SULEYMAN DEMIREL UNIVERSITY
DEPARTMENT OF POST-GRADUATE EDUCATION

UDK 004.415.25

ORYNBEKOVA KAMILA YERMUKHANBETOVNA

Development of system automating of sorting letters at the post office

Specialty: "6M070400 – Computing systems and software"

Academic Degree: Master of Technical Sciences

"Admitted to defense":
Director of the department of
Post-graduate education
_____ PhD. Mussabekov M.N.
«01» 06 2017 г.

Head of Department PhD. Alimanova M.O.

Scientific advisor PhD. Suliyev R.N.

Kaskelen, 2017

ANNOTATION

Development of postal service starts from 6 thousand years BC. Actually, first messages were send and service starts later, but these tam-tam drums laid the foundation of post service. During evolution of humanity people met problems, and then they tried to find solutions, such a smoke camp fire or pigeon.

Time decreasing consumption system proposes the most optimal rack to store the package, optimal by size and distance between rack and delivery stake. Algorithm is intended to decrease time for searching the empty racks and the racks, which have been already, stored the packages. It will make delivery and sorting process at receiving much faster and also will optimize consuming free space in racks.

Automating sorting algorithm is a system, which proposes the most optimal rack to store the package, optimal by size and distance between rack and delivery stake. Algorithm is intended to decrease time for searching the empty racks and the racks, which have been already, stored the packages. It will make delivery and sorting process at receiving much faster and also will optimize consuming free space in racks. During delivery, post official will exactly know, in which rack does the package stored and during sorting, it wouldn't be necessary to look for optimized rack to place package; in case if post official will not look for optimized rack, optimized consumption of space will not be actual. In developing of algorithm Mathematics and Statistics Science were used. And there are intentions for developing algorithm, working with Internet of Things methodology in future.

Key words: *post, post office, history, algorithm, optimize, time, database, statistics, development, evolution.*

ТҮЙІН

Б.з.б 6-ыншы мыңжылдықта пошта сервисінің дамуы өз бастауын алады, там-там барабандары хабарлама жіберудің негізін қалыптастырды. Эволюция процесінде адамзат көптеген қиындықтарға тап болды, сигналдық алаулар және пошта көгершіндері кейбірінің шешімі болып табылды.

Қолдану уақытын қысқарту жүйесі сәлемдемені ұтымды әрі ара қашықтығы және көлемі бойынша оңтайлы, сөреде сақтау әдісін ұсынады. Алгоритм оңтайланған бос немесе сәлемдемені сақтаудағы сөрені іздеудегі уақыттын төмендетуге бағытталған.Осылайша, сәлемдемелерді өткізу және қабылдап алу жылдамырақ жүзеге асырылатын болады, сондай-ақ қосымша сөрелерде кеңістікті тұтынуы оңтайландырылады.

Пошта бөлімшесіндегі орамахатты сұрыптау алгоритмы – бұл пошта жүйе мөлшерін, сондай-ақ беру бекетінің қашықтығын ескеретін ең оңтайлы сақтау сөресін ұсынады. Алгоритм бос және де орамахатты сақталынған сөрелерді іздеу уақытын қысқартуға арналған. Бұл орамахатты беру және сұрыптау жүйесін айтарлықтай жылдамдатады, сондай-ақ бос сөрелерді қолдануды оңтайландырады. Орамахатты беру кезінде пошта болімшесінің қызметкері орамахатының дәл қайда орналасқанын нақты білетін болады, сондай-ақ, сұрыптау кезінде қызметкерге орамахатты сақтауға арналған оңтайлы сөрені арнайы іздеуге қажет болмайды, егер қызметкер оңтайлы сөрелерді іздемеген жағдайда, бос сөрелерді оңтайландыру өзекті болмайды. Алгоритмди әзірлеген кезде Математика жане Статистика ғылымдары қолданылған. Ал болашақта Internet of Things әдістемесімен жұмыс істейтін алгоритмін әзірлеу жоспарлары бар.

Кілт сөздер: *пошта, пошта бөлімшесі,тарих, алгоритм, оңтайландырылған, уақыт, деректер базасы, статистика, дайындау, эволюция.*

АННОТАЦИЯ

Развитие почтового сервиса начинается с 6-ого тысячелетия до н.э., там-там барабаны положили основу отправки сообщений. В процессе эволюции человечества люди сталкивались с различными проблемами, решениями некоторых из них стали сигнальные костры и почтовые голуби.

Система сокращения времени потребления предлагает наиболее оптимальную полку для хранения посылки, оптимальная по размеру и по расстоянию. Алгоритм предназначен для уменьшения времени поиска оптимальной пустой или уже хранящей в себе посылок полки. Таким образом выдача и прием посылок будет осуществляться значительно быстрее, а также оптимизируется потребление пространства на полках.

Алгоритм сортировки посылок в почтовом отделении - это система, предлагающая самую оптимальную полку для хранения посылки, как по размеру, так и по расстоянию от поста выдачи. Алгоритм предназначен для уменьшения времени поиска пустых а так же уже хранящий в себе посылок полок. Это значительно ускорит процесс выдачи и сортировки при получении, а так же оптимизирует потребление свободных полок. При выдачи, сотрудник почтового отделения будет точно знать где лежит посылка, а при сортировке сотруднику не придется самому искать оптимальную полку для хранения посылки, в случае, если сотрудник не будет искать оптимальную полку, то оптимизация свободных полок не буде актуальна. При разработке алгоритма использовались науки Математика и Статистика.

Ключевые слова: *почта, Почтовое отделение,история, алгоритм, отимизация, время, база данных, статистика, разработка, эволюция.*

CONTENTS

ТҮЙІН

Б.з.б 6-ыншы мыңжылдықта пошта сервисінің дамуы өз бастауын алады, там-там барабандары хабарлама жіберудің негізін қалыптастырды. Эволюция процесінде адамзат көптеген қиындықтарға тап болды, сигналдық алаулар және пошта көгершіндері кейбірінің шешімі болып табылды.

Қолдану уақытын қысқарту жүйесі сәлемдемені ұтымды әрі ара қашықтығы және көлемі бойынша оңтайлы, сөреде сақтау әдісін ұсынады. Алгоритм оңтайланған бос немесе сәлемдемені сақтаудағы сөрені іздеудегі уақыттын төмендетуге бағытталған.Осылайша, сәлемдемелерді өткізу және қабылдап алу жылдамырақ жүзеге асырылатын болады, сондай-ақ қосымша сөрелерде кеңістікті тұтынуы оңтайландырылады.

Пошта бөлімшесіндегі орамахатты сұрыптау алгоритмы – бұл пошта жүйе мөлшерін, сондай-ақ беру бекетінің қашықтығын ескеретін ең оңтайлы сақтау сөресін ұсынады. Алгоритм бос және де орамахатты сақталынған сөрелерді іздеу уақытын қысқартуға арналған. Бұл орамахатты беру және сұрыптау жүйесін айтарлықтай жылдамдатады, сондай-ақ бос сөрелерді қолдануды оңтайландырады. Орамахатты беру кезінде пошта болімшесінің қызметкері орамахатының дәл қайда орналасқанын нақты білетін болады, сондай-ақ, сұрыптау кезінде қызметкерге орамахатты сақтауға арналған оңтайлы сөрені арнайы іздеуге қажет болмайды, егер кызметкер оңтайлы сөрелерді іздемеген жағдайда, бос сөрелерді оңтайландыру өзекті болмайды. Алгоритмди әзірлеген кезде Математика жане Статистика ғылымдары қолданылған. Ал болашақта Internet of Things әдістемесімен жұмыс істейтін алгоритмін әзірлеу жоспарлары бар.

Кілт сөздер: *пошта, пошта бөлімшесі,тарих, алгоритм, оңтайландырылған, уақыт, деректер базасы, статистика, дайындау, эволюция.*

АННОТАЦИЯ

Развитие почтового сервиса начинается с 6-ого тысячелетия до н.э., там-там барабаны положили основу отправки сообщений. В процессе эволюции человечества люди сталкивались с различными проблемами, решениями некоторых из них стали сигнальные костры и почтовые голуби.

Система сокращения времени потребления предлагает наиболее оптимальную полку для хранения посылки, оптимальная по размеру и по расстоянию. Алгоритм предназначен для уменьшения времени поиска оптимальной пустой или уже хранящей в себе посылок полки. Таким образом выдача и прием посылок будет осуществляться значительно быстрее, а также оптимизируется потребление пространства на полках.

Алгоритм сортировки посылок в почтовом отделении - это система, предлагающая самую оптимальную полку для хранения посылки, как по размеру, так и по расстоянию от поста выдачи. Алгоритм предназначен для уменьшения времени поиска пустых а так же уже хранящий в себе посылок полок. Это значительно ускорит процесс выдачи и сортировки при получении, а так же оптимизирует потребление свободных полок. При выдачи, сотрудник почтового отделения будет точно знать где лежит посылка, а при сортировке сотруднику не придется самому искать оптимальную полку для хранения посылки, в случае, если сотрудник не будет искать оптимальную полку, то оптимизация свободных полок не буде актуальна. При разработке алгоритма использовались науки Математика и Статистика.

Ключевые слова: *почта, Почтовое отделение,история, алгоритм, отимизация, время, база данных, статистика, разработка, эволюция.*

CONTENTS

INTRODUCTION

During evolution of humanity people met problems, and then they tried to find solutions. Development of modern post service is not differing. After up building huge cities, people trying to optimize their time consuming for secondary business, like sending or receiving post packages, so people count every minute spending in post office.

This thesis is dedicated to save post services` clients` time by performing sorting by algorithm. Automating sorting algorithm is a system, which proposes the most optimal rack to store the package, optimal by size and distance between rack and delivery stake. Algorithm is intended to decrease time for searching the empty racks and the racks, which have been already, stored the packages. It will make delivery and sorting process at receiving much faster and also will optimize consuming free space in racks.

Python programming language and SQLite database management system were chosen to implement this software.

Results of questioning

100 persons were asked one main question and answers in percents are below in Diagram 1:

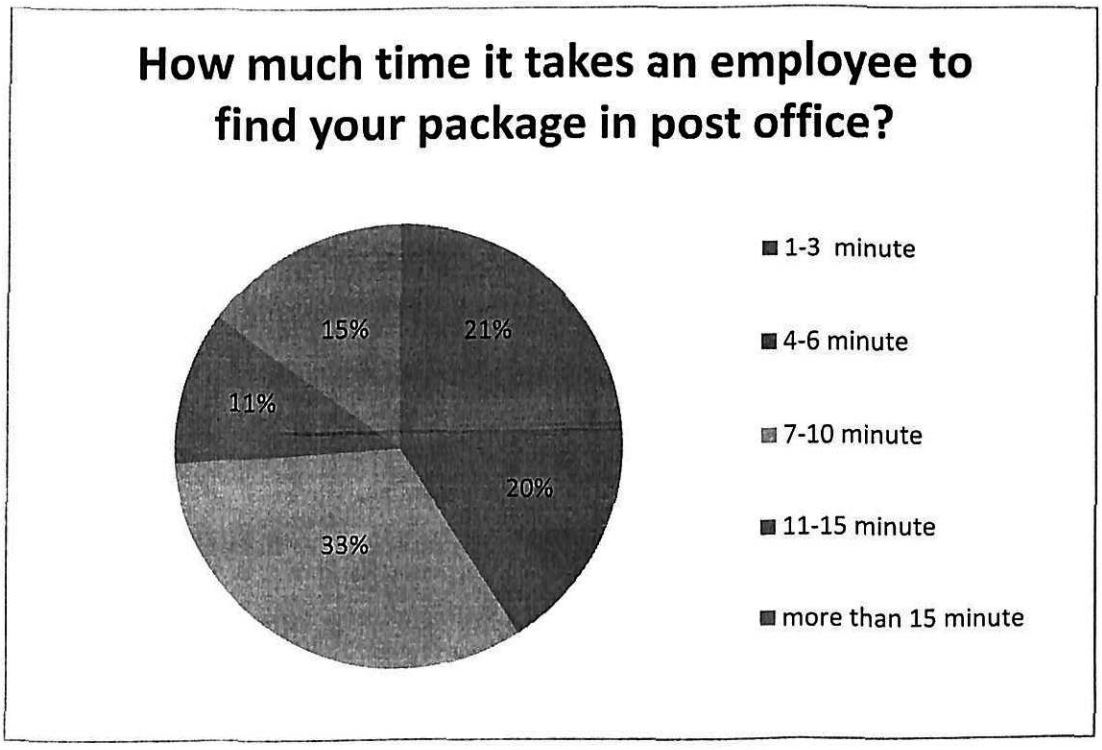

Diagram 1

Coming from questioning, you can see that relevance of this thesis is obvious. Even if there is 21% of fast finding, 33%, 11% and 15% are still being. So it is very big percentage and it must be decreased by automating sorting algorithm.

1. POST SERVICE

A post office is a customer service facility forming part of a national postal system. Post offices offer mail-related services such as acceptance of letters and parcels; provision of post office boxes; and sale of postage stamps, packaging, and stationery. In addition, many post offices offer additional services: providing and accepting government forms (such as passport applications), processing government services and fees (such as road tax), and banking services (such as savings accounts and money orders).

1.1 History of post services

Prehistoric people used voice to convey important information, which contributed to the emergence of articulate speech. However, the oral transmission of messages was imperfect, since the human voice is heard only at close range. To enhance the transmitted sound of a millennium ago, hollow tree trunks and later drums (about 6 thousand years BC) began to be used. With the help of conditional shock signals, news was transmitted from one settlement to another. In addition, people used to transmit news fire and smoke. Tam-tam drums are still used for long distance communications by African tribes [1], and smoke from campfires as far back as the twentieth century was used for the same purposes by the Indians of Canada [2].

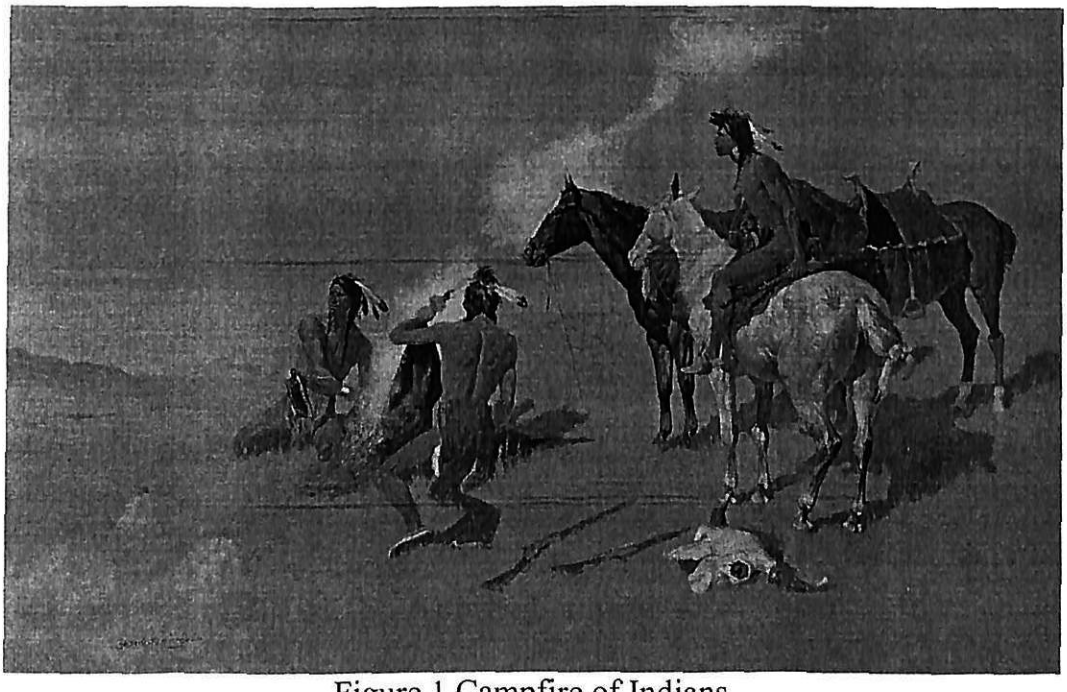

Figure 1 Campfire of Indians

The next type of connection in the history of mankind was the messengers - first foot, later mounted. In the ancient states of Mesopotamia, Egypt, Greece, Persia, China, the Roman Empire, there was a well-established state postal communication: written messages were sent with pedestrians and mounted messengers on the principle of a relay [3] [4]

Figure 2 Tam tam drums of Indians

The history of the post is closely related to the history of writing. With the nception of the latter information was sent in a written form, which marked the eginning of the postal service. Initially, this connection was episodic. With the mergence of slave states in the Ancient East, whose rulers needed constant nformation about the situation in their own country and in their territories, postal ommunication began to acquire an orderly character [5].

Figure 3 Mounted messenger

The first institutions of an orderly messaging service arose early enough in ncient times [5]. For the first time, a postal message appeared about 5,000 years

ago in Mesopotamia to transmit information imprinted in the form of a clay letter. No less ancient is the postal service in Egypt [4].

Initially, such services were used mainly for military purposes and were not intended for communication between civilians. Such were the communications services in Ancient Egypt, Assyria, Babylon and Persia. They were especially developed in Egypt and can be considered the forerunners of modern postal [5]. During the Fourth Dynasty of the Pharaohs (2,900-2,700 BC), a special hiking service (hikers), as well as horse riders, operated along the military roads with Libya, Ethiopia and Arabia [4].

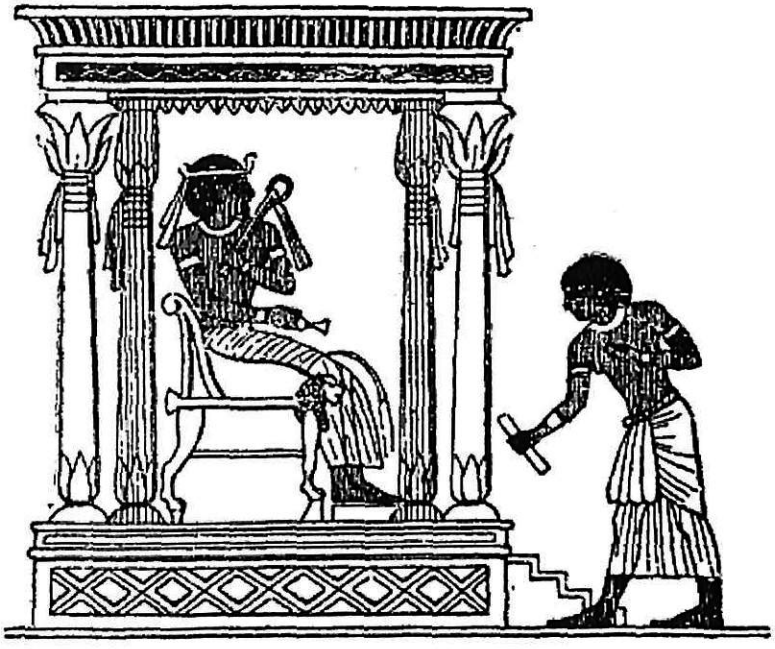

Figure 4 Messenger of Ancient Egypt

Ancient Egyptian mail was based primarily on the use of numerous pedestrians, thanks to which the pharaohs could easily communicate with remote

Figure 5 Egypt papyrus

provinces. In Beni-Hassan, on the wall of one of the tomb caves dating back to the Middle Kingdom, a messenger is depicted conveying a message to the official about the invasion of the enemy tribe. [2] It is known about the professional messengers that existed in Egypt during the XII dynasty (1985-1785 BC), which carried royal orders up to Asia. The chasers had to overcome long distances as soon as possible. For the transportation of letters, pigeons were also used.

