CZECH UNIVERSITY OF LIFE SCIENCES

PRAGUE

Faculty of Economics and Management

Informatics

Department of Information Engineering

Diploma Thesis

Application Development in Object-Oriented Languages

Doina Moraru

Declaration

I declare that I have worked on my diploma thesis titled “Application Development in Object-Oriented Languages” by myself and I have used only the sources mentioned at the end of the thesis. As the author of the diploma thesis, I declare that the thesis does not break copyrights of any third person.

In Prague on ……………………..   ……………………………………………………

Doina Moraru

Acknowledgement

I would like to thank to Ing. Vojtěch Merunka, Ph.D. for giving me a good guideline for assignment throughout numerous consultations. I also thank the University of Life Sciences in Prague for consent to include copyrighted pictures as a part of my paper.

Many people, classmates, especially Ing. Petr Benda, Ph.D., who have made valuable comment suggestions on this thesis which gave me an inspiration to improve my assignment. I thank all the people for their help to complete my diploma thesis.

Vývoj Aplikací v Objektově Orientovaných Jazycích

Application Development in Object-Oriented Languages

Souhrn

Tato práce se věnuje analýze a vyhodnocení současného stavu vývoje aplikací a způsobu fakturace drobného podniku z hlediska ziskovosti a použitelnosti. Mezi metody analýzy se řadí standard softwarového inženýrství UML, technologie relačních databází na základě standardu SQL, vývoj aplikací v prostředí Builder C++. Implementace databáze a desktopové aplikace lze nalézt v přílohách. Z výsledků analýzy dat vyplývá, že vytvořená aplikace umožňuje výměnu výrobků mezi drobnými podniky, snižuje dobu zpracování informací a implementačních nákladů. Výsledky práce říkají, že vyhlídky na drobné podnikání ve své současné pozici, nejsou pozitivní. Hlavní slabiny vyžadují další zkoumání. K dispozici jsou následující doporučení:

- Zavést automatizace pro zlepšení účinnosti a snížení nákladů
- Změňte aktuální databázové aplikace, aby se minimalizoval spouštěcí čas a zvýšila integrita dat.

Práce uznává, že analýza má svá omezení. Některá tato omezení jsou: druh podnikání neposkytuje dostatek podrobností o to reálné aktivitě, výsledky jsou tedy založeny na minulých výsledcích.

Klíčová slova: vývoj aplikací, nativní aplikace, OOP, rodák grafické uživatelské rozhraní, klient relační databáze, databázový systém řízení, systém faktura, drobné podnikání, grafické OOP

Summary

This thesis provides an analysis and evaluation of current state of application development and of the invoicing process at the micro scale business from perspective of profitability and usability. Methods of analysis include software engineering standard UML, relational database technology following the SQL standard, application development in Builder C++. Implementation of database and desktop application can be found in the appendices. Results of data analysis show that the created application make possible the exchange of products between micro scale businesses, decreasing the processing time of the information and implementation costs. The thesis results say that the prospects of the micro scale business in its current position are not positive. The main areas of weakness needs further investigation. There are following recommendations:

- Introduce automation to improve efficiency and reduce costs
- Change the current database application to minimize time execution and increase data integrity.

The thesis recognized the fact that the analysis has limitations. Some of the limitations include: the type of business does not provide enough details about it real activity i.e. results are based on past performances not present.

Keywords: application developement, native application, OOP, native graphical user interface, client of a relational database, database management system, invoice system, micro scale business, graphical OOP

TABLE OF CONTENT

Introduction

Nowadays, Object Oriented Programming (OOP) is very often used to resolve tasks with a high level of complexity. To develop an application using OOP we assume to organize application code to operate with objects. In this case, the application development will focus on analyzing, processing and creating objects in concordance with task specifications.

In first two chapters are done the analysis of current state of application development and most used OOP languages around the world. The basics about where are the starting points of the modern programing languages and how does existence of databases affect development of applications.

In practical part of presented work is described the steps in elaborating of an application in Object Oriented Language utilizing alternative database management system as a supporting database for desktop application. It include the elaboration of database that will improve quality, increase efficiency and transparency control of documents and records necessary for making invoices in a micro scale businesses, which encounter difficulties with developing or extending commerce transactions with other businesses. Making possible the exchange of products, services or information between businesses, the micro scale businesses will minimize transportation cost and improve the business relationships between each other.

A particularity of the done work is highlighting the positive aspects of a system of making invoices in a micro scale businesses. This task is difficult because it requires a high-level business culture (Tax Code, standards, etc.).

Present work does not represent a finished product, because it is only a simulation tool, a prototype in order to test its usefulness. The next stage requires the development of tools and an optimization method for documents administration providing services but also producing of goods, possible usage in ecommerce (e-invoicing with the bank system), websites.

Objectives

This thesis focuses on application development in Object Oriented Language utilizing alternative database management system as a supporting database for desktop application. Concept of object oriented programming together with relational database management systems slightly dominates the development, used to fix the nowadays problems.

Goals of the thesis are:

- Overview of available literature on the topic
- Comparison of OOP and different types of programming paradigm
- To review and select the type of database to be implemented
- To review and select the type of application to be implemented
- To develop an application for better fit of data in micro scale business
- To evaluate possible usage in micro scale business of created application

The aim of the project is to analyze information about current state of the making invoice process at the certain producing company in Republic of Moldova, to develop a desktop application for better data processing. To obtain a desktop application that satisfy the client(micro scale business) needs, connected with a database which will permit easy adding, changing and removing data from database, avoiding data duplication and inconsistent records.

METHOLOGY

Methodology of the thesis is based on analysis of available information sources and knowledge bases. The practical part aims to develop an OOP application provided by Builder C++ with relational database management system MYSQL to analyze performance, recognize compatibility issues, and evaluate their possible complementarity.

To visualize the design of the system have been used the UML (Unified Modelling Language) Technology, provided by StarUML 5.0.0.1570 software (The Open Source UML/MDA Platform) designing the static structure of the system, and the relations between objects (class diagram). To draw the state diagrams, use case diagrams, activity diagrams and sequence diagrams is used Visio Professional 2013 for Microsoft Office 2013, easy to create, customize, review, and modify diagrams, by adding and removing shapes.

To manipulate with the database was used the open source package XAMP 5.6.3, which handle with PHPMyAdmin to perform different tasks on MySQL database.

To make connection to the database was used ODBC Driver for connection of Builder C++ 6 application with MySQL Database, which provide access to a MySQL database using the industry standard Open Database Connectivity (ODBC) API.

Chapter I. Application Development in Object Oriented Language. Introduction

1.1 A Brief History of Application Development

Application development knows a significant progress beginning with the 1960s, starting with Assembler code, with its low CPU[1] and storage resource requirements. After, application development focus to COBOL[2] (WALDEN, DAVID AND NICKERSON, 2012). But many of these early Assembler and COBOL programs used, made programs difficult to debug and maintain.

The evolution of database management systems (DBMSs) and data dictionaries set a new step in application development. Structured programming concepts were introduced by Edsger Dijkstra, and formally developed by IBM Fellow Dr. Harlan Mills. In the early 1970s some IT development projects solidified the benefits of structured programming especially in debugging and application maintenance (MURPHY, 2012).

The introduction of relational technology brought in 1980s a new breath in IT application development. Relational systems allow to use tables for data storage while maintaining and enforcing certain data relationships by using Structured Query Language (SQL), which can be embedded in most high-level languages. In addition, object-oriented (OO) technology also began to be accepted. New OO languages such as C++ and Java grew in popularity (MURPHY, 2012). Since the 1990s, companies have adopted these technologies for application development, and till present an imposing application migration from legacy systems to these technologies take place, and is still in process.

[1] CPU – Central Processing Unit
[2] COBOL - Common Business-Oriented Language

1.2 Programming paradigms

Nowadays, programing languages are categorized according to the approach they use to solve a problem. A Programing paradigms represent models of programing based on different concepts which predetermined how the programmers design, manage and write programs in order to solve a problem. There four major programming paradigms represented in the Figure 1.

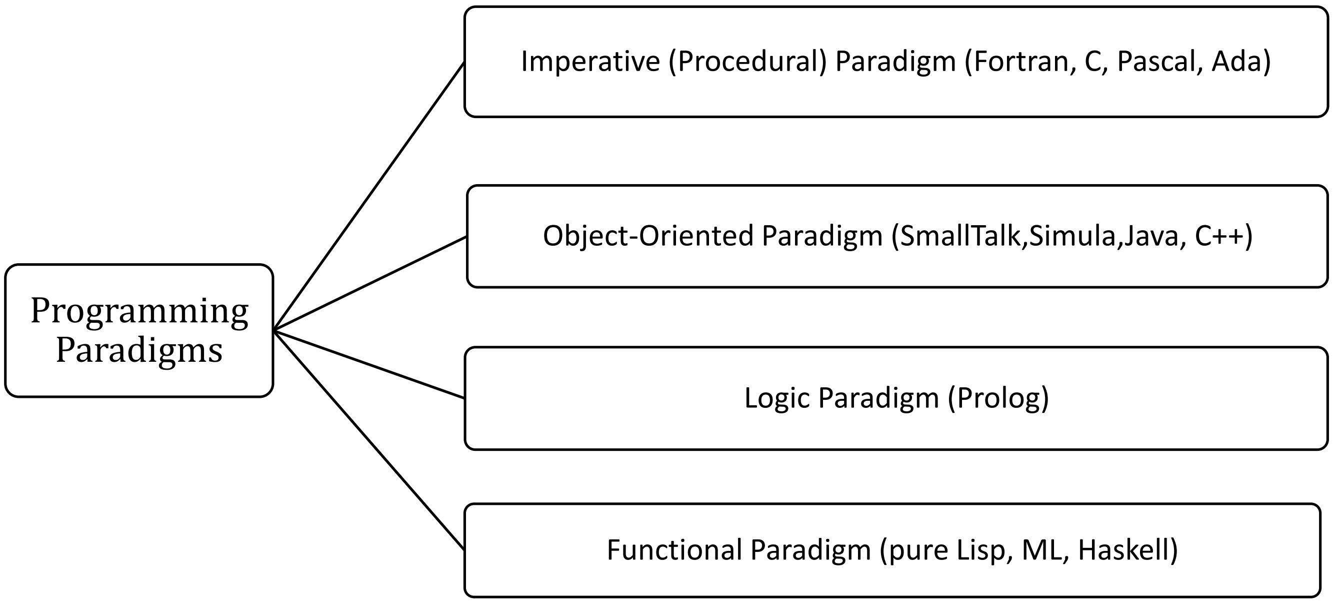

Programming Paradigms
Imperative (Procedural) Paradigm (Fortran, C, Pascal, Ada)
Object-Oriented Paradigm (SmallTalk,Simula,Java, C++)
Logic Paradigm (Prolog)
Functional Paradigm (pure Lisp, ML, Haskell)

Figure 1. Four major programming paradigm (Source: author)

A selected Programming Paradigm defines main property of a created software in concordance of programming language supporting the paradigm. In spite of that, a programing language can support more than one paradigm, these is so called multi paradigm programming languages. Indifferently which programming language and which programing paradigm support, developed application should satisfy several characteristics:

1. scalability
2. integrability/reusability
3. portability
4. performance
5. reliability
6. easy to develop

Imperative paradigm – Machine-model based, notion introduced by John von Neumann, works by changing the program state through assignment statements, the procedural abstraction and structured programming are its design techniques (PEOPLE.CS.AAU.DK, 2013). The phrase highlighted bellow represents the way how imperative programming works. Focus on incremental change of the program state as a function of time.

First do this and next do that

Functional paradigm – (Equations; Expression Evaluation) origins from mathematical discipline, focused on theory of functions. A function is a black box that marks a list of inputs to a list of outputs (PEOPLE.CS.AAU.DK, 2013). Time do not play an important role in comparison to the imperative paradigm. All computations are done by calling functions.

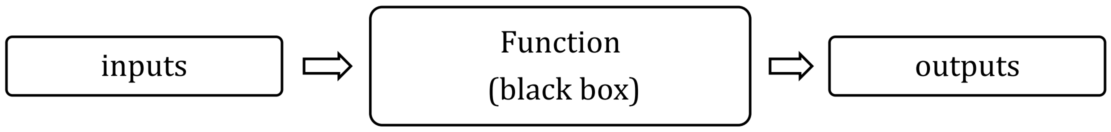


Figure 2. Functional Paradigm function (Source: author)

Logic Paradigm - First-order Logic Deduction, has a different approach then the other three paradigms, and is based on axioms, inference rules, and queries. The phrase highlighted bellow represents the main idea how logic programming works.

Answer a question via search for a solution

Object Oriented Paradigm – these paradigm knows a great popularity, data and operations are encapsulated in objects, information hiding is used to protect internal properties of an object, objects interact by means of message passing. Emphases on data abstraction, using a bottom-up design, presence of reusable libraries, suited for programming in the large.

Send messages between objects to simulate the temporal evolution of a set of real world phenomena

1.3 Introduction to Object Oriented programming

Object Oriented Programming **(OOP)** is a method of designing and implementing programs as a collection of objects, which interact with each other via messages. In OOP is defined not only the data type of a data structure, but also the types of operations that can be applied.

The OOP paradigm dates from SIMULA (Simulation of real systems) language that was developed in 1960 by researchers at the Norwegian Computing Center. Later in 1970, the research group at Xerox PARK (the Palo Alto Research Center) with Alan Kay developed the first pure OOPL: Smalltalk. In the same time, Bjarne Stroustrup at Bell Laboratories was also developing an extension to C language, called C++, which implements the OO concepts (JACOBSON, 1992). Beginning with the idea of OOP till now, are dozens of languages object oriented, like: Object C (1986), Eiffel (1988), Object Pascal (1986), JAVA (1995), etc.

The success of OOP as a programming methodology is due to how the key principle are respected in software engineering like the use of well-defined interfaces for communication between components, reuse components and the ability to define standard scheme of building components for specific purposes (PREDA, 2010). Object-oriented Programming is very popular also because of the special features like (reusability, encapsulation, and inheritance). It is very suitable for Graphic User Interface, GUI application development.

1.3.1 Basic concepts of OOP

The basic notions in OOP are:

- **Code** - sequences of computer instructions in a specified programing language.

- **Data** is representation of the perception of the real world. Any data has a degree of information. Data is considered stored in an electronic format and managed by an information system (AZUMA, VANíčEK J., 2001).
- **Class** is a template of object, it contain all information of an object data structure and code.
- **Object** is an instance of a class. Objects have state, identity and behavior.
- **Method** is the implementation of some behavior of an object.

The main concepts which stands on the base of OOP are represented in the Figure 3.

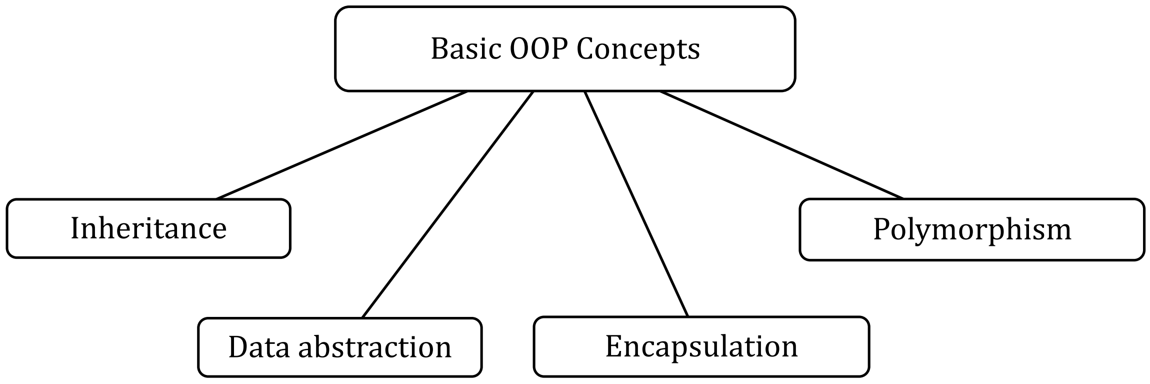


Figure 3. Basic OOP Concepts (Source: author)

Inheritance is the concept, that when a class of objects is defined, any subclass that is defined can inherit the definitions of one or more general classes (MEYER, 2013). In other words, inheritance characterize relationships among classes.

An object of the subclass is a kind of object of the superclass.

Subclasses make available for use specialized behaviors from common elements provided by the superclass. Through the use of inheritance, programmers can reuse the code the superclass many times.

Data abstraction - specifies behavior, based on the separation of interface and implementation. The user of application interface should not know how the application was implemented.

Encapsulation (called information hiding) is the process of making the detailed implementation of an object hidden from its user. Users sow objects as black boxes: knowing what operations may be requested of an object, but don't knowing how the specifics of how the operation are performed. Encapsulation hides detailed implementation of an object that can be vulnerable to malicious attack (CARDELLI, LUCA, WEGNER, 1985).

Polymorphism is a variable entity or data structure element, which will have the ability, at run time, to become attached to objects of different types, all controlled by the static declaration(MEYER, 2013) . The same operation can be defined for many different classes, and each can be implemented in their different way.

A general classification of polymorphism was introduced by Luca Cardelli and Peter Wegner according to this, there two general types of polymorphism (CARDELLI, LUCA, WEGNER, 1985), represented in the Figure 4.

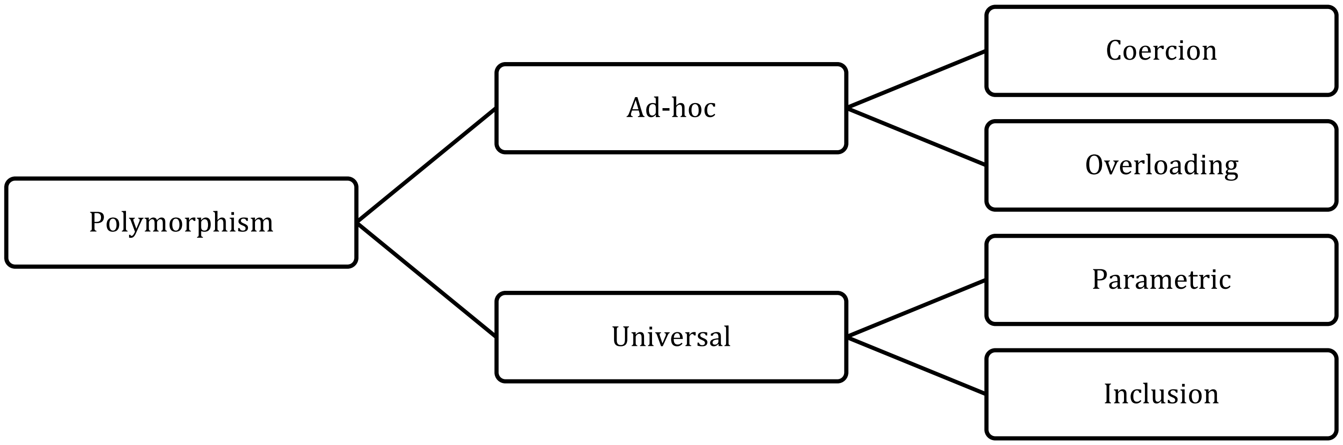


Figure 4. General Classification of Polymorphism. (Source: author)

Coercion represents implicit parameter type conversion to the type expected by a method or an operator, thereby avoiding type errors (SINTES, 2002).

