 728
Name : Y
Number : +40 757
Name : Vodafone
Number : +40720
Name : Sofer
Number : +40727
Name : A
Number : +40730
Name : Y

cases where the hard drive has defective components that need to be replaced or it is severely damaged and it can no longer be used under normal conditions (in which case a possible data recovery is done only by a specialized company).

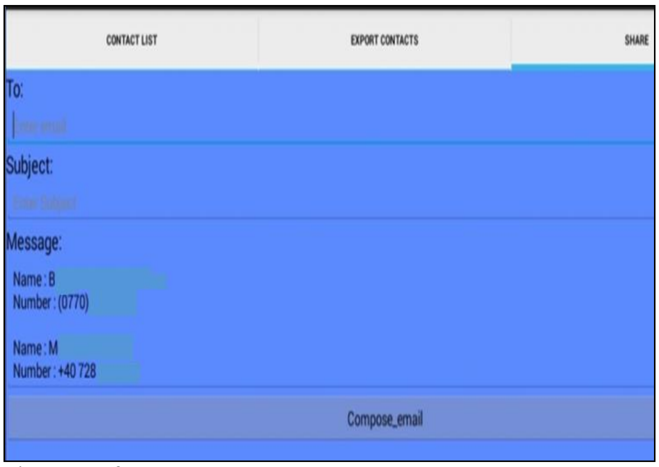
In the first phase, the *input/output* elements belonging to Java are loaded/imported, meaning the part of data writing and reading, *ArrayList*, the tables and other parameters with commands such as *import java.io.File*.

Also, through commands like *import android.provider.ContactsContract*, items for Android need to be imported, such as for the communication from external sources, importing contacts, database, filters, buttons, etc. Then, it is easier to write to an SD card [19].

The 3 tabs of the application are *Contacts*, *Export* and *Share*. The contact list is queried and displayed (Fig. 6).

Fig. 6 Main interface: contact list.

In the *Share* tab it is possible to send an e-mail with the desired contact and it is asked to enter the recipient, subject and message (Fig. 7).



To:

Subject:

Message:
Name : B
Number : (0770)

Name : M
Number : +40 728

Fig. 7 Interface: restore contacts.

4. Conclusions

The problem of recovering data from compromised information media is real and current, given the fact that most participants to the online educational process do not have newly manufactured devices.

In case of accidental deletion, on such devices, the data recovery programs presented in the introductory paragraph may be used, but they do not guarantee 100% the solution to the problem, as the content of the deleted files is not immediately removed from the storage device. In this case only the reference is removed from the folder system, and the space occupied by it is given for later reuse (being marked as empty space). For this reason, the data cannot always be recovered and rewritten. The exhaustive algorithm developed in the paper removes this disadvantage. The proposed application for PCs was tested on storage units such as external S*erial*ATA (SATA), PATA, connected via USB, eSATA, connected to the P*ersonal* C*omputer* with *Windows 7* and solves the problem of data recovery by software completion of the most common algorithm used in similar applications. The application is implemented in the C++ language and can also run on IDE/IDE-SATA hard disks with adapter.

Also, the *knowhow* for Linux partitions (*ext2, ext3, ext4* etc.) was incorporated.

Data recovery, using the developed application, has a much higher success rate if after the loss/deletion of certain files the storage space of that device is not overwritten (something else copied on the respective space sequence).

By using the application to recover contacts, there is no risk of losing a number of a contact person (colleague, teacher, secretary, etc.). If one or all contacts have been deleted by mistake, or are lost, then they can be restored by a single click.

The characteristics of the second developed application, compared to similar applications, consist in the cumulative realization of the restoration operations of all the deleted contacts from the address book and on the SIM card, as well as in the creation of a backup file for the contacts saved in the phonebook and on the SIM card.

References

[1] S.S. Demtsura, "Innovative methods, forms and technologies in the field of education" (Métodos, formas y tecnologías innovadoras en el campo de la educación), in Revista Espacios, Vol. 41 (46), Art.13, 2020, pp. 144-153.

[2] S.V. Pogorelaya and V.V. Solovyova, "Application of innovative technologies in the educational process: multimedia presentations". In Forum of young scientists, No. 5-2 (21), 2018, pp.1152-1155.

[3] W.O. Aparicio-Gómez, C.A. Aparicio-Gómez and J.F. Hernández Niño, "El aprendizaje móvil (m-learning) como herramienta formativa para la empresa", in Revista Internacional de Pedagogia e Innovacion Educativa,Vol.1(1), 2021, pp.69-102.

[4] P. Tamilarasan, S. Karthick and C.G. Anupama, "Mobile learning challenges and capabilities", in International Journal of Recent Technology and Engineering, 8(3), 2019, pp. 5358–5361, https://doi.org/10.35940/ijrte.C6883.098319

[5] A. Checko, H.L. Christiansen, Y. Yan, L. Scolari, G. Kardaras, M.S. Berger and L. Dittmann, "Cloud RAN for Mobile Networks - a Technology Overview", in IEEE Communications Surveys & Tutorials, Vol. 17(1), 2014, pp.405 – 426.

[6] L. Sri Istiyowati and P. Prati, "Mobile learning readiness in higher eduction based on the theory of planned behavior", in International Journal of Recent Technology and Engineering, 8(1C2), 2019, pp.831–836.

[7] K.A/P. Angamutu, N.A. Abd Rahman and N.N. Ain Nik Suki, "A Customized Data Recovery Tool", in Journal of Physics: Conference Series 1712, 2020, pp.1-9, https://iopscience.iop.org/article/10.1088/1742-6596/1712/1/012019/pdf

[8] H. Maryam, Q. Javaid, M.A. Shah and M. Kamran, "A Survey on Smartphones Systems for Emergency Management", in International Journal of Advanced Computer Science and Applications, Vol. 7, No. 6, 2016, pp.301-311.

[9] N. Castell, M. Kobernus, H.Y. Liu, P. Schneider, W. Lahoz, A.J. Berre and J. Noll, "Mobile technologies and services for environmental monitoring: The Citi-Sense-MOB approach", in Urban Climate, 2014, pp.1-14.

[10] A. Gholoobi and S. Stavrou, "RSS based localization using a new WKNN approach", in 7th International Conference on Computational Intelligence, Communication Systems and Networks, 2015, pp.27-30.

[11] S.J. Yang, J.H. Choi, K.B. Kim and T. Chang, "New acquisition method based on firmware update protocols for Android smartphones", in Digital Investigation, Vol.14, 2015 pp.568-576.

[12] N. Reddy, "Practical Cyber Forensics", in Linux Forensics, Springer, 2019, pp.69-100.

[13] *** QT Reference Documentation, Retrieved from http://doc.qt.nokia.com/4.7-snapshot/layout.html, accessed February 2018.

[14] *** QT Developer Network, http://qt-project.org/doc/qt-5.0/qmainwindow.html, accessed February 2018.

[15] R.K. Konoth, Victor van der Veen and H. Bos, **"**How Anywhere Computing Just Killed Your Phone-Based Two-Factor Authentication", in Financial Cryptography and Data Security, 2017, pp.405-421.

[16] *** Developer Android, Available: https://developer.android.com/reference/android/provider/ContactsContract.Contacts.html, accessed June 2020.

[17] D.P. Agrawal and Q.A. Zeng, Introduction to Wireless and Mobile Systems, Cengage Learning, Boston, 2014.

[18] A.C. Craiovan, "Data recovery application", *graduation thesis,* UVVG Arad (not published), 2013.

[19] F. Drig, *"*Android software for backup the contacts", *Student paper* - UVVG, Arad (not published), 2017.