The idea of a postal business in ancient Egypt can be obtained from a document on papyrus, dated about 255 BC. E. And containing the mail delivery account by one messenger. Such a highly developed postal system soon began to spread in other countries.

Figure 6 Pigeon messenger

Postal business in the Persian monarchy - in the form of a clear system of postal communication, known as the "angareion" - was introduced in the VI century BC. [4], at the time of King Cyrus II (550-529 BC). The transmission of messages was carried out mainly through horse messengers (hangars). However, there are indications that such a postal system existed in Persia much earlier. It is known from the descriptions of Herodotus and Xenophon [2] that, under Cyrus II, postal stations were set up on the most important roads, separated from each other at a uniform distance, which was approximately the horse's daytime run. These stations served couriers for recreation [6] [7].

In Greece, the postal system was well established in the form of land and sea mail services [4], but it could not significantly develop because of the multitude of city-states fighting among themselves. Governments for the transmission of

messages had, as a rule, at their disposal, the hoshest messengers. They were called hemedromes (Greek) [8]. Runners-runners overcame the distance of 55 stages (about 10 km) per hour and 400-500 stages for one flight [2].

Figure 7 Ancient Greek messenger

The most famous of these couriers was the Philippines, which according to the legend of Plutarch in 490 BC. E. Reported to Athens about the victory in the Battle of Marathon and died of exhaustion. This run was the first Marathon in history. Philipid gave only an oral message.

To send especially urgent messages, in the ancient times, they sent messengers. As Diodorus of Sicily writes, one of the commanders of Alexander the Great kept at his staff messengers - horsemen on camels [2].

In the Roman Republic, the postal order was probably borrowed from the Persians. In the beginning, only rich patricians who owned numerous slaves had their own messengers [2]. For government and private purposes, there were messengers, as well as private entrepreneurs who hired wagons and pack animals. Their colleges were united in one corporation. Gaius Julius Caesar laid the foundations for the creation of the state mail proper, which arose and was greatly developed under Emperor Augustus. In those days the post was called cursus publicus, was subordinated directly to the emperor and was not allowed for private messages. Thanks to a single postal network, there was a link between the separate parts of the Roman Empire [2]. It was a huge, branched postal system that operated according to strict regulations [4].

Postal shipments were carried out by land with the help of horses, by sea - by ships. In larger centers, post-office stations were established, which served for

rest and overnight travel of horsemen and drivers and usually stood one on the other for the day's journey. Standing and pack animals and, if necessary, wagons stood here [2]. Between each two mansiones (at a distance of 7-14 km), 6-8 smaller stations were constructed to change the horses. In those days they could say: "Statio posita in ..." which meant "station located in such and such a place". From the Latin word "posita", the most likely word was "post" [2].

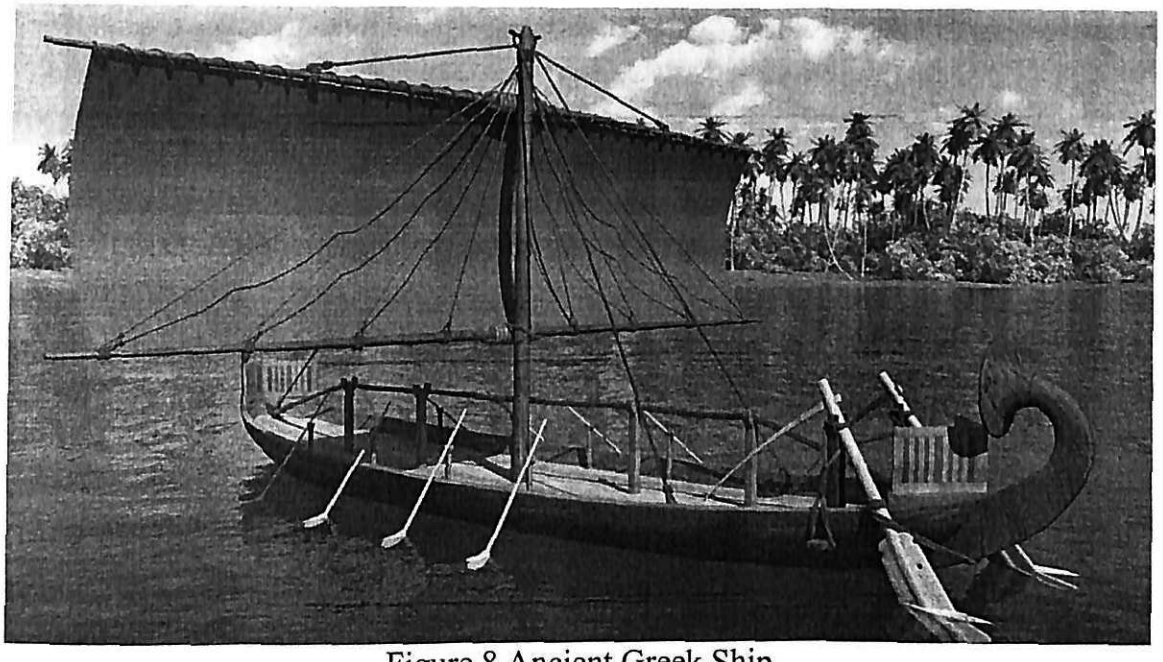

Figure 8 Ancient Greek Ship

Hasty packages were sent through horse couriers, traveling were transported in light wagons, various kinds of luggage - in carts. The use of state mail was allowed only for state purposes and certain officials. In urgent cases, entire military detachments were transported by means of cursus publicus. As an exception, and on the basis of special permits, other persons, traveling officials, especially veterans, and later - clerics could use state mail, which gave rise to various abuses.

The main management of the state mail was concentrated in the hands of one of the highest state officials: first prefect of the praetorian, and from the time of Constantine - the master of offices. Post office management in the provinces belonged to the governors, under which the technical part of the mail consisted of special prefects, later - procuratores cursus publici. The supply of horses, other vehicles and riders was a natural obligation of the surrounding population and residents of the conquered countries, where the content of the mail was a very heavy burden.

Although the activities of Roman state mail were limited to government needs, it was of great importance. Thanks to an excellent network of roads, security and order of communications, as well as extensive correspondence between civil and military authorities, an unusually lively movement developed at public post stations. Distances from Britain to the Balkans, the Caucasus, present-day Turkey, Syria, Palestine and Jordan and from the mouth of the Rhine to the Libyan Desert and Alexandria could be overcome relatively quickly. If Caesar,

using variable private horses, could do 100 miles a day, then Tiberius, with the help of cursus publicus, traveled twice as much a day. Of the most important provinces, the news was received daily in Rome. At stations along the busiest roads, there were 20-40 harnessed horses and mules. This organization lasted until the fall of the Western Roman Empire, with its fall, and cursus publicus. In the Eastern Roman Empire, the state post existed until about 520.

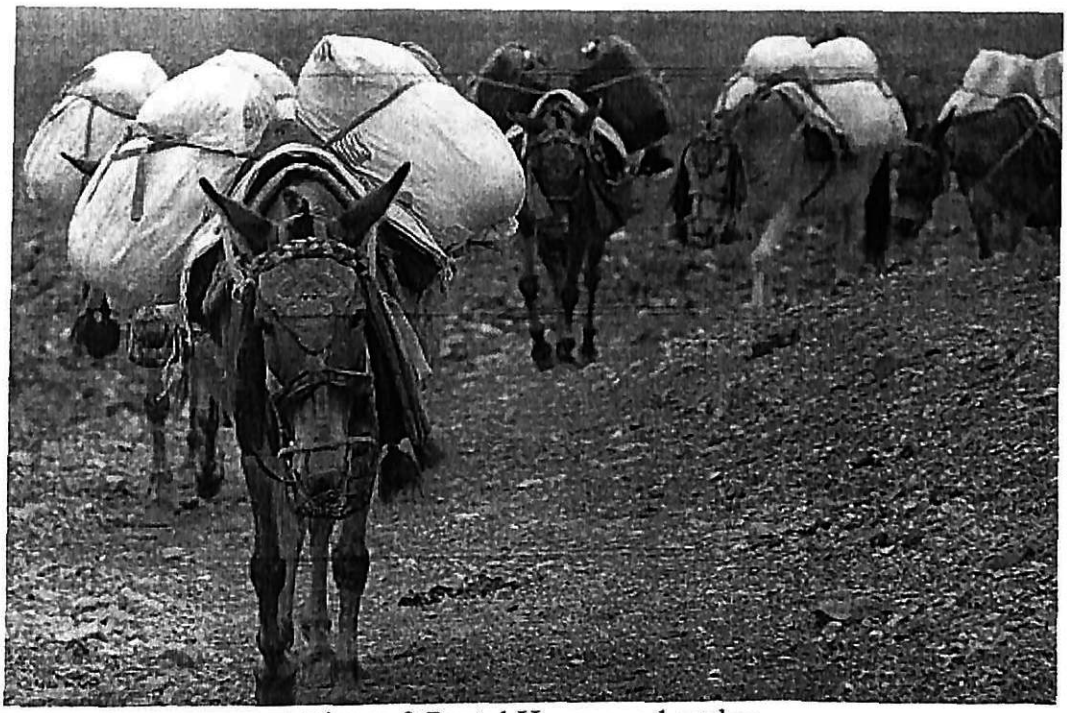

Figure 9 Postal Horses and mules

For private mail, they usually used the services of traveling friends, which caused a long delivery time. So, there is a case when someone Augustine received a letter in nine years. If the distance to the addressee was not very large, the Roman sent his slave, who on foot crossed the day to 75 km [6] [8].

Other highly developed cultures

In China, the postal service of foot and horse messengers arose quite early [4]; It was founded during the Chou dynasty (1123-249 BC). In those days, the postal

Figure 10 Chou dinasty

message was maintained with the help of 80 messengers and eight main couriers, for which at a distance of 5 km were arranged food for food and at a greater distance - items for overnight. This postal system was significantly expanded under the Qin dynasty (221-206 BC) and especially under the Han dynasty (206 BC-220).In the heyday of the Mayan culture, there was also a well-developed messenger service, but little is known about it [8].

Figure 11 Clovis - king of France

With the fall of the Western Roman Empire in Europe, there was hardly any functioning system for conveying news. Only Clovis (king of France from 482 to 511) tried - without much success - to recreate the postal connection from the remnants of Roman state mail. By the time of Charlemagne (768-814), messages were delivered with great difficulty. Charlemagne and his successors did not make serious attempts to restore Roman state mail. The establishment of the messengers, which existed under the Carolingians, adjoined the people's division into stamps and with the rapid disintegration of the monarchy did not receive wide development. The princes-feudal lords sent letters and things through the messengers and the drivers who were provided by their subjects.

Monastic and university mail

In the feudal medieval Europe of the 11th-15th centuries, with the fragmentation of state power, the dispatch of news was mainly undertaken by individual spiritual and secular corporations. In the exchange of thoughts, the church was most in need then, both because its structure rested on the beginning of centralization, and because for a long time it was the only bearer of the intellectual life of peoples. Archives of church institutions and the registers of the Roman Curia testify that at the very beginning of the Middle Ages there was a lively exchange of messages between the head of the Catholic hierarchy and its members; But there is no

indication of the existence of a special church institute of couriers or couriers. Only between numerous ramifications of spiritual orders were the right relations maintained through mediocre wandering monks, who served as messengers and took with them reports. Monasteries thus had their own system of communications - monastic mail. Monastic couriers maintained contact between individual monasteries and the head of the church in Rome, between monastic orders and their brotherhoods. In the lands of the German order for this purpose a special administration arose and stations for changing horses were established.

At universities, where students flocked from a variety of countries, corporations of professional messengers also enjoyed different privileges. In the XII-XIII centuries, the university's messengers were famous in Bologna, Salerno, Naples, Montpellier, Toulouse, later - the messengers of the Sorbonne University in Paris. University post messengers maintained contact between students and their families; Some of the university post for a fee delivered to private individuals.

The further development of society, primarily trade and crafts, and also of science and culture, contributed to an increase in interest in the transmission of messages and led to the emergence of numerous and varied services of messengers and mails of cities serving merchants and artisans. Gradually, the right to use these mails became available to other segments of the population.

Merchant mail was opened at large trading houses, which contained their own couriers. Soon, individual merchants began to borrow this idea and combined so that the collected mail could be transported to destination. The beginning of the merchant's mail can be found in the Republic of Venice. At the same time, there was still no single state mail.

Somewhat later, the so-called "butchers' mail" arose. The shop of butchers, who made extensive trips for their purchases, assumed the transportation of letters and parcels, in agreement with cities and merchant guilds. In some cities of southern Germany, it was imputed to the butcher shop as a duty, in exchange for which it was exempt from community service. Thus, a post of butchers, which existed until the end of the 17th century, was formed, and in places was given the status of a state institution.

With the development of urban liberties, one of the most important means of communication in the Middle Ages was the institution of city messengers, which from the fourteenth century existed almost everywhere, but developed especially in large shopping centers in Germany and Italy. Of the many regulations that have survived to the city's messengers in Cologne, Mainz, Nordhausen (XIV century), Strasbourg (1443), Augsburg (1552), Breslau (1573), etc., it can be seen that they were in charge of the city council, which Swearing oath to obey. They did not receive salaries from the community, or from individual corporations or merchant guilds. Speaking from the city on certain days, they were on horseback or on foot in a timely manner, delivered to the destination correspondence of the city administration, as well as letters and parcels of citizens, from which they charged a fee. The institute of urban messengers was firmly and widely developed thanks to the alliances of cities on the Rhine and in Lower Germany. Riders of the Rhine

Town Union supported the right messages from Cologne and Mainz via Frankfurt to Nuremberg. The couriers of the Hanseatic cities, maintaining the connections between Hamburg, Bremen, Amsterdam and Antwerp, and also to the east through Stettin, Danzig and Königsberg all the way to Riga, were famous for their accuracy in meeting deadlines. In southern Germany, the first place was occupied by the messengers of Augsburg. In addition to the lines to Nuremberg (three times a week), Lindau and Regensburg, they maintained communications with Italy; To Venice they arrived via Brenner in eight days.

Modern centralized mail originated with the strengthening of state power. In France, Louis XI edict June 19, 1464 established the royal couriers (fr. maîtres coureurs royaux). On all his possessions there was a network of stations for changing horses; At the head of the whole organization stood the grand maître. This post was intended solely for the needs of the government; Royal couriers under the death penalty were forbidden to carry out assignments of private individuals. In the Charles VIII patent of January 27, 1487, the royal couriers are named chevaucheurs en postes. Shortly thereafter, not only in France, but also in Germany and Italy, the name of the post office meant all the sets of regulations that were established by the state or under the control of the state for the forwarding of both governmental and private correspondence and for the carriage of passengers.

Figure 12 Franz von Taxis

The first experience in the organization of mail, in the most real sense of the word and on a broad international basis, was made by members of the Tasso family (later Tassis or Taxis, in addition to the name Tourn) from Bergamo, who accepted the messages between the Habsburg possessions. The post office of Turn-and-Taxis existed from the second half of the 15th century until 1867 and made a tremendous contribution to the development of postal communication in Europe [2] [5] [7] [8].

After the fall of the Western Roman Empire in its territory from Spain to Persia, the domination of the Arabs was established. In the 7th century, a well-organized regular postal service appeared in the Muslim caliphate, which private individuals could use to a limited extent. The existing postal services became even more developed. Postal couriers had special insignia to be recognized from afar. They were given special signs, idiosyncrasies, which were hung around the neck and over the shoulder with the help of yellow ribbons.

There are data on the Inca mail in Peru and the Aztecs in Mexico. Here, before the beginning of the 16th century, postal messengers existed, which, in addition to state communications, brought fresh fish, fruits and other products to the Tsar's table.

Ghosts of the Aztecs passed almost all the messages verbally. They wove red ribbons in their hair or waved a dagger at the joyous news (for example, about victory); Bad news was transmitted to the king, kneeling [2] [7] [8].

In the 16th-17th centuries, a centralized royal post appeared in France, Sweden, England and other countries. The idea of postal regalia, that is, the exclusive right of the government to keep postal institutions within the state territory, was first put forward at the end of the 16th century, and in the 17th century it began to be carried out in life. The first of the German princes, who founded the government post and recognized the nature of the monopoly, was the great Elector Frederick William (1646). His example was followed by other

Figure 13 Great Elector Frederic William

significant imperial ranks. At the same time, the content of mail began to be viewed not only as a right, but also as a duty of governments.

With the development of production relations and the emergence of capitalism, it became necessary to organize a regular and rapid postal communication both within countries and between countries. Already at the beginning of the 18th century, there were statesmen (for example, Friedrich-Wilhelm I in Prussia) who abandoned fiscal views of the post office and saw its task in reducing the cost of postal tariffs and possibly making postal communications more affordable for the population. Unlike France, where postal letters were declared a public monopoly, but along with government mail there were private enterprises for the transport of passengers, in the larger German states, the activity of government mail included both the forwarding of letters and goods, and Transportation of passengers.

In those days, the arrival of a postal coach to a small town was a whole event. About the approach the postman loudly blew in a post horn. News was delivered at a speed of 70 km per day - so much traveled the mail diligence [2].