Overloading afford to use the same operator or method name to denote multiple implementation of a method. Each method only differs in number and type of its parameters (SINTES, 2002).

An example of overloading in JAVA is shown below and the method max () takes two parameters and returns the maximum from those, whatever the parameters are integers or other type.

```
Public static int max(int a, int b);

Public static double max(double a, double b);
```

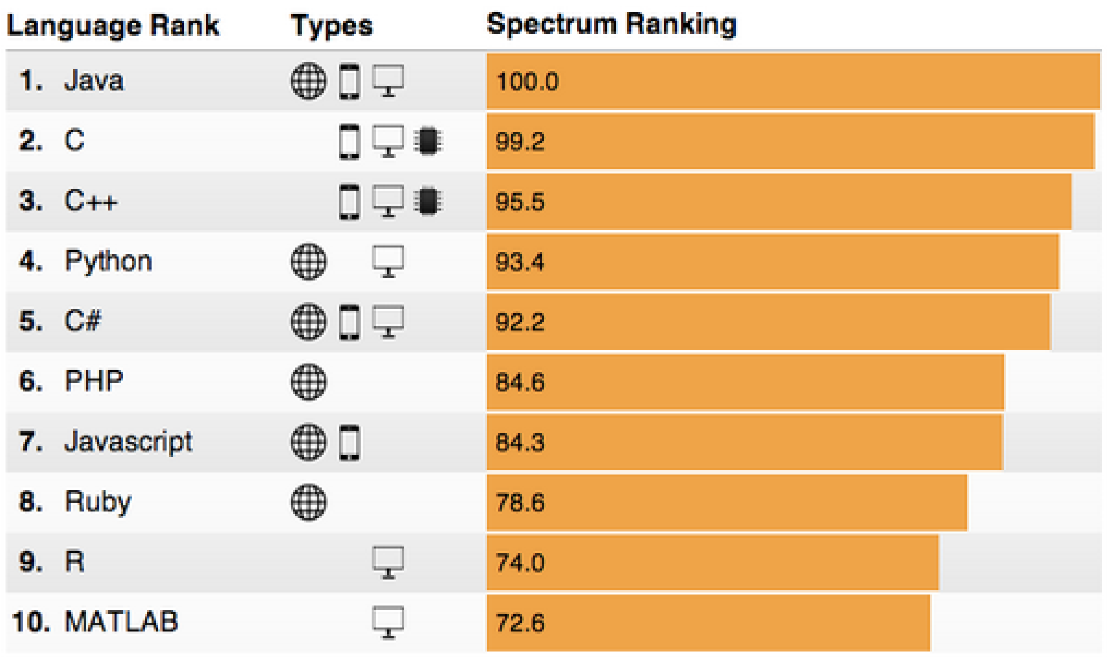
Parametric polymorphism allows the use of a single abstraction across many types. Coding something once and have it working with many different kinds of arguments.

Inclusion occurs in languages that allow subtypes and inheritance. An instance of an object of a given type can use the functions of its super type. Inclusion polymorphism achieves polymorphic behavior through an inclusion relation between types or sets of values. For many object-oriented languages the inclusion relation is a subtype relation (SINTES, 2002).

Not using the basic concepts of OPP does not mean that the software is not object oriented, but at a minimum it should be always used encapsulation. Without encapsulation it not possible to make an effective inheritance or polymorphism.

1.3.2 Object Oriented Programing Languages

Today, programming languages make things work on easiest way for human needs satisfaction. It's difficult to determine programming languages which are most popular, or most used, because particular languages are used for particular types of applications. IEEE — the world's largest professional association for the language of technology published its ranking of the popularity of programming languages on July, 2014. The ranking is based on 12 weighted factors, including Google search rankings and trends, social programming activity (GitHub and StackOverflow), job opportunities (Career Builder and Dice), social media chatter, aggregator posts (Reddit and Hacker news) and academic citations (SMITH, 2014). Looking at other object oriented languages, JAVA was ranked #1 and C++ at #3, C# ranked #5 (SMITH, 2014).



Language Rank	Types	Spectrum Ranking
1. Java	🌐 📱 🖥	100.0
2. C	📱 🖥 ▣	99.2
3. C++	📱 🖥 ▣	95.5
4. Python	🌐 🖥	93.4
5. C#	🌐 📱 🖥	92.2
6. PHP	🌐	84.6
7. Javascript	🌐 📱	84.3
8. Ruby	🌐	78.6
9. R	🖥	74.0
10. MATLAB	🖥	72.6

Figure 5. Ranking of the popularity of programming languages. (Source http://revolution-computing.typepad.com/.a/6a010534b1db25970b01a511df7b8e970c-pi)

1.4 Application Type Considerations

Application is a shorter form of application program. An application program is a program designed to perform directly the specific tasks for the user or, in some cases, for another application program. There are several general categories of applications used today, in the Figure 6 are represented the classification.

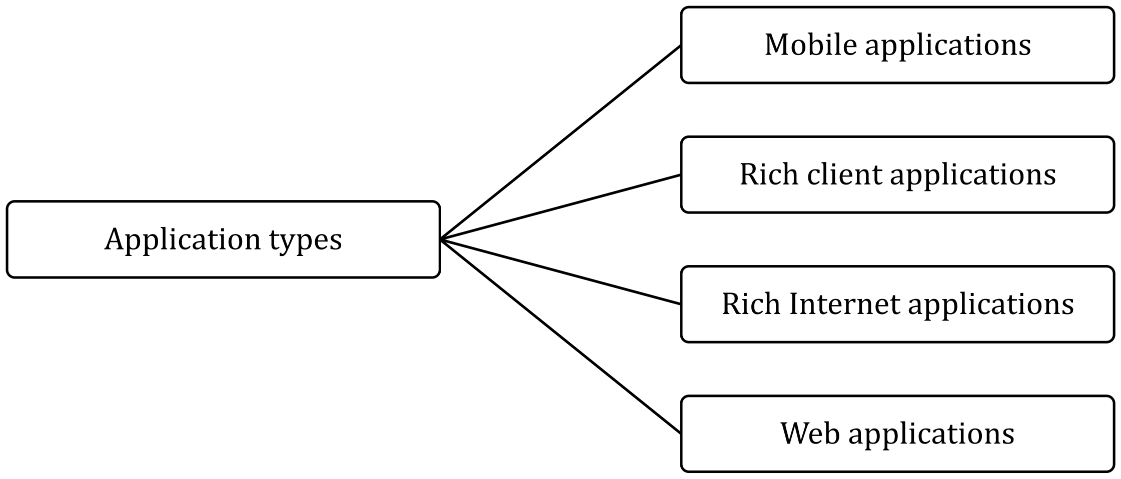


Figure 6. Application types (Source: author)

Mobile applications – programs designed to run on smartphones, tablet computers and other mobile devices. When is designed a mobile application, the device resources may prove to be a constraint. Mobile devices represents support for handled devices. On other hand, mobile applications have navigation limitations and limited screen area (MICROSOFT, 2009).

Rich client applications (RCA) – is client–server applications with a graphical user interface used to solve tasks with high level of complexity. One part of the task could be implemented on the client side, and another one on the server side. This type of applications permit disconnected and occasionally connected scenarios if is need to access remote data or functionality (MICROSOFT, 2009). Rich client application has rich UI functionality, and improved user experience. Support offline and occasionally connected scenarios. From other side, the RCA involve a complex development, also it is dependent on the platform, need to be updated over the time (MICROSOFT, 2009). Typical Rich Client architecture is represented in the figure 7. In the figure billow are represented detailed architecture, but in real life not all components are taken in consideration, this depends on the complexity of the application needed to be implemented.

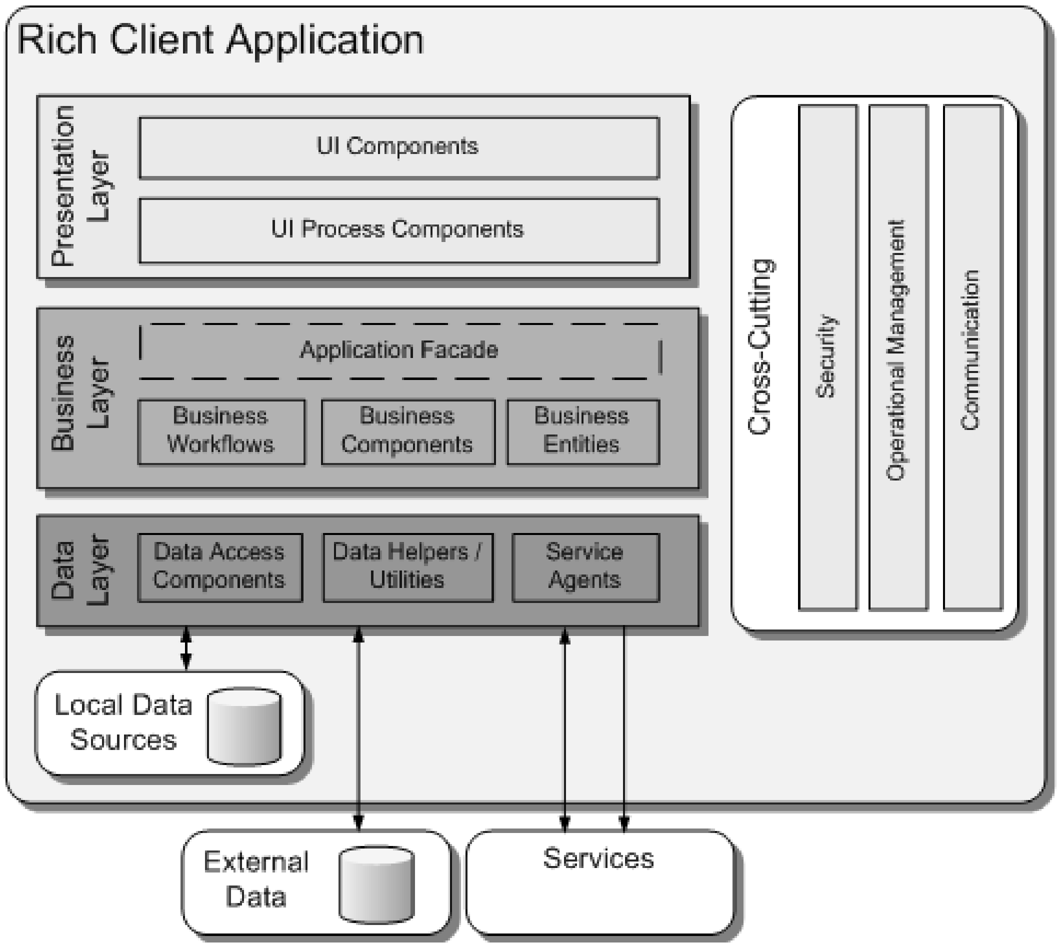


Figure 7. Typical Rich Client Architecture (Source: https://msdn.microsoft.com/en-us/library/ee658087.aspx)

Rich Internet applications – developed to support multiple platforms and multiple browsers, displaying rich media or graphical content (MICROSOFT, 2009). This type of application supports streaming media and graphical display and simple to develop. But requires a suitable runtime framework on the client.

Web applications – support connected scenarios and can support different browsers running on a range of operating systems and platforms (MICROSOFT, 2009). Dependent on network connectivity and do not provide a rich user interface.

Chapter II. Implementation of Database Management System

Today databases are used in every part of day-to-day life. Each company tend to extend amounts of information about business transactions, and depends on a file system (generally a database system) that allows for the access of specific information in a timely and cost effective manner. In following sections are described DBMS[3] and it components.

2.1 Introduction and Definitions

Today companies use huge amount of information about clients, personnel and business operations. This information is usually stored in files, records and fields.

A **file** is a collection of information relative to a number of related entities. For example, a company needs to collect various types of information about clients, personnel, sells, etc. Each one of these is a "sub package" of information, stored as a file. They may have 3 files, and Personnel File, a Clients File and invoice file which contain information about sells.

A **record** represents a single, implicitly structured data item in a table. If we look at the example above, the Clients File record contains information required by the company about each client, and stored for each client. For example, information contained in a client record may include at least first name, last name, id number, birth date. Each record in the file contain the same number and types of fields, the records have the same type.

A **field** is contained in a record. In the above example, the 4 pieces of information, first name, last name, id number and birth date are all fields.

Data value is the atomic content of an attribute. Is a qualitative or quantitative variable which describe a record.

2.2 Database Management System

Database Management System (DBMS) is a set of programs that permits users to manage a database structure. DBMS is focused on defining, designing, sharing and managing databases through different users and applications.

[3] **DBMS** – Database Management System

Today is possible to use a huge number of DBMSs to implement a database, if is needed one, so if is presented multiple processes (users/servers) modifying the data, then the database serves to prevent them from overwriting each other's changes, helps to structure data, when data is larger than memory. Nowadays with the memory we have available, this does indeed makes the use of databases in many applications out of date.

Functions of DBMS[4]:

1. **Defining –** specifying the data types, structures and constraints for the data to be stored in the database (SCHWARTZ, ZAITSEV, TKACHENKO, 2012).
2. **Constructing –** the process of storing the data itself on some storage medium that is controlled by the DBMS.
3. **Manipulating –** includes functions as querying the database to retrieve specific data, updating the database to reflect changes, and generating reports from the data.
4. **Sharing –** allows multiple users and programs to access the database concurrently.

Other functions include protecting the database and maintaining it over a long period of time:

1. **Protection –** includes system protection against software/hardware malfunction and security protection against unauthorized/malicious access.
2. **Maintaining –** the DBMS should allow the database to evolve as requirements change over time.

The following Figure 8 show the general database system overview.

[4] **DBMS** – Database Management System

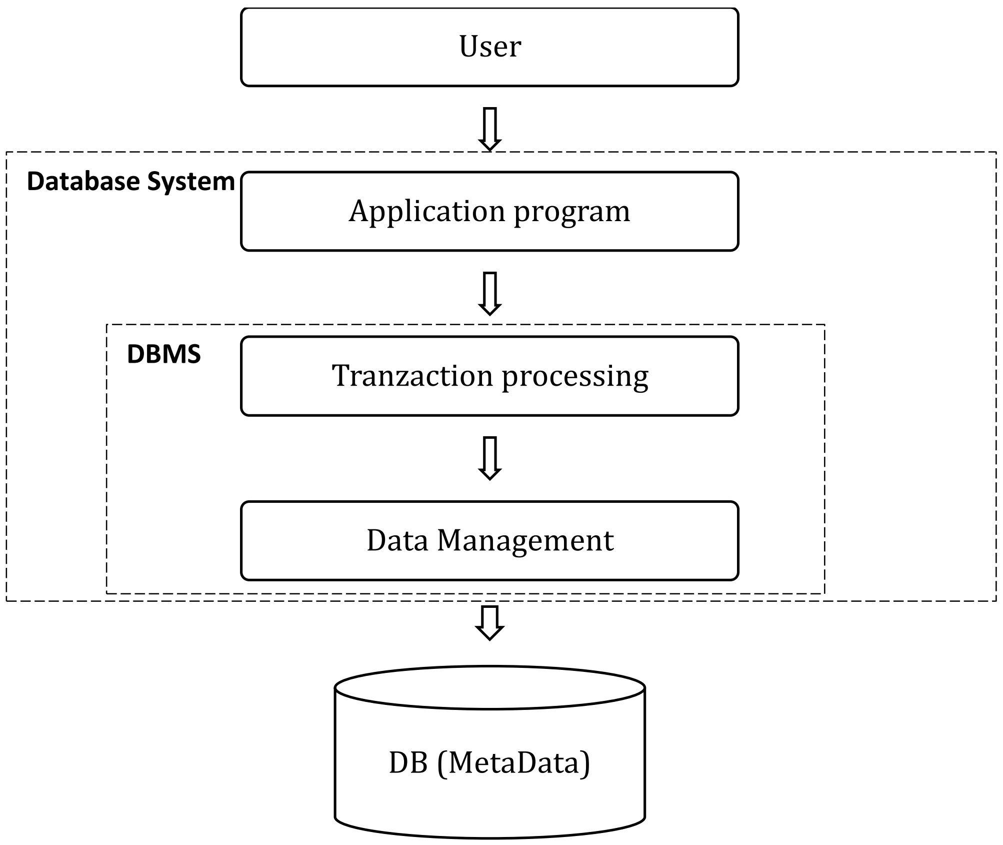


Figure 8. Overview of Database System (Source: author)

The DBMS administrates incoming data, processes it, and make it available for use, to be viewed, modified or extracted by users or other programs.

Besides, here are so many different DBMSs available, that permits to use one most suitable for needed solution for specific task.

Most important is to be a way for them to communicate with each other. In this case, most DBMS comes with an Open Database Connectivity (ODBC) driver that allows the database to communicate with other databases. For example, common SQL statements such as CREATE, SELECT, INSERT are recognized by other Database Management Systems.

Some DBMS examples include MySQL, Oracle, Microsoft Access, SQL Server, FoxPro and others.

To manipulate with the database I is used the open source package XAMP 5.6.3, which handle with PHPMyAdmin to perform different tasks on MySQL database, as a relational database management system implemented on operating systems Windows. This system can be used free of charge and is open source. This system is compatible with SQL standards, which make MySQL easy to use.

MySQL permits to set access rights to allow some or all privileges to individuals. Passwords are encrypted. Can handle almost any amount of data, up to approximately 50 million rows or more.

The default file size limit is about 4 GB. However, it is possible to increase this number to a theoretical limit of 8 TB of data (SCHWARTZ, ZAITSEV, TKACHENKO, 2012). It supports several development interfaces: which include ODBC, JDBC, and scripting (PHP and Perl), allowing creating database solutions that run on all major platforms, including Linux, UNIX, and Windows.

2.3 Implementing database "Invoice"

This section is based on a graphical representation of the design done by class diagram, state diagrams, use case diagram, activity diagrams and sequence diagrams. The technology used for design of the models is UML (Unified Modelling Language) and for creating diagrams is used StarUML 5.0.0.1570 software (The Open Source UML/MDA Platform). Each of diagrams has its own chapter and it's provided with short description of basic information about it.

2.3.1 Conceptual model. Class Diagram

The static structure of the system, with it's classes and their attributes, operations (or methods), and the relationships among objects, keep track of which invoices are processed from clients, and which contracts represent the agreement between the client and company, that still needs to be created, modified or updated.

