МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук

Кафедра інформатики

ДИПЛОМНА РОБОТА

Рівень вищої освіти бакалавр

на тему: Комп'ютерна система обліку первинних бухгалтерських документів

Виконала студентка 4 курсу групи К-42
Напрям підготовки 6.050101
Комп'ютерні науки

Апшай Олеся Михайлівна

Керівник к. т. н., доцент
Гнатовська Ганна Арнольдівна

Рецензент к. геогр. н., доцент
Кузніченко Світлана Дмитрівна

Одеса 2017

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ

Скорочення

API – Application Programming Interface – Прикладний програмний інтерфейс.

CGI – Common Gateway Interface – загальний інтерфейс шлюзу.

IDE – Integrated Development Environment – інтегроване середовище розробки програм.

HTML – HyperText Markup Language – мова гіпертекстової розмітки.

LCL – Lazarus Component Library – бібліотека візуальних компонентів.

ODBC – Open Database Connectivity – програмний інтерфейс, що забезпечує доступ до бази даних.

RAD –Rapid Application Development – середовище швидкої розробки додатків.

TCP/IP – Transmission Control Protocol (TCP) і Internet Protocol (IP) – набор мережевих протоколів передачі даних, який використовують в мережах

Терміни

Веб-інтерфейс – це сукупність засобів, за допомогою яких користувач взаємодіє з веб-ресурсом або будь-яким іншим додатком через браузер.

Проект Open Source – проект з відкритим вихідним кодом.

ADO-технологія –Active Data Objects – високорівневий компонент технології доступу до даних.

InterBase – система управління базами даних (СУБД).

ВСТУП

На сучасному етапі автоматизації управління суспільним виробництвом найбільш перспективним є автоматизація планово-управлінських функцій на базі персональних комп'ютерів, встановлених безпосередньо на робочих місцях фахівців. Ці системи набули широкого поширення в організаційному управлінні під назвою автоматизованих робочих місць (АРМ). Специфіка діяльності бухгалтерії дозволяє вибрати методом рішення створення АРМ. Це дозволить використовувати систему людям, які не мають спеціальних знань в області програмування, і одночасно дозволить доповнювати систему у міру потреби. Зайняти лідируюче положення на ринку, підвищити ефективність роботи персоналу, створити оптимальну структуру управління – ось першочерге завдання керівника підприємства. У бухгалтерській і банківської діяльності це особливо важливо, що обумовлює широке застосування бухгалтерських пакетів і програм, завдяки впровадженню яких підвищується оперативність обробки даних і ділової інформації, приймаються більш об'єктивні фінансові й управлінські рішення [1].

Автоматизація бухгалтерського обліку на підприємстві і підготовка фінансової звітності в податкові органи є однією з найбільш важливих завдань. Сам по собі бухгалтерський облік на підприємстві може розглядатися як внутрішня справа підприємства, а основою для оцінки фінансово-господарської діяльності підприємства з боку держави служить звітність, яка повинна щокварталу надаватися в податкову інспекцію за місцем реєстрації підприємства [2].

Метою даної роботи є створення програмного продукту для підготовки, формування і аналізу первинних банківських платіжних документів відповідно до прийнятих стандартів, регламентованими податковим законодавством України.

На ринку програмного забезпечення для автоматизації бухгалтерського обліку та фінансових документів, наведеними до правових норм України, основними аналогами є такі потужні програмні продукти як «Бухгалтерія 1С», «АРМ-звіт страхувальника», «БУМ», «М. Е. DOC» і т.п.

В даних програмних продуктах поєднуються складні системи автоматизованого обліку і простота у використанні, що робить їх доступними і незамінними на будь-якому підприємстві, незалежно від сфери його діяльності. Але наведене програмне забезпечення надає великий і складний набір послуг і функцій для бухгалтерського, фінансового та податкового обліку.

Для автоматизації бухгалтерського обліку малих підприємств України, недоцільно використання таких потужних і дорогих програмних продуктів як «Бухгалтерія 1С». До того ж при використанні цих програмних продуктів для ведення бухгалтерського і податкового обліку підприємств потрібно навчити і підготувати персонал, крім того, підприємство буде нести витрати по утриманню відповідного фахівця для обслуговування і супроводу даного ПЗ. А все це вельми великі витрати і невиправдані витрати, коли потрібно програма, яка виконує конкретний набір функцій з інтуїтивно зрозумілим інтерфейсом і вимагає на навчання досить мало часу [3].

Як правило, потрібні кілька програмних модулів для формування первинних фінансових платіжних документів і бланків податкової звітності.

Таким чином, розробка комп'ютерної системи обліку первинних бухгалтерських документів є актуальним завданням, і призначена для створення, аналізу, формування, обліку та зберігання первинних платіжних доручень, а так само і квитанцій-повідомлень.

Підготовка шаблонів платіжних документів виконана відповідно до вимог Національного банку України (Інструкція НБУ «Про безготівкові розрахунки в Україні в національній валюті», затверджена 21.01.2016 постановою Правління НБУ № 22, з урахуванням змін від 15.04.2016 та 18.10.2016).

Дипломна робота містіть 55 сторінок, 5 таблиць, 29 рисунків, 14 посилань та 1 додаток.

1 ОСНОВНІ ПРИНЦИПИ ОБЛІКУ ПЕРВИННИХ БУХГАЛТЕРСЬКИХ ДОКУМЕНТІВ

1.1 Бухгалтерські первинні документи

Первинний документ – документ, що включає вихідні відомості, отримані в процесі досліджень, розробок, спостережень та інших видів людської діяльності.

До бухгалтерії первинний документ здається або там же складається у момент здійснення господарської операції або, якщо це не надається можливим – безпосередньо після його закінчення і є першим свідченням, що відбулися факти здійснення операції. Первинний документ підтверджує юридичну силу виконаної господарської операції. Він встановлює відповідальність окремих виконавців за виконані ними господарські операції.

До первинних документів належать касовий ордер, накладна, довідка, акт, платіжне банківське доручення, квитанція-повідомлення і т. п.

Для більшості первинних документів введені уніфіковані форми, наприклад, для документів з обліку основних засобів, праці, торгових операцій, грошових розрахунків. Якщо офіційна форма документа не введена, то організація сама розробляє цю форму. При цьому документ повинен містити такі обов'язкові реквізити [1]:

- назва документу;
- дата складання документа;
- найменування організації;
- зміст господарської операції;
- вимірювачі операції;
- перелік відповідальних посадових осіб;
- особисті підписи зазначених осіб.

Бухгалтерський облік є документальним. Відображення господарських операцій на рахунках проводиться тільки на підставі документації. Документ – це письмовий доказ дійсного здійснення господарської операції і права на її вчинення. Якість бухгалтерського обліку залежить, перш за все, від правильності і своєчасності складання документів. Документи служать для попереднього і подальшого контролю за збереженням майна, за законністю і доцільністю господарських операцій [3].

Попередній контроль здійснюється керівними працівниками організацій при підписанні документа, на підставі якого здійснюється певна госпо-

дарська операція (прийом і видача матеріалів, виплата грошей з каси і т.д.). Підписуючи документ, працівник бере на себе відповідальність за законність і доцільність даної господарської операції. Це покладає на кожного працівника, який підписав документ, особисту відповідальність за даний дозвіл. Наступний контроль проводиться, головним чином, у формі документальних ревізій, шляхом перевірки в бухгалтерії всіх документів, що надходять, а також при аудиторській перевірці.

Первинная документація, як спосіб оформлення господарських операцій документами і обгрунтування бухгалтерських записів є одним з елементів методу бухгалтерського обліку. З документацією тісно пов'язані інші елементи методу бухгалтерського обліку, зокрема, рахунки і інвентаризація.

Первнинні документи є підставою для подальших записів операцій в системі рахунків. Вони широко використовуються для аналізу господарської діяльності організації, для оперативного керівництва та управління господарською діяльністю, так як служать виправданням тих чи інших дій оперативних працівників. Підставою для відображення господарських операцій в системі бухгалтерського обліку можуть бути тільки правильно оформлені документи. На підставі документів ведеться повсякденне спостереження за рухом товарно-матеріальних цінностей, готової продукції, грошових коштів, встановлюються законність і доцільність операцій. Важливе значення мають документи при ревізії фінансово-господарської діяльності, при аудиторській перевірці, при розгляді кримінальних і цивільних справ в процесі дізнання, слідства і суду, набуваючи юридичну силу доказу [2].

В організаціях в процесі виконання фінансово-господарської діяльності здійснюються різноманітні операції. Тому різні і документи, якими вони оформляються. Різноманітність документів пояснюється також особливостями їх в обліковій роботі.

Для вибору найбільш раціональних форм, видів при оформленні тих чи інших операцій всі документи класифікують (групують) за кількома ознаками [3]:

- за призначенням,
- по порядку відображення операцій,
- за способом охоплення операцій,
- за кількістю облікових записів,
- за місцем складання.

Ст. 9 «Про бухгалтерський облік в Україні» в останній чинній редакції від 3 січня 2017 року, говорить: «Всі господарські операції, що проводяться

організацією, повинні оформлятися виправдувальними документами, на підставі яких ведеться бухгалтерський облік. Первинні документи приймаються до обліку, якщо вони складені за формою, що міститься в альбомі уніфікованих форм первинної облікової документації, а документи, форма яких не передбачена в цих альбомах, повинні містити обов'язкові реквізити».

У повсякденній роботі організацій створюються документи з різних питань виробничої, господарської, фінансової і громадської діяльності. Це накази, рішення, листи, акти, договори, протоколи, заяви, телеграми, довідки та ін. Бухгалтерські документи групуються за такими розділами обліку [1]:

- сільськогосподарська продукція;
- праця і її оплата;
- основні засоби та нематеріальні активи;
- матеріали;
- роботи в капітальному будівництві;
- роботи будівельних машин і механізмів;
- роботи в автомобільному транспорті;
- результати інвентаризації;
- розрахунково-касові операції;
- торгові операції.

Документування управлінської діяльності полягає у фіксації різними способами (за встановленими формами) відповідної інформації, тобто створенні документів.

Первинні та зведені облікові документи можуть складатися на паперових і машинних носіях інформації. В останньому випадку організація зобов'язана виготовляти власним коштом копії таких документів на паперових носіях для інших учасників господарських операцій, а також на вимогу органів, що здійснюють контроль відповідно до законодавства, суду і прокуратури (ст.9 закону «Про бухгалтерський облік» від 03.01. 2017р. № 129-З). Документи необхідні для, того, щоб протягом певного часу можна було користуватися необхідної управлінської інформацією для самих різних цілей.

Рішення поставленої перед бухгалтерським обліком завдання із забезпечення інформацією внутрішніх і зовнішніх користувачів і показників діяльності організацій з метою достовірної оцінки їх фінансового та майнового стану призвело до зростання обсягу інформації та, в свою чергу, підвищення вимог до її якості.

Це вимагає встановлення дієвого контролю за достовірність облікової інформації як найважливішого джерела для прийняття обґрунтованих управ-

лінських рішень. У зв'язку з цим дослідження проблем вдосконалення документації і документообігу набуває практичне значення. Проте залишаються відкритими питання складання та аналізу документації та документообігу з метою підвищення ефективності цих процесів.

Актуальність проблеми підтверджується тим, що сьогодні організація бухгалтерського обліку і контролю повинна забезпечити достовірну інформацію не тільки по витраті бюджетних коштів, а й за кожним видом доходів і витрат від підприємницької та іншої діяльності [3].

Повна і достовірна інформація повинна бути сформована про всі сторони економічної і господарської діяльності як всередині організацій, так і в їхніх взаєминах з іншими господарськими організаціями та з фінансово-кредитною системою.