The governmental post of Saxony, Braunschweig-Hanover, Hessen, especially Brandenburg-Prussia, was famous for its expedient organization. On the main Prussian line Kleve - Memel post from 1655 went twice a week; From Koenigsberg to Berlin, she arrived at 4 days, from Koenigsberg to Kleve - in 10 days. This was an extraordinary speed for that time. In addition to the branches to Hamburg, Stettin, Leipzig and Breslau, postal messages were maintained in the west with Holland, in the east - with Warsaw and Swedish mail in Riga. In the business of passenger transportation, Prussia, however, already in the late 18th century was surpassed by countries with more comfortable roads. Moreover, the success achieved by Prussia in 1821, when so-called Germans were established, was struck by his contemporaries. Nagler'sche Schuellposten, with convenient for traveling crews.

In the XIX century a radical revolution in the postal business was caused by the spread of railways and shipping companies. The appearance at the beginning of the XIX century of a steam locomotive and steamer, and at the beginning of the 20th century the aircraft significantly increased the speed of sending mail. Postal communication became a nationwide one and began to serve the entire population.

Figure 14 Steamship line train

By the combination of rail and steamship lines, it was possible to establish correct post messages between the most remote countries. The first experience in this direction was made in 1835 by Lieutenant Veghorn, who was in the English service, who organized the Anglo-Indian mail, which was transported from Marseilles to Alexandria by steamers, from there first through the Mahmudie channel and then by rail to Suez, then again delivered by steamships In Bombay and Calcutta. At the beginning of the 20th century, this mail was delivered through the tunnel of Mont-Seni to Brindisi, from where it was directly transported by mail steamers through the Suez Canal to India and the countries of the Far East.

In 1820, an envelope was invented by a merchant of paper Brever in Brighton. An important milestone in the history of postal communication was the issue of a postage stamp in 1840 in the UK. Later, custom-made envelopes began to be used in England and its colonies.


Figure 15 USA stamp parcel

Stamp parcels appeared in 1857 in the United States of America, in 1864 - in the New South Wallis, in 1868 - in the North German Union; All such parcels were introduced later in 66 countries. Forms for closed letters were introduced in 55 countries, originally in 1879 in Paris; In Argentina and in France there are forms

with a paid answer. Forms for postal orders appeared in Braunschweig in 1865 and then were introduced in 14 countries; Only in three countries there were stamp envelopes for postal money orders.

The idea of the invention of an open letter (post-card, or postcard) belongs to the former German general-post director Henry von Stefan. At the 5th Conference of the German Postal Union in Karlsruhe in 1865, Stephan in a memo noted the inconvenience of the existing form of writing, which was not simple and concise and was associated with a loss of time in choosing paper, folding it, investing it in an envelope, The label of the brand, etc. In addition, it was not customary in an ordinary letter to be limited to short sentences, and under such conditions the speed of writing was not achieved. The initiative to introduce an open letter belongs to Austria, where in 1869 the first postcards appeared. Open letters with a paid response appeared in 1872 in the German Empire. In the future, open letters were introduced in 171 countries, with the answer paid - in 140.

Thanks to the timing of the arrival of the Atlantic steamers with the Pacific Railway trains in North America, and the latter with the shipping lines coming from Vancouver and San Francisco, it became possible to send a letter from Europe to Japan in 30-35 days. Being immediately sent from Japan further (to India), such a letter could make a round-the-world trip in 85 days [10]. With the end of the Great Siberian Railway at the beginning of the 20th century, the journey from Europe to Japan was reduced by six days, and the letter could bypass the globe in less than 80 days.

As the network of railways expanded and branched, and the number of daily trains increased, the number of mail arriving and departing in the area daily increased. This was supplemented by improvements introduced into the organization of the mailing device itself, for example, rural mail, the establishment of a cheap and uniform mail order, the introduction of a number of new postal operations.

With the invention of the Telegraph (1832), the telephone (1876) and the radio (1895), postal communication has not lost its important role of a means of communication of millions of people. In the telegraph office, the post found powerful assistance and completion, as a result of which almost all states, following the example of Germany, combined postal business with telegraphic to a great benefit for both departments.

Finally, international postal relations received a solid foundation and guarantee in the organization of the Universal Postal Union, which embraced all cultural countries. One of the founders of the Universal Postal Union was Heinrich von Stefan, who made a significant contribution to the development of German and international postal services in the 19th century.

In 1874, at the First World Postal Congress, 22 countries, including Russia, signed the Universal Single Postal Agreement and formed the Universal Postal Union (since 1878 - the Universal Postal Union). In 1878, the Universal Postal Convention regulating the exchange of correspondence was concluded, which contains written communications [5] [7] [8] [11].

It is impossible to describe in words how far the influence of evolution and revolutions has gone in the postal service, on land and on the sea. Mail for sure can be called a great achievement of mankind. The world of 1782, which followed the American War of Independence, was concluded only after negotiations lasting more than two years. Prussia and Austria fought in 1866. The campaign took seven days; Seven weeks have passed since the declaration of the war until the formal conclusion of peace. Obviously, the temporary difference in both cases was due only to the fact that in one case the news went on longer, and in the other case it was faster.

The market of unmanned aerial vehicles (UAV) is experiencing a huge rise. Now it has about the same potential as the production of cars 100 years ago. It is obvious that the most important innovative developments in the sphere of UAV and the explosive development of the industry are ahead, and who will offer the world the best technical solutions, he will get a chance to collect all the cream from the market. And it will happen in the very near future. In just two or three years, today popular multicopters and unmanned aircraft risk to make room before the new invention - convertoplan (combines the qualities of a helicopter - take off vertically and hang in the air, and the aircraft - to travel long distances). In any case, this scenario counts Novosibirsk company "Optiplein Unmanned Systems" - a developer of ultra-small UAVs with a variable aerodynamic scheme. Having invented, tested and received orders for the production of a hybrid model, the company intends to launch the world's first lightweight unmanned industrial envelope in mass production until mid-2017.

1.2 World of drones

Inspired by Novosibirsk technicians and inventors to create an unmanned convertoplan, the head and founder of the Amazon.com Internet company Jeff Bezos, simultaneously developing the aerospace company Blue Origin. At the end of 2013, an American entrepreneur announced a delivery service with the help of unmanned vehicles, called Amazon Prime Air.

Figure 16 Amazon postal drone

"The idea of creating a business for me appeared shortly after Bezos uttered his famous phrase - said that in the coming years, Amazon Prime Air will be as frequent in the sky as wagons on highways. After that, many companies around the world turned in the direction of drone. Interest in this market has multiplied ", - recalls the owner and CEO of the company" Optiplein Unmanned Systems "Kirill Yakovenko. And the Novosibirsk businessman wanted not only to make drones, but to create his own network of dronomats - not yet existing in the world network of automatic receiving and issuing small-size shipments of UAVs.

At the time of the creation of the business in 2014, the head of the Optiplein Unmanned Systems had a clear understanding that none of the ultra-small UAVs on the world market meets the requirements for fast, cheap and long-distance (tens and hundreds of kilometers) delivery of goods through the air. "We went to the market at the moment of colossal public interest in multi-copters. By this time, a large number of models appeared on the market, but if they met the requirements of private photo and video shooting, they were completely inadequate for the delivery of goods or for long-term monitoring. Multicopters have too low horizontal speed, short range and short duration of horizontal flight, - says Kirill Yakovenko.

1.3 Kazakhstan post service

The joint-stock company Kazpost is a Kazakh company, the operator of the Kazakh national postal network. The headquarters are in Astana. Member of the Universal Postal Union, a principal participant in international payment systems VISA International and MasterCard Worldwide.

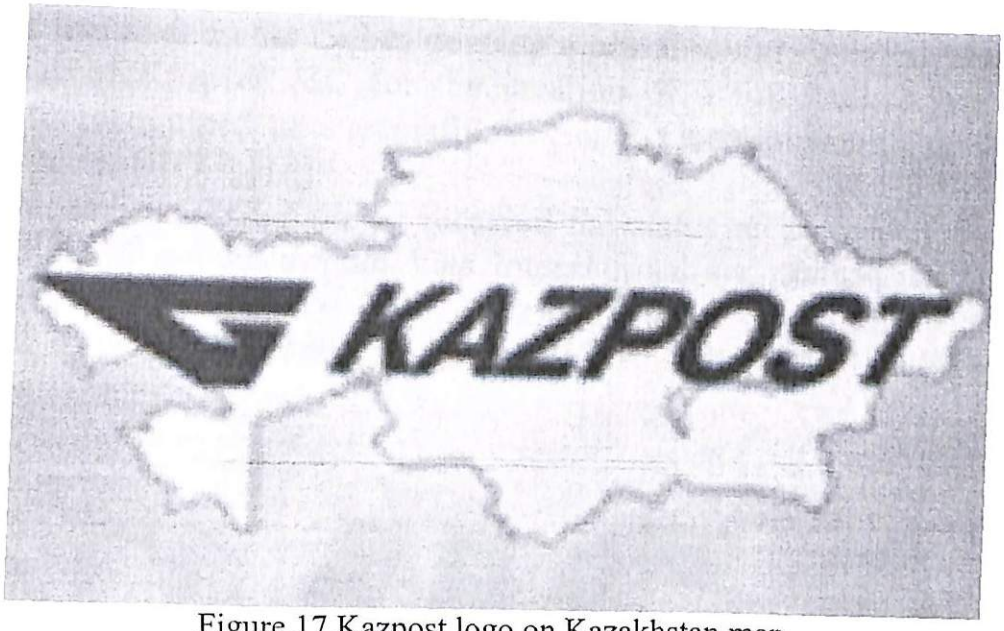

Figure 17 Kazpost logo on Kazakhstan map

The postal branch of Kazakhstan conducts its history from the pre-revolutionary and Soviet periods. With the acquisition of Kazakhstan's independence, the history of the formation and activities of Kazpost JSC as a national postal operator and legal entity began.

Since 1992, Kazakhstan is a full member of the Universal Postal Union (UPU) [9]. On April 5, 1993, according to the resolution of the Cabinet of Ministers of Kazakhstan "On improving the structure of management of the telecommunications industry in the Republic of Kazakhstan", mail and telecommunications were divided. In November 1995, the State Post became an independent economic entity, having been transformed into the Republican State Postal Communication Enterprise (RGPS) [10].

In the summer of 1999, a radical reform of the postal service began after the adoption of the government's decision of May 27, 1999, "On measures to stabilize and improve the financial health of the postal industry." On December 20, 1999, the RSEPS was transformed into Open Joint-Stock Company Kazpost with 100% state participation in the authorized capital; It was registered with a total capital of KZT903.66 million, the initial authorized capital of the company was formed exclusively in the form of buildings and equipment. The situation was exacerbated by the severe financial condition of the postal industry - the debt of the RSHPP for the period from 1993 to 2000 to accumulative pension funds, wages and taxes was KZT 140 million, the accumulated losses of past years for these years amounted to more than KZT250.6 million [10] .

The program for the development of the postal industry and the formation of the postal savings system for 2000-2003 marked the beginning of the creation of a postal savings system in Kazakhstan on the basis of a retail network of postal communication departments. As a source of financing for its implementation, a loan from the Islamic Development Bank in the amount of $ 9 million for state guarantees, the issue of domestic bonds worth KZT1.4 bn., As well as the constant increase of the company's authorized capital by the government. Kazakhstan

became the first state in the CIS to develop a postal savings system. The results of the activities of Kazpost JSC for the creation of a full-fledged postal savings system were recognized as successful Regional Commonwealth in the field of Communications (RCC) [10].

In December 2007, Kazpost received the status of an associate member of the international payment system Visa International and started issuing payment cards Visa [10]. In January 2013, Kazpost became a principal member of MasterCard Worldwide, since August 2013 - a principal member of Visa International. [11]

Figure 18 Visa Master Card logo

In February 2008, a subsidiary company, Kazpost GmbH, was established in Germany together with German partners to perform logistics functions, catalog trading, collecting and forwarding mail correspondence from Germany to Kazakhstan [10].

On December 19, 2012, within the framework of the postal network modernization carried out by Kazpost, the first 24-hour post office in Kazakhstan was opened in Astana. The first 24-hour customer service area "Post-24" provides postal and financial services, servicing employees of more than 30 ministries, departments, state holdings, national companies and residents. [12] [13].

The transformation program

The goal of the Transformation Program is to standardize and optimize business processes and conditions that ensure the quality of service delivery, increase the efficiency of the company's activities, create a single information space and corporate culture.

The transformation program is a complete cycle of transformations from a management system, development of a strategic vision, setting of processes and KPI, implementation of IT solutions to the management of changes and achievements of the final result jointly with the client.

Stages of the Transformation Program:

Stage 0 "Preparation and mobilization" (March - June 2014);
Stage 1 "Diagnostics and Design" (July 2014 - June 2015);
Stage 2 "Planning" (July - October 2015);
Stage 3 "Implementation" (beginning - 4th quarter 2015-2020).

In September 2014, in Almaty, Kazpost JSC put into commercial operation the first automated mail station - post-office.

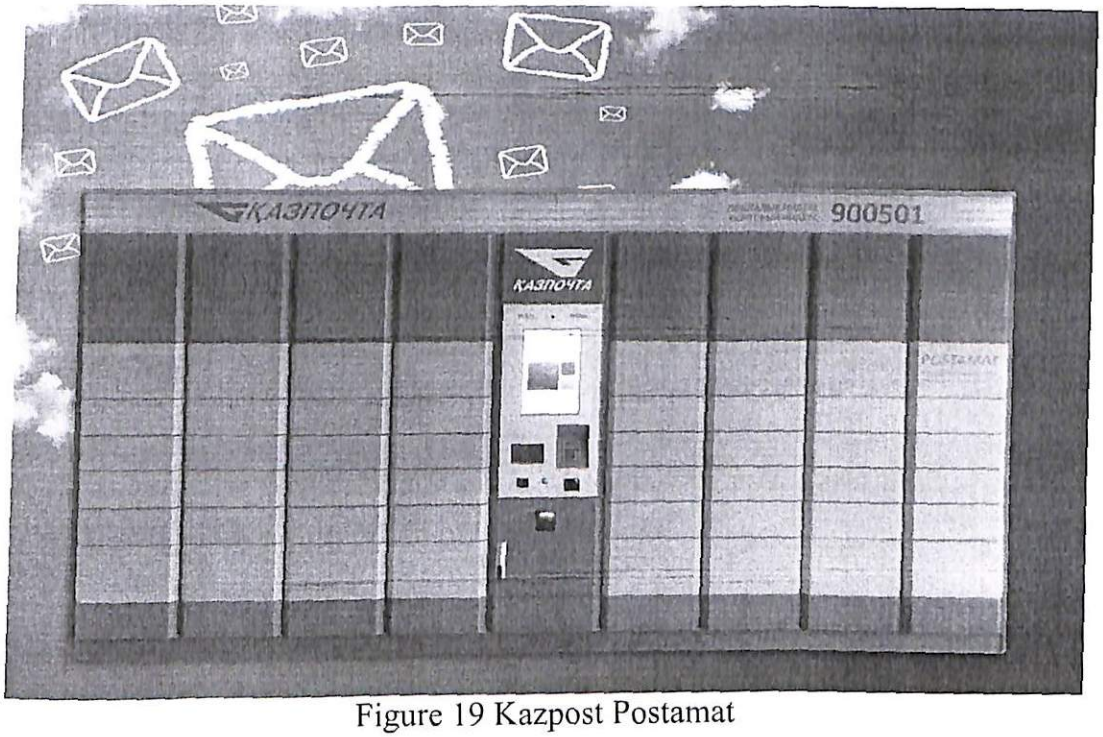

Figure 19 Kazpost Postamat

December 28, 2015, in Astana, Kazpost JSC opened the first supermarket of parcels.

Figure 20 Kazpost supermarket

2. USED TECHNOLOGIES

Python is a high-level general-purpose programming language designed to improve developer productivity and code readability. The syntax of the Python kernel is minimal. At the same time, the standard library includes a large amount of useful functions.

Python supports several programming paradigms, including structural, object-oriented, functional, imperative, and aspect-oriented. The main architectural features are dynamic typing, automatic memory management, full introspection, exception handling mechanism, support for multi-threaded computing and convenient high-level data structures. The code in Python is organized into functions and classes that can be combined into modules (they in turn can be combined into packages).

The reference implementation of Python is the CPython interpreter, which supports the most actively used platforms [14]. It is distributed under the free license of Python Software Foundation License, allowing it to be used without restrictions in any applications, including proprietary [14]. There are implementations of interpreters for JVM (with the ability to compile), MSIL (with the ability to compile), LLVM and others. The PyPy project offers a Python implementation using JIT compilation, which greatly increases the speed of execution of Python programs.

Python is an actively developing programming language, new versions (with the addition / modification of language properties) are published approximately every two and a half years. Because of this and some other reasons, Python does not have ANSI, ISO or other official standards, and CPython does.

Figure 21 Python Logo

2.1 Philosophy of Python

Python developers adhere to a certain programming philosophy called "The Zen of Python" ("Zen Python", or "Zen Python") [15]. Its text is given by the Python interpreter by the command "import this" (it works once per session). The author of this philosophy is Tim Peters.

The text of philosophy:

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Flat is better than nested.
- Sparse is better than dense.
- Readability counts.
- Special cases aren't special enough to break the rules.
- Although practicality beats purity.
- Errors should never pass silently.
- Unless explicitly silenced.
- In the face of ambiguity, refuse the temptation to guess.
- There should be one — and preferably only one — obvious way to do it.
- Although that way may not be obvious at first unless you're Dutch.
- Now is better than never.
- Although never is often better than 'right now'.
- If the implementation is hard to explain, it's a bad idea.
- If the implementation is easy to explain, it may be a good idea.
- Namespaces are one honking great idea — let's do more of those!

2.2 History of Python

The development of the Python language was started in the late 1980s [16] by the employee of the Dutch CWI Institute Guido van Rossum. For distributed OS Amoeba needed an extensible scripting language, and Guido began to write Python at his leisure, borrowing some lessons for the ABC language (Guido participated in the development of this language, focused on teaching programming). In February 1991, Guido published the source text in the newsgroup alt.sources [17]. From the beginning, Python was designed as an object-oriented language.

Figure 22 Old Logo of Python

The name of the language did not come from the species of reptiles. The author named the language in honor of the popular British comedy TV show of the 1970s "Monty Python's Flying Circus". However, still the name of the language is more often associated with the snake than with the transfer - the icons of files in KDE or in Microsoft Windows and even the emblem on the site python.org (before the release of version 2.5) depict snake heads. An important goal of Python developers is to make it fun to use. This is reflected in his name, which came from Monty Python, [18]. It is also reflected in the sometimes playful approach to tutorials and reference materials, such as usage examples that use the concepts of spam and eggs instead of the standard foo and bar.