The created system involves with next types of users presented in the Figure bellow. A user take on different roles while interacting with the system. There are next roles:

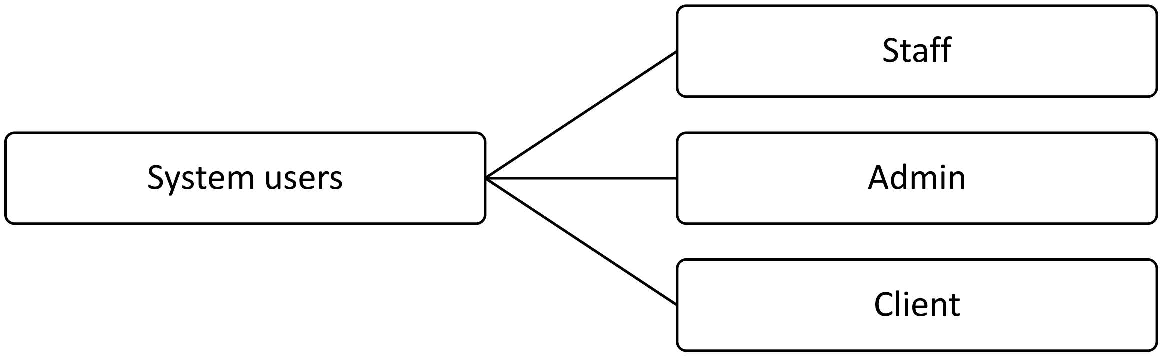


Figure 9. System Users (Source: author).

Staff role – plays a vital role in the invoice system. Staff have to control the specific entities of the database. The staff itself generates the report of the sold products. However, not the all company staff have access to the invoice system. The access rights to the system have only authorized users who must be employed or be the company staff. The staff is the only user thet can manage invoices, check the contract preconditions, post conditions and invariances.

Adim role – deals with all necessary changes which can appear in the functionality of the created system, also the admin manage staff who will work with the invoices directly.

Client role – describes the potential clients who will buy the company product. Any commercial enterprise especially small backeries, bars will be able to buy it like a clients.

Class diagram is used to maintain information concerning staff and clients in a invoice system. It describes the types of objects in the system and the various kinds of static relationships that exist among them (Figure 10).

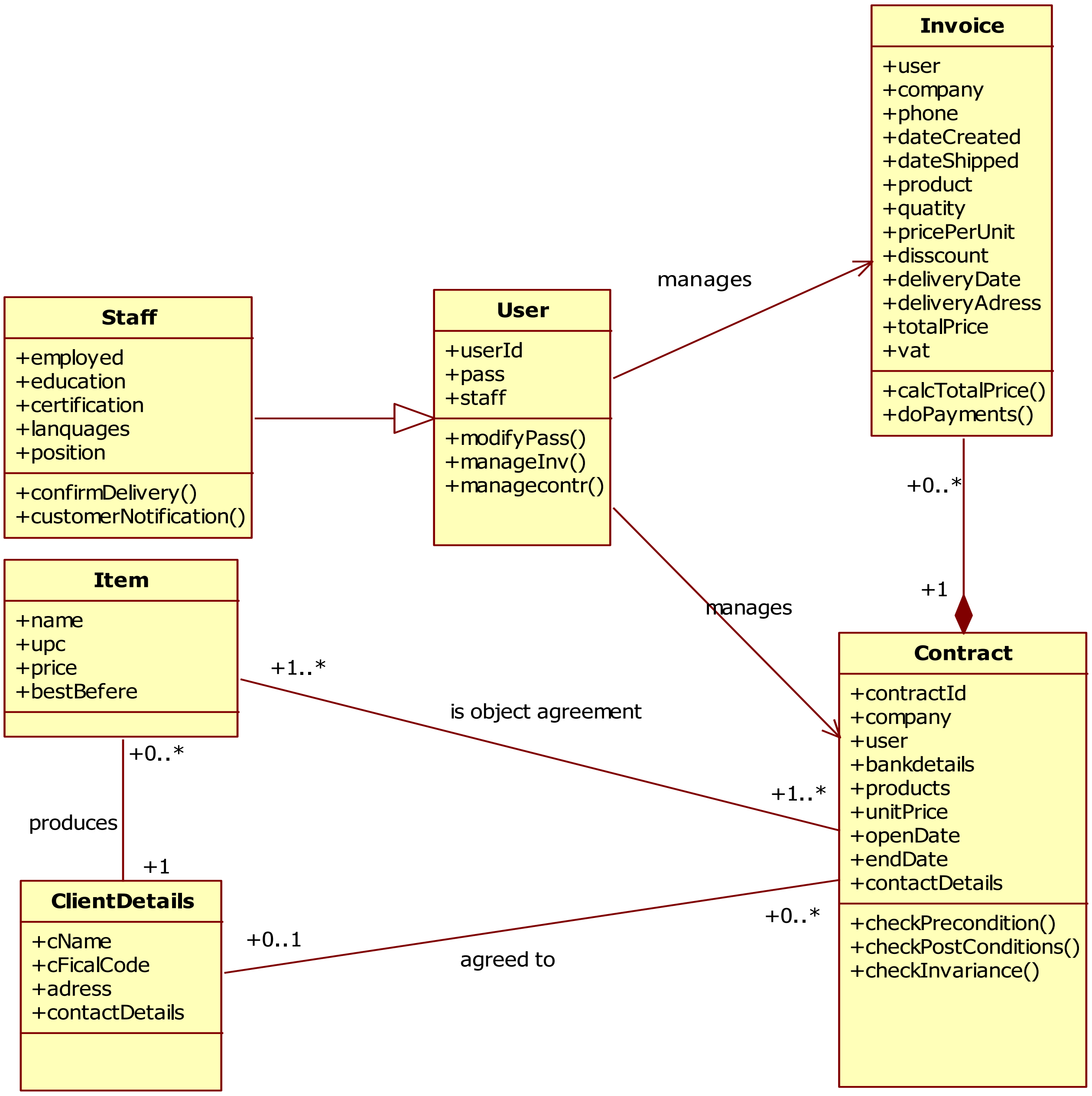


Figure 10. Class diagram. Types of objects in the invoice system (Source: author).

In the table below is the short description of the classes present in the class diagram.

Table 1. Description of the Classes presented in the Class Diagram (Source: author)

Class name	**User**	**It is an inherited class defining basic attributes information of the system administrator.**
Attributes	userID	This attribute contains the nicknames of the user.
	pass	We need this attribute in case of personal information security.
Operation	modifyPass ()	Method used to change the account password.
	manageinv()	Method used to manage the invoices.
	managecontr()	Method used to manage the contracts.
Class name	**Invoice**	**Characterize the attributes that describes the invoice.**
Attributes	user	Represents the person who will use the system.
	company	Client information to whom is shipped the invoice.
	phone	Indicates the phone number of the client with which we dill.
	dateCreated	In this attribute is the date of invoice creation
	dateShipped	In this attribute is the date of invoices shipping.
	product	In this attribute is the information about product which a client what to bay.
	quantity	In this attribute is specified the quantity of the product which a client what to bay.
	pricePerUnit	Represents the price for one unit of our product.
	discount	Indicates the discount percentage per one unit of product.
	deliveryDate	Date when the client what to take the bayed product.

	deliveryAddress	Indicates the delivery address.
	totalPrice	Indicates the total sum of the invoice.
Operations	calcTotalPrice()	Method used to calculate the total cost of the bayed products.
	doPayments()	Method used to make and verified if the payments have been done.
Class name	**Contract**	**This class characterize the agreement between our client and more parties for baying our product.**
Attributes	contractID	This attribute indicates the contract identification number.
	company	Client information to whom is shipped the invoice.
	user	Represents the person who will use the system in case of managing contracts.
	bankdetails	Contains the all bank information necessary for the payments.
	products	In this attribute is the information about product which a client what to bay. The subject of the contract.
	unitPrice	Represents the price for one unit of our product agreed for a period.
	opendate	Indicates the date from which the contract is valid.
	endDate	Indicates the date to which the contract is valid.
	contactDetails	Indicates the contact details of the client with which we dill. (Phone, fax, email).
Operation	checkPreconditions()	Method used to check if the preconditions where respected.

	checkPostConditions()	Method used to check if the pros conditions where respected.
	checkInvarience()	Method used to check if there are some invariance.
Class name	**ClientDetails**	**This class characterize the company with which we work.**
Attributes	cName	Specifies the company name details
	cFiscalCode	Specifies the public identification number.
	address	Indicates the company address
	contactDetails	Indicates the contact details of the company with which we dill. (Phone, fax, email).
Class name	**Item**	**This class describes item details**
Attributes	name	Specifies the product name details
	upc	Describes the Universal Product Code (UPC) that represents the barcode of the products.
	price	Contains price for one unit of our product agreed for a period.
	bestbefore	Indicates the date to which the product is valid to use.

In concordance with class diagram where specified next constrains for the database to be created:

1. The NOT NULL constraint which obligates the user to NOT introduce NULL values for the invoice system objects, especially until the all details of the invoices are not completed, in concordance with the contracts, the invoice status is "pending".
2. Each attribute has unique identification to exclude repetition, and once the invoice was completed and saved the identification number will not change or repeated.

3. The invoice cannot be managed until the contract is not agreed.
4. Only the users have access to the invoices and contracts, not the whole staff.
5. Admin cannot manage invoices, or other data stored in the database, admin takes care only on the expected functionality of the system and register staff to access database.
6. Into numeric columns is not possible to introduce string values. For example in the amount field is not possible to write “one”, only “1”.

2.3.2 Dynamic modeling

In this section is represented the dynamic model of the system, which consists from many state diagrams, one for main classes.

The first state diagram is describing the process of creating an invoice (Figure 11).

Creating invoice process steps:

1. First is needed to log into the system, to make possible to display the application Invoice form.
2. An empty invoice is created and data needs to be set.
3. After selecting items and quantity to be sold, the price is calculated and then stores to the price attribute of the invoice.
4. The prices are calculated with and without VAT and automatically saved into the specified fields.
5. Next step is on the client because the system is waiting for his confirmation.

There is a specific agreement between the client and company. If the agreement conditions is not respected then the invoice is set as cancelled. If the agreement conditions is respected then invoice is complete and the items are delivered to client (Figure 11).

To draw the diagrams is used Visio Professional 2013 for Microsoft Office 2013. To create, customize, review, and modify diagrams, with improved support for common activities (such as adding and removing shapes) (MICROSOFT, 2015).

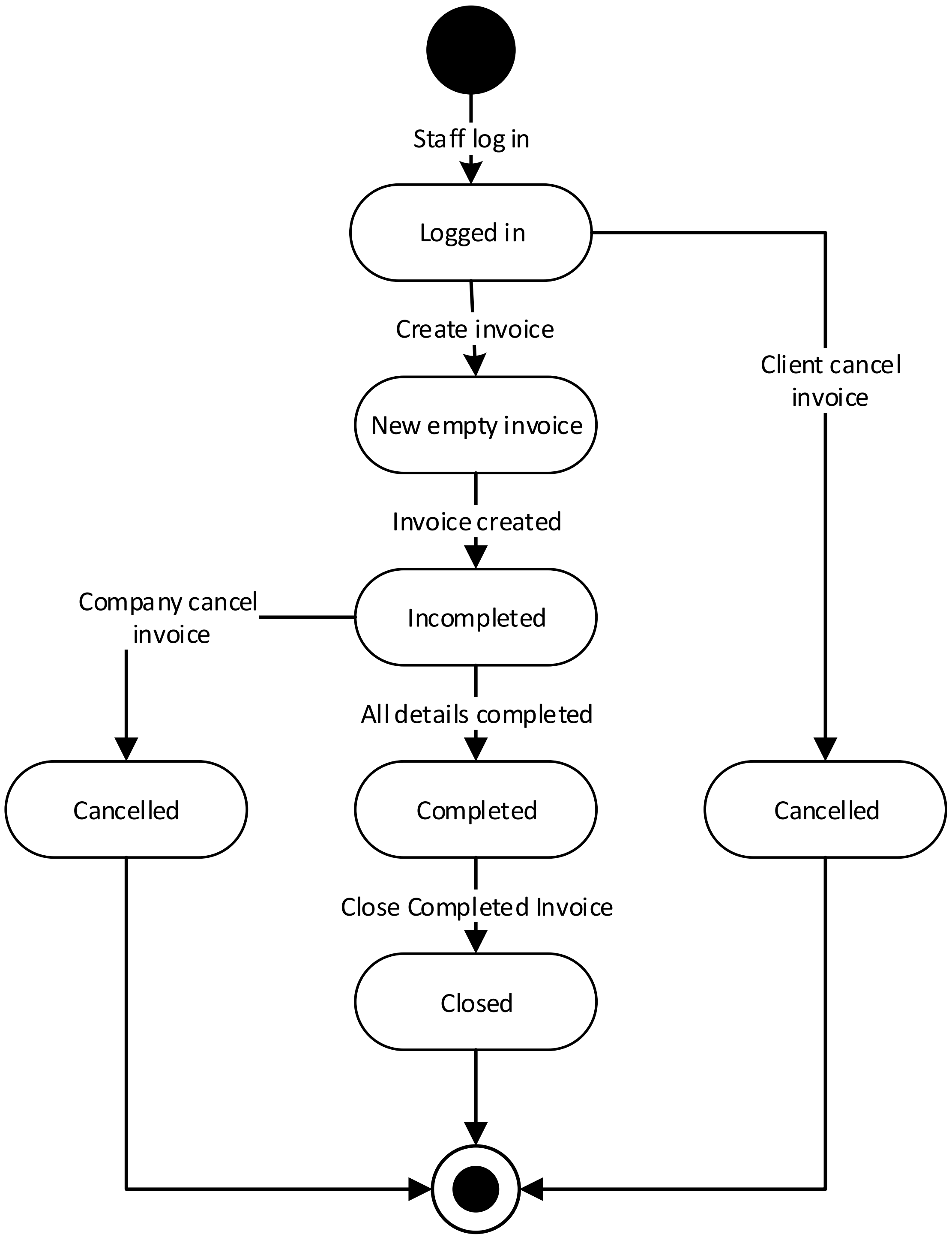


Figure 11. State diagram. Process of creating an Invoice (Source: author).

The second state diagram is describing the process of drawing up the contact between company and client (Figure 12).

Contract agreement process steps:

1. To create a new contract, the company staff should be logged into the system.
2. First, an empty contract is created and data needs to be set.
3. Setting agreement priorities regarding type of products to be purchase, price, delivery, and other details.
4. Contract is completed and company will complies with all conditions.

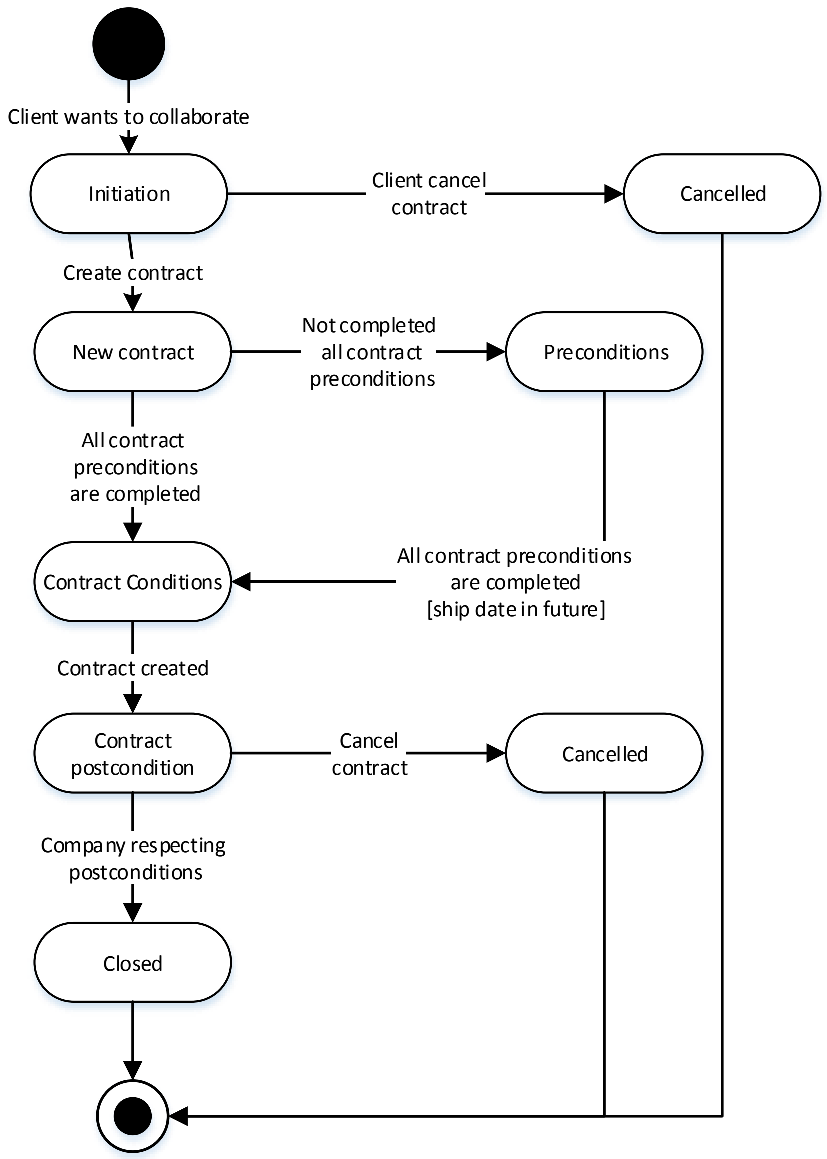

Client wants to collaborate
Initiation
Client cancel contract
Cancelled
Create contract
New contract
Not completed all contract preconditions
Preconditions
All contract preconditions are completed
Contract Conditions
All contract preconditions are completed [ship date in future]
Contract created
Contract postcondition
Cancel contract
Cancelled
Company respecting postconditions
Closed

Figure 12. State diagram contract (Source: author).

2.3.3 Interactions modeling

Interaction model represents the mode of how objects act to each other. Created diagrams shows the integral view over many objects. In this section is defined two use case diagrams, two sequence diagrams and one activity diagram.

Use case diagram describes interactions of a system with its environment. First use case diagram is "Invoice". Actors are Client and Company staff. In Figure 13 is shown the use case diagram "Invoice". Clients and company staff work together to create, edit and delete invoices and invoices components.