Практика переконує, що перебудова бухгалтерського обліку та звітності невіддільна від упорядкування документації і документообігу.

Посилення ролі первинних документів як основи бухгалтерського обліку випливає з того, що близько 90% необхідної для управління інформації становить облік їх діяльності – робота з документами.

Специфічною особливістю бухгалтерського обліку в єдиній системі облікового спостереження є суцільне за охопленням, безперервне за часом, документальне по реєстрації та узагальнююче по вимірюванню відображення об'єктів, що враховуються і процесів. Відбуваються якісні зміни документування та документаційного обслуговування фінансово-господарської діяльності організацій потребують удосконалення форм, методів і технології роботи з документами [4].

1.2 Вимоги до змісту та складанню бухгалтерських документів

Документи складаються з окремих елементів (показників), які називають реквізитами (від лат. Reguisitum – потрібне, необхідне).

Сукупність реквізитів документа зумовлює його форму [1].

Щоб документ відповідав своєму призначенню, він повинен бути складений відповідно до форми, прийнятої для даної категорії документів.

Від повноти і якості оформлення документів залежить їх доказова (юридична) сила, так як вони є свідченням, підтвердженням конкретних фактів. Всі реквізити поділяються на обов'язкові та додаткові.

Відповідно до вимог Положення по веденню бухгалтерського обліку і бухгалтерської звітності в Україні, первинні документи повинні містити такі обов'язкові реквізити [3]:

– найменування посад осіб, відповідальних за здійснення господарської операції і правильність її оформлення;
– особисті підписи і розшифровки (включаючи випадки створення документів із застосуванням засобів обчислювальної техніки).

У міру необхідності в зміст документа можуть бути включені додаткові реквізити: емблема організації, найменування вищестоящої організації, індекс підприємства зв'язку, поштова, телеграфна адреса, номер, посилання на індекс і дату вхідного документа, гриф обмеження доступу до документа, резолюція, відмітка про контроль, візи, друк, прізвище виконавця і номер його телефону, позначка про перенесення даних на машинний носій і інші, за рішенням організації.

Сукупність розташованих у встановленій послідовності реквізитів документа називається його формуляром.

Формуляр для певного виду документів називається типовим (наприклад, для наказів, протоколів і т.п.). Такий формуляр характеризується встановленим набором реквізитів, розташованих у суворій послідовності.

Всі первинні документи повинні бути складені в момент здійснення господарської операції, якщо це неможливо з яких-небудь причин – безпосередньо після закінчення операції.,

Первинні та зведені облікові документи можуть складатися на паперових і машинних носіях. В останньому випадку організація зобов'язана виготовляти власним коштом копії таких документів на паперових носіях для інших учасників господарських операцій, а також на вимогу органів, що здійснюють контроль відповідно до законодавства України, суду і прокуратури (ст.9 п.7 № 129 «Про бухгалтерський облік»).

Форми первинних облікових документів, застосовуваних для оформлення господарських операцій, по яких не передбачені типові форми первинних облікових документів, а також форми документів для внутрішньої бухгалтерської звітності повинні затверджуватися наказом керівника організації разом з обліковою політикою [1].

Документи по фінансовим, кредитним і фінансовим операціям повинні мати підписи керівника організації та головного бухгалтера або уповноваженими ними особами.

Список осіб, які мають право першого та другого підпису, оформляється наказом по організації.

Без підпису головного бухгалтера грошові та розрахункові документи, фінансові і кредитні зобов'язання вважаються недійсними і не повинні прийматися до виконання.

Первинні документи повинні бути складені в момент здійснення операції, а якщо це не представляється можливим, то безпосередньо після закінчення операції.

За достовірність які у документі даних, доброякісність його складання відповідальність несуть посадові особи, які підписали документ. На документах не повинно бути не обумовлених виправлень, а на банківських і касових документах виправлення взагалі не допускаються, навіть обумовлені.

Помилки в документах виправляються зачеркиванием неправильного запису (однією рисою, щоб можна було прочитати закреслене) і обумовлені підписами осіб, які підписали документ, із зазначенням дати виправлення [2].

Комплекс документів визначається:

- колом питань, що вирішуються організацією в процесі своєї діяльності;
- порядком вирішення питань (на основі колегіальності або єдиноначальності);
- об'ємом і характером зовнішніх зв'язків.

Управлінський документ повинен відповідати наступним вимогам:

- бути складеним за встановленою формою, а в ряді випадків відповідати стандартам;
- видаватися відповідним компетентним органом або посадовою особою, а так само і особами, яким таке право надається законом або відповідним директивним зазначенням;
- видаватися на виконання норм права і не суперечити їм.

Текст документа повинен бути ясним і точним. Укладач повинен завжди пам'ятати, що документи повинні відповідати чинному законодавству, тому в разі необхідності слід звіряти документи з законами і підзаконними актами.

Особливо це важливо враховувати при підготовці рішень, наказів та інших розпорядчих документів.

У документах треба використовувати тільки перевірені, достовірні факти. Технологія проходження первинних документів представлена на рис. 1.1.

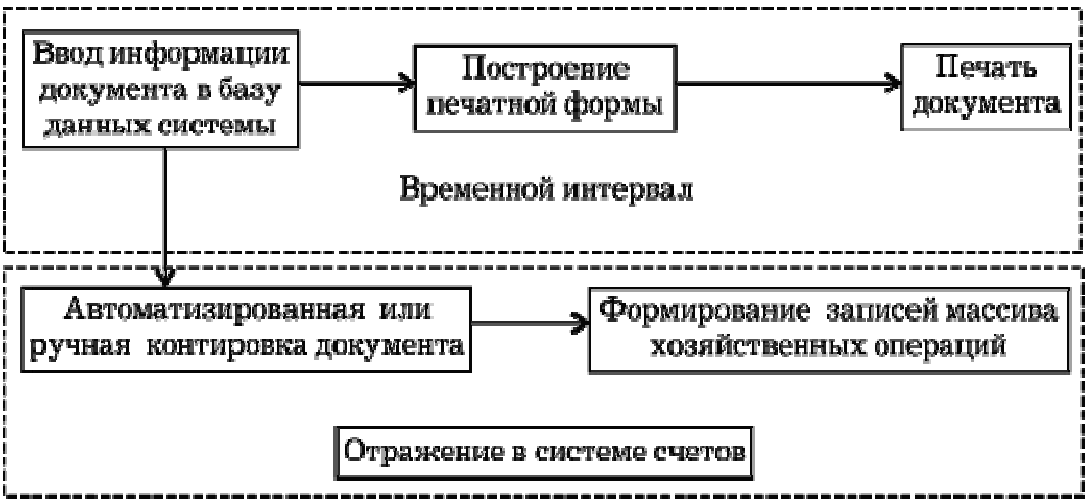


Рисунок 1.1 – Технологія проходження первинних документів

Підставою для бухобліку господарських операцій є первинні документи. Для контролю та впорядкування оброблення даних на підставі первинних документів можуть складатися зведені облікові документи.

Первинні документи підлягають обов'язковій перевірці працівниками, які ведуть бухоблік, за формою і змістом, т. п. Перевіряється наявність у документі обов'язкових реквізитів та відповідність господарської операції законодавству в сфері бух обліку [3].

В Законі про бухоблік зафіксували можливість складання первинних документів в електронній формі. Первинні документи, складені в електронній формі, застосовуються в бухгалтерському обліку за умови дотримання вимог законодавства про електронні документи та електронний документообіг (п. 2 ст. 9 Закону про бухоблік).

1.3 Визначення функцій та вимог системи обліку первинних бухгалтерських документів

Завданням дипломної роботи є розробка комп’ютерної системи обліку первинних бухгалтерських платіжних документів, призначена для створення, аналізу і формування банківських платіжних доручень, а так само квитанцій-повідомлень. Розробка даної системи дозволяє підвищити ефективність роботи персоналу. Крім того, підвищується оперативність обробки даних і вірогідність ділової інформації, і як наслідок – приймаються більш об'єктивні фінансові й управлінські рішення [4]. Схема автоматизованого складання інформації наведена на рис. 1.2.

Рисунок 1.2 – Автоматизоване заповнення первинної бухгалтерських даних

Зберігання документів у вигляді комп'ютерних даних на диску безумовно доцільніше, ніж зберігання їх в класичному вигляді, тобто у вигляді паперів. Значно спрощується пошук потрібного документа, є можливість зберігати дані за багато років і не плутатися в них, сильно спрощується зміна будь-якого документа, складання численних довідок [4].

1.4 Вимоги до програмного забезпечення комп’ютерної системи

Програмне забезпечення комп’ютерної системи обліку первинних бухгалтерських документів та складання бухгалтерських платіжних документів для представлення в установи банків, що відповідають вимогам Національного банку України, має забезпечувати виконання таких завдань:

- ведення бази платників і одержувачів з їхніми рахунками банків, платежів і, безпосередньо, платіжних доручень;
- попередній перегляд і друк платіжного доручення (квитанцій);
- фільтрація списку платіжних доручень за всіма реквізитами (квитанцій);
- автоматична нумерація платіжного доручення при формуванні нового доручення чи квитанцій;

- автоматична підстановка рахунку і банківських реквізитів при виборі платника або одержувача в ході формування нового доручення (квитанцій);
- пошук платника або одержувача за кодом ОКПО або найменуванням (рекурсивний пошук);
- формування суми платежу прописом;
- конструктор (дизайнер) форми платіжного доручення (квитанцій);
- ведення архіву бази;
- пошук доручення в базі: за номером, сумою, датою;
- створення нового типу платежу на основі поточного доручення;
- забезпечення макроподстановкі в полі «Призначення платежу».

При підготовці платіжних документів за допомогою даної програми забезпечуються:

- запам'ятовування всіх реквізитів платника;
- послідовна нумерація платіжних документів;
- адаптивне ведення довідника платежів.

Крім того, система дозволяє вести облік одержувачів і платників (постачальників і покупців товарів і послуг), реалізуючи наступні функції:

- довідник одержувачів з можливістю фільтрації, пошуку;
- довідник платників з індивідуальними настройками;
- висновок примірників платіжного доручення довільного формату;
- банківські реквізити – індивідуальні значення поля «Призначення платежу» для кожного одержувача з можливістю додавання в доручення (квитанцію);
- прості налаштування інтерфейсу форми роботи з платіжними дорученнями і квитанціями;
- формування документа «квитанція-сповіщення».

У зв'язку з частим внесенням поправок до податкового законодавства України і як наслідок зміною форм платіжних доручень і квитанцій-повідомлень в системі повинен бути реалізовані:

- конструктор форм (шаблонів) платіжних доручень;
- конструктор форм (шаблонів) квитанцій.

Конструктор форм повинен забезпечувати наступні можливості:

- створення нових форм (шаблонів)документів,
- редагування існуючих форм (шаблонів)документів;
- форматування існуючих шаблонів друкарських форм платіжних документів.

2 ВИБІР АРХІТЕКТУРИ ТА ПРОГРАМНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ СИСТЕМИ

2.1 Обгрунтування вибору програмних засобів реалізації

Різні розробники по різному підходять до вибору програм і технологій створення систем. Розглянемо переваги і недоліки різних підходів в реалізації конкретних систем обліку. Одним з частих підходів є використання систем сервер-клієнта, в яких в якості клієнта виступає браузер, а як сервер-зв'язка з Web-сервера і сервера додатків [5].

Мова HTML орієнтована на представлення даних, а не на їх створення, занадто ускладнює редагування вже створених даних. Це є проблемою багатьох систем заснованих на Web-інтерфейсі, крім того відсутня можливість використання при підготовці платіжних документів технології OLE, що ускладнює вставку складних об'єктів (формул, полів і ін.).