Having a friendly, responsive user community is considered along with Guidot's design intuition as one of the factors in the success of Python. Language development takes place according to the strictly regulated process of creating, discussing, selecting and implementing PEP documents (English Python Enhancement Proposal) - proposals for the development of Python.

On December 3, 2008, after a long test, the first version of Python 3000 (or Python 3.0, also using the Py3k abbreviation) was released. Python 3000 eliminates many architectural flaws with the maximum possible (but not complete) compatibility with older versions of Python. Both branches of development are supported today (Python 3.x and 2.x).

2.3 The impact of other languages on Python

Appearing relatively late, Python was created under the influence of many programming languages:

- ABC - indents for grouping operators, high-level data structures (map) (Python was actually created as an attempt to correct errors made in the design of ABC);
- Modula-3 - packages, modules, use else together with try and except, named function arguments (this was also affected by Common Lisp);
- C, C ++ - some syntactic constructions (as Guido van Rossum himself writes - he used the most consistent designs from C, so as not to cause hostility from C programmers to Python);
- Smalltalk - object-oriented programming;
- Lisp - individual features of functional programming (lambda, map, reduce, filter and others);
- Fortran - slices of arrays, complex arithmetic;
- Miranda - list expressions;
- Java - modules logging, unittest, threading (some of the functionality of the original module is not implemented), xml.sax of the standard library, sharing finally and except when handling exceptions, using @ for decorators;
- Icon - generators.

- Most of the other Python features (for example, byte-compiling source code) have also been implemented earlier in other languages.

Portability

Python is ported and works on almost all known platforms - from PDAs to mainframes. There are ports under Microsoft Windows, almost all versions of UNIX (including FreeBSD and Linux), Plan 9, Mac OS and Mac OS X, iPhone OS 2.0 and above, Palm OS, OS / 2, Amiga, HaikuOS, AS / 400 and even OS / 390, Windows Mobile, Symbian and Android.

As the platform becomes obsolete, its support in the main branch of the language stops. For example, with the 2.6 series, support for Windows 95, Windows 98 and Windows ME is no longer supported. However, on these platforms, you can use the previous versions of Python - at the moment the community actively supports Python versions starting from 2.3 (for them there are fixes).

In this case, unlike many portable systems, for all major platforms, Python has support for specific technologies for this platform (for example, Microsoft COM / DCOM). Moreover, there is a special version of Python for the Java virtual machine - Jython, which allows the interpreter to run on any system that supports Java, while Java classes can be directly used from Python and even be written in Python. Also, several projects provide integration with the Microsoft .NET platform, the main ones of which are IronPython and Python.Net.

Data Types and Structures

Python supports dynamic typing, that is, the type of the variable is defined only at runtime. Therefore, instead of "assigning a value to a variable," it's better to talk about "associating a value with a name". In Python, there are built-in types: Boolean, string, Unicode string, an integer of arbitrary precision, a floating-point number, a complex number, and some others. From collections in Python built: list, tuple (unchanged list), dictionary, set and others. All values are objects, including functions, methods, modules, classes.

You can add a new type either by writing a class, or by defining a new type in the extension module (for example, written in C). The class system supports inheritance (single and multiple) and metaprogramming. It is possible to inherit from most built-in types and extension types.

All objects are divided into reference and atomic. To atomic are int, long (in version 3 any number int, since in version 3 there is no size limit), complex and some others. When assigning atomic objects, their value is copied, while for reference only the object pointer is copied, so both variables after the assignment use the same value. Reference objects are mutable and unchangeable. For example, strings and tuples are immutable, and lists, dictionaries and many other objects are mutable. A tuple in Python is, in fact, an unchangeable list. In many cases, tuples work faster than lists, so if you do not plan to change the sequence, it's best to use them.

Syntax and semantics

The language has a clear and consistent syntax, thoughtful modularity and scalability, so that the source code written in Python programs is easy to read. When passing arguments to functions, Python uses a call-by-sharing call.

Operators

The set of operators is quite traditional.

Conditional statement if. Alternate block after else (otherwise). If there are several conditions and alternatives, you can use elif (abbreviated from else if). Operators of the while (while) and for (for). Inside the loop, you can use break and continue to interrupt the loop and go directly to the next iteration, respectively.

The class definition operator.

Operator of definition of function, method or generator def. Internally, it is possible to use return to return from a function or method, and in the case of a generator, yield.

The try-except-else or try-finally exception handling operator (since version 2.5, you can use finally, except, and else in one block).

The pass statement does nothing. Used for empty blocks of code.

One of the interesting syntactic features of the language is the selection of blocks of code using indents (spaces or tabs), so in Python there are no parentheses for begin / end, as in Pascal, or braces, as in C. This "trick" allows you to reduce the number of lines and characters in the program and accustoms to a "good" programming style. On the other hand, the behavior and even the correctness of the program may depend on the initial gaps in the text. To some, this behavior may seem unintuitive and inconvenient.

Object-oriented programming

The design of the Python language is built around an object-oriented programming model. The implementation of OOP in Python is elegant, powerful and well thought out, but at the same time quite specific in comparison with other object-oriented languages.

Opportunities and features:

- Classes are simultaneously objects with all the possibilities listed below.
- Inheritance, including multiple.
- Polymorphism (all functions are virtual).
- Encapsulation (two levels - public and hidden methods and fields). Feature - hidden members are available for use and are marked as hidden only by special names.
- Special methods that control the life cycle of an object: constructors, destructors, memory allocators.
- Overloading operators (all except is, '.', '=' And character logical ones).
- Properties (simulation of the field with the help of functions).
- Controlling access to fields (emulation of fields and methods, partial access, etc.).
- Methods for managing the most common operations (truth value, len (), deep copying, serialization, iteration over an object, ...)

- Metaprogramming (managing the creation of classes, triggers for creating classes, etc.)
- Full introspection.
- Class and static methods, class fields.
- Classes nested in functions and classes.

Functional Programming

Python supports the paradigm of functional programming, in particular:

- Function is an object;
- Functions of higher orders;
- Recursion;
- Advanced list processing (list expressions, operations on sequences, iterators);
- Analogue of closures;
- Partial application of the function;
- The possibility of realizing other means in the language itself (for example, karring).

Modules and packages

The software (application or library) in Python is designed in the form of modules, which in turn can be collected in packages. Modules can be located both in directories and in ZIP-archives. Modules can be of two types in their origin: modules written in pure Python, and extension modules written in other programming languages. For example, in the standard library there is a "clean" module pickle and its analogue in C: cPickle. The module is made in the form of a separate file, and the package - in the form of a separate directory. The module is connected to the program by the import statement. After the import, the module is represented by a separate object, giving access to the namespace of the module. During the execution of the program, the module can be reloaded with the reload() function.

Iterators

Iterators are widely used in Python programs. The for loop can work with both the sequence and the iterator. All collections, as a rule, provide an iterator. Objects of a user-defined class can also be iterators. More information about iterators can be found in the section on functional programming. The itertools module of the standard library contains many useful functions to work with iterators.

Standard library

Python comes with "batteries included."

A rich standard library is one of the attractive aspects of Python. There are tools for working with many network protocols and Internet formats, for example, modules for writing HTTP-servers and clients, for parsing and creating e-mail messages, for working with XML, etc. A set of modules for working with the

operating system allows you to write a cross -platform applications. There are modules for working with regular expressions, text encodings, multimedia formats, cryptographic protocols, archives, data serialization, support for unit testing, etc.

Expansion modules and software interfaces

In addition to the standard library, there are many libraries that provide an interface to all system calls on different platforms; In particular, all Win32 API calls are supported on the Win32 platform, as well as COM in the amount not less than in Visual Basic or Delphi. The number of application libraries for Python in a variety of areas is enormous (web, database, image processing, word processing, numerical methods, operating system applications, etc.) without exaggeration.

For Python, the specification of the program interface to the DB-API 2 databases has been adopted and packages for accessing various DBMSs have been developed: Oracle, MySQL, PostgreSQL, Sybase, Firebird (Interbase), Informix, Microsoft SQL Server and SQLite. On the Windows platform, access to the database is possible through ADO (ADOdb). Commercial package mxODBC for access to DBMS through ODBC for Windows and UNIX platforms is developed by eGenix. Python has written a lot of ORM (SQLObject, SQLAlchemy, Dejavu, Django), software frameworks for developing web applications (Django, Pylons, Pyramid).

NumPy library for working with multidimensional arrays allows to achieve the productivity of scientific calculations, comparable with specialized packages. SciPy uses NumPy and provides access to a wide range of mathematical algorithms (matrix algebra - BLAS levels 1-3, LAPACK, FFT ...). Numarray is specially designed for operations with large volumes of scientific data.

WSGI - gateway interface with a web server (Python Web Server Gateway Interface).

Python provides a simple and user-friendly programming interface C API for writing its own modules in C and C ++. A tool like SWIG allows you to almost automatically get bindings for using C / C ++ libraries in Python code. The capabilities of this and other tools range from the automatic generation (C / C ++ / Fortran) -Python interfaces on special files (SWIG, pyste, SIP, pyfort), to providing more convenient APIs (boost :: python, CXX, Pyhrol, etc.). . The ctypes standard library tool allows Python programs to directly access dynamic libraries / DLLs written in C. There are modules that allow you to embed C / C ++ code directly into your Python source files, creating extensions on the fly (pyinline, weave).

Another approach is to embed the Python interpreter in applications. Python easily integrates into programs in Java, C / C ++, OCaml. Interaction of Python-applications with other systems is also possible using CORBA, XML-RPC, SOAP, COM.

With the help of the Cython project, it is possible to translate programs written in Python and Pyrex (English) into C code, and then compile into machine code. Cython is used to simplify the writing of Python libraries, using it accelerates the code and reduces overhead.

The Shedskin pilot project involves creating a compiler to transform implicitly typed Python programs into optimized C ++ code. Starting with version 0.22 Shedskin allows you to compile individual functions into extension modules.

Python and the vast majority of libraries to it are free and come in source code. Moreover, unlike many open systems, the license does not limit the use of Python in commercial development and imposes no obligations other than the indication of copyrights.

One of the channels for distributing and updating packages for Python is PyPI (English Python Package Index).

Graphic libraries

With Python, a tkinter library is provided based on Tcl / Tk for creating cross-platform programs with a graphical interface.

There are extensions that allow to use all the main GUI libraries - wxPython, based on the wxWidgets library, PyGTK for Gtk, PyQt and PySide for Qt, and others. Some of them also provide ample opportunities for working with databases, graphics and networks, using all the capabilities of the library on which they are based.

To create games and applications that require a non-standard interface, you can use the Pygame library. It also provides extensive means of working with multimedia: with its help you can control sound and images, play video. Provided pygame hardware graphics acceleration OpenGL has a higher-level interface compared to PyOpenGL, which copies the semantics of the C library for OpenGL. There is also PyOgre, which provides a binding to Ogre, a high-level object-oriented library of 3D graphics. In addition, there is a pythonOCC library that provides a binding to the 3D modeling environment and OpenCascade simulations.

To work with raster graphics, the Python Imaging Library is used.

Type checking and overloading functions

There are modules that allow you to control the types of function parameters at runtime, for example, typecheck or method signature checking decorators. An optional type declaration for function parameters is added in Python 3, the interpreter does not check types, but only adds relevant information to the function's metadata for subsequent use of this information by extension modules.

Overloading functions are implemented by various third-party libraries, including PEAK. Unaccepted plans to support overload in Python3000 were partially implemented in the library overloading-lib.

Profiling and code optimization

In the standard Python library, there is a profiler (profile module), which you can use to collect statistics about the time of operation of individual functions. To solve the question of which version of the code works faster, you can use the timeit module. The measurements made in the following program allow to find out which of the variants of string concatenation is more effective:

As in any programming language, Python has its own code optimization techniques. You can optimize the code based on various (often competing with each other) criteria (increase in speed, decrease in the amount of required RAM,

compactness of the source code, etc.). Most often, the programs are optimized for runtime.

There are several obvious rules.

- Do not optimize the program if its speed is sufficient.
- The algorithm used has a certain time complexity, so before optimizing the program code, it is necessary to first reconsider the algorithm.
- It's worth using ready-made and debugged functions and modules, even if you need to process the data a little. For example, in Python, there is a built-in sort () function.
- Profiling will help to clarify the bottlenecks. Optimization must begin with them.

Python has the following features and related optimization rules.

- Calling functions is quite a costly operation, so inside the nested loops one should try to avoid calling functions or, for example, to transfer the loop to functions. A function that processes a sequence is more efficient than processing the same sequence in a loop by calling a function.
- Try to take out of the deep nested cycle everything that can be calculated in external cycles. Access to local variables is faster than to global variables, or than access to fields.
- The psyco optimizer can help speed up the operation of the program module, provided that the module does not use the dynamic properties of the Python language.
- In the event that the module carries out massive processing of data and optimization of the algorithm and code does not help, you can rewrite critical sections, say, in C or Pyrex.

A tool called Pychecker will help you analyze the source code in Python and give recommendations on the problems found (for example, unused names, changing the method signature when it is overloaded, etc.). In this static analysis of the source code, errors can also be detected. Pylint is designed to solve similar problems, but it has a bias toward checking code style, searching for code with a smell.

Comparison with other languages

Most often Python is compared with Perl and Ruby. These languages are also interpretable and have roughly the same speed of program execution. Like Perl, Python can be used successfully to write scripts. Like Ruby, Python is a well-designed system for OOP.

Functional programming tools are partially borrowed from Scheme and Icon.

In a commercial application environment, the speed of executing programs in Python is often compared to Java applications.

Despite the fact that Python has a rather original syntax, one of the principles of design of this language is the principle of least surprise.

Low speed

Classic Python, like many other interpreted languages that do not use, for example, JIT-compilers, have a common drawback - a relatively low speed of execution of programs. Saving the bytecode (.pyc extension, and up to version 3.5, .pyo) allows the interpreter to not spend too much time recompiling the module code every time it starts.

There are implementations of the Python language that introduce high-performance virtual machines (VMs) as back-end compilers. Examples of such implementations are PyPy, based on RPython; An earlier initiative is the Parrot project. It is expected that the use of an LLVM VM will lead to the same results as the use of similar approaches for implementations of the Java language, where low computational performance is largely overcome.

Many programs / libraries for integration with other programming languages (see above) provide the opportunity to use another language for writing critical sections.

In the most popular implementation of the Python language, the interpreter is quite large and more resource-intensive than in the popular popular implementations of Tcl, Forth, LISP or Lua, which limits its use in embedded systems. Nevertheless, Python has found application in PDAs and some models of mobile phones.

Inability to modify built-in classes

Compared to Ruby and some other languages, Python does not have the ability to modify built-in classes, such as int, str, float, list, and others, which, however, allows Python to consume less RAM and run faster. Another reason for introducing such a restriction is the need for coordination with expansion modules. Many modules (in order to optimize performance) convert Python objects of elementary types to corresponding C-types instead of manipulating them with C-API. Also, this avoids many potential errors with uncontrolled dynamic override of built-in types.

Global interpreter lock (GIL)

GIL (Global Interpreter Lock) is a feature inherent in CPython, Stackless and PyPy, but absent in Jython and IronPython. In its work, the main Python interpreter constantly uses a large amount of thread-safe data. Basically, these are dictionaries in which the attributes of objects are stored, and calls to an external code (written in C, etc.). In order to avoid the destruction of these data when co-modified from different threads, before the execution of several instructions (by default 100), the interpreter thread grabs the GIL and releases it at the end. Because of this, at any one time in a single Python interpreter process, only one Python code stream can be executed, even if the computer has multiple processors or processor cores (the GIL is also released for the duration of blocking operations such as I / O, changes / checks The state of the synchronizing primitives and others - thus, if one thread is blocked, others can be executed). An attempt was made to move to more granular synchronizations, but because of the frequent locking / unlocking of locks, this implementation was too slow on single-threaded

applications. In the near future, the transition from GIL to other techniques is not expected due to the fact that, according to the author of the Python language Guido Rossum, single-threaded programs accelerated by GIL are larger than multithreaded ones, which GIL brakes. However, there is python-safethread - CPython without GIL and with some other changes (according to its authors, on single-threaded applications the speed corresponds to 60-65% of the speed of the original CPython).

This problem has two main solutions. The first is to refuse to share the data and external code calls. In this case, the data is duplicated in the threads and the need to ensure their synchronization (if necessary) lies on the programmer. This approach leads to an increase in the consumption of RAM (however, not so much as when using processes in Windows - as for Unix this is not critical, the processes are small).

The second approach is to provide more granular synchronization for individual data structures. In this case, the performance degrades due to the increase in the number of lock releases / captures.

If you need parallel execution of several Python code streams, you can use processes, for example, a processing module that mimics the semantics of the standard threading module, but uses processes instead of threads. Since version 2.6, the processing module has been added to the standard library and renamed to multiprocessing. There are many modules that simplify the writing of parallel and / or distributed applications in Python, such as parallelpython, Pypar, pympi and others. GIL is freed when executing the code of most extensions, for example, NumPy / SciPy, allowing other threads to execute during calculations. Another solution might be to use IronPython or Jython, which is devoid of this feature of the classic Python.

Version 3.2 introduced the modified GIL, and there were other innovations.

- Switching by timeout, and not by the number of opcodes - in the previous version, GIL was released after every 100 operations. But the operation can be performed from nanoseconds to fractions of seconds. In the new version, the lock will be removed every 5 milliseconds.
- Reducing overhead and increasing efficiency in the case of frequent switching - on some systems (mainly MacOS X), there is a problem with the effectiveness of locking when GIL is already captured: system calls become expensive. In the new version, the delays will be reduced.
- Reducing delays in switching threads due to the introduction of the mechanism of forced flow switching and priority requests (when the thread executes a priority request, the GIL is released as soon as possible).

Realizations

CPython is the main, but not the only implementation of the Python programming language. There are also other implementations.