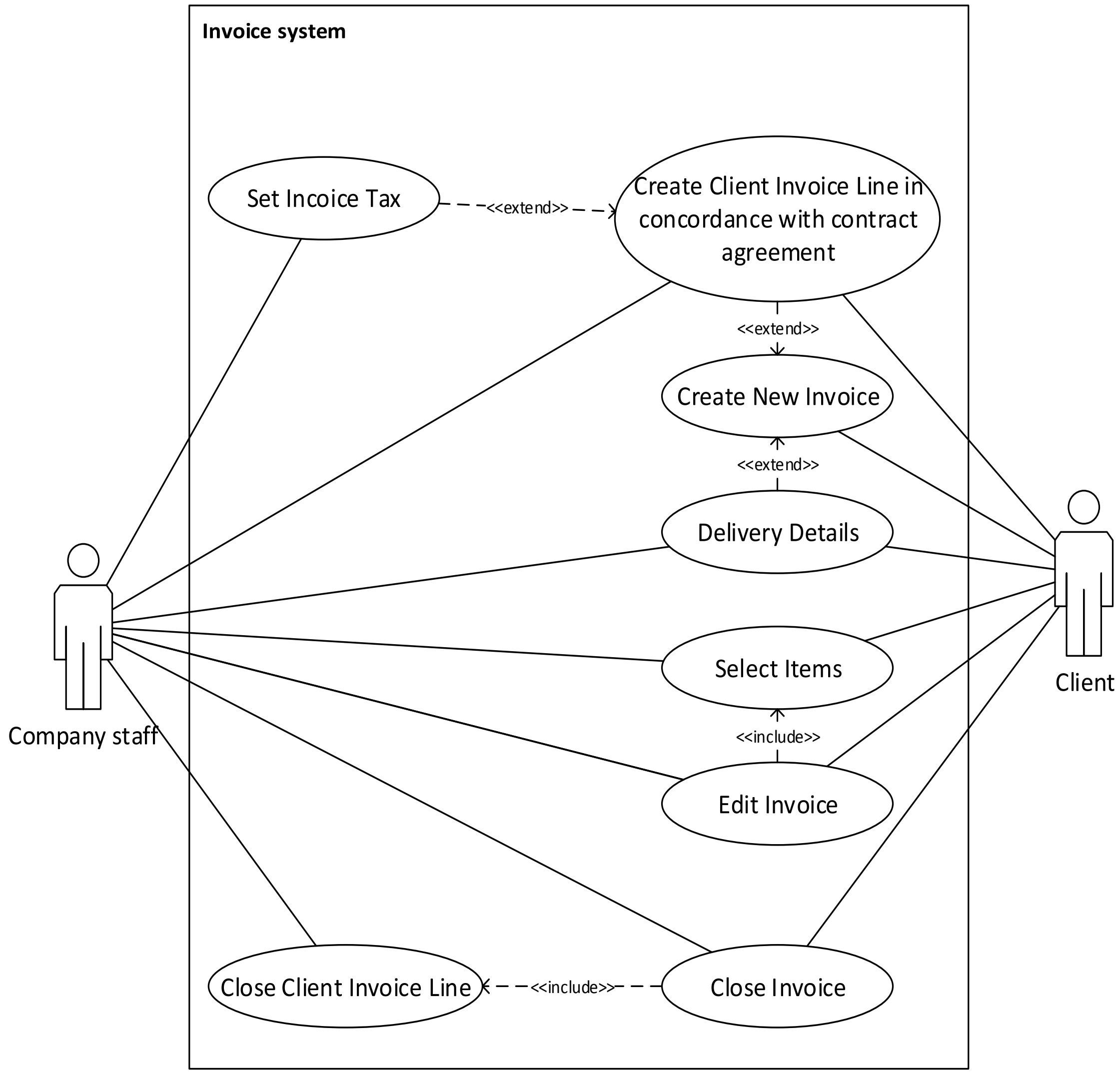


Figure 13. Use case diagram Invoice (Source: author).

The second diagram is use case diagram “New contract”. Actors are Client and Company staff. Functional requirements (Figure 14) are:

1. Client specifies the details for contract with the company.
2. Company staff check the information which client gives.
3. The perfected contract is confirmed and sign be client.
4. Company staff create introduce contract details to system.

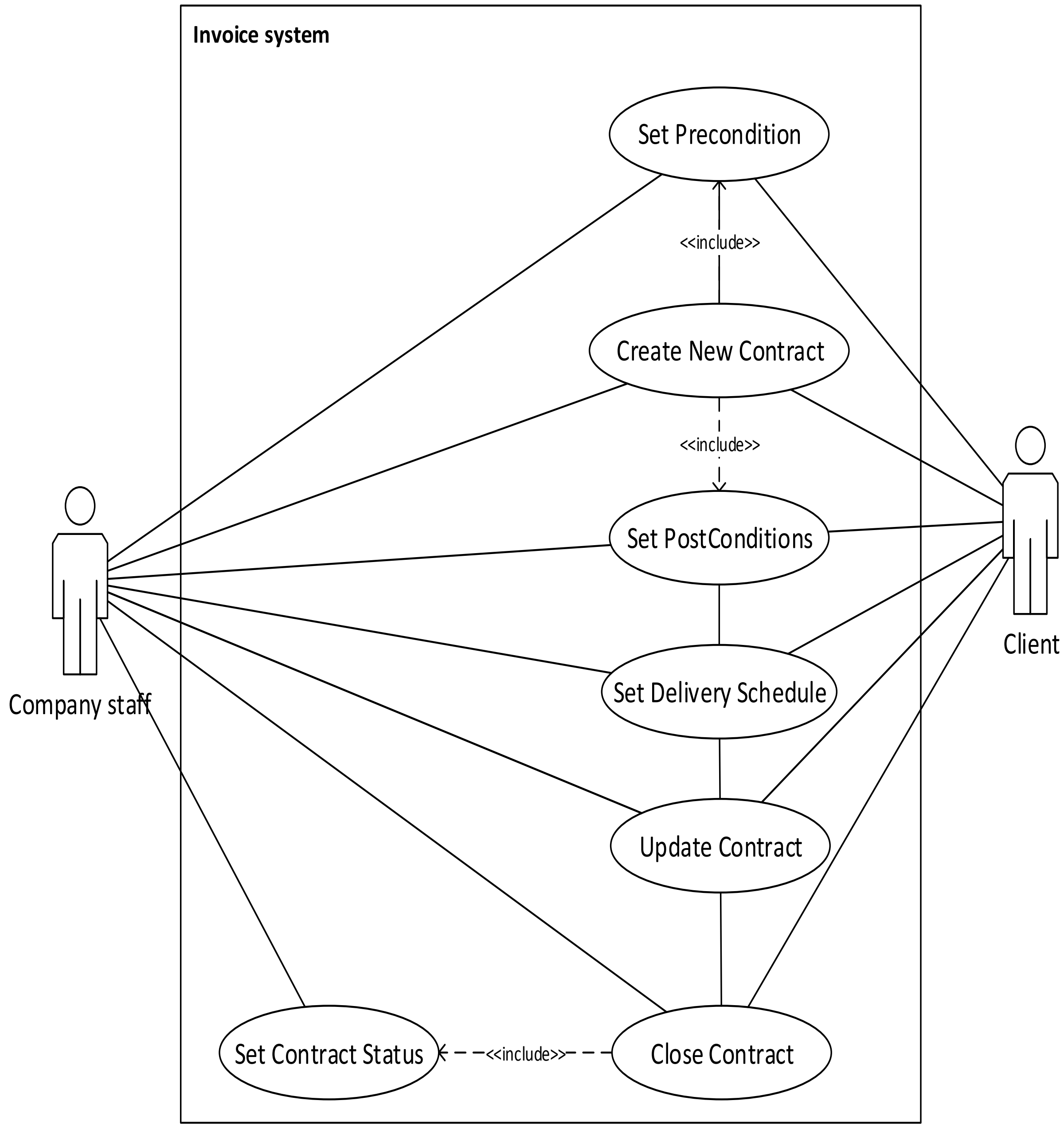


Figure 14. Use case diagram “New contract” (Source: author)

2.3.4 Sequence diagrams

Sequence diagram describes an exchange of messages in time within a set of objects. It is suitable to describe sequences of behavior from a perspective of a system user. This type of sequence diagram gives the possibility to represent the majority of objects state modified in time, as usually objects are passive and they are activated only by calling. They return control back to a calling object after completing their operations and become inactive again.

The first sequence diagram (Figure 15) shows how user (company staff) login into the system by interacting with various classes. The user will enter his/her login & password and then will log into account. The system gets login and user details from user class instance and verifies the login and password. If user details are invalid then user will not be allowed to login and an appropriate error message will be displayed, otherwise the user is logged in and the work session will be initiate.

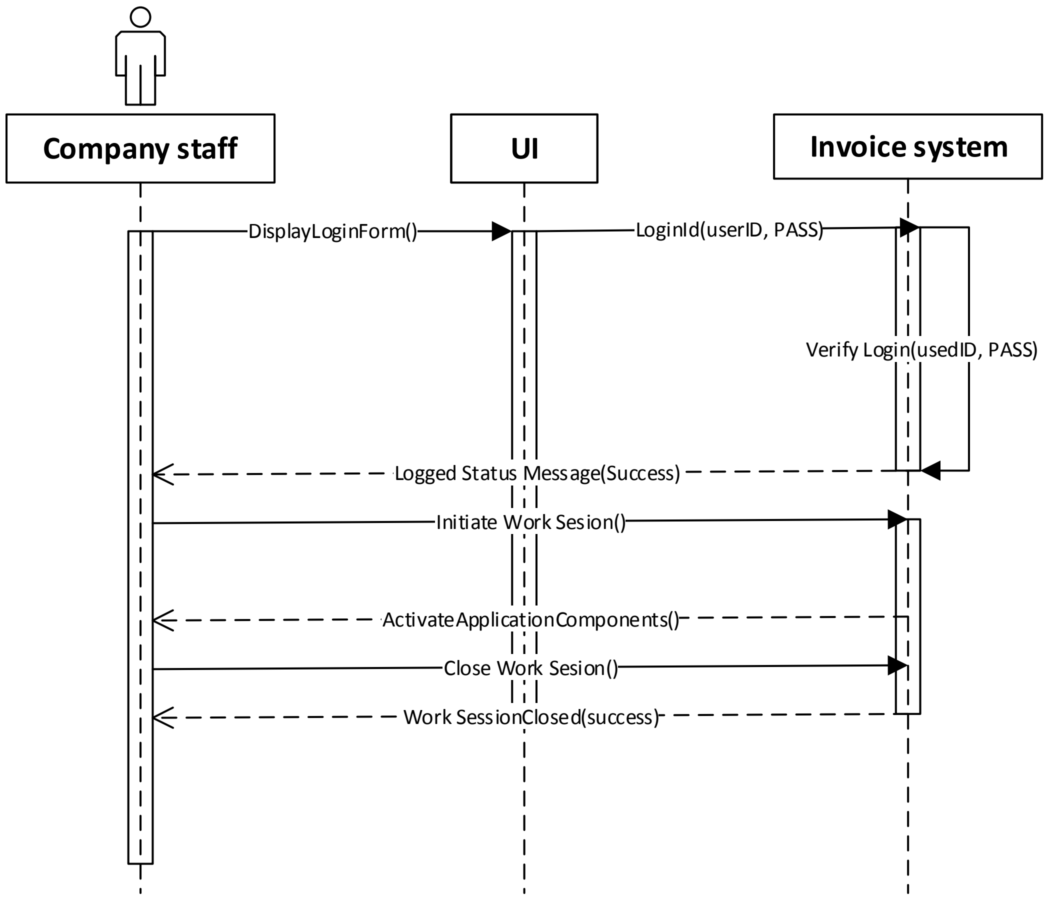

Company staff
UI
Invoice system
DisplayLoginForm()
LoginId(userID, PASS)
Verify Login(usedID, PASS)
Logged Status Message(Success)
Initiate Work Sesion()
ActivateApplicationComponents()
Close Work Sesion()
Work SessionClosed(success)

Figure 15. Sequence diagram Login In (Source: author)

The second sequence diagram (Figure 16) shows a successful invoice transaction between client and company.

Specifications: The client will select the needed item, after the company staff will create invoice and insert all necessary details about transaction. Some of the methods, like calcPrice which is called to calculate the total amount after all the products are selected, will automate the business operations. When a client checks out a product, the prices of all products selected are calculated, including the shipping charges. Then is placed Invoice confirmation message and it is sent to the client.

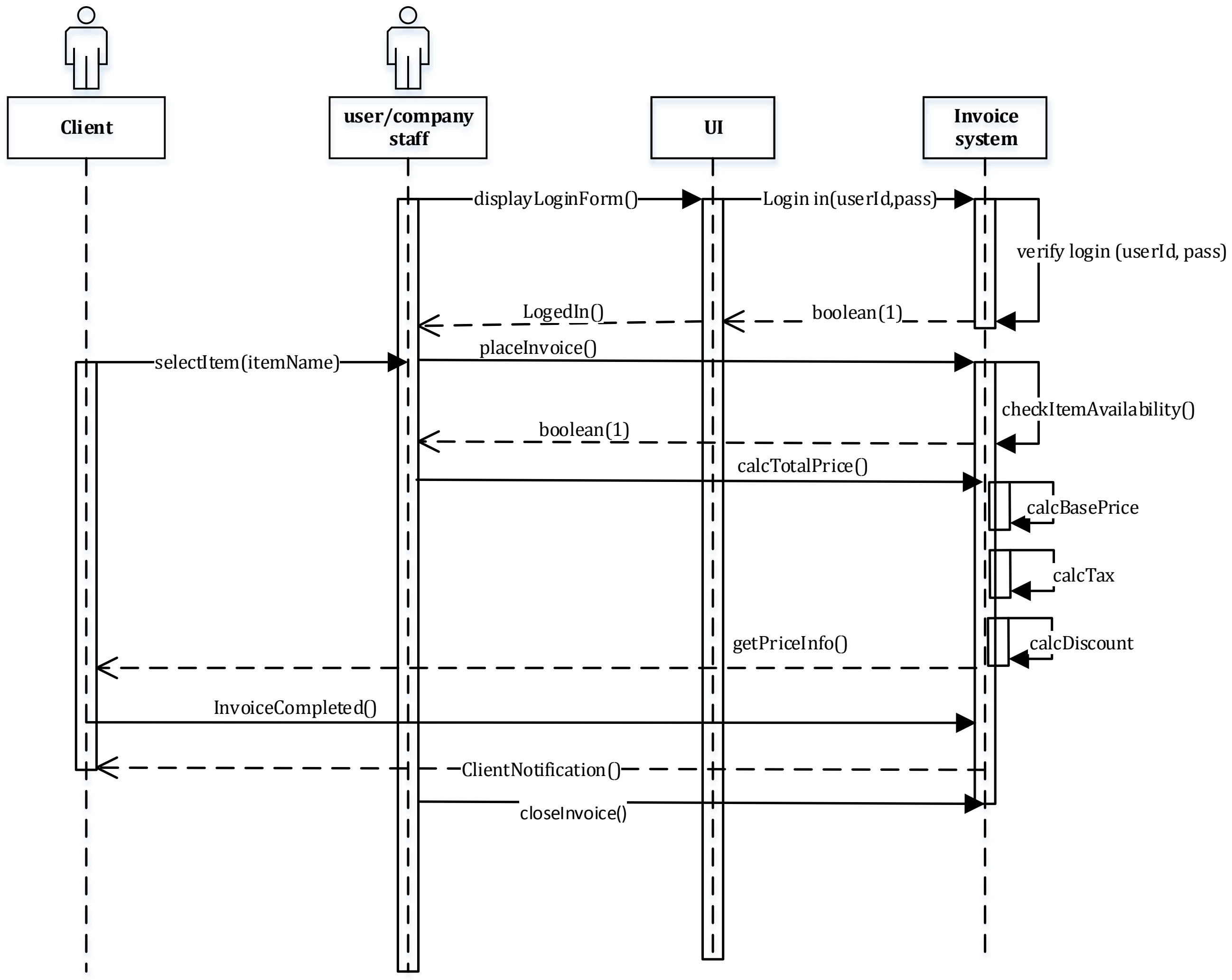


Figure 16. Sequence diagram Invoice (Source: author)

2.3.5 Activity diagram

An activity diagram (Figure 17) is used to display the sequence of activities. Activity diagrams show the workflow from a start point to the finish point detailing the many decision paths that exist in the progression of events contained in the activity. They may be used to detail situations where parallel processing may occur in the execution of some activities (OBJECT MANAGEMENT GROUP, 2015). Example of an activity diagram is shown below.

The diagram shows the “New Invoice”.

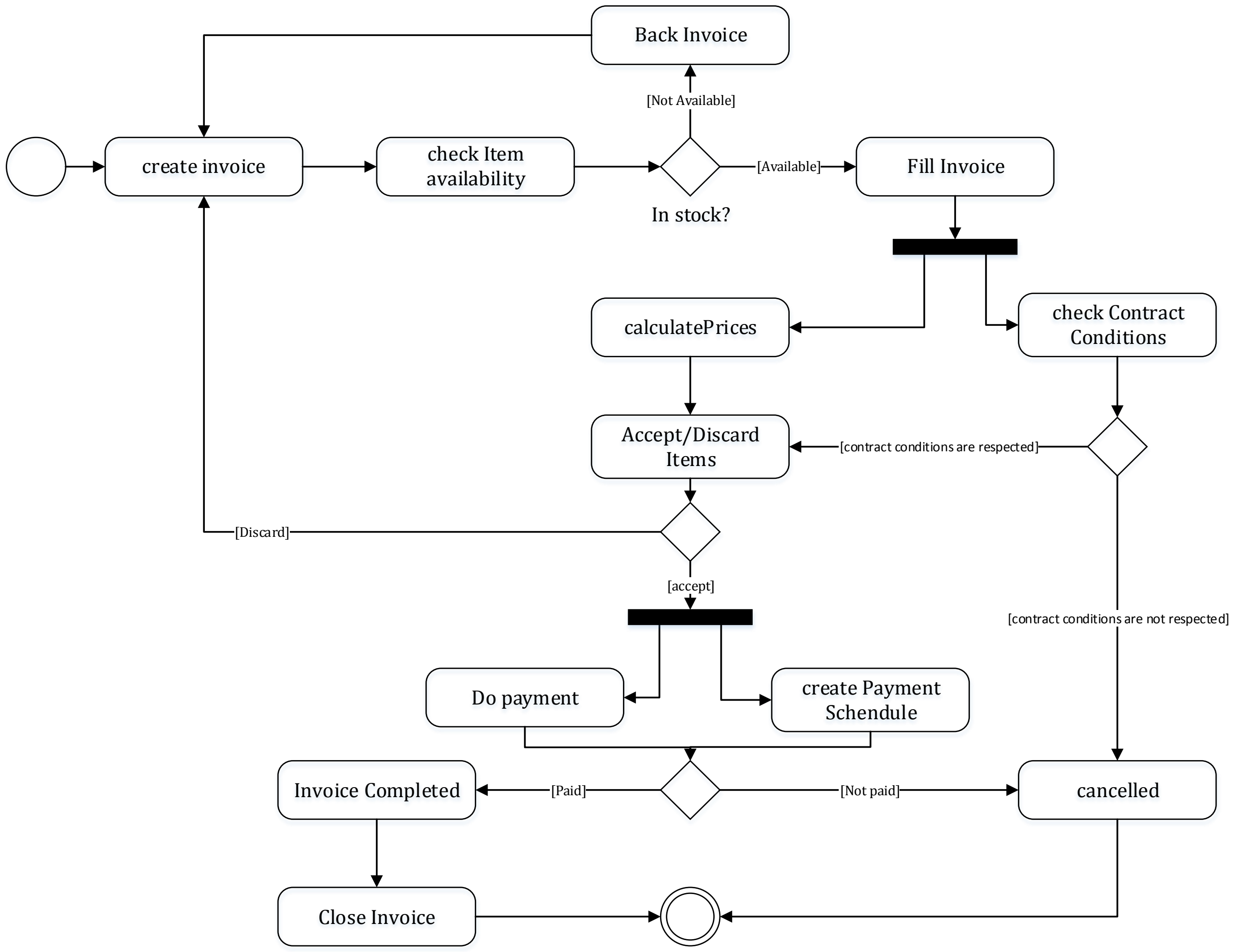


Figure 17. Activity diagram “New Invoice” (Source: author)

2.3.6 Relational Database implementation

In recent years, development of database systems becomes one of the most important component in Information Technology, having a decisive impact on the mode how an institution are organized and work.