Перевагами такого підходу є:

– відсутня залежність від операційної системи;
– відсутня необхідність в установці і настройці клієнтської частини;
– мова HTML має великі можливості за поданням даних;
– відсутність проблем з передачею даних в мережі Інтернет.

Альтернативою є використання спеціалізованого програмного забезпечення для створення програмного продукту для обліку та складання первинних бухгалтерських платіжних документів. Даний підхід має наступні переваги:

– зручність створення і редагування платіжних документів;
– автономність – не потрібне наявність підключення до мережі.

Використання спеціалізованого програмного забезпечення для обліку первинних платіжних документів дозволяє [4]:

– мінімізувати обсяг переданння даних;
– реалізувати складні структури документів.

Недоліки спеціалізованого програмного забезпечення:

– можливі обмеження за форматами представлення інформації;
– необхідність установки спеціального клієнтського додатка;
– проблема оновлення версій клієнтів при вдосконаленні програмного забезпечення.

При аналізі аналогів програм для ведення автоматизованого обліку та складання первинних фінансових і бухгалтерських документів на підприємс-

твах, більшість з них вибрали використання спеціалізованого програмного забезпечення, так як цей напрямок має значні переваги, такі як: при проектуванні комп’ютерної системи обліку та складання первинних бухгалтерських платіжних документів планувалося мінімізувати обсяг переданої інформації; також, повинна бути врахована автономність, тобто відсутність залежності від наявності зв'язку; зручність і простота у використанні і налаштуванні [4].

Для реалізації комп’ютерної системи підготовки та обліку первинних платіжних документів обрана програмну оболонку Lazarus з компілятором Free Pascal та мовою Object Pascal, для створення бази даних СУБД InterBase, для проектування і створення дизайну Adobe Photoshop.

2.2 Вибір архітектури системи

В основі широкого розповсюдження локальних мереж комп'ютерів лежить відома ідея поділу ресурсів. Висока пропускна здатність локальних мереж забезпечує ефективний доступ з одного вузла локальної мережі до ресурсів, що знаходяться в інших вузлах.

Розвиток цієї ідеї призводить до функціонального виділення компонентів мережі. Розумно мати не тільки доступ до ресурсів віддаленого комп'ютера, але також отримувати від цього комп'ютера деякий сервіс, специфічний для ресурсів даного роду і програмні засоби для забезпечення, яких недоцільно дублювати їх в декількох вузлів, тобто відбувається поділ робочих станцій і серверів локальної мережі.

Робоча станція призначена для безпосередньої роботи користувача або категорії користувачів і володіє ресурсами, відповідними локальним потребам даного користувача. Сервер локальної мережі повинен володіти ресурсами, відповідними його функціональному призначенню і потребам мережі [6].

Практично всі сучасні СУБД використовують у своїй роботі технологію «клієнт-сервер» і СУБД InterBase не є винятком. «Клієнт-сервер» – це модель взаємодії комп'ютерів в мережі.

Щодо систем баз даних архітектура «клієнт-сервер» цікава і актуальна головним чином тому, що забезпечує просте і відносно дешеве рішення проблеми колективного доступу до баз даних в локальній мережі.

У деякому роді системи баз даних, засновані на архітектурі «клієнт-сервер», є наближенням до розподілених систем баз даних. У мережі один і той же комп'ютер може виконувати роль як клієнта, так і сервера.

У загальному випадку, щоб прикладна програма, яка виконується на робочій станції, могла використовувати послугу деякого сервера, як мінімум потрібно деякий інтерфейсний програмний код, що підтримує такого роду взаємодію. З цього, власне, і випливають основні принципи системної архітектури «клієнт-сервер» [7].

Система розбивається на дві частини, які можуть виконуватися в різних вузлах мережі, клієнтську і серверну частини.

Прикладна програма або кінцевий користувач взаємодіють з клієнтською частиною системи, яка в найпростішому випадку забезпечує просто міжмережевий інтерфейс. Клієнтська частина системи при потребі звертається по мережі до серверної частини.

Термін «сервер баз даних» зазвичай використовують для позначення всієї СУБД, заснованої на архітектурі «клієнт-сервер», включаючи і серверну, і клієнтську частини. Такі системи призначені для зберігання і забезпечення доступу до баз даних. Модель функціонування сервера баз даних наведена на рис. 2.1.

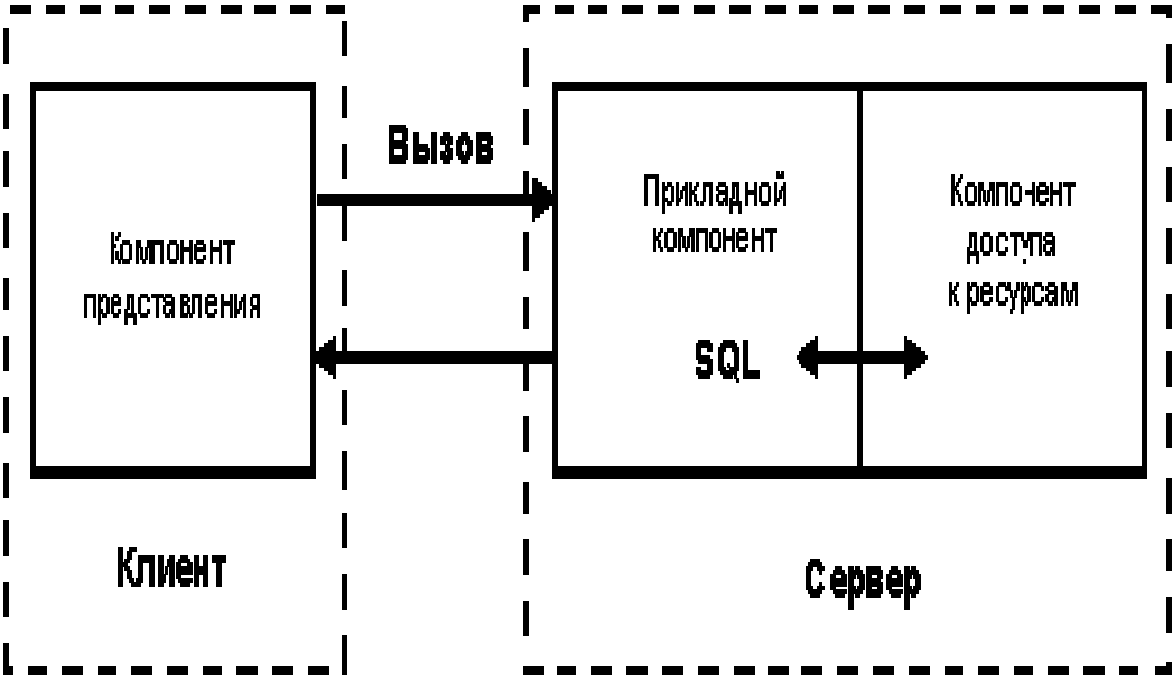


Рисунок 2.1 – Модель функціонування сервера бази даних

Хоча зазвичай одна база даних цілком зберігається у одному вузлі мережі і підтримується одним сервером, сервери баз даних є простим і дешевим наближенням до розподілених баз даних, оскільки загальна база даних доступна для всіх користувачів локальної мережі.

2.3 Забезпечення доступу до бази даних

Підключення до баз даних відбувається за допомогою ADO (Active Data Objects) – це високорівнева компонент технології доступу до даних від Microsoft. Даними для ADO можуть бути як звичні таблиці Access або серверні бази MS SQL або Oracle, так і Microsoft Active Directory Service, XML-файли і т.п. ADO- технологія, працює через інтерфейс OLE DB [8].

Середовище швидкої розробки додатків Lazarus забезпечує пряму взаємодію з SQL-сервером InterBase і дозволяє підтримувати повний життєвий цикл інформаційного додатка, починаючи від розробки простого прототипу додатка і закінчуючи його супроводом і модернізацією [9].

Доступ до бази даних від прикладної програми або користувача виробляється шляхом звернення до клієнтської частини системи. Як основний інтерфейс між клієнтської і серверної частинами виступає мова баз даних SQL.

Це мова по суті справи є поточним стандартом інтерфейсу СУБД у відкритих системах. Збірна назва SQL-сервер відноситься до всіх серверів баз даних, заснованих на SQL. Дотримуючись обережності при програмуванні, можна створювати прикладні інформаційні системи, мобільні в класі SQL-серверів.

Сервери баз даних, інтерфейс яких базується виключно на мові SQL, мають свої переваги і свої недоліки. Очевидна перевага стандартність інтерфейсу. В ідеалі, клієнтські частини будь-якої SQL-орієнтованої СУБД могли б працювати з будь-яким SQL-сервером незалежно від того, хто його створив. Недолік теж досить очевидний. При такому високому рівні інтерфейсу між клієнтської і серверної частинами системи на стороні клієнта працює занадто мало програм СУБД. Це нормально, якщо на стороні клієнта використовується малопотужна робоча станція. Але якщо клієнтський комп'ютер має достатню потужність, то часто виникає бажання покласти на нього більше функцій управління базами даних, розвантаживши сервер, який є слабким місцем всієї системи [10].

Одним з перспективних напрямків СУБД є гнучка конфігурація системи, при якій розподіл функцій між клієнтською і призначеної для користувача частинами СУБД визначається при установці системи. У типовому випадку на стороні клієнта СУБД працює тільки таке програмне забезпечення, яке не має безпосереднього доступу до баз даних, а звертається для цього до сервера з використанням мови SQL.

СУБД InterBase – це система управління базами даних, що поставляється корпорацією Borland для побудови додатків з архітектурою клієнт-сервер довільного масштабу. Сервер InterBase – це кросплатформова СУБД, що підтримує більшість операційних систем: Windows, Linux, Unix, Solaris, Mac OS та інші. Сервер InterBase має цілу низку переваг, що вигідно відрізняють його від інших СУБД [11]:

- оновлювані уявлення View;
- двофазне підтвердження транзакцій;
- ефективний механізм тригерів;
- серверна обробка Blob-полів (Blob-filters);
- події (повідомлення);

Повнофункціональна СУБД InterBase, забезпечує легку масштабованість, не вимагає складного адміністрування, підтримує Unicode, відповідає стандартам SQL, кросплатформова. Вона підтримує найширший діапазон способів використання – від окремого вбудованого пристрою до складних корпоративних рішень з сотнями користувачів.

InterBase виконує шифрування стовпців даних, відстеження змін на рівні полів, вирішує завдання представлення даних для конкретного користувача, проводить журналізацію, здійснює відновлення даних на конкретний момент часу. Всі ці можливості надають легкий, компактний та простий у використанні та адмініструванні засіб для реалізації розподілених систем.

Ця СУБД має мінімальні вимоги до простору на диску і розміру пам'яті, що робить її невимогливою при розгортанні де завгодно – як на серверах з Windows або Linux (на своєму сервері або в хмарі), так і на мобільних пристроях. Кросплатформена дискова архітектура зберігання бази даних InterBase принципово знижує проблеми з розгортанням систем [11].

При використанні InterBase дуже легко здійснювати переноси даних між базами для розробки, тестування і експлуатації систем – не залежно від того, під якими операційними системами працюють ці БД. InterBase повністю відповідає вимогам ACID (атомарність, узгодженість, ізольованість, довговічність).

InterBase підтримує всі основні мови програмування, включаючи Java, C, C ++, C#, .NET, Delphi, PHP і Ruby. Він тісно інтегрується з C ++ Builder, RAD Studio, Lazarus та Delphi [12]. При розгортанні систем на будь-якій платформі, СУБД InterBase може автоматично розгортатися разом з цим ПО. Якщо для розробки систем використана інтегрована середа Visual Studio, то

працювати з InterBase можливо через ADO.NET або ODBC, що забезпечить вам легкий доступ до даних.