- PyPy is a Python implementation written in RPython (a subset of Python that has much less dynamic capabilities). Allows you to easily check new features. In PyPy, in addition to the standard CPython, Stackless, Psyco, AST modification on the fly and more are included. The project integrates the capabilities of Python code analysis and translation into other languages and bytecodes of virtual machines (C, LLVM, Javascript, .NET from version 0.9.9). Starting with 0.9.0, fully automatic translation of RPython in C is possible, which results in a speed acceptable for use (2-3 times slower than CPython with JIT disabled for version 0.9.9). By default, PyPy comes with a built-in JIT compiler, with which it can run much faster than CPython.
- PyS60 - the implementation of the language for Nokia smartphones on the Series 60 platform.
- IronPython - Python for .NET Framework and Mono. Compiles the Python program into MSIL, thus providing full integration with the .NET system.
- Stackless is also a Python implementation written in C. This is not a full implementation, but patches for CPython. Provides advanced multi-threaded programming capabilities and a much greater depth of recursion.
- Python for .NET is another Python implementation for .NET. Unlike IronPython, this implementation does not compile Python code in MSIL, but only provides an interpreter written in C #. Allows using .NET assemblies from Python code.
- Jython is a Python implementation that uses the JVM as the runtime environment. Allows you to transparently use Java libraries.
- Python-safethread is a version of CPython without GIL, which allows you to simultaneously execute Python threads on all available processors. Some other changes have also been made.
- Unladen Swallow - Google's project to develop a highly efficient, maximally compatible with the CPython JIT compiler based on LLVM. According to the Python development plans, it was planned to migrate the Unladen Swallow source code to CPython in version 3.3. But PEP-3146 was canceled due to a lack of interest in Unladen Swallow from Google, the main development sponsor.
- Tinypy is a minimalist version of Python. Some of the features of CPython are not implemented.
- Micro Python is an efficient implementation of Python 3 for embedded systems with a small amount of RAM.
- Brython - the implementation of the language in client-side JavaScript, which allows you to write browser scripts in Python 3.
- QPython - Implement python for Android. The project is still at the testing stage, however, some of the most necessary libraries have already been ported to qpython. Allows and works in an interactive mode. There is, also, Qpython3.

2.4 Database Managemement System

SQLite is a compact embedded relational database. The source code of the library is transferred to the public domain. In 2005, the project received the Google-O'Reilly Open Source Awards [19].

Figure 23 SQLite Logo

Design

The word "embedded" means that SQLite does not use the client-server paradigm, that is, the SQLite engine is not a separately running process that the program interacts with but provides the library with which the program is linked and the engine becomes an integral part of the program. Thus, the function calls (APIs) of the SQLite library are used as the communication protocol. This approach reduces overhead, response time and simplifies the program. SQLite stores the entire database (including definitions, tables, indexes and data) in a single standard file on the computer on which the program runs. Simplicity of implementation is achieved due to the fact that before the execution of the record transaction, the entire file storing the database is blocked; ACID functions are also achieved by creating a log file.

Several processes or threads can simultaneously read data from one database without any problems. Recording to the database can be performed only if no other requests are currently being processed; Otherwise, the write attempt fails and the error code is returned to the program. Another variant of the development of events is the automatic retry of attempts to record during a given time interval.

In the delivery package there is also a functional client part in the form of an executable file sqlite3, which demonstrates the implementation of the functions of the main library. The client part works from the command line, allows you to access the database file based on typical OS functions.

Thanks to the architecture of the engine, it is possible to use SQLite both on embedded systems and on dedicated machines with gigabyte data arrays.

Restrictions

Old versions of SQLite were designed without any restrictions, the only condition was that the database fit into memory, in which all calculations were performed using 32-bit integers. This created certain problems. Due to the fact that

the upper limits have not been determined and therefore properly tested, errors were often found when using SQLite under rather extreme conditions. Therefore, in new versions of SQLite, limits have been entered that are now checked along with a common set of tests.

During the compilation of the SQLite library, the following restrictions are set, which, if necessary, you can increase:

1 000 000 000	SQLITE_MAX_LENGTH
2 000	SQLITE_MAX_COLUMN
1 000 000	SQLITE_MAX_SQL_LENGTH
1 000	SQLITE_MAX_EXPR_DEPTH
100	SQLITE_MAX_FUNCTION_ARG
500	SQLITE_MAX_COMPOUND_SELECT
50 000	SQLITE_MAX_LIKE_PATTERN_LENGTH
999	SQLITE_MAX_VARIABLE_NUMBER
1 000	SQLITE_MAX_TRIGGER_DEPTH
10	SQLITE_MAX_ATTACHED
32 768	SQLITE_MAX_PAGE_SIZE
1 073 741 823	SQLITE_MAX_PAGE_COUNT

Currently, only the SQLITE_MAX_PAGE_SIZE value can not be greater than the default value. Thus, without changing SQLITE_MAX_PAGE_COUNT, we can say that the maximum file size of the database is approximately 32 TB (35'184'372'056'064 B).

Some restrictions can be changed in the direction of decreasing during the execution of the program with the help of the category definition and the corresponding value of the function sqlite3_limit ():

*Int sqlite3_limit (sqlite3 *, int id, int newVal)*

3.1 Algorithm of Automating Sorting System

How much time it takes for postal official to find your package in the post office? Some of clients remain contended, but for someone it is hard to stay calm, when post official seeking for package a long. To decrease such situations let's assume that system store every package`s rack and by entering some id of client or package postal official can easily find it, and it will make this process much faster. Algorithm of automating sorting system explained below:

Previously, database must be filled by racks with their sizes and quantity: parameters will be id of rack, name of rack, X, Y, Z, volume. In addition, there must be accessory fields: x free, y free, z free, volume free 1, volume free 2 and volume free 3. Just received new package will have parameters like: id, tracking code, sender id, receiver id, x, y, z, volume.

Then, system makes query for data base to get from it all racks, which`s at least one free volume is greater than package`s volume. It takes all, because even if volume is greater it may not be appropriate by shape. Here, system starts to look over every rack`s x free, y free and z free are they greater or not. But package may be placed horizontally or vertically, by front side or back side, and in that case xyz parameters of package will not be actual. To find actual xyz parameters let`s remind Statistics Permutation. Permutations in Statistics refer to different ways of arranging a given set of variables. Package has 3 parameters: x y z.

Factorial property, which used by Permutation is $n! = n(n-1)!$, n is number of variables.

$$1! = 1 = 1$$

$$2! = 2 \times 1 = 2$$

$$3! = 3 \times 2 \times 1 = 6$$

$$4! = 4 \times 3 \times 2 \times 1 = 24$$

In case x y z, n will be equal to 3. Accordingly to 3! the result is 6, number of permutations will be six permutations. Let`s see them: xyz, xzy, yxz, yzx, zxy, zyx. So, there are six possible positions of package, in which it can be placed in rack [20]. Which suitable position will be first met, that xyz permutation system will store as actual resultant xyz. For example, suitable is yzx, in that case, y will be stored as x, z as y and x as z.

In Figure 25 shown example of one package in the rack. Blue one is rack, red one is package. X, Y, Z are parameters of rack, xr, yr and zr are resultant parameters of package.

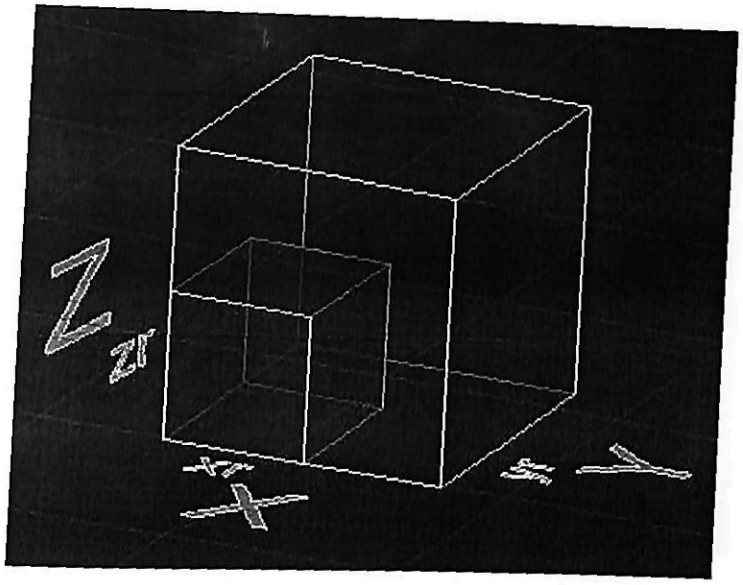

Figure 24. One package in the rack.

After placing one package rack will store 3 possible free volumes. Here they are:

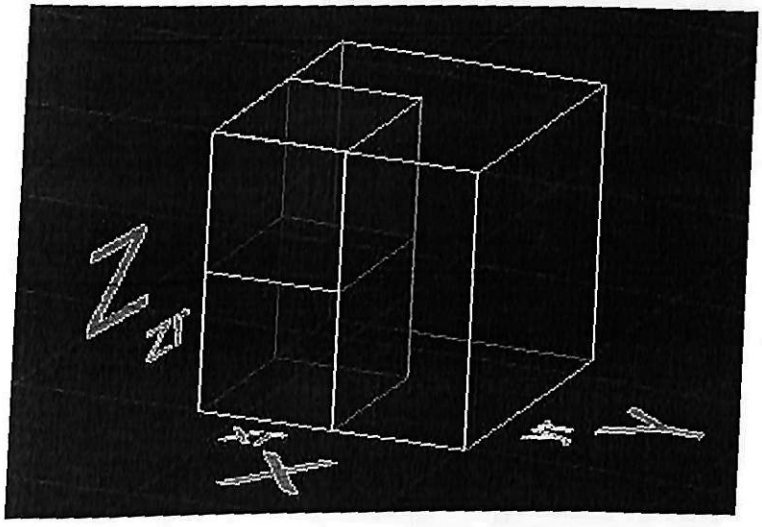

Figure 25. First possible free volume.

In Figure 26, by green shape shown first possible free volume. Also, system must store it`s xyz parameters, in this case they are:

Volume free 1 = (X – xr) * (Y - yr) * (Z – zr)

In Figure 27, by green shape shown second possible free volume. Also, system must store it`s xyz parameters, in this case they are:

Volume free 2 = X * (Y - yr) * Z

In Figure 28, by green shape shown third possible free volume. Also, system must store it`s xyz parameters, in this case they are:

Volume free 3 = (X – xr) * Y * Z

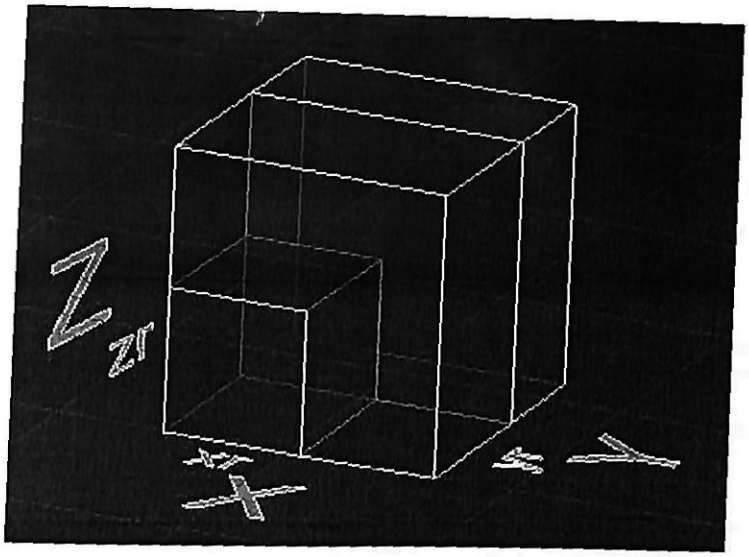

Figure 26. Second possible free volume.

Optimized free volume will be the first met greater and according parameters volume because the set of volumes is sorted and starts from the least.

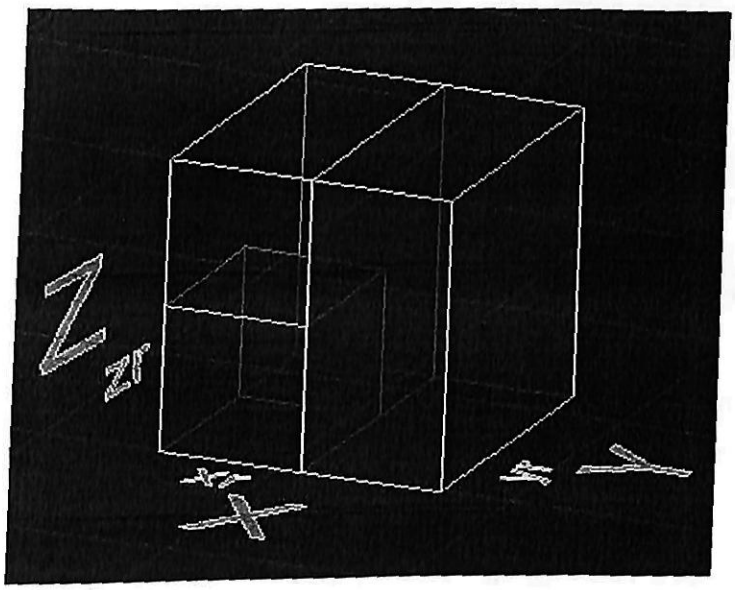

Figure 27. Third possible free volume.

After placing second package free volumes will be changed by same logic.

3.2 Graphical User Interface

GUI is a A program interface that takes advantage of the computer's graphics capabilities to make the program easier to use. Well-designed graphical user interfaces can free the user from learning complex command languages. On the other hand, many users find that they work more effectively with a command-driven interface, especially if they already know the command language.

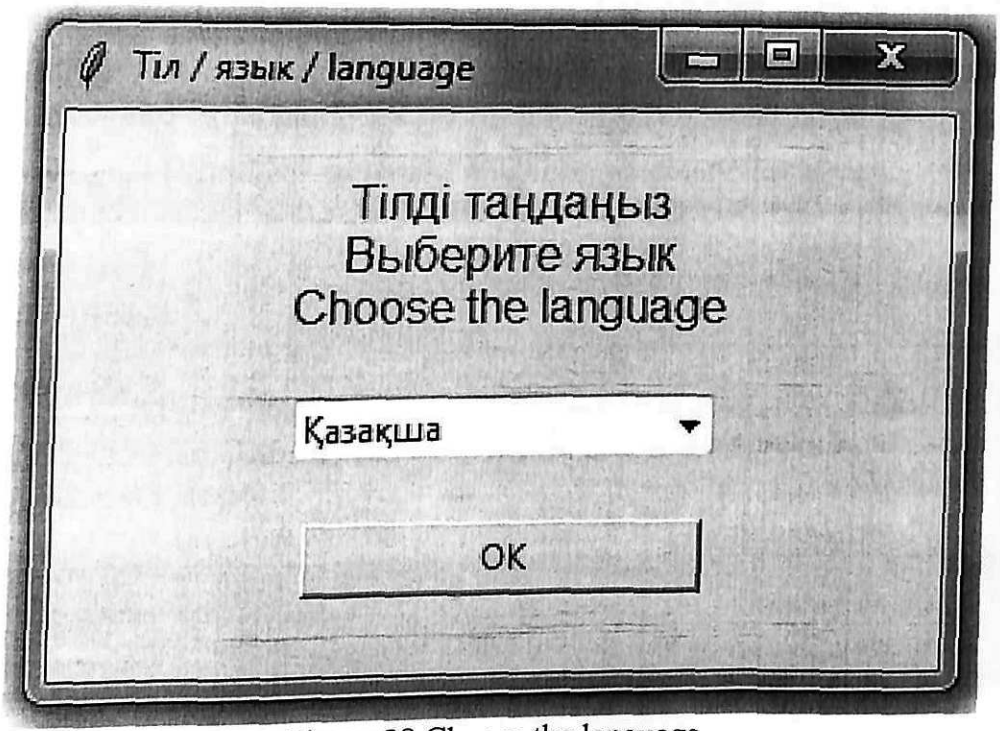

Figure 28 Choose the language

Firstly, because of multilingualism of software, user must choose the language of interface from Combobox.

Graphical User Interfaces with Tk

Tk/Tcl has long been an integral part of Python. It provides a robust and platform independent windowing toolkit, that is available to Python programmers using the tkinter package, and its extension, the tkinter.tix and the tkinter.ttk modules.

The tkinter package is a thin object-oriented layer on top of Tcl/Tk. To use tkinter, you don't need to write Tcl code, but you will need to consult the Tk documentation, and occasionally the Tcl documentation. tkinter is a set of wrappers that implement the Tk widgets as Python classes. In addition, the internal module _tkinter provides a threadsafe mechanism which allows Python and Tcl to interact.

tkinter's chief virtues are that it is fast, and that it usually comes bundled with Python. Although its standard documentation is weak, good material is available, which includes: references, tutorials, a book and others. tkinter is also famous for having an outdated look and feel, which has been vastly improved in Tk 8.5. Nevertheless, there are many other GUI libraries that you could be

interested in. For more information about alternatives, see the Other Graphical User Interface Packages section.

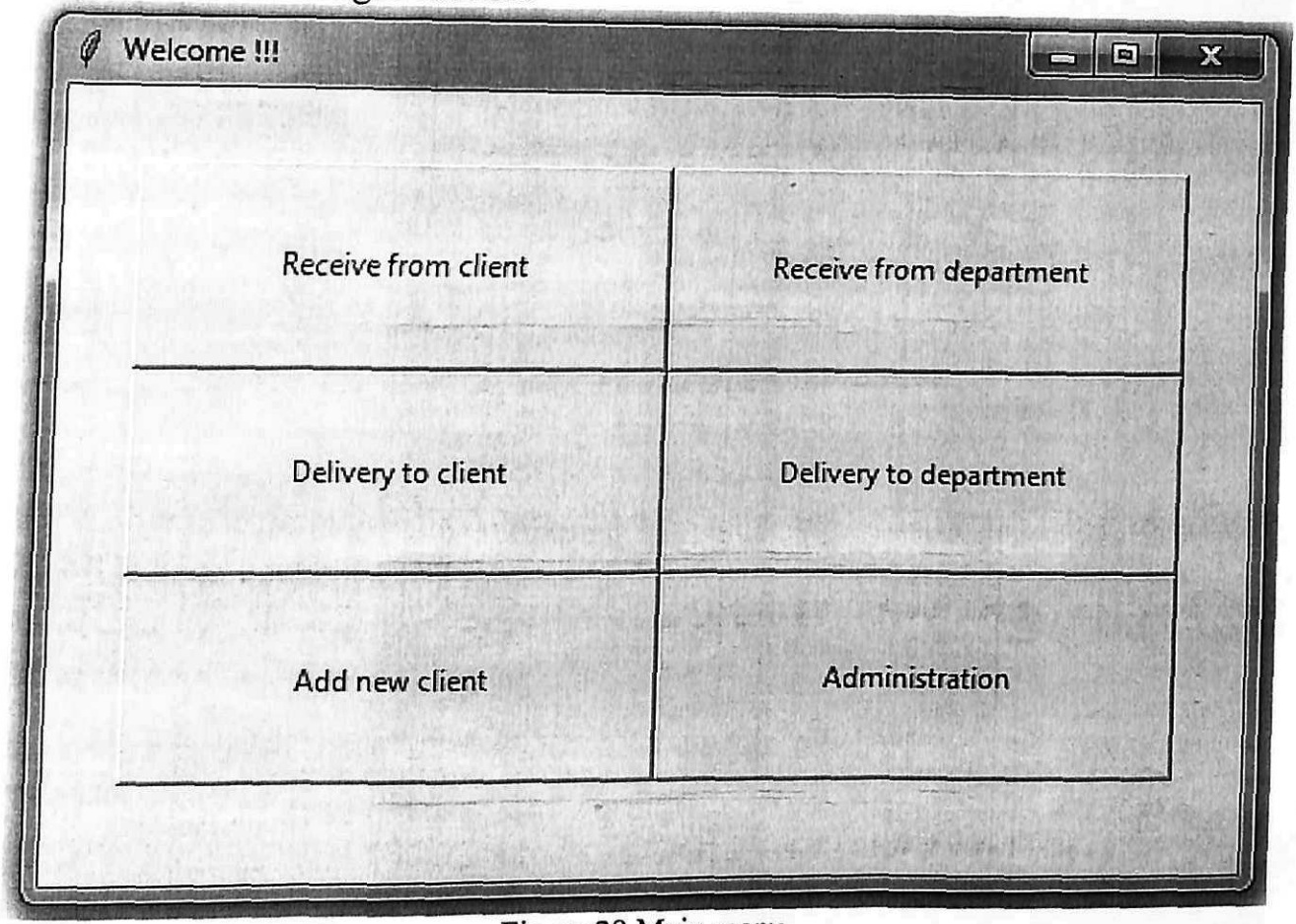

Figure 29 Main menu

The main menu showed In Figure 29. Consisting six buttons: Receive from client – receive package from client; receive from department – receive package from another department; delivery to client – deliver package to client; delivery to department – deliver package to another department; add new client – add new client; administration – administration.