Day by day, almost all people have activities that involve interaction with a database, like different management systems implemented to be used by employees in a company, a large variety of database systems for education etc.

Vast majority of existing database systems are currently relational or object oriented and there are a large number of commercial systems that can be purchased.

The relational databases model was introduced by E. F. Codd, IBM Company, in 1970 in this work "A Relational Model of Data for Large Shared Data Banks". This model is based on mathematical relationships, which correspond to entities of the same type and has an understandable representation, and are easy to manipulate. Consists of a two-dimensional table composed of lines and columns. Each line in the table represents an entity and is composed of a set of attribute values, each attribute corresponds to a column of the table **(GLASS, 2004).**

The user see data as tables, which is easy to use in almost all domains of activities.

Besides the benefit of an accurate and simple data model, the relational database systems also has advantage from a recognized and accepted programming language, SQL (Structured Query Language), for which was developed a large number of standards by the International Organization for Standardization (ISO).

Object oriented data model is a unifying concept in computer science, being applicable in programming, hardware design implementation, user interface design, in databases etc.

There are some areas, especially which handle with complex data types such as geographic information systems, in medicine etc., are most suitable for this type of databases, here relational model proved to be insufficiently expressive and with reduced execution performance.Important characteristics of object-oriented model are: abstraction, inheritance, encapsulation, and polymorphism possess characteristics described in the chapter I OOP introduction **(SCHWARTZ, ZAITSEV, TKACHENKO, 2012).**

In object oriented programming, programs are organized as collections of cooperating objects, each object is an instance of a class. Each class is an entity type abstraction of reality shaped and classes are members of a hierarchy of classes, interrelated through inheritance relationships. Every object is encapsulated, meaning that his representation (the internal structure of the object) is not visible to users who have access only to the functions (methods) on the object which they are able to perform. Classes and object oriented program objects are grouped into modules that can be compiled separately and which have boundaries defined and documented, which reduces the complexity of handling data (in other words the presence of polymorphism).

Object-relational data model is extending the relational model with characteristics of object-oriented model, extending the databases for defining and processing complex data types. In essence, the model object-relational data structure maintains relations (represented as tables), but adds the possibility to define new data types for fields of attribute values.

In hierarchical data model a database is represented by a hierarchical structure of data (records) connected by links (links). The hierarchical model was first used for the databases development. The conceptual structure of hierarchical database model is represented by a certain number of trees. A tree is a directed and is represented on multiple levels, in which the nodes are the types of records and arcs are the types of links. Each node (with except the root node) has a single connection to a node on a higher level (parent node) and each node (except leaf nodes) has one or more links to nodes of the level immediately below (nodes sons).

The Network Model uses a graph structure to define the conceptual schema of the database, nodes of the graph are the types of entities (records) and edges of the graph are explicitly associations (links) between entity types. Like the hierarchical model, the main disadvantage of network model is that each query must be provided early in design stage by storing explicit links between entity types. In addition, the complexity of network data representation model is particularly high.

Independently of the type of database, there tree levels of database architecture, implemented by ANSI/X3/SPARC (1975) standard. The levels of database architecture are focused on enabling users to access the same data but with a personalized view of it.

The internal level responsible for physically storage of data, not visible for simple users. This level also permit to change the database storage structures without affecting the users' views.

1. **Conceptual level** describes the structure of the whole database for a community of users. On the conceptual level is the full description of the database, without the details of physical storage, focusing on describing entities, data types, relations between them and the associated restrictions. The details described on this level usually are used in implementation.
2. **External level includes** a collection of external diagrams that describe the database through the various users. Each user group describes this database through their own interests. At this level a user group could hide details from other groups of users who are not interested.
3. **For database implementation** is used Database Management Systems (DBMS). DBMS is focused on defining, designing, sharing and managing databases through different users and applications.

Nowadays, the market has a very large DBMSs, from those which could be used for free (unlicensed or licensed public), to high-performance systems, the use of which requires paid licenses. For these systems exist, on websites, managed by producers, called trial version of DBMS, to use it a limited number of days (30, 60 days, depending on the manufacturer) for free after which is decided to by license or not.

Microsoft SQL Server database management system is multi-user relational database developed for Microsoft Windows operating systems. There were several versions, the current one being SQL Server 2014. In all versions, the database system supports SQL2 standard, the implementation of efficiency, advanced storage features and data processing.

There is also GUI for user interaction, for using all options: export / import data, create and handling tables, create queries, store procedures, triggers etc. **(MICROSOFT, 2009).**

MySQL is a relational database management system implemented on operating systems like Linux, UNIX, and Windows. This system can be used free of charge and is open source. The latest version and documentation on MySQL can be downloaded at http://www.mysql.com. This system is compatible with SQL2 standard, but some provisions of the standard being implemented partially.

DBMS Oracle is a multi-user database management system, with implementations on all platforms (Windows, Linux, UNIX), which provides both high execution performance and a high degree of protection and data security. In all versions, Oracle offers complete implementation of the characteristics of the relational model, SQL2 standard, and the last version (Oracle8i, Oracle9i, etc.) are object-relational management systems distributed, implementing object oriented extensions provided in SQL3 standard and enabling the development of bases distributed data. The license terms allow free use of these systems for non-commercial purposes, for commercial use there are paid appropriate licenses (**ORACLE, 2014**).

Most systems management of current relational database implement different version of the standard for SQL, like called SQL: 2006 (working with XML data) and others.

Below are all necessary DDL[5] and DML[6] commands required for relational database implementation. Both of these categories contain far more statements than is presented here, and each of the statements is far more complex then is shown in this section.

Used tools to implement database are:

1. **XAMPP 5.6.3**
   - phpmyadmin version 4.2.11
   - http://www.phpmyadmin.net

[5] **DDL -** Data Definition Language (DDL) is a vocabulary used to define data structures in SQL Server 2014. Use these statements to create, alter, or drop data structures in an instance of SQL Server.

[6] **DML -** Data Manipulation Language (DML) is a vocabulary used to retrieve and work with data in SQL Server 2014. Use these statements to add, modify, query, or remove data from a SQL Server database.

- Host: 127.0.0.1
- Server version: 5.6.21
- PHP Version: 5.6.3

2. **Database server:**
   - Server: 127.0.0.1 via TCP/IP
   - Server type: MySQL
   - Server version: 5.6.21 - MySQL Community Server (GPL)
   - Protocol version: 10
   - User: root@localhost
   - Server charset: UTF-8 Unicode (utf8)

In the Figure 18 is the implemented design of the relational database “Invoice”.

DDL statements, used to build and modify the structure of tables and other objects in the database, and DML statements, are.

First statement is creation of database itself, which will consists of all needed tables for invoice system presented below.

```
CREATE DATABASE IF NOT EXISTS `invoice` DEFAULT CHARACTER SET latin1 COLLATE
latin1_swedish_ci;
```

To use created database is needed to call the function use:

```
USE `invoice`;
```

Second statement is to create table structure for table `accounts`:

```
CREATE TABLE IF NOT EXISTS `accounts` ( `acc_id` varchar(15) NOT NULL,  `p_d`
varchar(250) NOT NULL COMMENT 'Position description ',  PRIMARY KEY (`acc_id`))
ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

To see the souse code of the implementation of the relational database using DBMS MySQL look at APPENDIX 4.

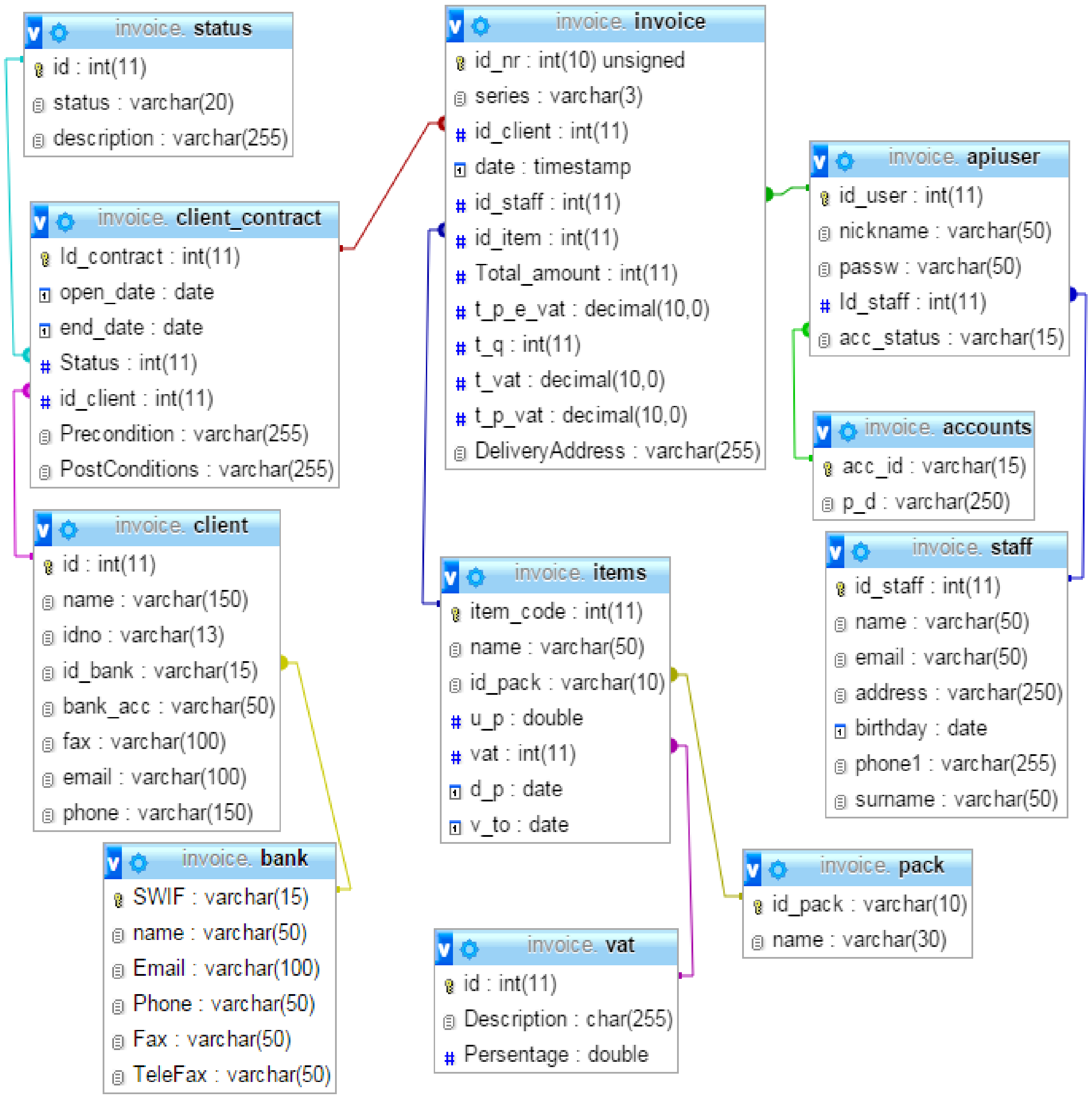


Figure 18. Entity-Relation Diagram of the "Invoice" Database (Source: author).

Chapter III. Implementation of Application in Object Oriented Language

A good desktop API[7] design focuses on the functionality that it provides to the user. The desktop API presents a logical view of the functionality of the real system which usually is needed to be automated.

The way of how is done and what mechanism is being used to accomplish it, the user no need to know. The user will be able to use a desktop API if it would be designed in simplest way, efficient, and clear, in the way that the user will be able to use it intuitively. Today using graphical environment, we can exactly know what we are going to design or build, and to get immediate feedback about what is working and what is not working.

In this chapter is implemented a desktop API which will work with the invoice system for micro scale businesses. The application starts with the concept, with the idea about what is going to be implemented and what is needed to do to get it.

Problem formulation:

Micro scale businesses in the villages, like small shops, bakeries, bars use excel databases to work with their accounting documentation, but this is not the biggest problem, because is no need to work with large data, and excel is enough for this. The problem is that they can work only with consumer in other words possible only B2C[8], no possibility to sell to other business (B2B[9]). To make possible to sell to anther businesses the micro scale business should implement accounting software with agreement from Fiscal Police. As an example of this type of software is 1C: Enterprise, which have a lot of modules which is no need for micro companies. The License price a quite high, and not all individual business can afford it.

Proposed solution:

If it was possible just to implement an invoice system which will permit only generation of invoices and save information about clients, it will be low cost and increase the processing time of the information. In this case, will be implemented a desktop application

[7] API – Application Program Interface
[8] B2C – Business to consumer
[9] B2B – Business to Business

which will permit generation and evidence of invoices, also will save information about clients (B2B), using different tools described below.

To identify what a Micro Scale businesses presents, is extracted some basic information/ characteristics from Low of Republic of Moldova Number 619, from 08 November 2001 with changes in currents years:

1. Micro-scale business revenue is generally lower than companies that operate on a larger scale.
2. The maximum revenue allowance for the small business designation is set at 3 million lei per year (LEGE Nr. 619, 08.11.2001).
3. Maximum number of employees for small scale business is 9 persons (LEGE Nr. 619, 08.11.2001).

Proposed solution tools:

Implementation of a simple invoice system in which will be saved data about what was sold, about clients and information about items (goods) which are going to be sold. **For implementation is used following tools:**

1. **XAMPP[10] 5.6.3**
   - phpmyadmin version 4.2.11
   - http://www.phpmyadmin.net
   - Host: 127.0.0.1
   - Server version: 5.6.21
   - PHP Version: 5.6.3

   a. **Database server:**
   - Server: 127.0.0.1 via TCP/IP
   - Server type: MySQL
   - Server version: 5.6.21 - MySQL Community Server (GPL)
   - Protocol version: 10
   - User: root@localhost
   - Server charset: UTF-8 Unicode (utf8)

[10] **XAMP** – **X** (to be read as "cross", meaning cross-platform) **A**pache HTTP Server **M**ySQL **P**HP **P**erl

2. **Connector/ODBC 5.3 for connection between database and desktop API**
3. **Borland C++ Builder Enterprise Suite version 6**

C++ Builder is the multi-device, standards-based C++ solution, used for faster building high performance, natively compiled apps. C++Builder uses tools that permit drag-and-drop visual development making the process of creating graphical user interface more easy and faster (EMBARCADERO TECHNOLOGIES, 2014).

To create an application using C++ Builder first of all is needed to know some introductory notions like:

Forms – are most frequent objects used in the creation of an application, they are defined as C++ classes and are graphical windows on which are positioned almost all components needed to work with database. Forms could be created dynamically or statically, in dependence on what is going to be build. The form is implemented by **TForm Class**, and when is created a new form is called the **CreateForm()** method of the **TApplication Class** (EMBARCADERO TECHNOLOGIES, 2014).

```
void __fastcall CreateForm(System::TMetaClass* InstanceClass, void *Reference);
```

When the application is started or ended, initialization and finalization is carried out by Terminate() and Initialize() functions. So, the initialization, will run the main form after checking for error occurrence.

```
Application->Terminate(); or  Application-> Initialize();
```

When CreateForm() method is called, it requires to set the name of the form which will be used in following work. This is done by the operator **_classid() ,** which plays the role of identification key for the new created form.

```
Application->CreateForm(__classid(TForm1), &Form1);
```

After is specified the name of the source file, that holds the implementation of the form, using **USEFORM** macro.