InterBase підтримує роботу з використовуючи кілька мережевих протоколів: TCP/IP, NetNEUI, IPX/SPX, та забезпечує роботу механізму оптимістичного блокування на рівні запису. Це означає, що сервер блокує тільки ті записи, які реально були змінені користувачем, і не блокує всю сторінку даних цілком. Ця особливість ще більше знижує ймовірність конфліктів при багато користувальницькому режимі.

У порівнянні з багатьма іншими СУБД, InterBase надає дуже ефективний механізм тригерів: кожна таблиця може мати велику кількість тригерів, які виконуються автоматично при вставці, зміні або видаленні кожного окремого запису, до або після цих подій. Багато функції існуючих СУБД були вперше реалізовані в InterBase – це, зокрема, оновлювані уявлення, події (event alerters), багатовимірні масиви і BLOB-поля [11].

Важливою особливістю сервера InterBase є можливість розширення стандартного набору SQL-функцій за допомогою призначених для користувача бібліотек – User Defined Functions, а також механізми обробки BLOB-полів на сервері за допомогою BLOB-фільтрів. InterBase відрізняється значною стійкістю, оскільки спеціально був спроектований для застосування в Intranet-додатках, додатках для мобільних пристроїв і вбудованих додатках баз даних.

2.4 Вибір інтегрованого середовища розробки інтерфейсу системи

Lazarus – це IDE (Integrated Development Environment) – інтегроване середовище розробки програм, що використовує компілятор FPC (Free Pascal Compiler), редактори коду, форм, Інспектор Об'єктів, відладчик і багато інших інструментів. Середовище Lazarus – це RAD (Rapid Application Development) – середовище швидкої розробки додатків [12].

Інтегроване середовище розробки програм Lazarus став першою (і поки єдиною) безкоштовною IDE, доступною освітнім і державним установам абсолютно безкоштовно.

Lazarus є проектом Open Source – проектом з відкритим вихідним кодом. Багато програмістів по всьому світу беруть участь в його розвитку, вихідний код Lazarus доступний для вивчення і модифікації. Lazarus має підтримку безлічі мов, в тому числі і російською та українською, що вигідно відрізняє його від інших IDE. FPC – кросплатформовий інструмент, що під-

тримує величезну кількість платформ. Серед них – AmigaOS, DOS, Linux, BSD, OS/2, MacOS і Win32.

Інтегроване середовище розробки (Integrated Development Environment) IDE Lazarus – це вільне середовище розробки програмного забезпечення для компілятора Free Pascal мовою Object Pascal і поширюється під вільними ліцензіями GNU General Public License (GNU GPL) і GNU Lesser General Public License (GNU LGPL). Вона являє собою середовище з графічним інтерфейсом для розробки програм, схожа на Delphi, і базується на кросплатформової бібліотеці візуальних і не візуальних компонентів LCL (Lazarus Component Library), сумісних з VCL Delphi. Такого набору компонентів Lazarus досить для створення додатків з графічним інтерфейсом, які працюють з базами даних та Інтернетом. В даний час Lazarus працює під управлінням операційних систем (ОС) Linux, Mac OS X, UNIX-подібних, Windows, Android, в результаті чого додатки, створені під одну ОС, без праці можуть бути пере компільовані під іншу ОС. Тобто, використовуючи середу Lazarus, можна створювати кросплатформові додатки. Крім того середовище Lazarus має досить багато локалізацій, в тому числі російську та українську та надає можливість кроссплатформової розробки додатків в Delphi-подібному оточенні. Дозволяє досить нескладно переносити Delphi-програми з графічним інтерфейсом в різні операційні системи: Linux, FreeBSD, Mac OS X, Microsoft Windows. Починаючи з Delphi XE2 в самому Delphi є можливість компіляції програм для Mac OS X і OS. Lazarus заснований на бібліотеці візуальних компонентів Lazarus Component Library (LCL) [13].

Lazarus має інтерфейс налагодження (використовується зовнішній відладчик GDB) і простий перехід для Delphi програмістів завдяки близькості LCL до VCL. Пакет має повністю Юнікодний інтерфейс (UTF-8)і редактор, тому відсутні проблеми з імпортом коду, що містить національні символи. У Lazarus є потужний редактор коду, що включає систему підказок, гіпертекстову навігацію по вихідним текстам, а також форматування коду з використанням механізмів Jedi Code Format.

У ньому здійснена підтримка двох стилів асемблера: Intel і AT & T (підтримуються з боку компілятора). У Lazarus здійснена підтримка безлічі типів синтаксису Pascal: Object Pascal, Turbo Pascal, Mac Pascal, Delphi, підтримуваних з боку компілятора. Він має власний формат управління пакетами. Недоліками середовища Lazarus є відсутність повної сумісності з Delphi (хоча на відміну від Delphi надає можливість створювати більш кросплатфо-

рмені додатки), а також відсутність повної документації, хоча вона доступна у вигляді Wiki-підручників, які можуть редагувати самі користувачі [13].

Бази даних займають важливу роль в сучасному світі, їх використовують для створення більшості сайтів, інформаційних систем, різних програм. У світі налічується величезна кількість СУБД (система управління базою даних), як комерційних MsSql, Orcle, Interbase так і безкоштовних начебто MySQl, PostgreSQL, Firebird, Sqlite.

Lazarus дозволяє працювати з більшістю з них, для цього в середовище необхідно встановити компоненти для роботи з відповідними СУБД. В основі більшості баз даних лежить мова SQL (мова структурованих запитів), яка дозволяє отримувати та обробляти набори даних. Стандартний набір компонентів для роботи з базою даних в Lazarus можливо застосувати з вкладки SQLdb.

3 ПРОЕКТУВАННЯ КОМПЬЮТЕРНОЇ СИСТЕМИ

3.1 Визначення інформаційних потреб системи

Перший крок полягає у визначенні інформаційних потреб інформації, яка буде знаходитись у комп’ютерній системі для здійснення обліку первинних документів підприємства. Він включає в себе опитування майбутніх користувачів для того, щоб зрозуміти і задокументувати їх вимоги. Слід з'ясувати наступні питання:

- які дані використовуються різними додатками;
- чи зможуть нові додатки спільно використовувати будь-які з цих даних;
- хто буде вводити дані в базу і в якій формі; як часто будуть змінюватися дані;
- чи достатньо буде однієї бази або буде потрібно кілька баз даних з різними структурами;
- яка інформація є найбільш чутливою до швидкості її вилучення та зміни.

Дослідження предметної області показало, що для зберігання бази даних комп’ютерної системи обліку первинних бухгалтерських платіжних документів досить одного файлу, який буде поміщений на SQL-сервер. База даних імовірно буде не дуже великою, що не знизить швидкість вилучення інформації. Система повинна містити дані про покупців і продавців (постачальників або покупців товарів і послуг). Для підвищення інформативності системи необхідно буде реалізувати у системі Довідник одержувачів з можливістю фільтрації і пошуку та Довідник платників з індивідуальними настройками. Наявність таких Довідників дозволяє вести облік як постачальників так і покупців товарів і послуг. Крім того, на підставі даних цих Довідників формуються платіжні доручення та квитанції.

3.2 Проектування концептуальної моделі бази даних системи

Цей етап включає в себе аналіз об'єктів реального світу, які необхідно змоделювати в базі даних і складається з наступних кроків [8]:

- ідентифікація функціональної діяльності предметної області. В даному випадку мова йде про діяльність підприємства і в якості фун-

кціональної діяльності можна розглядати ведення обліку платіжних документів;

- ідентифікацію об'єктів, які здійснюють цю функціональну діяльність і ідентифікувати всі сутності і взаємозв'язку між ними.
- ідентифікацію характеристик цих сутностей;
- ідентифікацію взаємозв'язків між сутностями.

Процес «ведення обліку первинних платіжних документів» ідентифікує такі сутності: Платник; Одержувач; Тип платежу; Платіж; Місто.

Сутність Платник може включати такі характеристики як:

- ідентифікатор;
- імя,
- № розрахункового рахунку,
- ОКПО – індивідуальний однозначно ідентифікує номер підприємства або фірми;
- МФО – код банку;
- назва банку платника;
- місто, де розташований банківський рахунок Платника;
- тип написання суми (гривні або інша валюта);
- тип написання копійок,
- № примірника платіжного доручення або квитанції.

Сутність Одержувач може включати такі характеристики як:

- ідентифікатор,
- назва;
- ОКПО – індивідуальний номер, який однозначно ідентифікує номер підприємства або фірми;
- № розрахункового рахунку;
- МФО – код банку;
- назва банку отримувача;
- місто, де розташований банківський рахунок отримувача;

Сутність Тип платежу може включати такі характеристики як:

- ідентифікатор;
- назва;
- номер;
- сума;
- дата;
- ідентифікатор платника;

– ідентифікатор одержувача;
– коментар.

Сутність Платіж може включати такі характеристики як:

– ідентифікатор;
– № рахунку;
– сума;
– дата;
– ідентифікатор платника;
– ідентифікатор одержувача;
– коментар;
– ідентифікатор типу платежу.

Сутність Місто може включати такі характеристики як: ідентифікатор; назва.

Концептуальна модель бази даних комп'ютерної системи обліку первинних бухгалтерських документів побудована за допомогою CASE-засоба Power Designer та представлена на рис. 3.1.

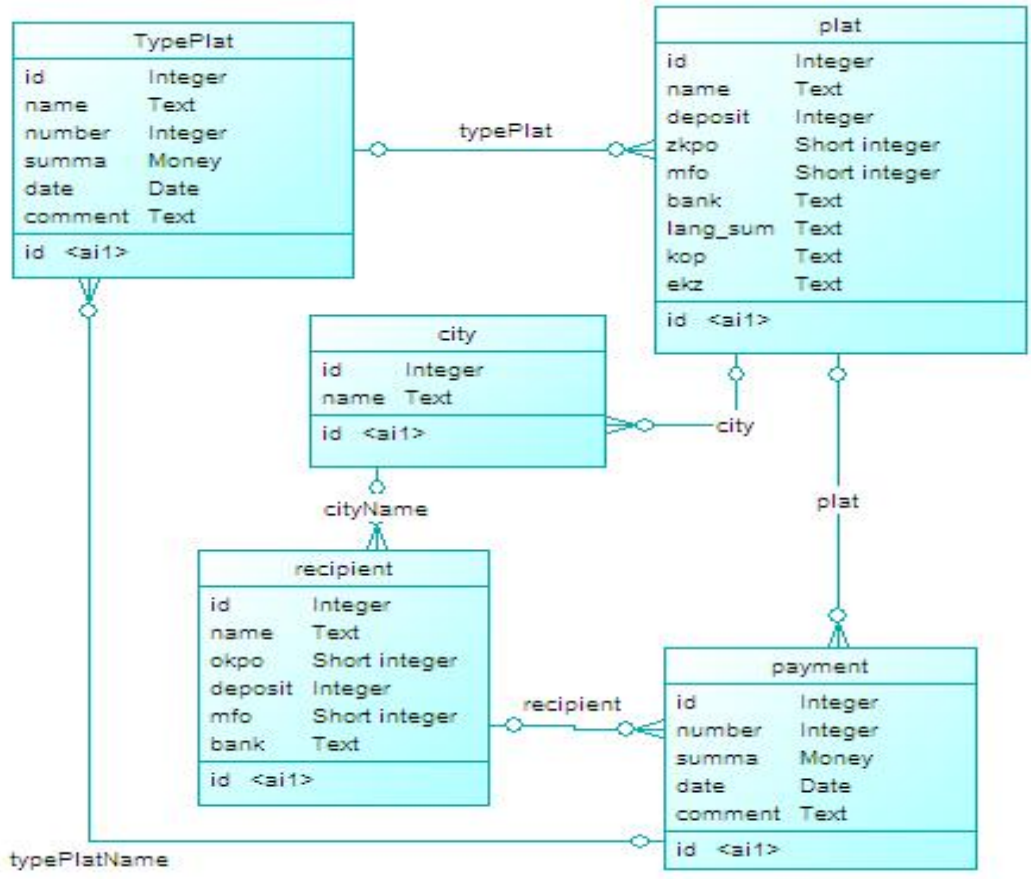


Рисунок 3.1 – Концептуальна модель бази даних системи

На основі концептуальної моделі формується фізична модель даних комп’ютерної системи обліку первинних бухгалтерських документів, яка представлена на рис. 3.2.