Receive from client. In Figure 30, shown form, which must be filled for receive package from client. First block is intended for sender information, namely IIN. Further placed verify button, to identify does the IIN is correct and exist in database. Also, there is a label, which shows did the IIN verified. If there is no such client, there is exist button to add new one.

Second block is intended for receiver information, namely IIN. Further placed verify button, to identify does the IIN is correct and exist in database. Also, there is a label, which shows did the IIN verified. If there is no such client, there is exist button to add new one.

Third block is combobox with list of addresses from data base. And button of adding new address.

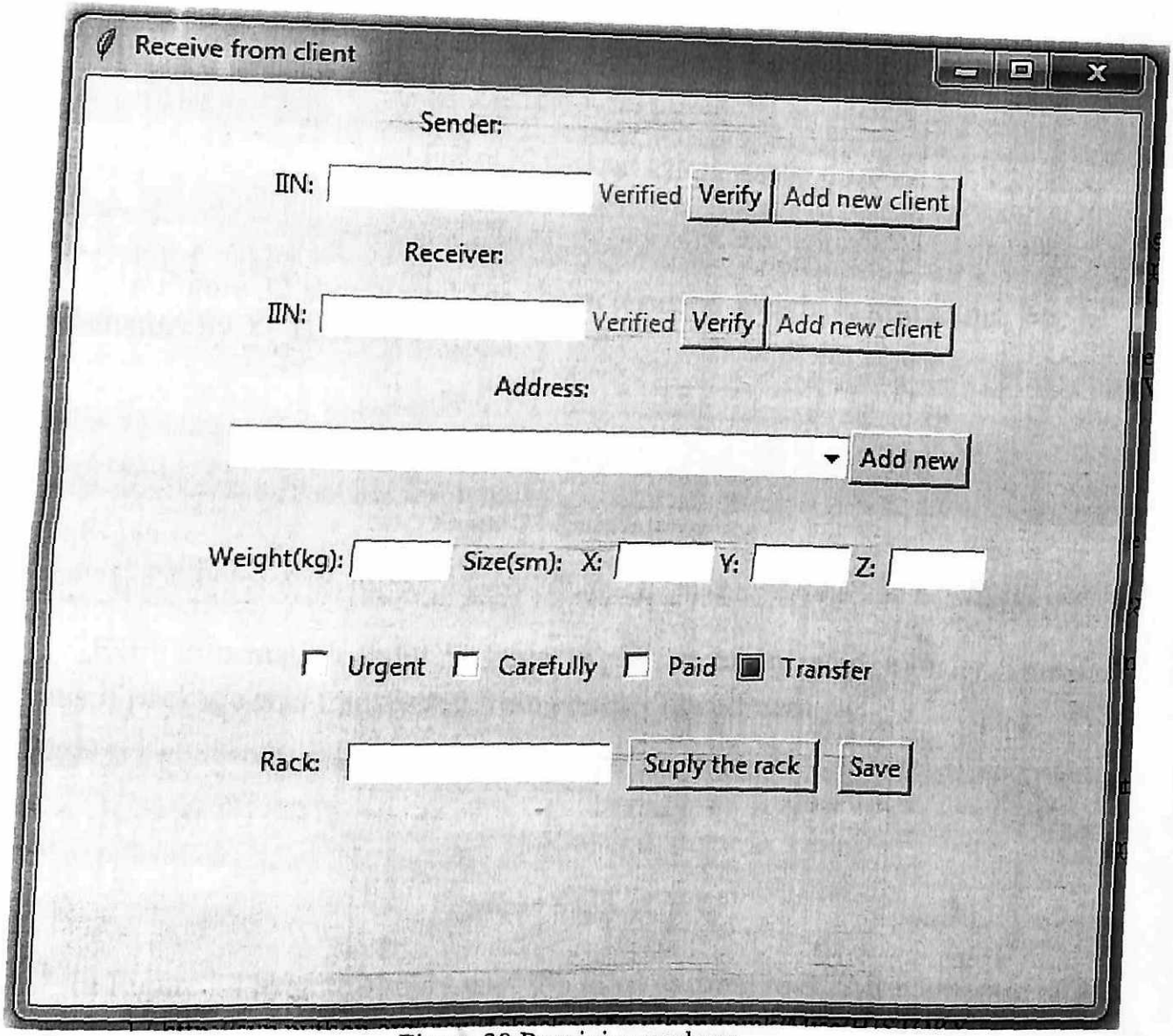

Figure 30 Receiving package

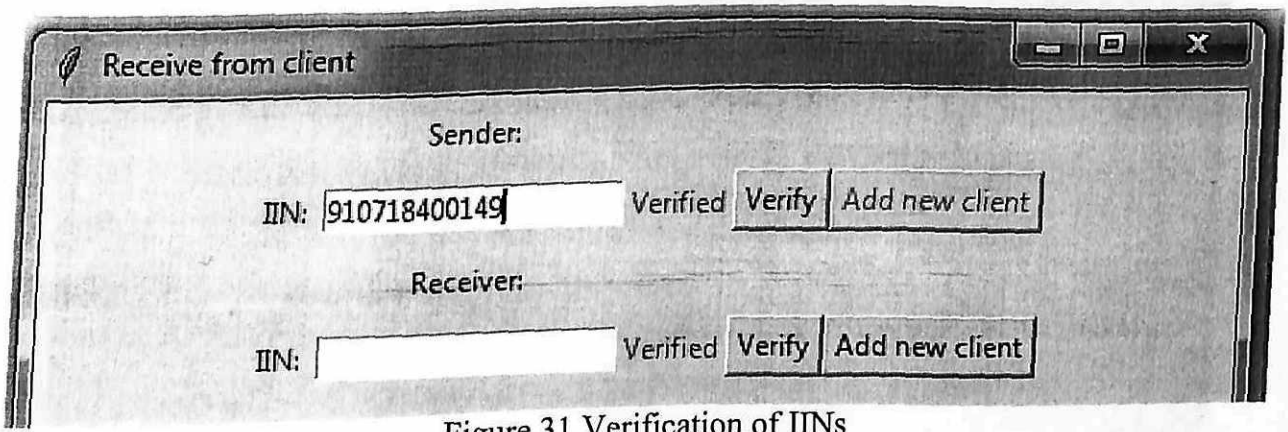

Figure 31 Verification of IINs

Label changes its color for green if verification succeed and red if otherwise. And adding new client button is frozen till label verify became red.

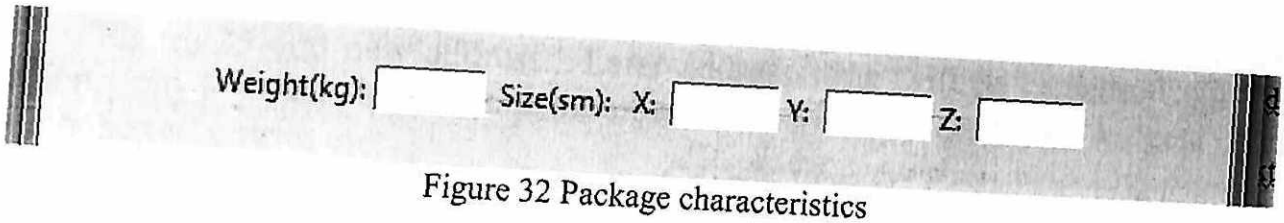

Figure 32 Package characteristics

In Figure 32 shown package characteristics: weight in kilograms, size in centimeters by X, Y, Z.

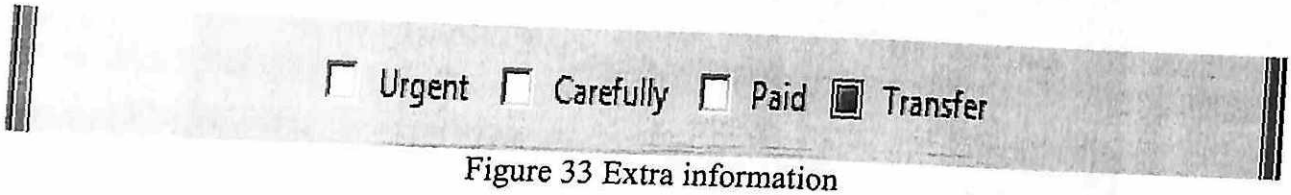

Figure 33 Extra information

Extra information about package: urgent, carefully, paid, transfer. Transfer means if package was transferred from another department.

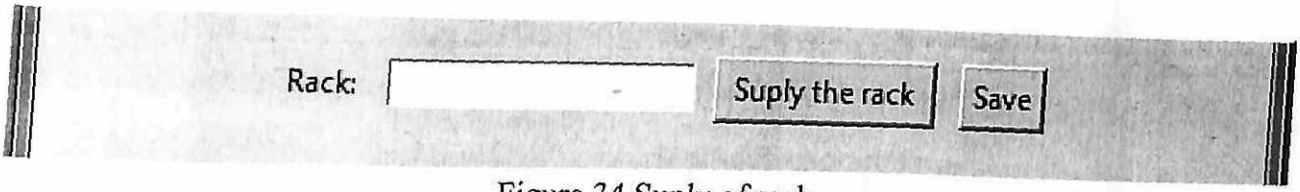

Figure 34 Suply of rack

In Figure 34 system supply user the most optimal rack. After entering rack and pressing save button system will store all information.

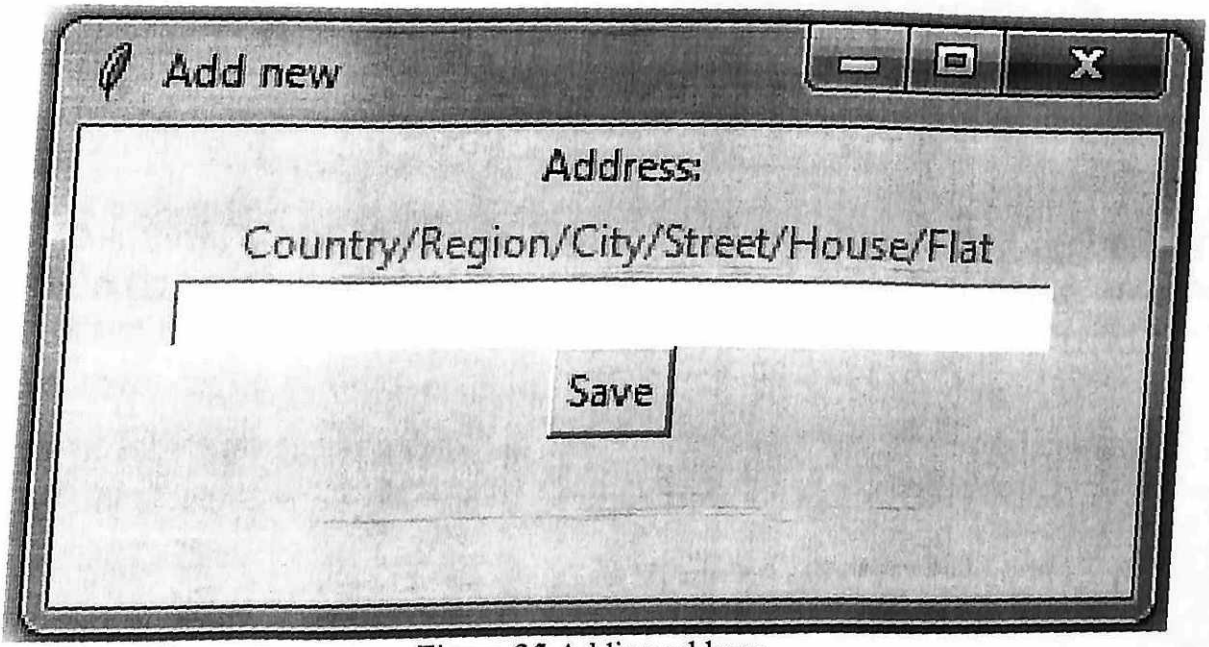

Figure 35 Adding address

Figure 35 add new address. Later address form will be remade. Country, region, city, street, house and number flat will be separate widgets like comboboxes, entries etc.

Figure 36 shows form of adding new client, which consist of information of client: surname, name, IIN, mobile phone number.

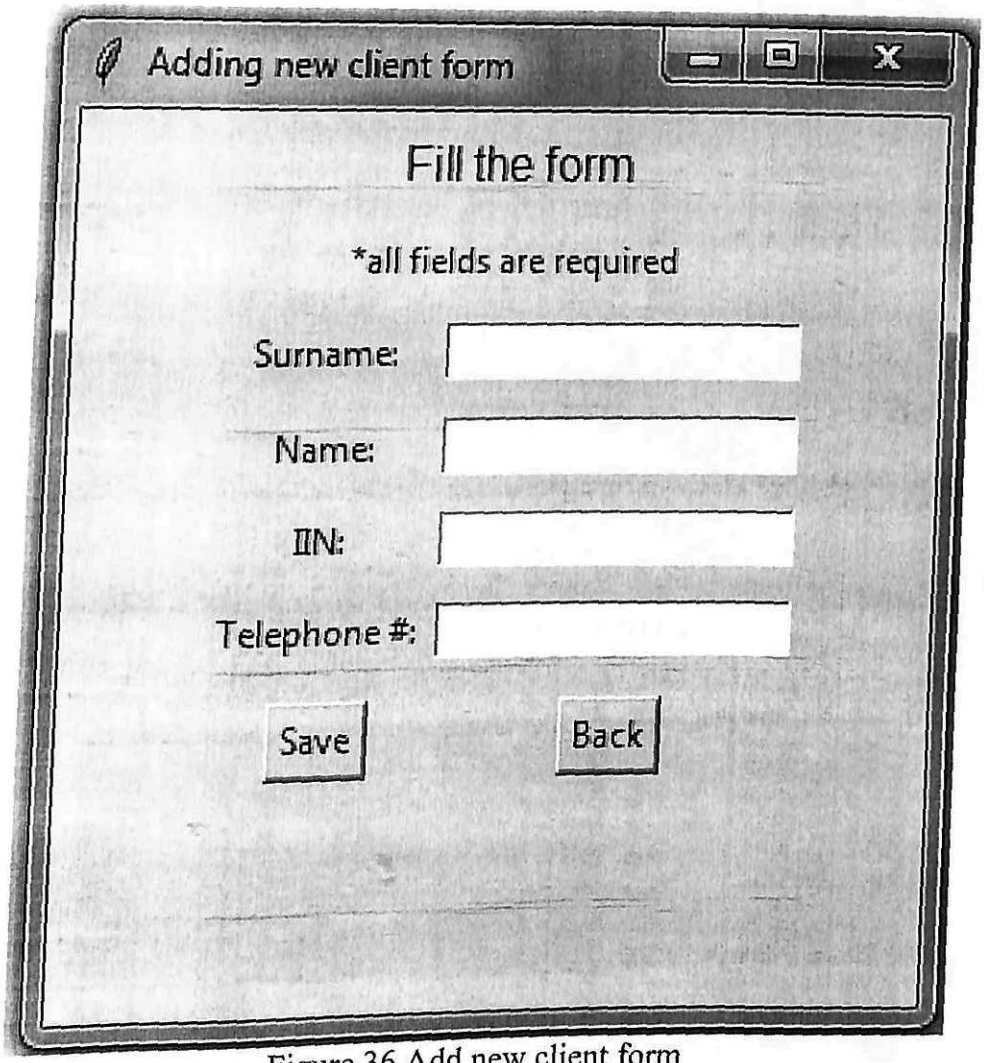

Figure 36 Add new client form

This form, which shown in Figure 37, allows user to find package by its tracking code or receiver's IIN. Then it checks for right rack and send, and delete this package from rack.