```
USEFORM("Unit1.cpp", Form1);
```

Automatically when is saved a form also is saved Project1.cpp with all implemented forms and source files names (see APPENDICS). In the applications created are implemented almost all forms as static forms which can be accessed outside the application, it was also created a dynamical form which is locally created, and available only in the event or the function in which is created (Login form).

First, application begins with the database login form(Figure 18) in which is specified if you are not a user of the used database (see Chapter II) the application would be not opened.

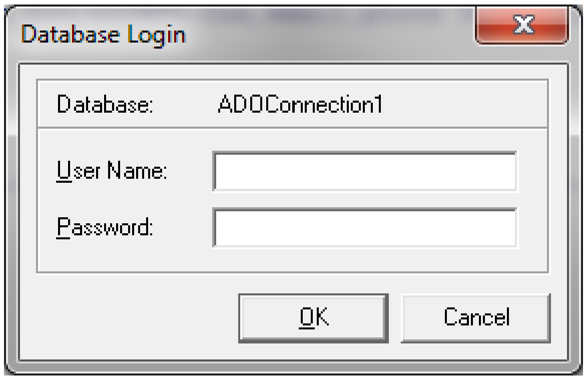


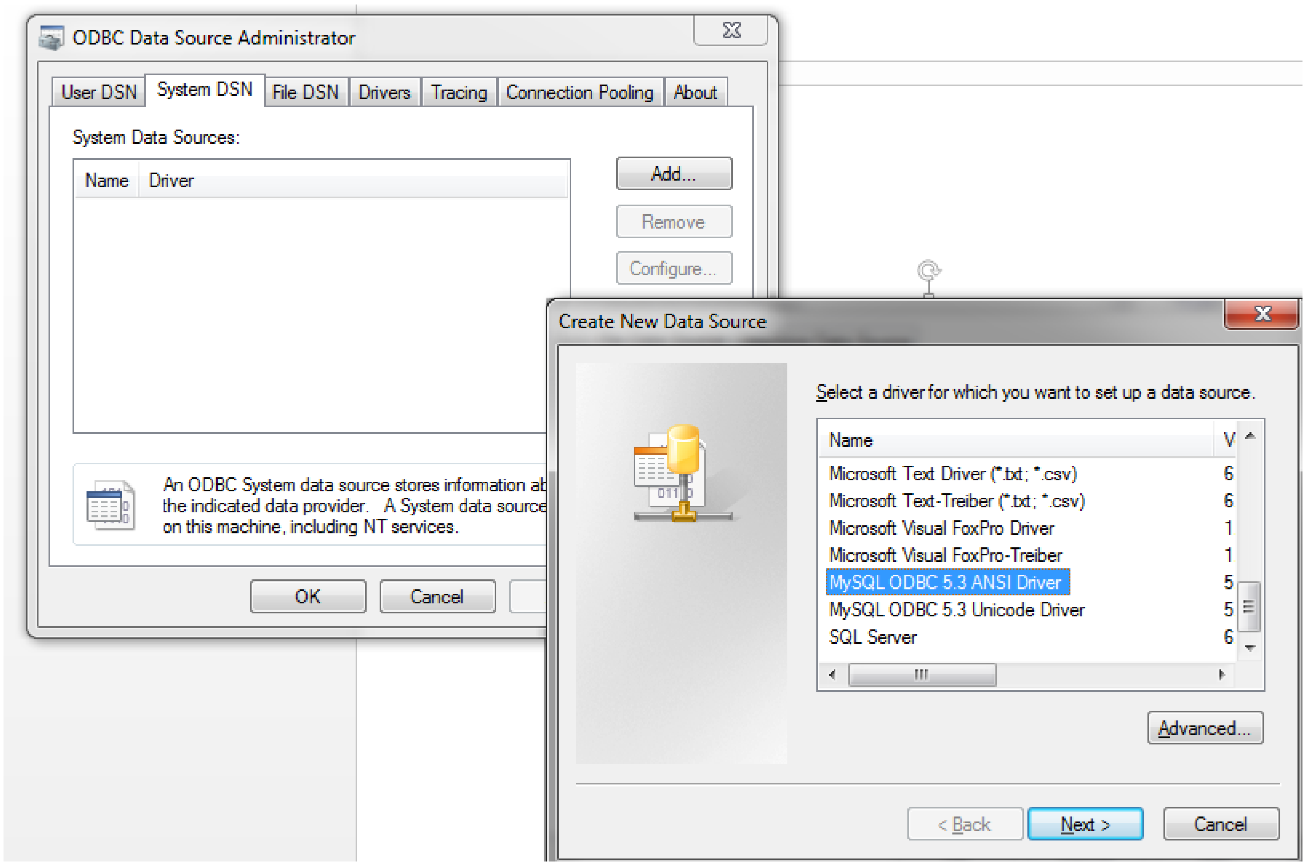
Figure 18. Login form (Source: author)

To make connection to the database was used ODBC Driver for connection of Builder C++ 6 application with MySQL Database. ODBC Driver can be used on all major platforms supported by MySQL. MySQL Connector/ODBC provide access to a MySQL database using the industry standard Open Database Connectivity (ODBC) API (ORACLE, 2014).

It was used Connector/ODBC 5.3, which includes the functionality of the Unicode driver and the ANSI driver, which formerly were split between Connector/ODBC 5.1 and Connector/ODBC 3.5. Also it provides both driver-manager based and native interfaces to the MySQL database, with full support for MySQL functionality, including stored procedures, transactions and, with Connector/ODBC 5.1 and higher, full Unicode compliance (ORACLE, 2014).

To create the connection with MySQL database using ODBC, first I create a Data Source Name (DNS). Connection steps:

1. Using ODBC Data Source Administrator was crated new data source for MySQL (Figure 20).



An ODBC System data source stores information ab
the indicated data provider. A System data source
on this machine, including NT services.

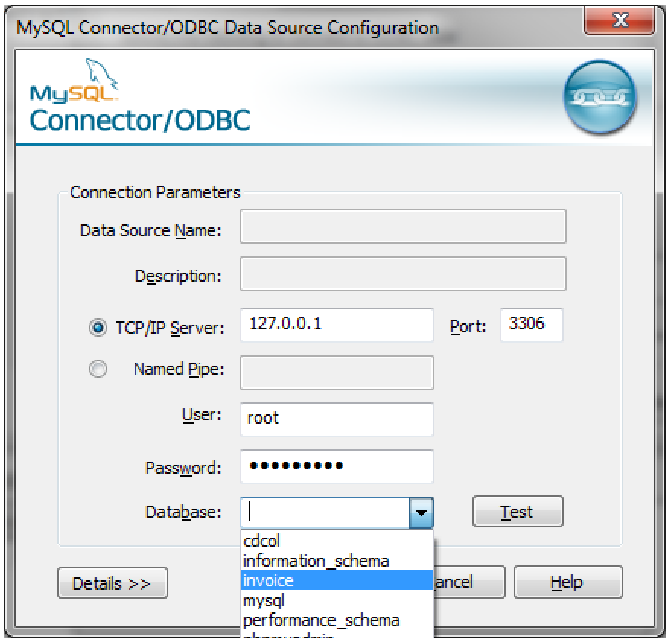
Select a driver for which you want to set up a data source.

Name	V
Microsoft Text Driver (*.txt; *.csv)	6
Microsoft Text-Treiber (*.txt; *.csv)	6
Microsoft Visual FoxPro Driver	1
Microsoft Visual FoxPro-Treiber	1
MySQL ODBC 5.3 ANSI Driver	5
MySQL ODBC 5.3 Unicode Driver	5
SQL Server	6

Figure 19. ODBC Data Source Administrator window (Source: author work)

2. After running ODBC Data Source Administrator window is configured the specific fields for the DSN[11].

[11] DSN – Data Source Name


MySQL Connector/ODBC Data Source Configuration

Connection Parameters

Data Source Name:

Description:

(●) TCP/IP Server: 127.0.0.1 Port: 3306

( ) Named Pipe:

User: root

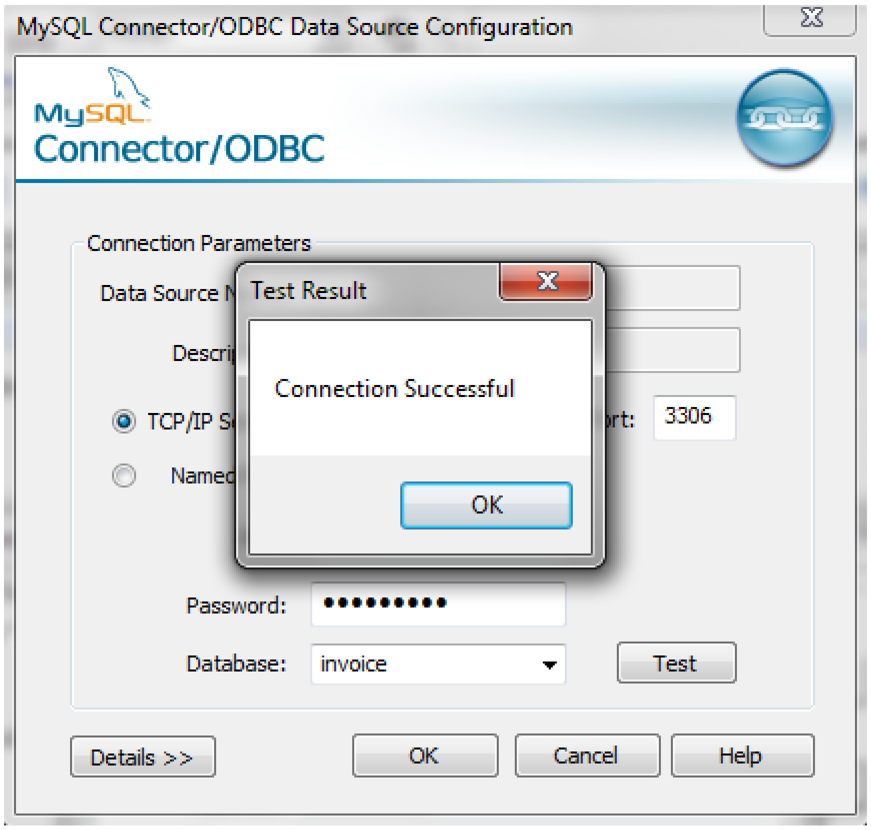
Password: •••••••••

Database: |

cdcol
information_schema
invoice
mysql
performance_schema

Figure 20. MySQL Connector/ODBC Data Source Configuration (Source: author work)

3. After configuring the specific fields for the DSN was tested the connection and the results of connection are successful.


MySQL Connector/ODBC Data Source Configuration

Connection Parameters

Test Result

Connection Successful

(●) TCP/IP S

Port: 3306

Password: ••••••••••

Database: invoice

Figure 21. MySQL Connector/ODBC Data Source Configuration. Testing connection (Source: author work)

To apply created connection in the application, is used TADOConnection component from the ADO tab of the Component Palette (Figure 23).

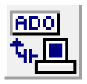
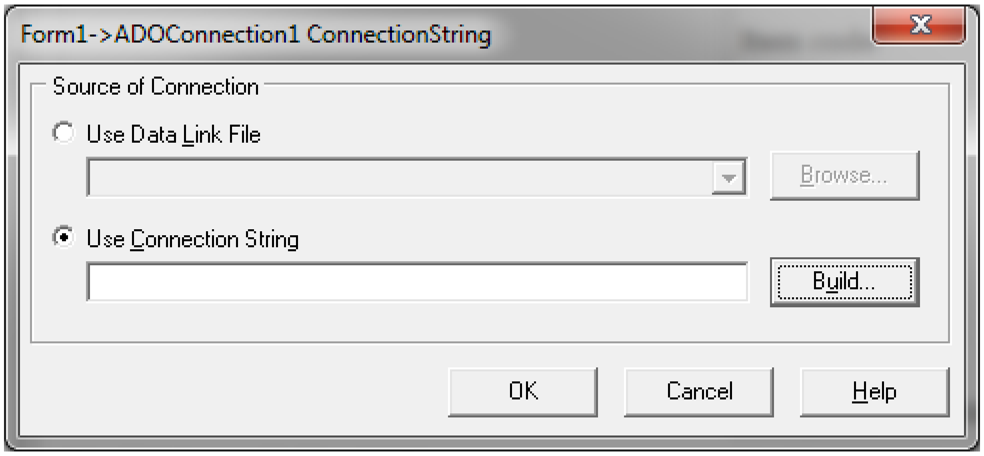


Figure 22. TADOConnection component from the ADO tab of the Component Palette. (Source: author work)

On the Object Inspector -> Proprieties, is selected ConnectionString, cliking on which is displayed ADOTConnection -> ConnectionString dialog box (Figure 24).


Form1->ADOConnection1 ConnectionString

Source of Connection

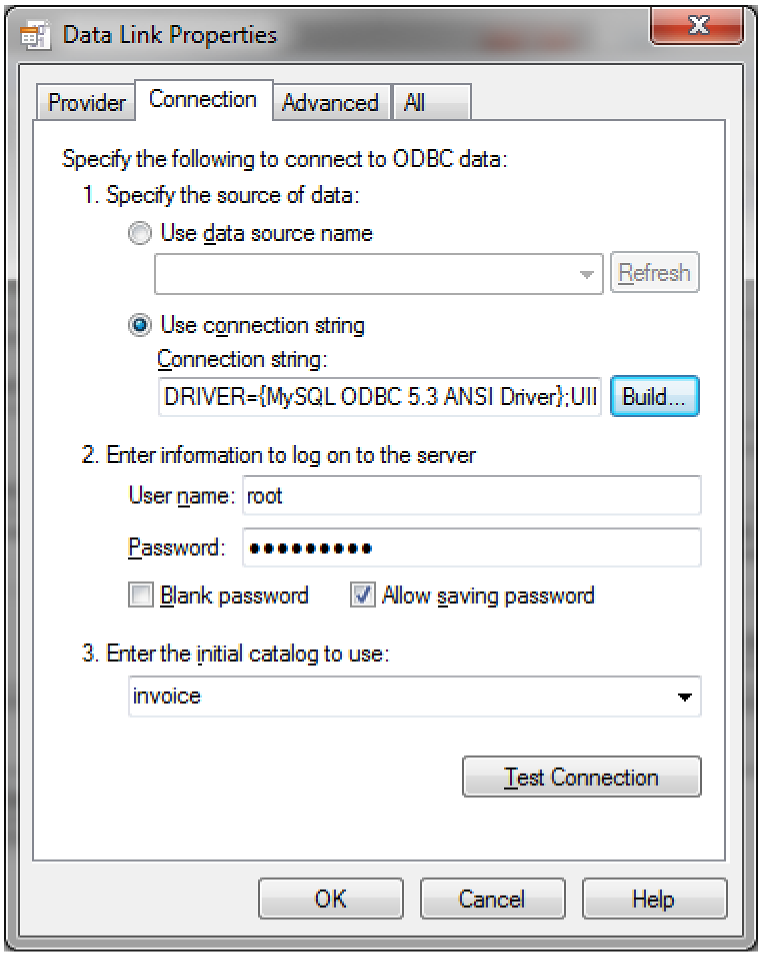
Use Data Link File

Use Connection String

Figure 23. Connection String dialog box (Source: author work).

Pressing button

is displaying Data Link Window dialog box, in which was specified details with created DSN.


Data Link Properties

Provider | Connection | Advanced | All

Specify the following to connect to ODBC data:

1. Specify the source of data:
   - ( ) Use data source name
   - (●) Use connection string

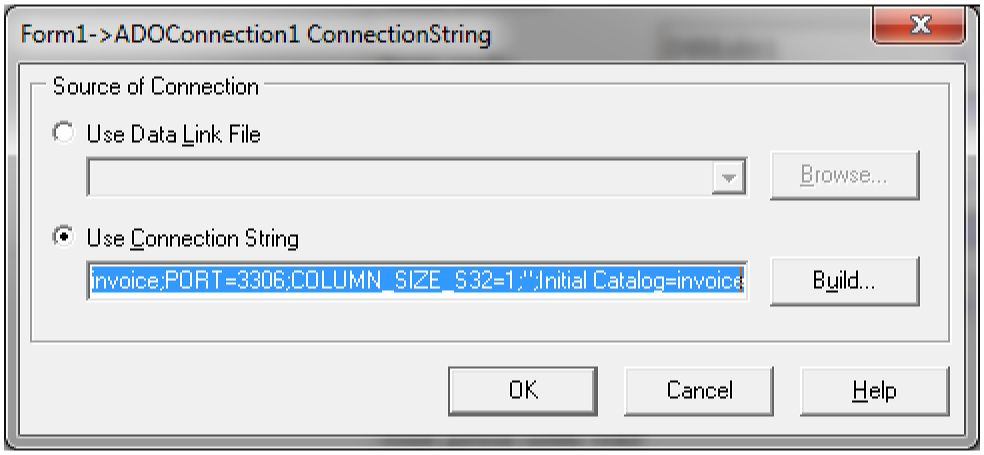
   Connection string: DRIVER={MySQL ODBC 5.3 ANSI Driver};UII

2. Enter information to log on to the server
   - User name: root
   - Password: ●●●●●●●●●●
   - [ ] Blank password [✓] Allow saving password

3. Enter the initial catalog to use: invoice

Figure 24. Data Link Window dialog box (Source: author work).

After the all information is completed, like information to log to the server, database which is going to be used, connections string with DSN, is needed to press button OK and Connection String dialog box is completed, the connection between database and application is created.


Form1->ADOConnection1 ConnectionString

Source of Connection
- ( ) Use Data Link File
- (●) Use Connection String

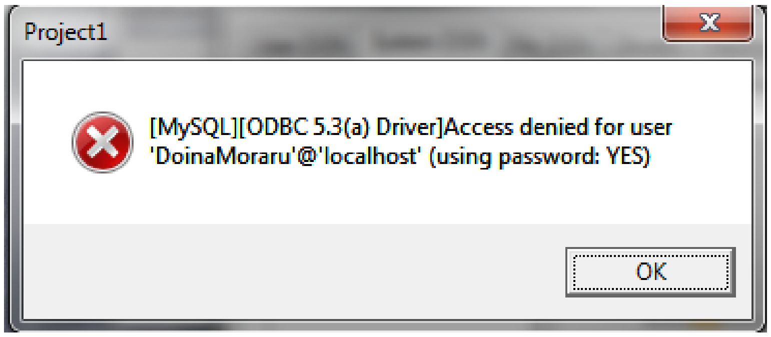
nvoice;PORT=3306;COLUMN_SIZE_S32=1;";Initial Catalog=invoice

Figure 25. Completed Connection String dialog box (Source: author work).

The connection string looks like:

```
Provider=MSDASQL.1;Password=068003524;Persist Security Info=True;
UserID=root; ExtendedProperties="DRIVER={MySQL ODBC 5.3 ANSI
Driver};UID=root;
PWD=068003524;SERVER=127.0.0.1;DATABASE=invoice;PORT=3306;COLUMN_SI
ZE_S32=1;";Initial Catalog=invoice
```

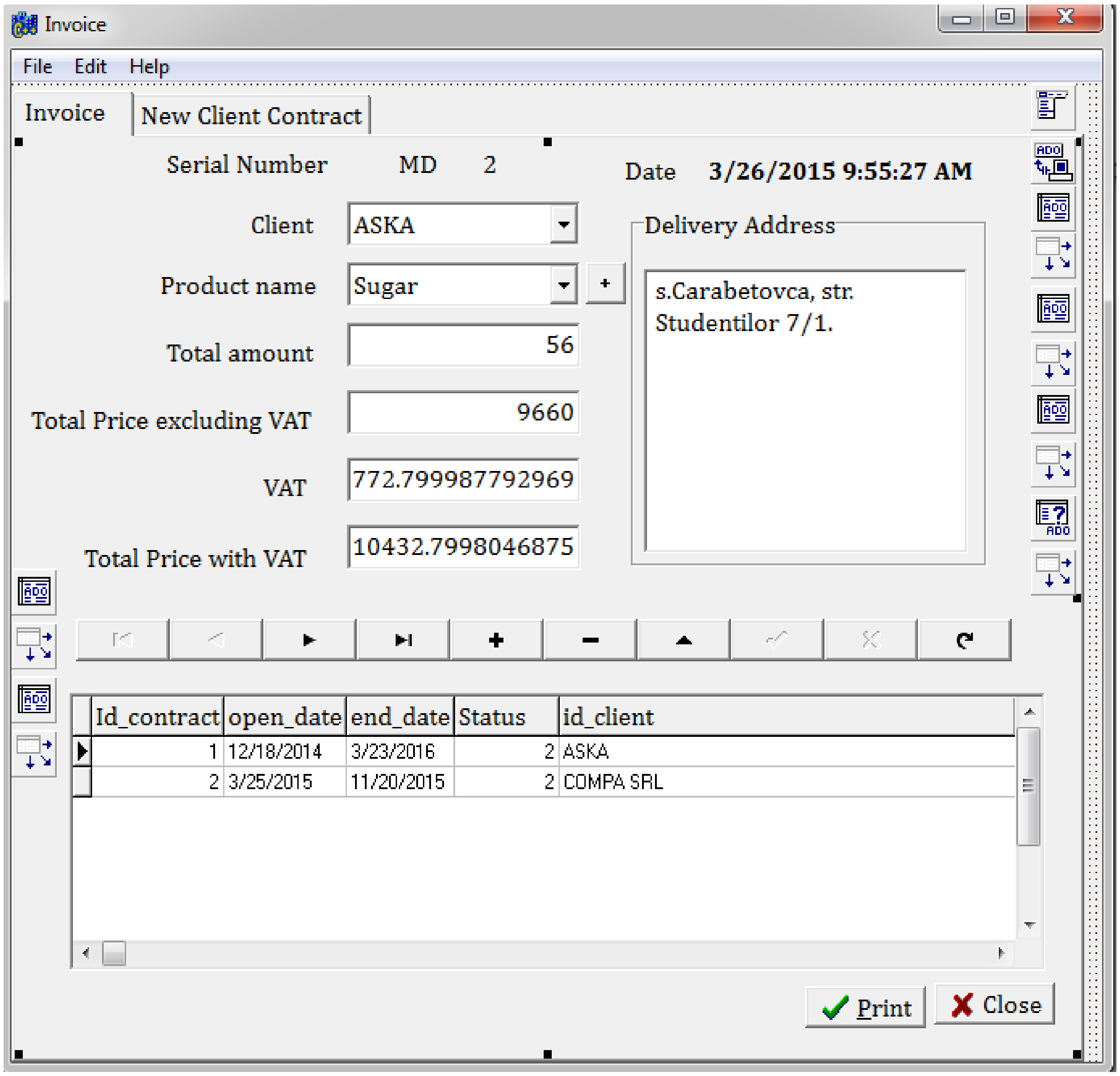
When the application is running is displayed Login Form (see Figure 19). If the user have entered a wrong password or username, authorization fails and is displayed the alert window with corresponding message (Figure 27). Is verified if it matching Password and User ID from connection string specified before.



[MySQL][ODBC 5.3(a) Driver]Access denied for user 'DoinaMoraru'@'localhost' (using password: YES)

Figure 26. Alert Window displaying Authorization Fail (Source: author work)

If the login into the database is successful, is displayed the invoice/client new contract form with the all necessary components for creating invoices (Figure 27).



Serial Number MD 2

Date 3/26/2015 9:55:27 AM

Client: ASKA

Product name: Sugar

Total amount: 56

Total Price excluding VAT: 9660

VAT: 772.799987792969

Total Price with VAT: 10432.7998046875

Delivery Address: s.Carabetovca, str. Studentilor 7/1.

Id_contract	open_date	end_date	Status	id_client
1	12/18/2014	3/23/2016	2	ASKA
2	3/25/2015	11/20/2015	2	COMPA SRL

Figure 27. Invoice form (Source: author).

To calculate an item's purchase price based on the item's store price added the tax, with the tax rate in percentage value is needed to transform tax rate from 8 % in C++ terms, and it will be equal to 0.08 (because $^{8\,\%}/_{100\,\%} = 0.08$). In this case the tax amount collected on a purchase is taken from an item's price formula:

$$VAT_{item} = P_{Item} * \frac{VAT_{rate}}{100\,\%}\ ;\ \ (\boldsymbol{Equation\ 1})$$

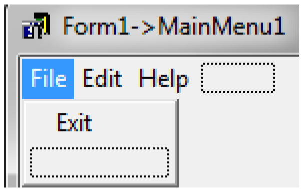
Where:

- VAT$_{item}$ – Value Added Tax per item represents the tax amount for the unit item price.
- P$_{Item}$ – Price per Item.
- VAT$_{rate}$ – Value Added Tax in percentages.

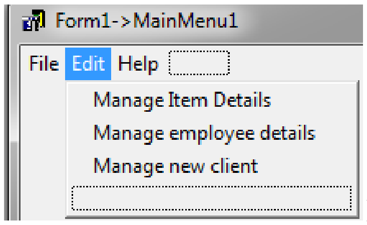
The formula of calculating the final price of an item is:

$$TP_{Item} = P_{Item} + VAT_{Item}; \quad \textbf{\textit{(Equation 2)}}$$

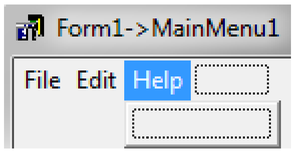
Menu bar is part component of almost every commercial application, and using C++Builder's Menu Designer to create it in concordance with user's expectations are timeless and easy. To construct a menu bar is used The MainMenu component from Standard Pallet, which permit this, containing all functionality (properties, methods, and events) for forms menu bar and associates dropdown menus and submenus with accelerators and shortcuts. On the Invoice Form is presented menu bar with next submenus:



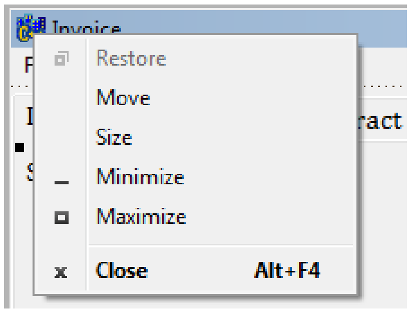
File menu has a submenu Exit which permit to close the all forms opened to work with.



Menu Edit allow user to call forms for introduction new information details about items, employee, clients. This form could be called also from the others forms using buttons + to add something new.



Menu Help does not have submenu, it directly calls the form with details about how to use the application.



To call system menu is needed to set click right the icon used on the title bar possesses. The menu displayed allows to perform the common actions of a regular Windows form.

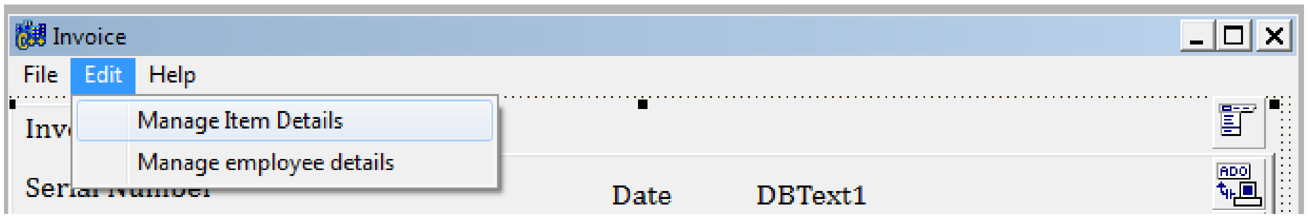
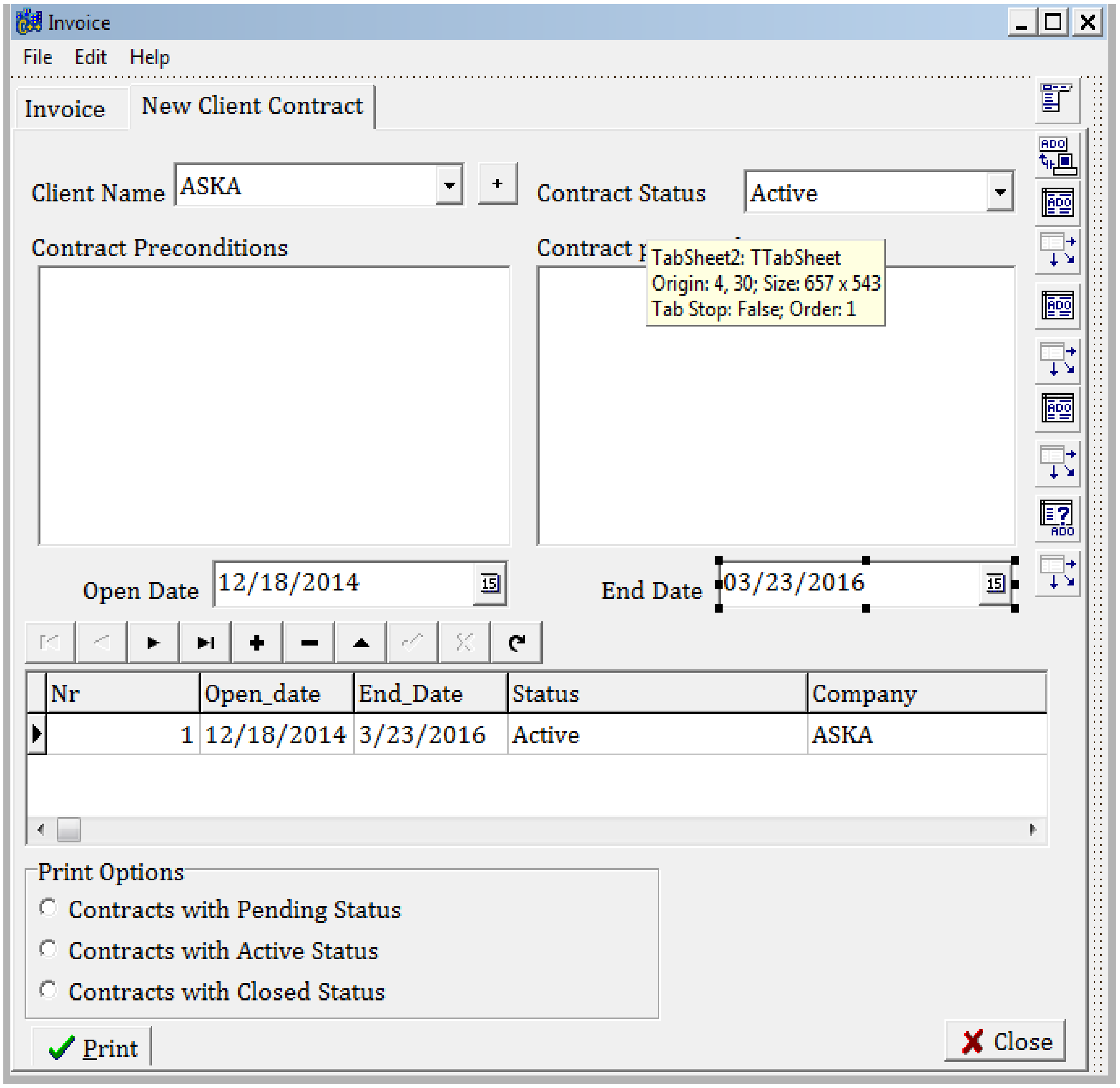


Figure 28. Invoice form (Source: author work)

On the same form is component TabSheet2 (Figure 29), which show the all-necessary components for making new clients contracts and set their company details.



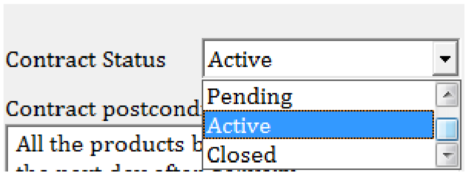
Client Name: ASKA
Contract Status: Active
Contract Preconditions
Contract p
TabSheet2: TTabSheet
Origin: 4, 30; Size: 657 x 543
Tab Stop: False; Order: 1

Open Date: 12/18/2014
End Date: 03/23/2016

Nr	Open_date	End_Date	Status	Company
1	12/18/2014	3/23/2016	Active	ASKA

Print Options
- Contracts with Pending Status
- Contracts with Active Status
- Contracts with Closed Status

Figure 29. Component TabSheet2 – New Client Contract (Source: author).


Contract Status: Active
Contract postcond
All the products b
- Pending
- Active
- Closed

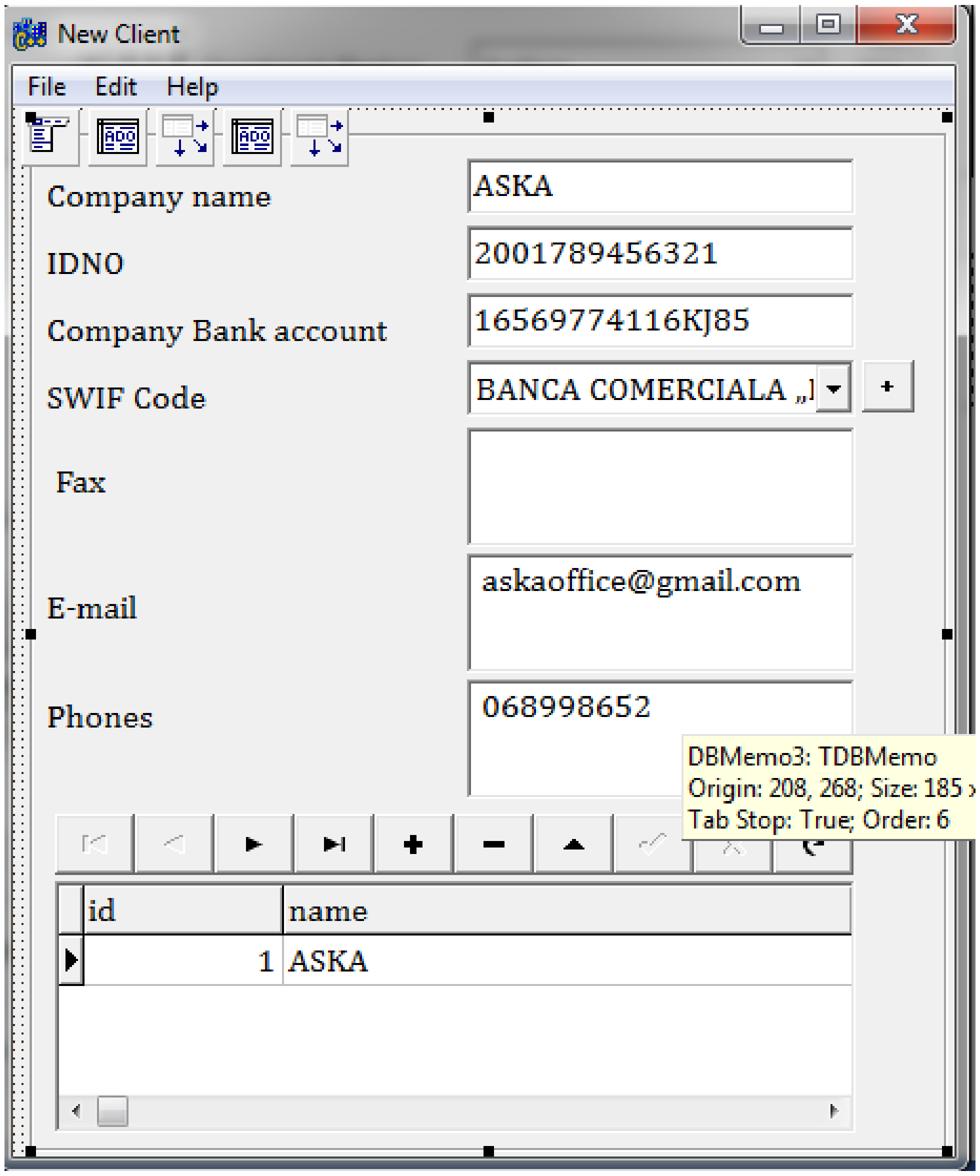
The contract status in the “New Contract Form” is a system-defined value which every contract that is created is associated with. Each new contract has automatically set default status Pending and in the table below are the list of statuses which I am working with and their characteristics.

Table 2. Processing status of a contract. Source (ORACLE, 2014).

№	Status	Description	Characteristics
1	Pending	Default processing status of a new contract. This status indicates that the contract was created and that some data was entered.	Data fields are available for entry. Minimal system data validation was performed to validate the entered data.
2	Active	All contract data is entered into the system and has passed system data validation. A contract must be in a status of Active for any contract element to be available for application processing.	Controlled data fields are available for entry. The system has confirmed the entry of required fields and has validated the contract data.
3	Closed	The contract was terminated or the contract was completed, all contractual obligations were fulfilled and all entitlements were received.	Data fields are not available for entry. The contract passed system data validation to verify that all processing related to this contract was completed.

Note: Once is defined a contract status, the remap of the contract status to a different processing status will cause a change for all mapped invoices to that contract.

Before creating the new contract is needed to have the client information in the system. To introduce new client information,on the component TabSheet2 –> New Client Contract near the component object DBLookupComboBox2 component from class TDBLookupComboBox are placed button + which permit to display form New Clinet (Figure 30).


New Client

Company name	ASKA
IDNO	2001789456321
Company Bank account	16569774116KJ85
SWIF Code	BANCA COMERCIALA „I
Fax	
E-mail	askaoffice@gmail.com
Phones	068998652

DBMemo3: TDBMemo
Origin: 208, 268; Size: 185 x
Tab Stop: True; Order: 6

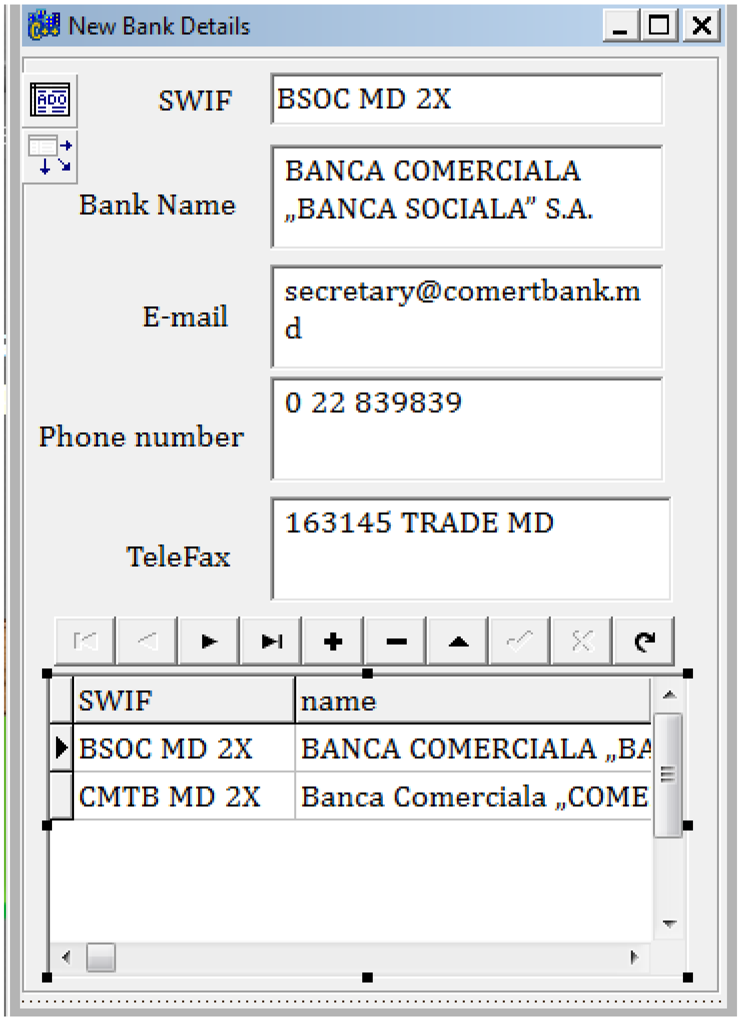
id	name
1	ASKA

Figure 30. Form for registration new clients.

There are all necessary Edit Boxes for registration new client. In concordance with Article 161 of the Tax Code, the tax authorities exercise state registration of the taxpayer by way of assignment of fiscal code and maintenance of the tax registry.

The state identification number (IDNO) – is unique numeric code assigned by registration authority to the entrepreneurs at the time of state registration, which serves to identify them in the information systems of the Republic of Moldova. It is assigned only once, irrespective of the provisions of the tax regulations concerning the establishment and discharge of tax obligations. An organization, which was not assigned a fiscal code, may not be registered as a taxpayer. The IDNO code of a resident or non-resident organization is considered to be the personal code indicated at the moment of state registration.

Also to introduce new information about bank which work with the client company is needed to press button +, and will be displayed the Form "New Bank Details" (Figure 31).


New Bank Details

SWIF	BSOC MD 2X
Bank Name	BANCA COMERCIALA „BANCA SOCIALA" S.A.
E-mail	secretary@comertbank.md
Phone number	0 22 839839
TeleFax	163145 TRADE MD

SWIF	name
BSOC MD 2X	BANCA COMERCIALA „BA
CMTB MD 2X	Banca Comerciala „COME

Figure 31. Form "New Bank Details" (Source author).

On the figure 29 is seen, on the left side down of the form New Client Contract, the print options which permits to print the reports based on status of the contract, to see which of them are active and which are not (Figure 32).

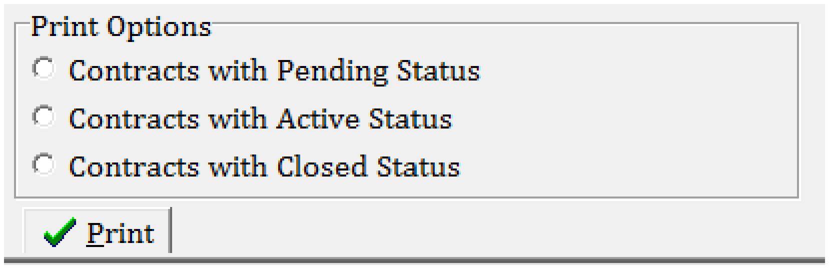
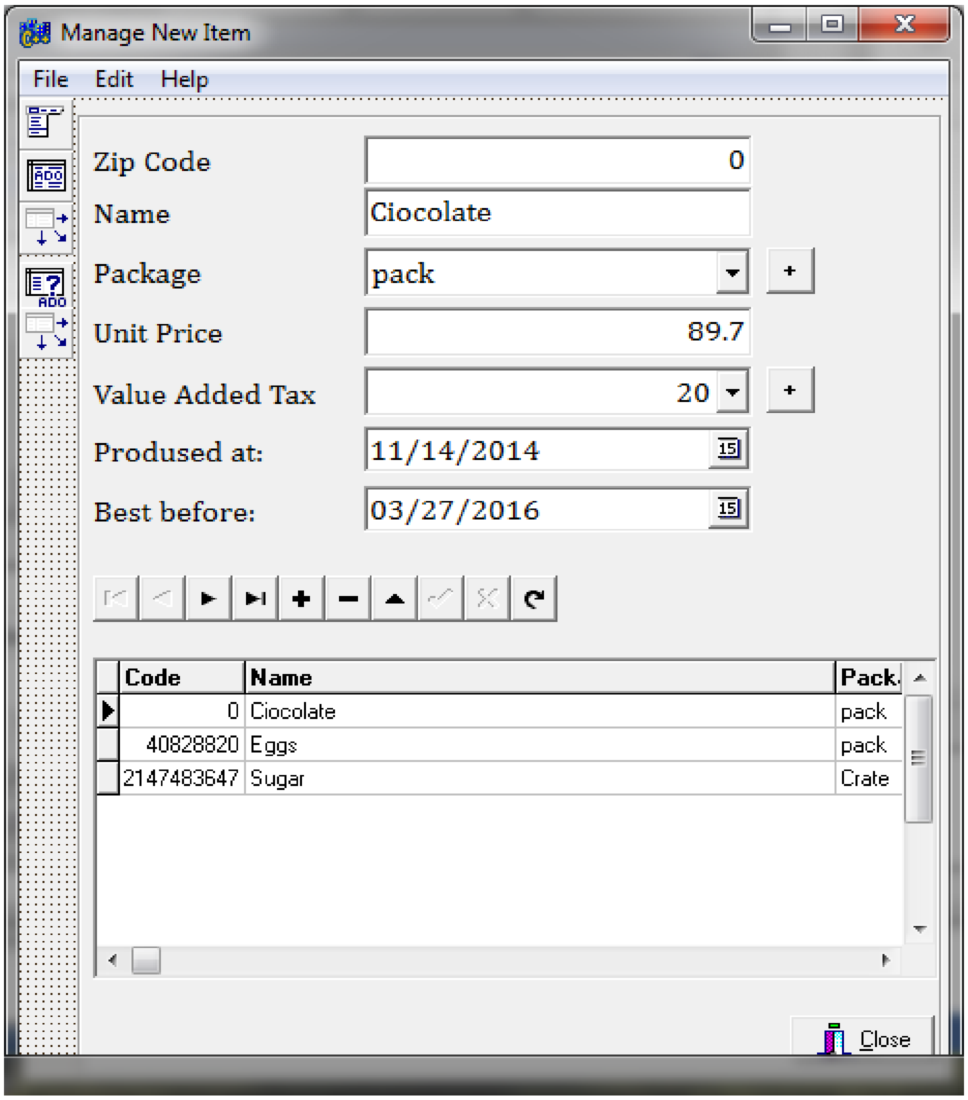

Print Options
Contracts with Pending Status
Contracts with Active Status
Contracts with Closed Status
Print

Figure 32. Printing options for new client Contract Form.

Each of this options activate the specific form called by the BitBtn2 on click event. The source code to call the form are presented bellow using if statement.