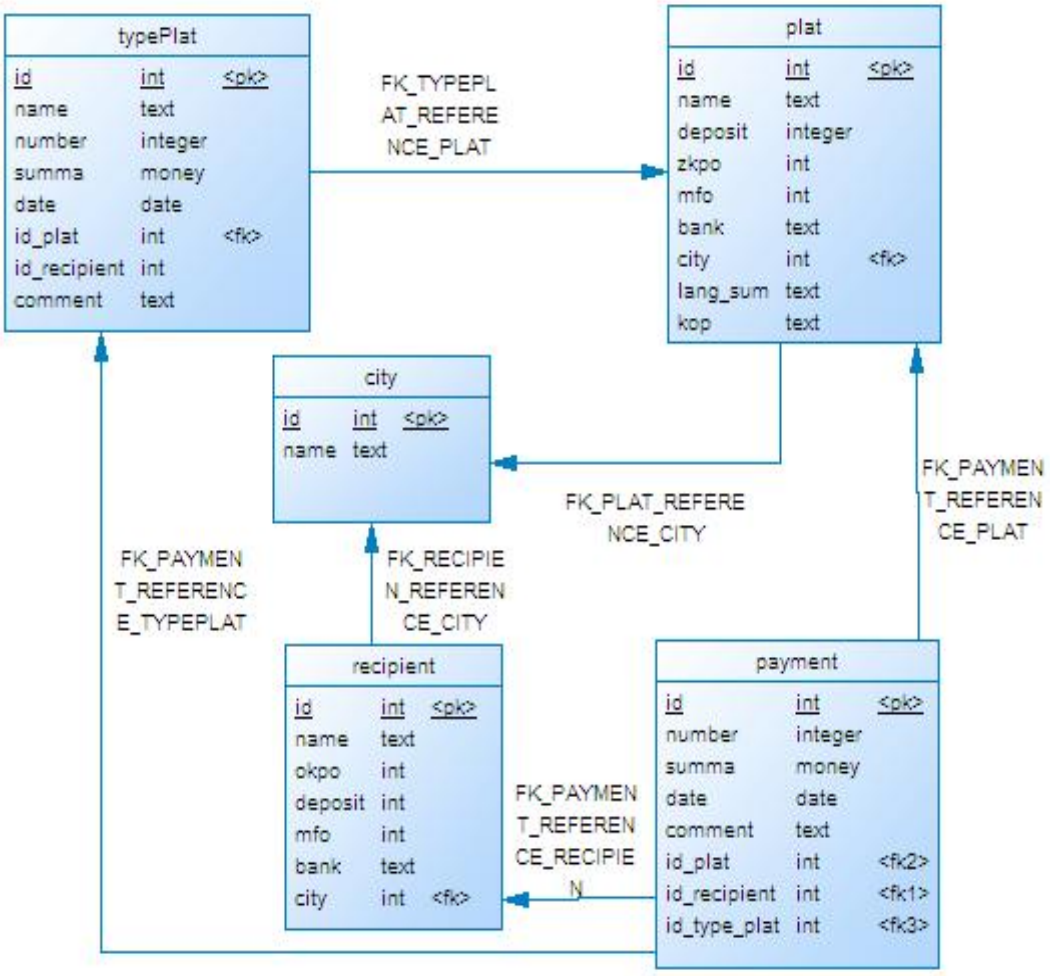


Рисунок 3.2 – Фізична модель бази даних системи

3.3 Визначення зв'язків між сутностями бази даних

Наступний етап проектування БД полягає у встановленні відповідності між сутностями і характеристиками предметної області і відносинами і атрибутами в нотації обраної СУБД. Оскільки кожна сутність реального світу володіє якимись характеристиками, в сукупності утворюють повну картину її прояви, можна поставити їм у відповідність набір відносин (таблиць) і їх атрибутів (полів).

Перерахувавши всі відносини і їх атрибути, вже на цьому етапі можна почати усувати зайві позиції. Кожен атрибут повинен з'являтися тільки один раз і треба вирішити, яке відношення буде власником якого набору атрибутів.

Потім визначаються атрибути, які унікальним чином ідентифікують кожен об'єкт. Це необхідно для того, щоб система могла отримати будь-яку одиничну рядок таблиці.

При проектуванні бази даних, наступним етапом є вироблення правил, які будуть встановлювати і підтримувати цілісність даних. На наступному етапі встановлюються зв'язки між об'єктами (таблицями і стовпцями) і виробляється дуже важлива операція для виключення надмірності даних – нормалізація таблиць [8].

Існує кілька типів зв'язків: зв'язок «один-до-одного»; зв'язок «один-до-багатьох»; зв'язок «багато-до-багатьох».

Зв'язок «один-до-одного» являє собою найпростіший вид зв'язку даних, коли первинний ключ таблиці є в той же час зовнішнім ключем, що посилаються на первинний ключ іншої таблиці.

Зв'язок «один-до-багатьох» в більшості випадків відображає реальну взаємозв'язок сутностей в предметної області. Вона реалізується вже описаною парою "зовнішній ключ – первинний ключ", тобто коли визначено зовнішній ключ, що посилається на первинний ключ іншої таблиці.

Зв'язок «багато-до-багатьох» в явному вигляді в реляційних базах даних не підтримується. Однак є ряд способів непрямої реалізації такого зв'язку, які з успіхом відшкодовують її відсутність. Один з найбільш поширених способів полягає у введенні додаткової таблиці, рядки якої складаються із зовнішніх ключів, що посилаються на первинні ключі двох таблиць. У базі даних, що спроектована для реалізації комп’ютерної системи обіку первинних бухгалтерських документів використовується зв'язок «один-до-багатьох».

Після визначення таблиць, полів, індексів і зв'язків між таблицями слід нормалізувати проектовану базу даних. Важливість нормалізації полягає в тому, що вона дозволяє розбити великі відносини, які містять велику надмірність інформації, на більш дрібні логічні одиниці, що групують тільки дані, об'єднані «по-природі» [8].

Таким чином, ідея нормалізації полягає в наступному. Кожна таблиця в реляційній базі даних задовольняє умові, відповідно до якого в позиції на перетині кожного рядка і стовпця таблиці завжди знаходиться єдине значення, і ніколи не може бути безлічі таких значень. Процес нормалізації включає:

- усунення повторюваних груп (приведення до 1НФ);
- видалення частково залежних атрибутів (приведення до 2НФ);
- видалення транзитивно залежних атрибутів (приведення до 3НФ).

Після проведення всіх вище перерахованих етапів проектування бази даних комп'ютерної системи обліку первинних бухгалтерських документів, схема бази даних представлена на рис. 3.3.

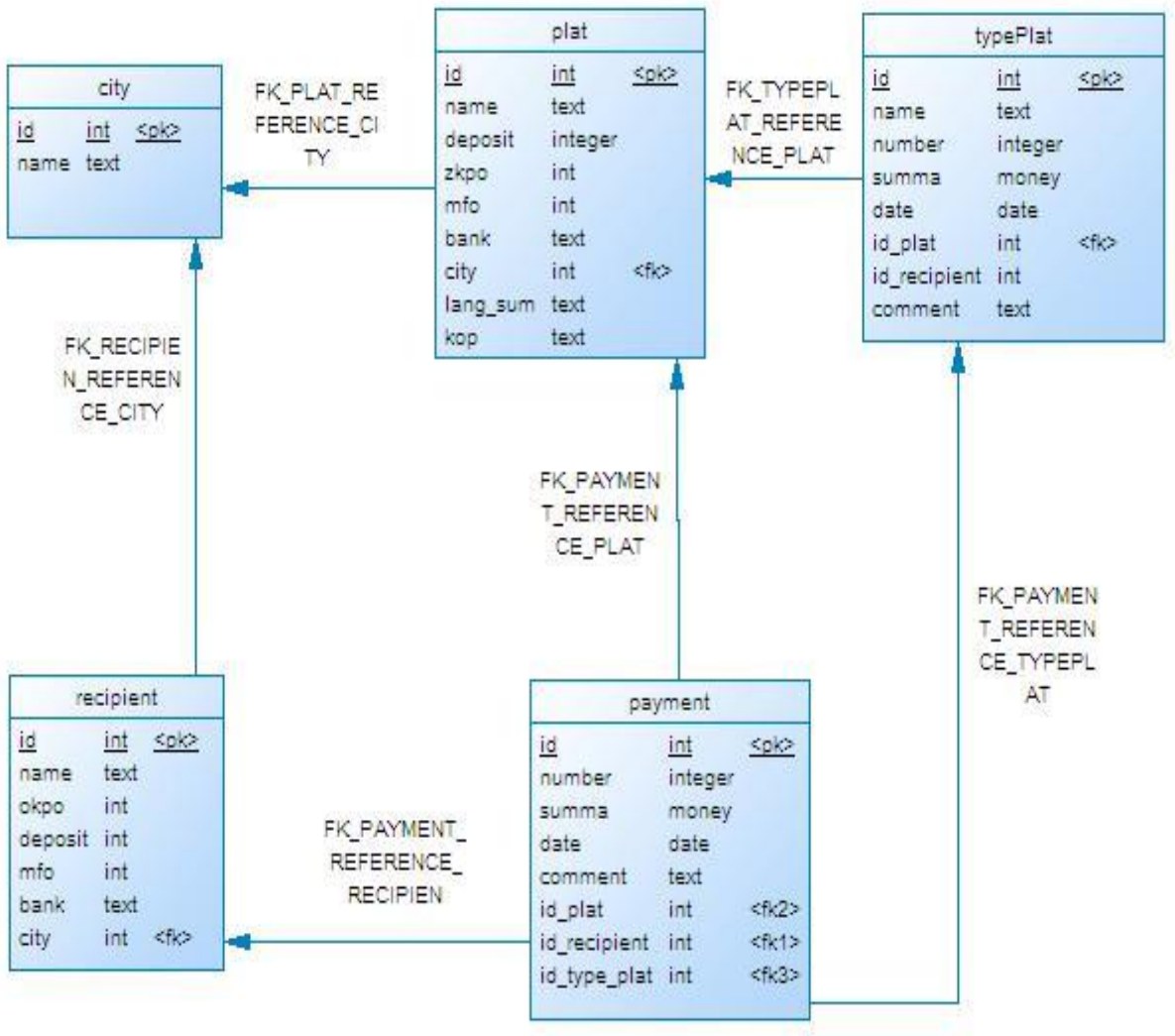


Рисунок 3.3 – Схема бази даних комп'ютерної системи

3.4 Опис таблиць бази даних і зв'язків між ними

Виходячи з вимог до розробки комп'ютерної системи обліку первинних бухгалтерських документів, а також враховуючи вимоги до інформації, що зберігається, було отримано ряд таблиць реляційної бази даних. В якості СУБД, для бази даних комп'ютерної системи обліку первинних бухгалтерських документів буде використовуватися SQL-сервер InterBase, якій дозволяє зберігати і обробляти великі обсяги інформації в умовах одночасної роботи з БД множини клієнтських додатків. На підставі врахування всіх вимог до сис-

теми створені таблиці бази даних, які мають наступні поля (табл. 3.1 – 3.5). Таблиця «Платник» – містить інформацію про всіх платників наявних в базі даних підприємства.