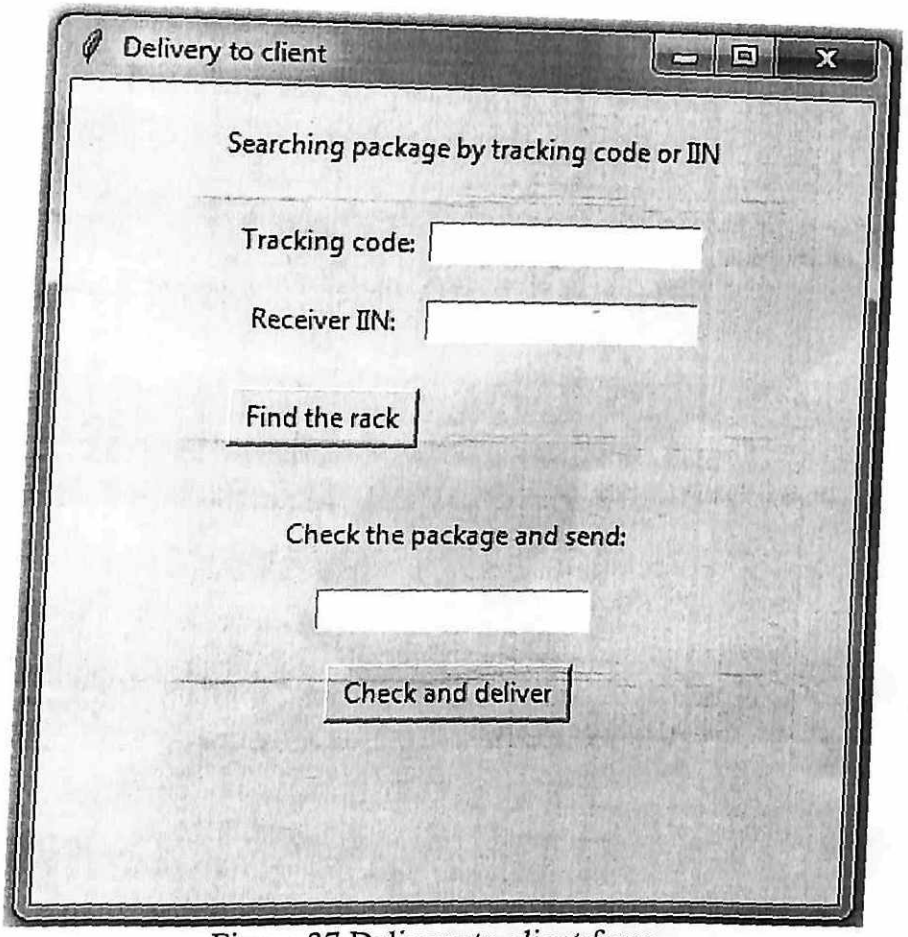

Figure 37 Delivery to client form

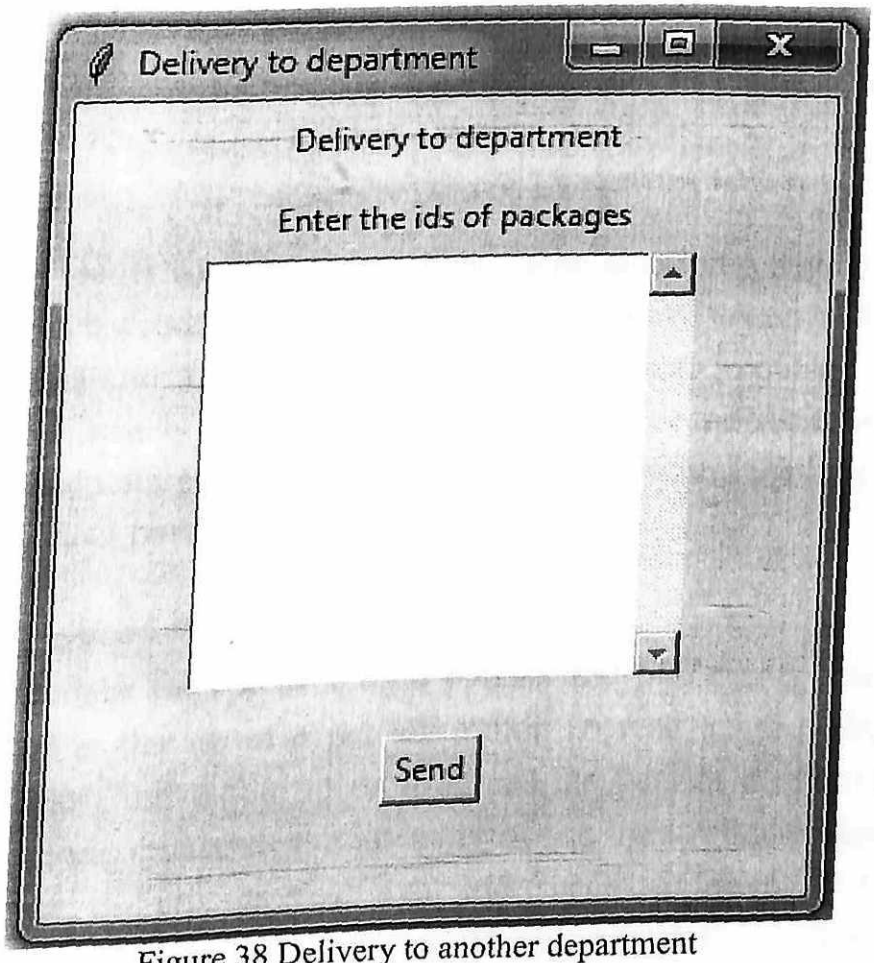

Figure 38 Delivery to another department

In Figure 38 shown form, this can deliver to another department several packages at ones. Entering ids of packages by barcode reader which shown in Figure 39.

Figure 39 Barcode reader

A barcode reader (or barcode scanner) is an electronic device that can read and output printed barcodes to a computer. Like a flatbed scanner, it consists of a light source, a lens and a light sensor translating optical impulses into electrical ones. Additionally, nearly all barcode readers contain decoder circuitry analyzing the barcode's image data provided by the sensor and sending the barcode's content to the scanner's output port.

Pen-type Scanners

Pen-type readers consist of a light source and photodiode that are placed next to each other in the tip of a pen or wand. To read a bar code, the person holding the pen must move the tip of it across the bars at a relatively uniform speed. The photodiode measures the intensity of the light reflected back from the light source as the tip crosses each bar and space in the printed code. The photodiode generates a waveform that is used to measure the widths of the bars

and spaces in the bar code. Dark bars in the bar code absorb light and white spaces reflect light so that the voltage waveform generated by the photodiode is a representation of the bar and space pattern in the bar code. This waveform is decoded by the scanner in a manner similar to the way Morse code dots and dashes are decoded.

Laser scanners

Laser scanners work the same way as pen type readers except that they use a laser beam as the light source and typically employ either a reciprocating mirror or a rotating prism to scan the laser beam back and forth across the bar code. As with the pen type reader, a photo-diode is used to measure the intensity of the light reflected back from the bar code. In both pen readers and laser scanners, the light emitted by the reader is rapidly varied in brightness with a data pattern and the photo-diode receive circuitry is designed to detect only signals with the same modulated pattern.

CCD readers (also known as LED scanners)

CCD readers use an array of hundreds of tiny light sensors lined up in a row in the head of the reader. Each sensor measures the intensity of the light immediately in front of it. Each individual light sensor in the CCD reader is extremely small and because there are hundreds of sensors lined up in a row, a voltage pattern identical to the pattern in a bar code is generated in the reader by sequentially measuring the voltages across each sensor in the row. The important difference between a CCD reader and a pen or laser scanner is that the CCD reader is measuring emitted ambient light from the bar code whereas pen or laser scanners are measuring reflected light of a specific frequency originating from the scanner itself.

Camera-based readers

Two-dimensional imaging scanners are a newer type of bar code reader. They use a camera and image processing techniques to decode the bar code.

Video camera readers use small video cameras with the same CCD technology as in a CCD bar code reader except that instead of having a single row of sensors, a video camera has hundreds of rows of sensors arranged in a two dimensional array so that they can generate an image.

Large field-of-view readers use high resolution industrial cameras to capture multiple bar codes simultaneously. All the bar codes appearing in the photo are

decoded instantly (ImageID patents and code creation tools) or by use of plugins (e.g. the Barcodepedia used a flash application and some web cam for querying a database), have been realized options for resolving the given tasks.

Omnidirectional barcode scanners

Omnidirectional scanning uses "series of straight or curved scanning lines of varying directions in the form of a starburst, a Lissajous pattern, or other multiangle arrangement are projected at the symbol and one or more of them will be able to cross all of the symbol's bars and spaces, no matter what the orientation.[1] almost all use a laser. Unlike the simpler single-line laser scanners, they produce a pattern of beams in varying orientations allowing them to read barcodes presented to it at different angles. Most of them use a single rotating polygonal mirror and an arrangement of several fixed mirrors to generate their complex scan patterns.

Omnidirectional scanners are most familiar through the horizontal scanners in supermarkets, where packages are slid over a glass or sapphire window. There are a range of different omnidirectional units available which can be used for differing scanning applications, ranging from retail type applications with the barcodes read only a few centimetres away from the scanner to industrial conveyor scanning where the unit can be a couple of metres away or more from the code. Omnidirectional scanners are also better at reading poorly printed, wrinkled, or even torn barcodes.

Cell phone cameras

While cell phone cameras without auto-focus are not ideal for reading some common barcode formats, there are 2D barcodes which are optimized for cell phones, as well as QR Codes and Data Matrix codes which can be read quickly and accurately with or without auto-focus.

Cell phone cameras open up a number of applications for consumers:

- Movies: DVD/VHS movie catalogs.
- Music: CD catalogs – play MP3 when scanned.
- Book catalogs and device.
- Groceries, nutrition information, making shopping lists when the last of an item is used, etc.
- Personal Property inventory (for insurance and other purposes)ode scanned into personal finance software when entering. Later, scanned receipt images can then be automatically associated with the appropriate entries. Later, the

bar codes can be used to rapidly weed out paper copies not required to be retained for tax or asset inventory purposes.

- If retailers put barcodes on receipts that allowed downloading an electronic copy or encoded the entire receipt in a 2D barcode, consumers could easily import data into personal finance, property inventory, and grocery management software. Receipts scanned on a scanner could be automatically identified and associated with the appropriate entries in finance and property inventory software.
- Consumer tracking from the retailer perspective (for example, loyalty card programs that track consumers purchases at the point of sale by having them scan a QR code).

A number of enterprise applications using cell phones are appearing:

- Access control (for example, ticket validation at venues), inventory reporting (for example, tracking deliveries), asset tracking (for example, anti-counterfeiting).

Smartphones

- Smartphones can be used in Google's mobile Android operating system via both their own Google Goggles application. Nokia's Symbian operating system features a barcode scanner which can scan barcodes, while mbarcode is a barcode reader for the Maemo operating system. In the Apple iOS, a barcode reader is not automatically included, but there are more than fifty free or paid apps available with both scanning capabilities and hard-linking to URI. With BlackBerry devices, the App World application can natively scan barcodes. Windows Phone 8 is able to scan barcodes through the Bing search app.

Stationary scanner

Wall- or table-mounted scanners that the barcode is passed under or beside. These are commonly found at the checkout counters of supermarkets and other retailers.

Fixed-position scanner

An industrial barcode reader, used to identify products during manufacture or logistics. Often used on conveyor tracks to identify cartons or pallets which need to be routed to another process or shipping location. Another application joins holographic scanners with a checkweigher to read bar codes of any orientation or placement, and weighs the package. Systems like this are used in factory and farm automation for quality management and shipping.

Methods of connection

Early serial interfaces

Early barcode scanners, of all formats, almost universally used the then-common RS-232 serial interface. This was an electrically simple means of connection and the software to access it is also relatively simple, although needing to be written for specific computers and their serial ports.

Proprietary interfaces

There are a few other less common interfaces. These were used in large EPOS systems with dedicated hardware, rather than attaching to existing commodity computers. In some of these interfaces, the scanning device returned a "raw" signal proportional to the intensities seen while scanning the barcode. This was then decoded by the host device. In some cases the scanning device would convert the symbology of the barcode to one that could be recognized by the host device, such as Code 39.

Keyboard wedge (e.g. PS/2)

With the popularity of the PC and its standard keyboard interface, it became ever easier to connect physical hardware to a PC and so there was commercial demand similarly to reduce the complexity of the associated software. "Keyboard wedge" (PS/2) hardware plugged between the PC and its normal keyboard, with characters from the barcode scanner appearing exactly as if they had been typed at the keyboard. This made the addition of simple barcode reading abilities to existing programs very easy, without any need to change them, although it did require some care by the user and could be restrictive in the content of the barcodes that could be handled.

USB

Later barcode readers began to use USB connectors rather than the keyboard port, as this became a more convenient hardware option. To retain the easy integration with existing programs, a device driver called a "software wedge" could be used, to emulate the keyboard-impersonating behavior of the old "keyboard wedge" hardware.

In many cases, a choice of USB interface types (HID, CDC) are provided. Some have PoweredUSB.

Wireless networking

Some modern handheld barcode readers can be operated in wireless networks according to IEEE 802.11g (WLAN) or IEEE 802.15.1 (Bluetooth). Some barcode readers also support radio frequencies viz. 433 MHz or 910 MHz. Readers without external power sources require their batteries be recharged occasionally, which may make them unsuitable for some uses.

Resolution

The scanner resolution is measured by the size of the dot of light emitted by the reader. If this dot of light is wider than any bar or space in the bar code, then it will overlap two elements (two spaces or two bars) and it may produce wrong output. On the other hand, if a too small dot of light is used, then it can misinterpret any spot on the bar code making the final output wrong.

The most commonly used dimension is 13 thou (0.013 in or 0.33 mm), although some scanners can read codes with dimensions as small as 3 thou (0.003 in or 0.075 mm). Most manufacturers advertise bar code resolution in mil, which is interchangeable with thou. Smaller bar codes must be printed at high resolution to be read accurately.

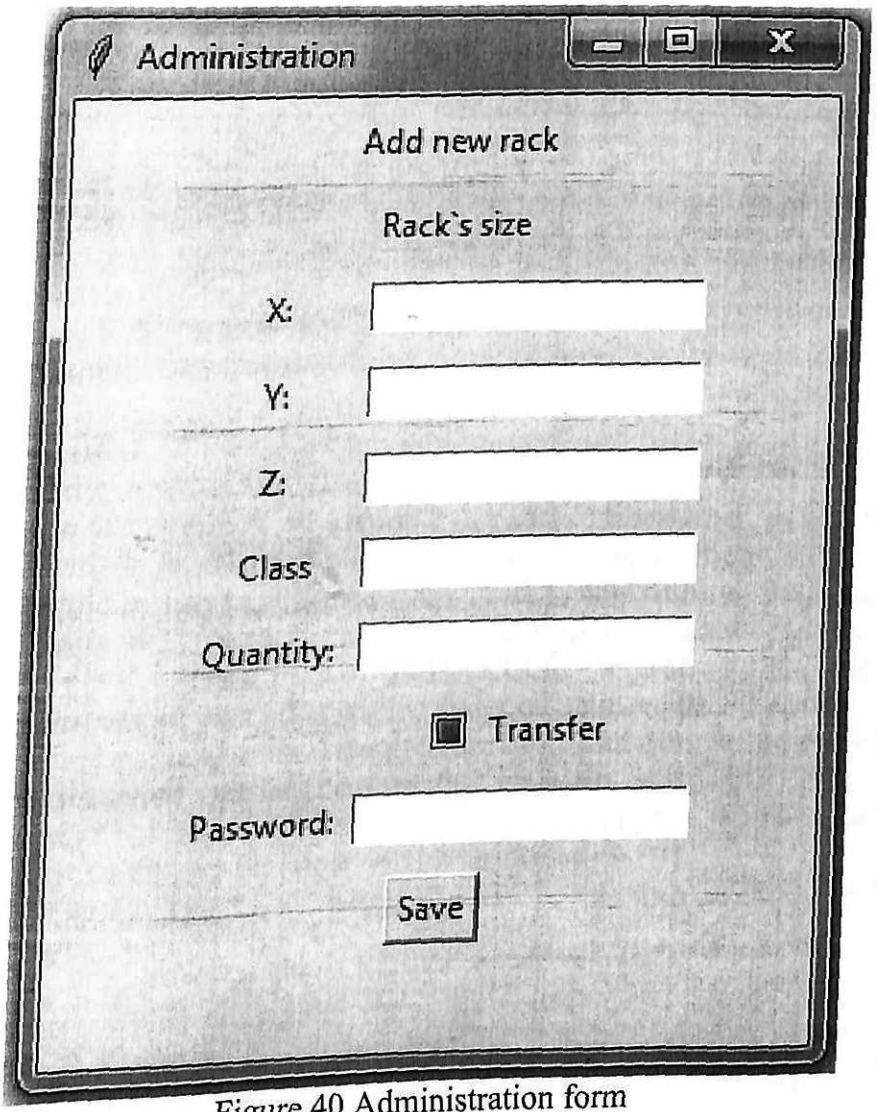

Figure 40 Administration form

In Figure 40 shown administration form, where administrator add new racks into the system.