```
void __fastcall TForm1::BitBtn2Click(TObject *Sender)
{
if (RadioGroup1->ItemIndex==0) Form9->Show();
else if (RadioGroup1->ItemIndex==1) Form10->Show();
else if (RadioGroup1->ItemIndex==2) Form11->Show();
}
```

Returning back to the invoice form, is important to make registration of the new client but also to specify which product item the client want. To introduce a new product item into the system is possible using the menu Edit -> Manage Item Details, but also using button + from form invoice which on activate event will call the Form "Manage new Item" (Figure 33).



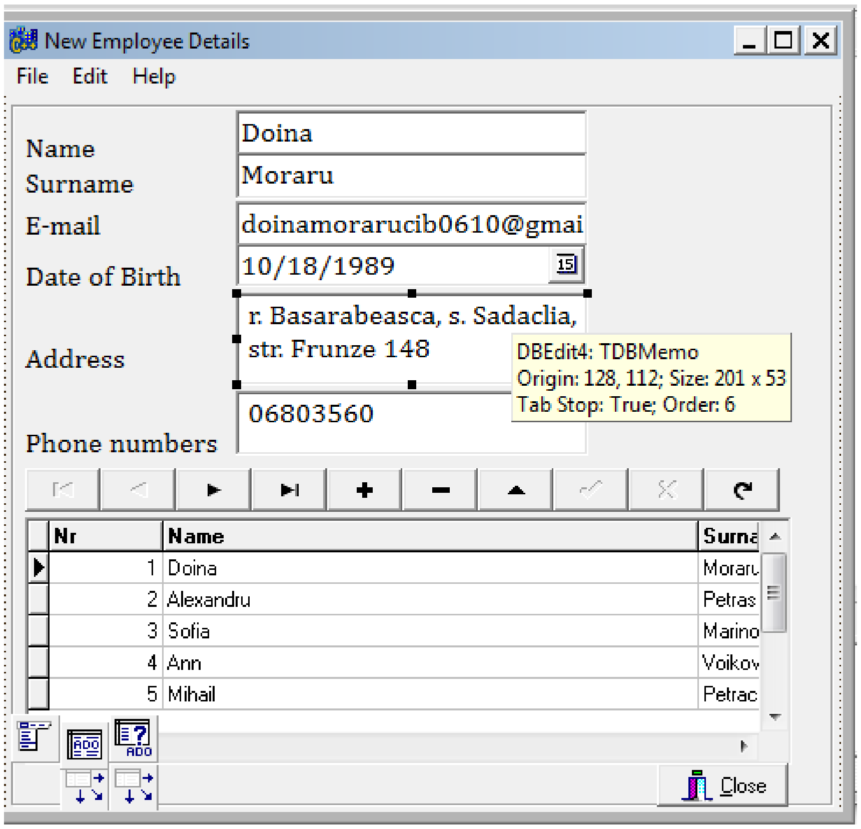
Zip Code	0
Name	Ciocolate
Package	pack
Unit Price	89.7
Value Added Tax	20
Prodused at:	11/14/2014
Best before:	03/27/2016

Code	Name	Pack.
0	Ciocolate	pack
40828820	Eggs	pack
2147483647	Sugar	Crate

Figure 33. Manage New Item Form (Source: author).

The presence of the button "+" on every form or Tab Sheet Component assume the possibility of displaying the some additional forms which helps with the inserting date into the database. As an example on the TForm1 is located "+" near the product Edit component, this button helps to display the Form for inserting the new product of the company (Figure 29).

The employee or staff have a huge role in the activity of the application, only the staff can manage and work with invoice system directly. To manage employee/staff (Figure 34) details introducing into the system using application, is needed to use Menu Edit-> Manage Employee Details, but even after introduction information about new employee, the new employed person do not have access to the application forms. To get the access right is needed the admin involvement. Only admin gives the access right to the new employed person.



Name	Doina
Surname	Moraru
E-mail	doinamorarucib0610@gmai
Date of Birth	10/18/1989
Address	r. Basarabeasca, s. Sadaclia, str. Frunze 148
Phone numbers	06803560

DBEdit4: TDBMemo
Origin: 128, 112; Size: 201 x 53
Tab Stop: True; Order: 6

Nr	Name	Surna
1	Doina	Moraru
2	Alexandru	Petras
3	Sofia	Marino
4	Ann	Voikov
5	Mihail	Petrac

Figure 34. New employee Details Form (Source: author).

The implemented application source code are partial included in the appendix 5. The total amount of the pages with application source code exceeds 100 pages, because of the DFM[12] files in which are include graphical objects characteristics.

[12] DFM - Contains the properties of objects contained in a Borland Delphi form, could be saved in either binary or text format, information from DFM files is loaded into the final executable (.EXE) file.

CONCLUSION

This thesis is focuses on application development in Object Oriented Language utilizing alternative database management system as a supporting database for desktop application. To achieve the aim of the project which was set, was analyzed information about current state of the making invoice process at the micro scale business in Republic of Moldova, in concordance with the Article 161 and Article 117 of the Tax Code, and there is developed a prototype of desktop application for better data processing.

To implement the prototype of the desktop application was led on the software development life cycle – prototype model. From elaborating requirements till the developing initial prototype, was used UML (Unified Modelling Language) Technology, provided by StarUML 5.0.0.1570 software (The Open Source UML/MDA Platform) to design class diagram. To draw the state diagrams, use case diagrams, activity diagrams and sequence diagrams was used Visio Professional 2013 for Microsoft Office 2013. To manipulate with the database was used the open source package XAMP 5.6.3, which handle with PHPMyAdmin to perform different tasks on MySQL database. To make connection to the database was used ODBC Driver for connection of Builder C++ 6 application with MySQL Database, which provide access to a MySQL database using the industry standard Open Database Connectivity (ODBC) API.

The company which activity was analyzed, wanted to be anonymous, and is used the notion of company when is spoken about it. The company is micro scale businesses in the village, a small shop which want to collaborate with bakery and bars, but because of impossibility to use bills to sell to other companies the B2B is not possible. In concordance with the law the simple bills a not recognize as accounting document used for reports to Fiscal Office. To make possible to sell to other businesses the micro scale business should implement accounting software with agreement from Fiscal Police. As an example of this type of software is 1C: Enterprise, which have a lot of modules which is no need for micro companies. The License price a quite high, and not all individual business can afford it.

What was implemented is invoice system which permit only generation, evidence of invoices and save information about clients. The application is created based on the 2 main articles from Fiscal Office, where is described the all necessary components of invoices and gives a general knowledge about the types of businesses and their contribution to Fiscal Office. Created prototype of desktop application meets the exact needs of the user and:

- Make possible to create invoice agreed by Fiscal Office and recognized as significant for accounting reports.
- Low cost to implement (in order to 100$ per year paying for Licenses).
- Decrease the processing time of the information.
- Permit multiple access.
- Restrict access to improve the security of the information stored.
- Properly documented.
- Efficient, quick, easy, intuitively access to all forms, following instructions.

Futures of the application:

- Implementation on other platforms except Microsoft Window.
- Collaboration with the web application based on MySQL database.
- Diversification of the modules contained by created prototype (Summarizing reports for Fiscal Police, Salary etc.).

There are many technologies, which could be used to develop an application prototype, the desktop applications is not the only possibility to resolve the task. This thesis focused on desktop application, using Graphical Object Oriented Programming, which is a component part of rich client application type, it has rich UI functionality, and improved user experience. It support offline and occasionally connected scenarios, according to the client's desires. Together with the databases management systems, can process a huge amount of data with minimum time consumption, automating paper work and multiple calculations. For this prototype was used MySQL DBMS because it is an open source, it has easy statements for processing data and it is easy to understand the structure of the created database (using tables and foreign keys). It is working with structured data which is optimal for small companies.

In comparison to the NoSQL database model, which is more useful for analyzing big data stores, and is non-relational DBMS, MySQL has enough power and flexibility to work with small and medium scale companies. MySQL can handle around 8 TB of data, having many reporting tools, which can help to prove application validity, and uses standardization, which is a significant point in the database industry.

LIST OF SOURCES USED

[1] MURPHY, NATE. IBM: (2012) *A Brief History of Application Development: Exploring the capabilities of IBM InfoSphere Optim data lifecycle management solutions* [online]. [cit. 2015-02-10]. Available at: http://ibmdatamag.com/2012/08/a-brief-history-of-application-development

[2] MICROSOFT: (2009) *.NET application architecture guide. 2nd ed. Redmond, Wash.: 524 p. Patterns & practices.* ISBN 073562710x. Available at: https://msdn.microsoft.com/en-us/library/ff650706.aspx

[3] GLASS MICHAEL: (2004). *Beginning PHP, Apache, MySQL web development* [online]. Indianapolis, Ind.: Wiley, 2004, xx, 700 p. [cit. 2015-03-20]. 720 pages. ISBN 07-645-5744-0. Available at: http://www.wrox.com/WileyCDA/WroxTitle/Beginning-PHP-Apache-MySQL-Web-Development.productCd-0764557440,descCd-DOWNLOAD.html

[4] AZUMA M., VANÍČEK J.: (2001). *SQuaRE: Next Generation of ISO/IEC 9126 & 14598. In: EurOpen CZ. XVIII konference Dolní Malá Úpa: 1–16 pages.*

[5] MEYER BERTRAND: (2013). *Touch of class: learning to program well with objects.* New York: Springer p. cm. ISBN 9783540921448.

[6] PREDA MIRCEA CEZAR: (2010). *Introducere in programarea orientata obiect. Concepte fundamentale din perspectiva ingineriei software*. Romania, Iași: Polirom, 01 Jan 2010. ISBN 9789734616299.

[7] JACOBSON IVAR: (1992). *Object-oriented Software Engineering: A Use Case Driven Approach.* New York: ACM.

[8] CARDELLI LUCA and PETER WEGNER: (1985). *On understanding types, data abstraction, and polymorphism.* ACM Computing Surveys. vol. 17, issue 4, s. 471-523. DOI: 10.1145/6041.6042. Available at: http://portal.acm.org/citation.cfm?doid=6041.6042

[9] DARIESCU, NADIA: (2009) *Drept procedural fiscal. Romania: Editura Lumen*. 233 pages. ISBN 9789731661438. Available at: Https://books.google.cz/books?id=V3z7U3E0ZfAC&amp;dq=Cod+de+Identificare+Fiscal%C4%83&amp;source=gbs_navlinks_s

[10] HAILPERN, B.: (1986) *Guest Editor's Introduction Multiparadigm Languages and Environments. IEEE Software.* 1986, vol. 3, issue 1, s. 6-9. DOI: 10.1109/MS.1986.232426. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1695465

[11] ORACLE (2014). *PeopleSoft Online Help: Defining Contract Statuses* [online]. 2014 [cit. 2015-03-26]. Available at: http://docs.oracle.com/cd/E52390_01/fscm92pbr3/eng/fscm/fcam/task_DefiningContractStatuses-9f6074.html

[12] *Overview of the four main programming paradigms* [online]. July 2, 2013, July 2013 [cit. 2015-03-02]. Available at: http://people.cs.aau.dk/~normark/prog3-3/html/notes/paradigms_themes-paradigm-overview-section.html#paradigms_imperative-paradigm-overview_title_1

[13] KAISLER, STEPHEN H.:(2005) *Software Paradigms*. United States: John Wiley & Sons. ISBN 0471703575. Available at: https://books.google.cz/books?id=kfGHwo2E0FYC&dq=functional+paradigm+characteristics+ppt&source=gbs_navlinks_s\

[14] SCHWARTZ, Baron, Peter ZAITSEV a Vadim TKACHENKO: (2012) *High Performance MySQL: Optimization, Backups, and Replication*. 3. vyd. United States: "O'Reilly Media, Inc.", 2012. 828 pages. ISBN 1449332498, 9781449332495.

[15] Republic of Moldova. *LEGE Nr. 619 din 08.11.2001 pentru modificarea şi completarea Legii nr.112-XIII din 20 mai 1994 cu privire la susţinerea şi protecţia micului business. In: LP112/1994. Republic of Moldova: PARLIAMENT, 2001*. Accessible at:

http://lex.justice.md/viewdoc.php?action=view&amp;view=doc&amp;id=33325&amp;lang=1

[16] ORACLE USA, Inc.: (2014) *MySQL Connector/ODBC Developer Guide: Chapter 1 Introduction to MySQL Connector/ODBC* [online]. Oracle Corporation, 2005, 2014 [cit. 2015-03-11]. Available at: http://dev.mysql.com/doc/connector-odbc/en/preface.html

[17] EMBARCADERO TECHNOLOGIES: (2014) Inc. *Turbo C++ Community: C++Builder XE7* [online]. Embarcadero Technologies, Inc., [cit. 2015-03-13]. Available at: http://www.turboexplorer.com/video/cpp

[18] SMITH, David: (2014). *Revolutions. Learn more about using open source R for big data analysis, predictive modeling, data science and more from the staff of Revolution Analytics: IEEE ranks R #9 amongst all languages* [online]. [cit. 2015-03-25]. Available at: http://blog.revolutionanalytics.com/2014/07/ieee-ranks-r-9-amongst-all-languages.html

[19] MICROSOFT, (2015). *Office: Visio Professional 2013* [online]. United States: Microsoft [cit. 2015-03-25]. Available at: http://products.office.com/en-us/Visio/visio-professional-2013-business-and-diagram-software

[20] OBJECT MANAGEMENT GROUP: (2015), Inc. *Unified Modeling Language™ (UML®) Resource Page: Introduction to Unified Modeling Language* [online]. Object Management Group, Inc., 1997-2015 [cit. 2015-03-25]. Available at: http://www.omg.org/gettingstarted/what_is_uml.htm

LIST OF FIGURES

LIST OF TABLES