Таблиця 3.1 – Сутність «Платник»

Ім'я поля	Тип
ID	лічильник
Name	Varchar (35)
Deposit	Integet
Zkpo	Integer
mfo	Integer
Bank	Varchar (40)
City	integer
Lang_sum	Varchar (20)
Kop	Varchar (20)
Ekz	Varchar (20)

Таблиця «Одержувач» – містить інформацію про всіх одержувачів платежів наявних в базі даних системи.

Таблиця 3.2 – Сутність «Одержувач»

Ім'я поля	Тип
ID	лічильник
Name	Varchar (35)
Deposit	Integet
Okpo	Integer
mfo	integer
Bank	Varchar (40)
City	integer

Кожне підприємство в залежності від виду господарської діяльності, яку застосовує підприємство може мати специфічні типи платежем, де-які з

котрих можуть бути і регулярними (повторюючимися). Таблиця «Тип платежу» – містить інформацію про всіх наявних типах платежів в базі даних підприємства.

Таблиця 3.3 – Сутність «Тип платежу»

Ім'я поля	Тип
ID	лічильник
Name	Varchar (35)
number	integer
Summa	integer
Date	datetime
Id_plat	integer
Id_recipient	integer
Comment	Memo

Здійснення платежів підприємством здійснюється через розрахункові рахунки в установах банків. Підставою для проведення платежу банком є сформоване платіжне доручення. Таблиця «Платежі» – містить інформацію про всі наявні платежі в базі даних підприємства.

Таблиця 3.4 – Сутність «Платежі»

Ім'я поля	Тип
ID	лічильник
Number	integer
Summa	integer
Date	datetime
Id_plat	integer
Id_recipient	integer
Comment	Memo
Id_type_plat	integer

Таблиця «Місто» – містить назви міст, де розташовані банківські установи через які здійснюється оплата.

Таблиця 3.5 – Сутність «Місто»

Ім'я поля	Тип
ID	лічильник
Name	Varchar (35)

Таким чином, цілісність даних бази даних системи встановлена.

Заключним етапом проектування бази даних є вирішення питання надійності даних і, при необхідності, збереження секретності інформації. Для цього необхідно відповісти на наступні питання: хто буде мати права на використання бази даних, хто буде мати права на модифікацію, вставку і видалення даних, чи потрібно робити розмежування прав доступу у системі; яким чином забезпечити загальний режим захисту інформації [10].

Компьютерна система обліку первинних бухгалтерських документів, не є мережевою і передбачає однокористувальницький режим, тому і установку розмежувань прав доступу проводити не доцільно.

3.5 Розробка архітектури додатку системи

Архітектура програмного забезпечення – це сукупність узгоджених технічних рішень спрямована на задоволення вимог, що пред'являються до програмного забезпечення.

Архітектура ПЗ оперує програмними модулями – елементами структури системи, які вирішують деякі підзадачі в рамках загальних завдань системи і взаємодіючими з оточенням через певний інтерфейс.

Програмні модулі є одиницями збірки і конфігураційного управління, – файли з кодом на якійсь мові, бінарні файли, будь-які документи, що входять до складу системи і представляють собою одиниці розгортання системи [14].

Компютерна система обліку первинних бухгалтерських документів буде містити декілька основних модулів, які в свою чергу будуть поділятися на більш дрібні і спеціалізовані.

Основні модулі такі:

– майстер настройки програми;
– модуль для роботи з клієнтами;
– менеджер платіжних документів;

– довідник платників;
– довідник одержувачів;
– довідник типових платежів;
– конструктор шаблонів.

Структура взаємодії програмних модулі, що реалізують функції системи обліку первинних бухгалтерських документів представлена на рис. 3.4.

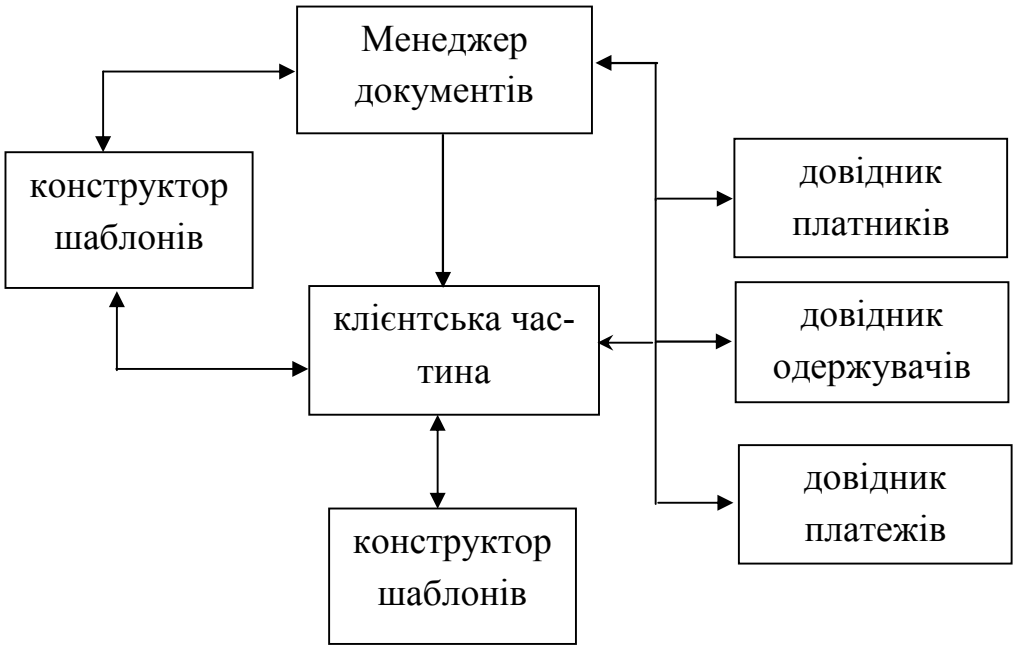


Рисунок 3.4 – Структура взаємодії модулів програми

3.6 Проектування інтерфейсу користувачів системи

Основні фактори, за допомогою яких можна оцінити доцільність впровадження компютерної системи обліку первинних бухгалтерських платіжних документів, такі.

Адекватність інтерфейсу. Адекватність призначеного для користувача інтерфейсу програми – це відповідність тим завданням, які користувачі повинні і хотіли б вирішувати з його допомогою. Це відповідність має два аспекти [7]:

– по-перше, всі потрібні користувачам завдання повинні бути розв'язані;
– по-друге, ті дії, які користувачі виконують частіше, повинні вимагати менше зусиль.

Продуктивність роботи користувачів. Це кількість однотипних реальних завдань, які користувач може вирішити за допомогою програмного забезпечення за одиницю часу.

Швидкість навчання нових користувачів. Це кількість завдань, виконання яких новий користувач самостійно навчається за одиницю часу.

Крім того, важливим показником є відповідність навчання частоті виникнення завдань – чим частіше на практиці виникає необхідність вирішити певне завдання, тим швидше користувач повинен навчитися робити це.

Ефективність запобігання та подолання помилок користувачів. Цей показник тим краще, чим рідше користувачі помиляються при роботі з даним інтерфейсом і чим менше часу і зусиль потрібно для подолання наслідків вже зроблених помилок. Велике значення має також ризик, пов'язаний з виникненням помилки.

Зручне ПЗ не повинно вимагати від користувача серйозних зусиль на вчинення дій, помилки в яких не дуже накладні; але якщо наслідки помилки катастрофічні, необхідно всіляко перешкоджати її здійсненню.

Суб'єктивне задоволення користувачів. Цей фактор визначає, наскільки інтерфейс добре сприймався користувачами. Вважається, що суб'єктивне задоволення користувачів майже завжди підвищується в наступних випадках [7].

Якщо інтерфейс програми естетичний і елегантний, тобто побудований на небагатьох і близьких до гармонійного співвідношення між розмірами окремих елементів і відстанями між ними, на м'яких, непомітних кольорах і не ріжучих очей їх поєднаннях, невеликій кількості контрастів, злегка згладжених кутах, на акуратному вирівнюванні окремих елементів.

У роботі системи не повинно виникати довгих пауз, під час яких користувачі не знають, чим зайнятися. Навіть якщо системі потрібно багато часу для виконання якихось дій, які відображаються в цей час картинки і надається додаткова інформація можуть знизити суб'єктивну тривалість очікування.

Правило доступності. Система повинна бути настільки зрозумілою, щоб користувач, ніколи раніше не бачив її, але добре розбирається в предметній області, міг без будь-якого навчання почати її використовувати. Це правило служить деяким ідеалом, до якого треба прагнути, оскільки на практиці досягти такої міри зрозумілості майже ніколи не вдається.

Правило ефективності. Система не повинна перешкоджати ефективній роботі досвідчених користувачів, які працюють з нею довгий час.

Правило підтримки. Система повинна сприяти більш простому і швидкому вирішенню завдань користувача. Це означає, перш за все, що система повинна дійсно вирішувати завдання користувача.

Правило дотримання контексту. Система повинна бути узгоджена з контекстом, в якому їй належить працювати. Це правило вимагає від системи бути працездатною не «взагалі», а саме в тому оточенні, в якому нею будуть користуватися. У контекст можуть входити специфіка і обсяги вхідних і вихідних даних, тип і цілі організацій, в яких система повинна працювати, рівень користувачів тощо.

Представлені вище правила визначають загальні вимоги, яким повинен задовольняти зручний інтерфейс. Крім них при розробці програмного забезпечення необхідно мати деякі підказки, що дозволяють зробити його більш зручним.

Наступні принципи дозволяють знаходити рішення, що підвищують зручність для користувача інтерфейсу [7].

Принцип структуризації. Інтерфейс повинен бути доцільно структурований. Близькі за змістом, родинні його частини повинні бути пов'язані видимим чином, а незалежні – розділені; схожі елементи повинні виглядати схоже, а несхожі – відрізнятися.

Принцип простоти. Найбільш поширені операції повинні виконуватися максимально просто. При цьому повинні бути видимі посилання на більш складні процедури. Принцип видимості. Всі функції і дані, необхідні для вирішення певної задачі, повинні бути видні, коли користувач намагається її вирішити.

Принцип зворотного зв'язку. Користувач повинен отримувати повідомлення про дії системи і про важливі події всередині неї. Повідомлення повинні бути інформативними, короткими, однозначними і написаними на мові, зрозумілій користувачеві.

Принцип толерантності. Інтерфейс повинен бути гнучким і терпимим до помилок користувача. Збиток від помилок повинен знижуватися за рахунок можливості скасування і затримки дій і за рахунок розумної інтерпретації будь-яких розумних дій користувача і введених їм даних.

Принцип повторного використання. Слід намагатися використовувати багаторазово внутрішні і зовнішні компоненти, забезпечуючи тим самим уніфікованість інтерфейсу і схожість між його елементами.