3.3 Implementation

Importing modules and setting connections with database:

```
import sqlite3
import multilang
from tkinter import *
from tkinter import ttk
from tkinter import messagebox
import random

til = 0
lang = multilang.languages
conn = sqlite3.connect('first.db')
cursor = conn.cursor()
```

Main window, after user chooses language, main menu appears:

```
def chLan():
til = cb.current()
    wnLan.destroy()
    wn = Tk()
    wn.title(lang[0][til])
    wn.geometry('500x330')
    f = Frame(wn)
    f.pack()
    fr = Frame(wn)
    fr.pack(pady = 30)
    b1 = Button(fr, text = lang[2][til], command = addNewOwn,
width = 30, height = 5).grid(row = 2, column = 0)
    b2 = Button(fr,text = lang[19][til], width = 30, height = 5,
command = adminpanel).grid(row = 2, column = 1)
    b3 = Button(fr, text = lang[4][til], width = 30, height = 5,
command = recPackage).grid(row = 0, column = 0)
    b4 = Button(fr, text = lang[5][til], width = 30, height = 5,
command = recPackage).grid(row = 0, column = 1)
    b5 = Button(fr, text = lang[6][til], width = 30, height = 5,
command = clientGet).grid(row = 1, column = 0)
    b6 = Button(fr, text = lang[7][til], width = 30, height = 5,
command = kurier).grid(row = 1, column = 1)

wnLan = Tk()
wnLan.title('Тіл / язык / language')
wnLan.geometry('300x190')

lab = Label(wnLan, text = '\nТілді таңдаңыз\nВыберите
язык\nChoose the language\n', font = '18')
lab.pack()
listLan = ['Қазақша','Русский','English']
cb = ttk.Combobox(wnLan, values = listLan)
cb.current(0)
```

```
cb.pack()
lll = Label(wnLan, text = '').pack()
b = Button(wnLan, text = 'OK', command = chLan, width =
18).pack()
wnLan.mainloop()
```

Adding new owner form, by sample INSERT query into database, such data like surname, name, iin, telephone number:

```
def addNewOwn():
    def saveOwner():
        name = enNm.get()
        surname = enSn.get()
        iin = int(eniin.get())
        tel = entel.get()
        cursor.execute("INSERT INTO owner (surname, name,
iin,tel) VALUES (?,?,?,?);", (surname, name, iin, tel))
        conn.commit()
        lsave = Label(wnAddNewOwn, text = lang[8][til], fg =
'green').pack()
        enNm.delete(0, END)
        enSn.delete(0, END)
        eniin.delete(0, END)
        entel.delete(0, END)
    def closewin():
        ans =
messagebox.askquestion(lang[17][til],lang[18][til])
        if ans == 'yes':
            wnAddNewOwn.destroy()
    wnAddNewOwn = Tk()
    wnAddNewOwn.title(lang[9][til])
    wnAddNewOwn.geometry('300x300')
    lab1 = Label(wnAddNewOwn, text = lang[10][til], fg = 'blue',
font = '16').pack(pady= 5)
    lab2 = Label(wnAddNewOwn, text = lang[11][til]).pack(pady=5)
    fr1 = Frame(wnAddNewOwn)
    fr1.pack()
    labSn = Label(fr1, text = lang[12][til]).grid(row = 0,
column = 0,pady= 5)
    labNm = Label(fr1, text = lang[13][til]).grid(row = 1,
column = 0, pady= 5)
    enSn = Entry(fr1)
    enSn.grid(row = 0, column = 1)
    enNm = Entry(fr1)
    enNm.grid(row = 1, column = 1)
    labiin = Label(fr1, text = lang[14][til]).grid(row = 2,
column = 0, pady= 5)
    labtel = Label(fr1, text = lang[15][til]).grid(row = 3,
column = 0, pady= 5)
    eniin = Entry(fr1)
    eniin.grid(row = 2, column = 1)
    entel = Entry(fr1)
```

```
    entel.grid(row = 3, column = 1)
    but1 = Button(fr1, text = lang[16][til], command =
saveOwner).grid(row = 4, column = 0, pady= 7)
    but2 = Button(fr1, text = lang[17][til], command =
closewin).grid(row = 4, column = 1)
```

Receiving package from client, process starts will filling characteristics of sender and receiver, than characteristics of package, and in the end system proposes the most optimal rack:

```
def recPackage():
    def verifyOwner():
        owner_iin = int(iinen.get())
        cursor.execute('SELECT COUNT(iin) FROM owner WHERE iin =
(?)',(owner_iin,))
        res = cursor.fetchone()
        print (res[0])
        if res[0] != 0:
            lverify.config(fg = 'green')
            badd.config(state = DISABLED)
        else:
            lverify.config(fg = 'red')
            badd.config(state = NORMAL)
    def verifyOwner2():
        owner_iin = int(iinen2.get())
        cursor.execute('SELECT COUNT(iin) FROM owner WHERE iin =
(?)',(owner_iin,))
        res = cursor.fetchone()
        print (res[0])
        if res[0] != 0:
            lverify2.config(fg = 'green')
            badd2.config(state = DISABLED)
        else:
            lverify2.config(fg = 'red')
            badd2.config(state = NORMAL)
    def addAdr():
        def saveadres():
            adres = enadres.get()
            cursor.execute('INSERT INTO address (name) VALUES
(?);',(adres,))
            conn.commit()
            enadres.delete(0,END)
            cursor.execute('SELECT name FROM address')
            adrList = cursor.fetchall()
            wAddAdr.destroy()
            recPackage()
            wPackAdd.destroy()
            wAddAdr = Tk()
            wAddAdr.title(lang[27][til])
            wAddAdr.geometry('300x130')
```

```
            ladres = Label(wAddAdr, text = lang[28][til]).pack()
            lput = Label(wAddAdr, text = lang[38][til], fg =
'grey').pack()
            enadres = Entry(wAddAdr, width = 40)
            enadres.pack()
            badres = Button(wAddAdr, text = lang[16][til],
command = saveadres).pack()
    def proprack():
        global xr, yr, zr
        xr = int(enx.get())
        yr = int(eny.get())
        zr = int(enz.get())
        volr = xr*yr*zr
        rackTr = chtran.state()
        print(rackTr)
        global fnamer
        fnamer = ['None']
        global index
        index = -1
        global tr
        if 'selected' in rackTr:
            tr = 1
            cursor.execute('SELECT name,xfree,yfree,zfree FROM
rackT WHERE volfree >= (?)',(volr,))
            fnamer = cursor.fetchall()
            print (fnamer)
        elif 'alternate' not in rackTr:
            tr = 2
            cursor.execute('SELECT name,xfree,yfree,zfree FROM
racks WHERE volfree >= (?)', (volr,))
            fnamer = cursor.fetchall()
            print (fnamer)
        elif 'alternate' in rackTr:
            tr = 3
            fnamer = [['0','0','0','0'],['0','0','0','0']]
        for i in range(0,len(fnamer)):
            print (i)
            print(len(fnamer))
            print (fnamer[i][1])
            print (fnamer[i][1]) >= xr and int(fnamer[i][2]) >=
            if (int(fnamer[i][3]) >= zr):
yr and int(fnamer[i][3])
                index = i
                break
            elif (int(fnamer[i][1]) >= xr and int(fnamer[i][3])
>= yr and int(fnamer[i][2]) >= zr):
                index = i
                p = yr
                yr = zr
                zr = p
                break
            elif (int(fnamer[i][2]) >= xr and int(fnamer[i][1])
>= yr and int(fnamer[i][3]) >= zr):
```

```
                index = i
                p = xr
                xr = yr
                yr = p
                break
            elif (int(fnamer[i][2]) >= xr and int(fnamer[i][3])
>= yr and int(fnamer[i][1]) >= zr):
                index = i
                p = xr
                xr = yr
                yr = zr
                zr = p
                break
            elif (int(fnamer[i][3]) >= xr and int(fnamer[i][1])
>= yr and int(fnamer[i][2]) >= zr):
                index = i
                p = xr
                xr = zr
                zr = yr
                yr = p
                break
            elif (int(fnamer[i][3]) >= xr and int(fnamer[i][2])
>= yr and int(fnamer[i][1]) >= zr):
                index = i
                p = xr
                xr = zr
                zr = p
                break
        global name
        name = fnamer[index][0]
        proplab.config(text = name)
    def saverack():
        nameR = enpolka.get()
        if nameR == name:
            print (fnamer)
            nx = fnamer[index][1] - xr
            ny = fnamer[index][2] - yr
            nz = fnamer[index][3] - zr
            nvol = nx*ny*nz
            own_iin = iinen.get()
            rec_iin = iinen2.get()
            adr = comboadres.get()
            massiv = '0123456789'
            massivA = 'ABCDEFGHIJKLMNOP'
            track = 'KZ'
            for i in range(0,9):
                j = random.randint(0,9)
                track = massiv[j] + track
            k = random.randint(0,15)
            track = massivA[k]+track
            k = random.randint(0,15)
            track = massivA[k]+track
```

```
            print (track)
            if tr == 1:
                cursor.execute('UPDATE rackT SET xfree = (?),
yfree = (?), zfree = (?), volfree = (?) WHERE name = (?)',
(nx,ny,nz,nvol,nameR,))
            conn.commit()
            cursor.execute('INSERT INTO package
(rack_name,owner_iin,receiver_iin,destination,tracking_code,x,y,
z,transfer) VALUES (?,?,?,?,?,?,?,?,?);',
(nameR,own_iin,rec_iin,adr,track,xr,yr,zr,tr))
            conn.commit()
            elif tr == 2:
                pass
                cursor.execute('UPDATE racks SET xfree = (?),
yfree = (?), zfree = (?), volfree = (?) WHERE name = (?)',
(nx,ny,nz,nvol,nameR,))
            conn.commit()
            cursor.execute('INSERT INTO package
(rack_name,owner_iin,receiver_iin,destination,tracking_code,x,y,
z,transfer) VALUES (?,?,?,?,?,?,?,?,?);',
(nameR,own_iin,rec_iin,adr,track,xr,yr,zr,tr))
            conn.commit()
        wPackAdd = Tk()
        wPackAdd.title(lang[4][til])
        wPackAdd.geometry('500x430')
        adrList = []
        cursor.execute('SELECT name FROM address')
        adrList = cursor.fetchall()
        #print (adrList)
        fiin = Frame(wPackAdd)
        fiin.pack()
        lsender = Label(fiin, text = lang[23][til]).grid(row =
0, column = 1, pady = 5)
        liin = Label(fiin, text = lang[14][til]).grid(row = 1,
column = 0, pady = 5)
        iinen = Entry(fiin)
        iinen.grid(row = 1, column = 1)
        lverify = Label(fiin, text = lang[24][til], fg = 'grey')
        lverify.grid(row = 1,column = 2)
        bverify = Button(fiin, text = lang[25][til], command =
verifyOwner).grid(row = 1, column = 3)
        badd = Button(fiin, text = lang[2][til], state =
DISABLED, command = addNewOwn)
        badd.grid(row = 1, column = 4)
        lrecive = Label(fiin, text = lang[26][til]).grid(row =
2, column = 1, pady = 5)
        liin2 = Label(fiin, text = lang[14][til])
        liin2.grid(row = 3, column = 0,pady = 5)
        iinen2 = Entry(fiin)
        iinen2.grid(row = 3, column = 1)
        lverify2 = Label(fiin, text = lang[24][til], fg =
'grey')
```

```
        lverify2.grid(row = 3,column = 2)
        bverify2 = Button(fiin, text = lang[25][til], command =
verifyOwner2)
        bverify2.grid(row = 3, column = 3)
        badd2 = Button(fiin, text = lang[2][til], state =
DISABLED, command = addNewOwn)
        badd2.grid(row = 3, column = 4)
        fadres = Frame(wPackAdd)
        fadres.pack()
        ladr=Label(fadres, text = lang[28][til]).grid(row = 0,
column = 0, pady = 5)
        comboadres = ttk.Combobox(fadres, values = adrList,
width = 45)
        comboadres.grid(row = 1,column = 0, pady = 5)
        badr = Button(fadres, text = lang[27][til], command =
addAdr).grid(row = 1, column = 1)
        frazmer = Frame(wPackAdd)
        frazmer.pack()
        lprobel = Label(frazmer, text = '').grid(row=0,column=0)
        lves = Label(frazmer, text =
lang[29][til]).grid(row=1,column=0,pady=5)
        enves = Entry(frazmer, width = 7).grid(row=1,column=1)
        lsize = Label(frazmer, text =
lang[30][til]).grid(row=1,column=2)
        lx = Label(frazmer, text = 'X: ').grid(row=1, column=3)
        enx = Entry(frazmer, width = 7)
        enx.grid(row = 1, column = 4) # X
        ly = Label(frazmer, text = 'Y: ').grid(row = 1,column=5)
        eny = Entry(frazmer, width = 7)
        eny.grid(row = 1, column = 6) # Y
        lz = Label(frazmer, text = 'Z: ').grid(row = 1,column=7)
        enz = Entry(frazmer, width = 7)
        enz.grid(row = 1, column = 8) # Z
        fch = Frame(wPackAdd)
        fch.pack()
        lprobel2 = Label(fch, text = ' ').grid(row =0, column=0)
        churj = Checkbutton(fch, text = lang[31][til]).grid(row
= 1 , column = 0)
        chcare = Checkbutton(fch, text = lang[32][til]).grid(row
= 1 , column = 1)
        chpaid = Checkbutton(fch, text = lang[33][til]).grid(row
= 1 , column = 2)
        chtran = ttk.Checkbutton(fch, text = lang[34][til])
        chtran.grid(row = 1 , column = 3) # tran
        fpo = Frame(wPackAdd)
        fpo.pack()
        lprobel3 = Label(fpo, text = ' ').grid(row =0, column=0)
        lpolka2 = Label(fpo, text = lang[35][til]).grid(row = 1,
column = 0,padx = 3)
        enpolka = Entry(fpo, width = 20)
        enpolka.grid(row = 1, column = 1,padx = 3)
```

```
        bpol = Button(fpo,text = lang[36][til], command =
proprack).grid(row = 1, column = 2, padx = 5)
        bsave = Button(fpo, text = lang[16][til], command =
saverack).grid(row = 1, column = 3, padx = 5)
        proplab = Label(fpo, text = '')
        proplab.grid(row = 2, column = 1, pady = 5)
```

Administration panel, here user adds new racks and confirm adding by password. Racks are clisified by classes and size:

```
def adminpanel():
    def addRack():
        x = int(ex.get())
        y = int(ey.get())
        z = int(ez.get())
        volfree = x*y*z
        kol = int(ekol.get())
        rackT = chtrans.state()
        print(rackT)
        last_id = ['None']
        if 'selected' in rackT:
           cursor.execute('SELECT max(id) FROM rackT')
            last_id = cursor.fetchone()
        elif 'alternate' not in rackT:
            cursor.execute('SELECT max(id) FROM racks')
            last_id = cursor.fetchone()
        lid = str(last_id[0])
        print (lid)
        rtype = eklass.get()
        if lid == 'None':
            lid = 1
         else:
            lid = int(lid)+1
        for i in range(lid,lid+kol):
           rack =
rtype+'x'+str(x)+'x'+str(y)+'x'+str(z)+'x'+str(i)
            if 'selected' in rackT:
                print ('selectedddd')
                cursor.execute("INSERT INTO rackT
(name,x,y,z,xfree,yfree,zfree,volfree) VALUES
(?,?,?,?,?,?,?,?);", (rack,x,y,z,x,y,z,volfree))
                conn.commit()
            elif 'alternate' not in rackT:
                print ('not selectedddd')
                cursor.execute("INSERT INTO racks
(name,x,y,z,xfree,yfree,zfree,volfree) VALUES
(?,?,?,?,?,?,?,?);", (rack,x,y,z,x,y,z,volfree))
                conn.commit()
```

Deliver to another department, here post office employee select several packages by barcode reader and then system send them to another department:

```
def kurier():
    def sendkurier():
        content = spisok.get(1.0,END)
        #print (content)
        packs = content.split('\n')
        #print (packs)
        info = []
        for i in range(0,len(packs)-2):
            cursor.execute('SELECT rack_name,x,y,z FROM package
WHERE tracking_code = (?)',(packs[i],))
            odin = cursor.fetchall()
            print (odin[0][0])
            xt = int(odin[0][1])
            yt = int(odin[0][2])
            zt = int(odin[0][3])
            volt = xt*yt*zt
            trackcode = odin[0][0]
            print (odin)
            cursor.execute('UPDATE rackT SET xfree = xfree +
(?), yfree = yfree + (?), zfree = zfree + (?), volfree =
xfree*yfree*zfree WHERE name = (?)',(xt,yt,zt,trackcode,))
            conn.commit()
            cursor.execute('UPDATE rackT SET volfree =
xfree*yfree*zfree WHERE name = (?)',(trackcode,))
            conn.commit()
            cursor.execute('DELETE FROM package WHERE
tracking_code = (?)',(packs[i],))
            conn.commit()
    wKurier = Tk()
    wKurier.title(lang[7][til])
    wKurier.geometry('280x310')
    lkur = Label(wKurier, text = lang[7][til]).pack(pady = 5)
    lkur2 = Label(wKurier, text = lang[39][til]).pack(pady = 5)
    frtext = Frame(wKurier)
    frtext.pack()
    spisok = Text(frtext, width = 20, height = 10)
    spisok.grid(row = 0, column = 0, sticky="nsew")
    scrl = Scrollbar(frtext, command = spisok.yview)
    scrl.grid(row = 0, column = 1, sticky="nsew")
    spisok['yscrollcommand'] = scrl.set
    lprobel4 = Label(wKurier, text = '').pack()
    bkur = Button(wKurier, text = lang[40][til], command =
sendkurier)
    bkur.pack()
```

CONCLUSION

Nowadays post services system is becoming more and more optimized every day. A lot of people around the world work for making human life better, more comfortable and easier. Specially - information technology scientists. Hundred years ago decreasing months and weeks of delivery was discussed, but now every minute and second is important. Even if this algorithm is a very small portion of that huge system, it would help to save time, make clients stay calm and use space optimized. According to development of post service, starting from basics, it seems that every second will be important and consumed optimized in next 10-20 years.

Nowadays post services system is becoming more and more optimized every day. A lot of people around the world work for making human life better, more comfortable and easier. Specially - information technology scientists. They are developing methodology of Internet of Things, which in its turn will make the World smart.

Even if this algorithm is a very small portion of that huge system, it would help to save time, make clients stay calm and use space optimized. In future algorithm will be supplemented by parameters of package such as 'carefully', 'urgent', 'costly' etc. And the most exciting that, algorithm will work with Internet of Things technology

REFERENCES

1 Kisin B. M. "Philately country", 240 p., Enlightenment, 1969

2 E. Seton-Thompson "Little Savages"

3 The Oldest Posts // The Great Philatelic Dictionary / N.I. Vladinets, L.I. Il'ichev, I. Ya. Levitas, P.F. Mazur, I.N. Merkulov, I.A. Morosanov, Yu. K. Myakota, S.A. Panasyan, Yu. M. Rudnikov, M.B. Slutsky, V.A. Jacobs; "Under the Society"; N.I. Vladints and VA Jacobs "Radio and Communication", 1988. 76p, 320 p., ISBN 5-256-00175-2

4 The idea of the possibility of making a round-the-world trip at that time in less than three months was the basis of the adventure novel of the French writer Jules Verne "Around the World in Eighty Days" (1873).

5 Vladinets N.I. Filatelia / / The Great Soviet Encyclopedia.

6 Mail // Encyclopedic Dictionary of Brockhaus and Efron: 86 t. (82 t. And 4 ext.). - St. Petersburg, 1890-1907.

7 This section uses information translated into Russian from a German article "Geschichte der Post".

8 Postage Stamps // Encyclopaedic Dictionary of Brockhaus and Efron: 86 t. (82 t. And 4 ext.). - St. Petersburg, 1890-1907

9 Kazakhstan has ratified the Universal Postal Union document. IA News-Kazakhstan (January 18, 2011).

10 Natalia Todorova. Kazpost: 15 years of equalization of the country's independence. Kazakhstan's truth (February 8, 2008).

11 JSC "Kazpost" received the status of the Principal member of the international payment system Visa International. Kazinform (August 16, 2013).

12 The first in Kazakhstan 24-hour post office "Post-24" was opened in Astana. Kazinform (December 19, 2012).

13 A 24-hour post office opened in Kazakhstan. 24.KZ (December 19, 2012).

14 https://www.python.org/about/

15 https://www.python.org/dev/peps/pep-0020/

16 http://www.artima.com/intv/pythonP.html

17 http://svn.python.org/view/*checkout*/python/trunk/Misc/HISTORY

18 https://docs.python.org/2/faq/general.html#why-is-it-called-python

19 Google-O'Reilly Open Source Awards — Hall of Fame — Google Code

20 Charalambides Ch A. Enumerative Combinatorics, p. 42. 2002. ISBN 978-1-58488-290-9.