Схема інтерфейсу, призначеного для користувачів комп’ютерної системи обліку первинних бухгалтерських документів наведена на рис. 3.5

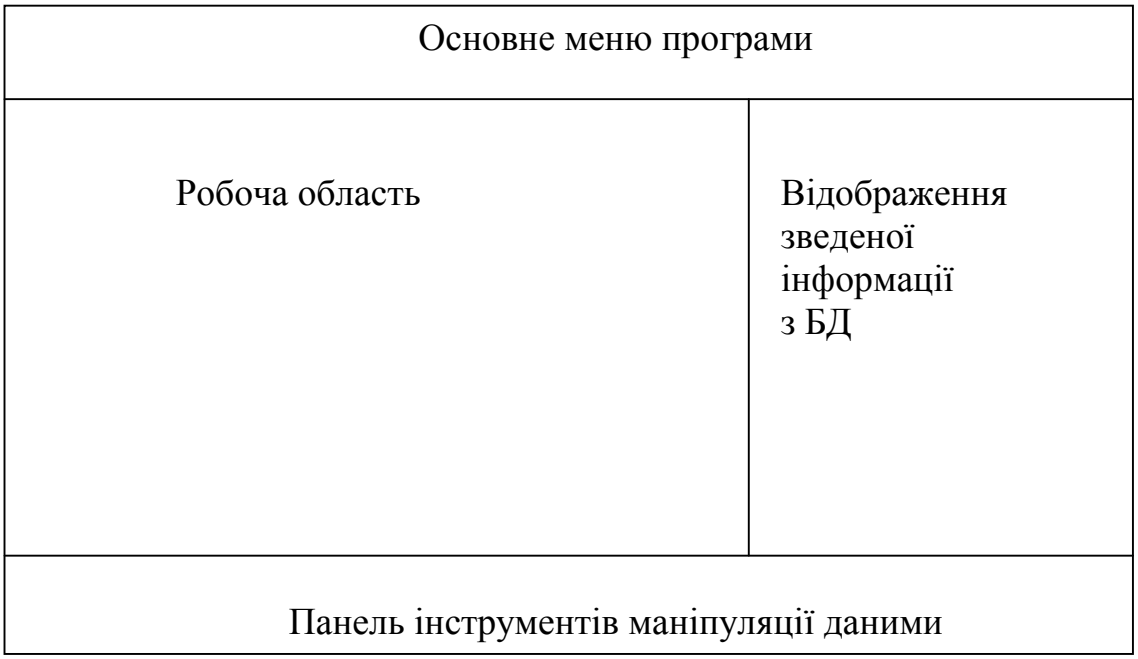


Рисунок 3.5 – Схема інтерфейсу користувачів компютерної системи

Важливим атрибутом будь-якого програмного продукту є зручний і інтуїтивно зрозумілий інтерфейс, приємний, але ненав'язливий зовнішній вигляд програми.

Також важливими елементами є логотип (графічний знак, який ідентифікує компанію) і назва програмного продукту. Ці елементи не відіграють важливої ролі у функціонуванні програми, але є важливою частиною програми, саме яку запам'ятовує клієнт і змушує його повернутися до цього програмного забезпечення.

4 ОПИС РОБОТИ СИСТЕМИ ОБЛІКУ БУХГАЛТЕРСЬКИХ ДОКУМЕНТІВ

4.1 Опис завантаження системи

Програма реалізована у вигляді пов'язаних модулів (форм), кожен відповідає за свій конкретний набір дій і описує кожну форму в програмних модулях (Unit).

Перший модуль (UkrDoc) відповідає за завантаження програми, тобто перевірку основних параметрів і перевірку підключення до бази даних. Програмний код перевіряючий підключення до бази даних:

```
try
        DM.ConnectUkrDocMdb.Connected: = true;
        DM.tblPlat.Active:=true;
        DM.tblCity.Active:=true;
        DM.tblPayment.Active:=true;
        DM.tblTypePlat.Active:=true;
        DM.tblRecipient.Active:=true;
    except
      on e: Exception do
       begin
          MessageDlg ('Немає доступу до бази даних, перевір-
те ее наявність !!!',
                      mtInformation, [mbOk], 0);
          DM.ConnectUkrDocMdb.Connected: = false;
          Close ();
          end;

          end;
          LoadApplicationPart2;
```

При запуску програма активує підключення до бази даних і якщо з'єднання не встановлено, тобто виникає помилка підключення, проводиться висновок повідомлення про відсутність бази даних або помилковому підключенні.

Якщо підключитися не вдалося, програма завершує свою роботу. Підключення здійснюється через рядок підключення:

```
Provider = Microsoft.Jet.OLEDB.4.0;
Data Source = ukr_doc.mdb; Persist Security Info = False
```

На рис. 4.1 представлений зовнішній вигляд завантажувача програми, який відображає у процентах виконання загрузки.

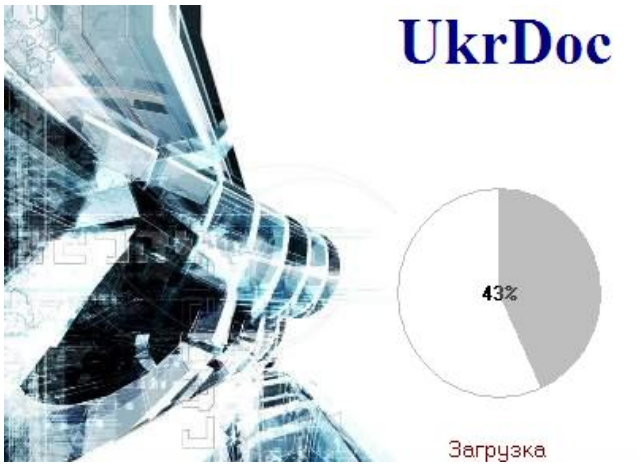


Рисунок 4.1 – Завантажувач програми

Якщо завантаження програми успішно завершено, запускається Майстер налаштування програми.

4.2 Робота з Майстером налаштування системи

Модуль Майстера налаштування програми пропонує користувачу системи виконати декілька кроків, які здійснюються засобами діалогових вікон для здійснення необхідних бухгалтерських налаштувань. На рис. 4.2 представлений зовнішній вигляд першого кроку Майстра настройки.

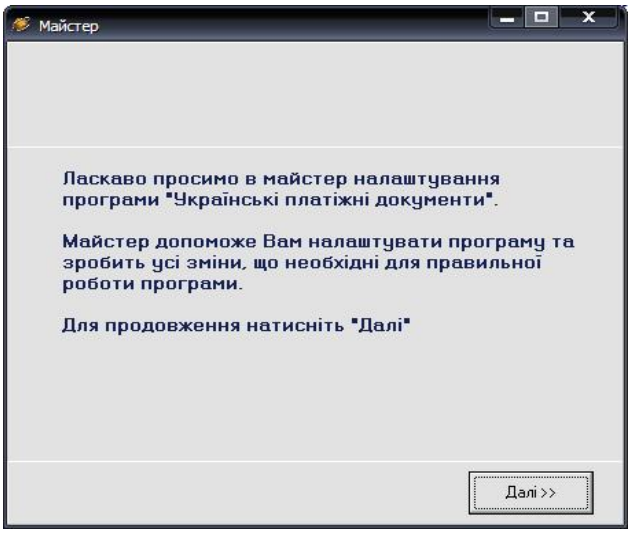


Рисунок 4.2 – Майстер налаштування Крок 1

Після першого кроку, Майстера налаштування відбувається перехід до наступного кроку, де користувачеві пропонується ввести банківські реквізити підприємства, яке в подальшому буде виступати в ролі Платника, а введені дані будуть зберігатися в Довіднику Платників (рис. 4.3).

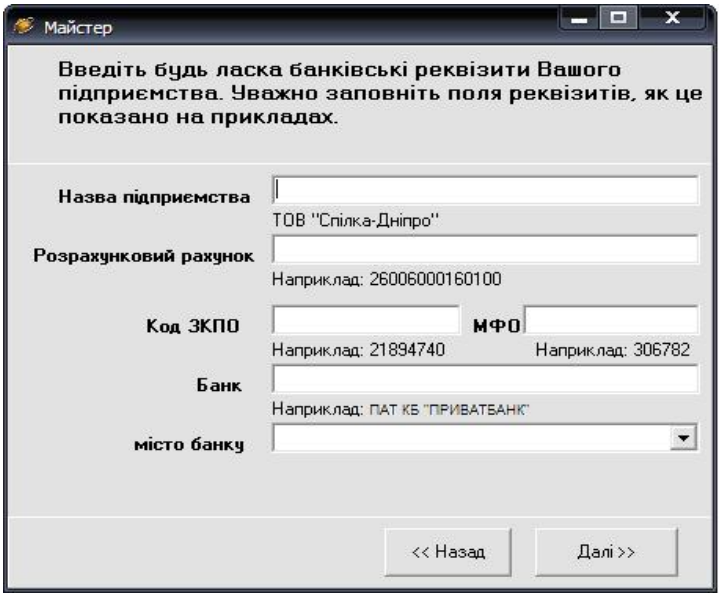


Рисунок 4.3 – Майстер налаштування Крок 2

Після введення всіх реквізитів користувач переходить до останнього третього кроку Майстра налаштування, де вводяться заключні налаштування (рис. 4.4), після чого при натисканні на кнопку «Далі» виводиться заключне повідомлення, що інформує користувача про необхідність перевірити всі дані, що вводяться і підтвердити їх правильність.

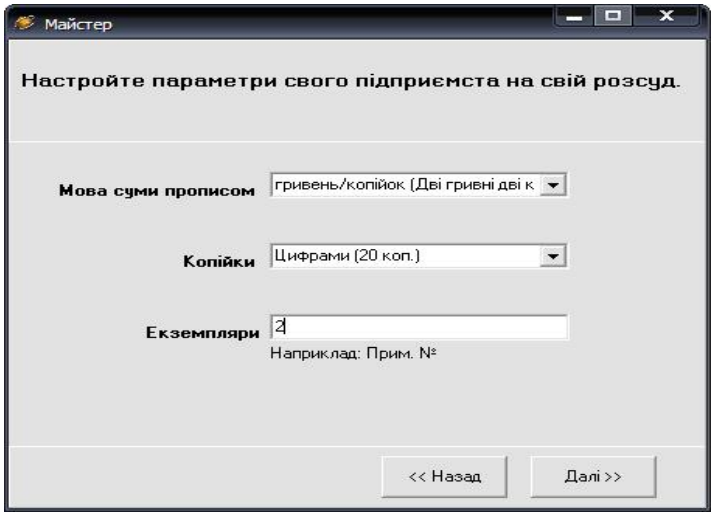


Рисунок 4.4 – Майстер налаштування Крок 3

Якщо якісь дані наведені не вірно до них можна повернутися за допомогою кнопки «Назад» і змінити. Якщо користувач підтверджує всі введені дані, вони зберігаються в базі даних, і відкривається головне вікно програми.

4.3 Робота у системі обліку первинних бухгалтерських документів

В головному вікні програми комп'ютерної системи обліку первинних бухгалтерських документів представлені такі елементи, як: Головне меню програми, представлене розділами – Документи, Довідники, Настройки, Допомога, Вихід (рис. 4.5).

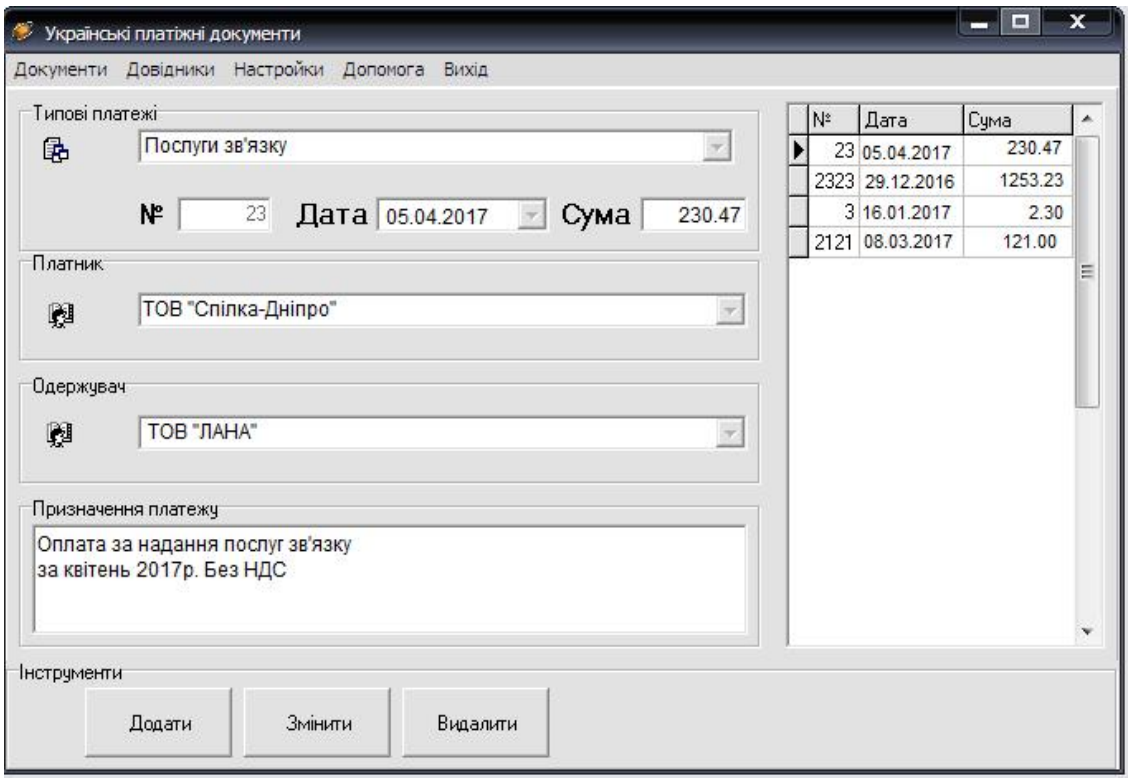


Рисунок 4.5 – Головне вікно програми

В основній робочій області розташовані елементи для створення платіжного доручення, а в правій частині – розташована таблиця зі зведеною інформацією з бази даних. Інформація відображає раніше створені платіжні доручення.

В нижній частині Головного вікна програми розташовані керуючі кнопки для виконання дій з платіжними дорученнями – Додати, Змінити, Видалити. Для додавання нового платіжного доручення потрібно натиснути на кнопку «Додати», при цьому неактивні елементи стають активними для вве-

дення інформації, а кнопки «Додати» і «Змінити» змінюються на кнопки «Відмінити» і «Зберегти».

У таблиці зі зведеною інформацією при натисканні правої кнопки миші викликається контекстне меню представлене на рис. 4.6.

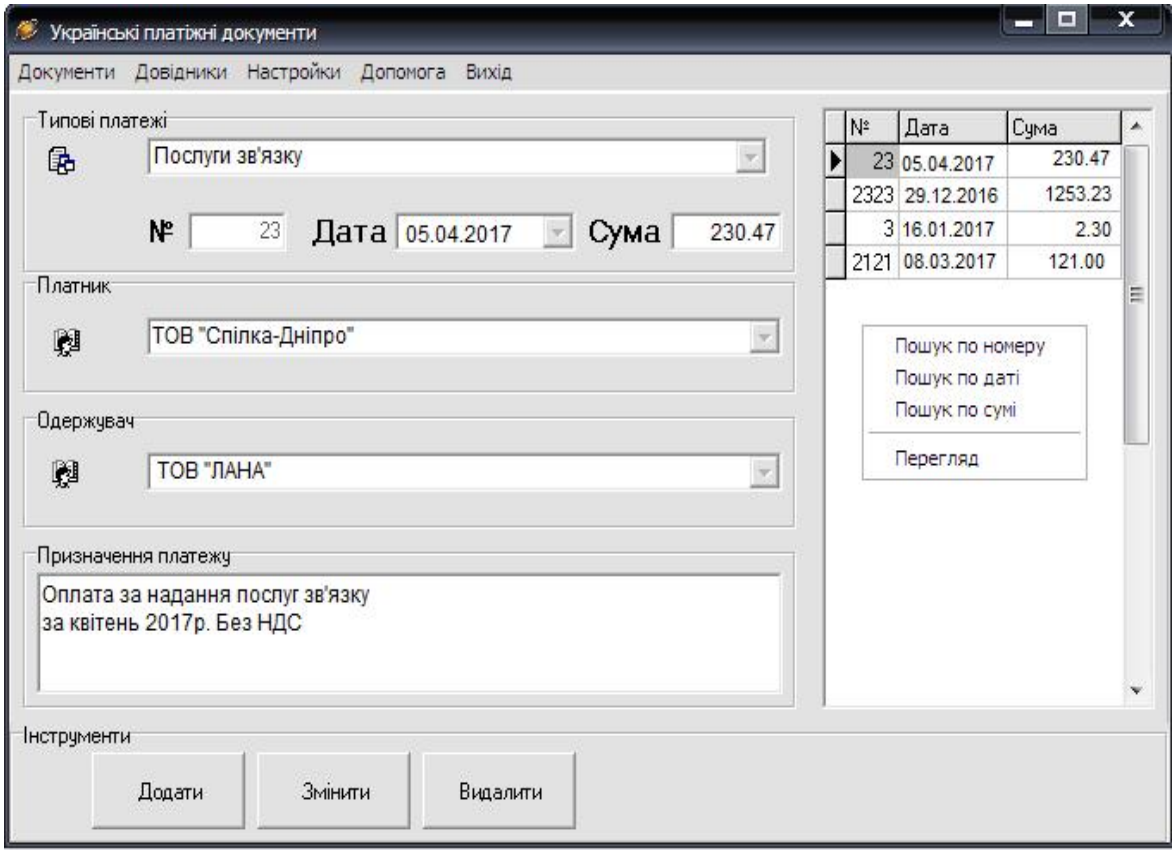


Рисунок 4.6 – Виклик контекстного меню

За допомогою пунктів цього контекстного меню надається можливість пошуку раніше створеного платіжного доручення за такими критеріями як:

– пошук за номером платіжного доручення або квитанції (рис. 4.7);
– пошук за датою формування документа (рис. 4.8);
– пошук по сумі, що зазначена в платіжному дорученні або квитанції (рис. 4.9).

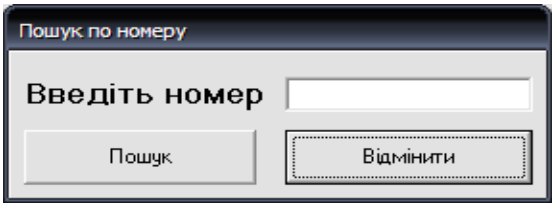


Рисунок 4.7 – Діалогове вікно пошуку документа за номером

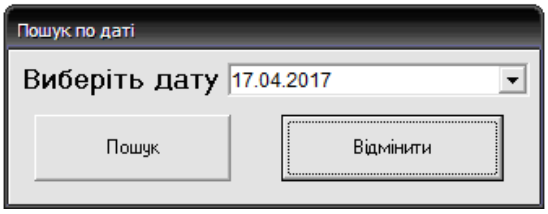


Рисунок 4.8 – Діалогове вікно пошуку документа за датою

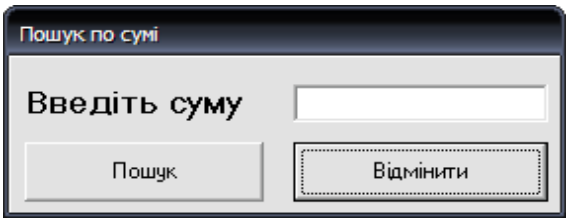


Рисунок 4.9 – Діалогове вікно пошуку документа по сумі

За допомогою пункту меню «Перегляд» можна переглянути створений документ. Документ буде відображений з урахуванням розстановок полів, заданих в шаблоні документа (Конструкторі форм). У вікні перегляду можна побачити сформоване платіжне доручення або квитанцію в версії для друку.

Розроблена комп'ютерна система обліку первинних бухгалтерських документів надає також можливість створити такий платіжний документ, як «Квитанція-доручення», форма для створення якої представлена на рис. 4.10.

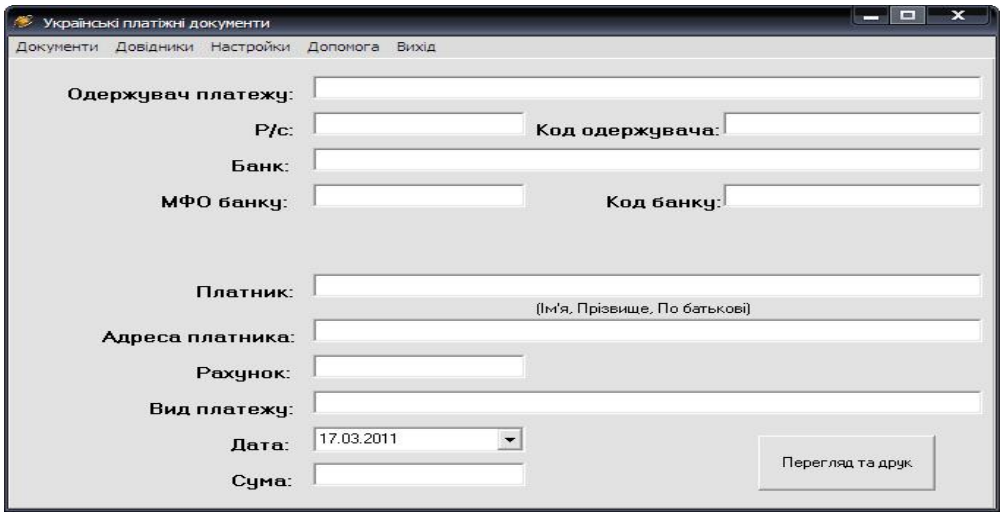


Рисунок 4.10 – Форма для здійснення формування Квитанції-доручення

У комп'ютерній системі обліку первинних бухгалтерських документів, згідно з визначеними вимогами реалізовані наступні прграмні модулі:

– Довідник Платників;
– Довідник Одержувачів;
– Довідник Типових платежів.

Довідник платників дозволяє додати в базу даних реквізити нових платників (підприємств, які здійснюєть оплату). Заповнивши один раз дані цього підприємства можливо і надалі використовувати ці дані при складанні платіжних документів, обираючи це підприємство у базі даних, що відображено у правій частині Головного вікна програми (рис. 4.11).

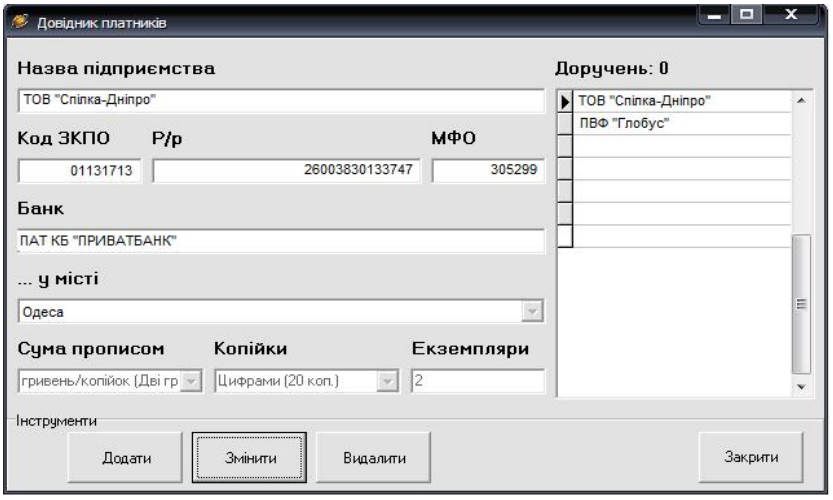


Рисунок 4.11 – Довідник платників

Довідник одержувачів дозволяє внести всі необхідні данні про підприємство одержувача платежу з подальшим збереженням даних у базі даних (рис. 4.12).

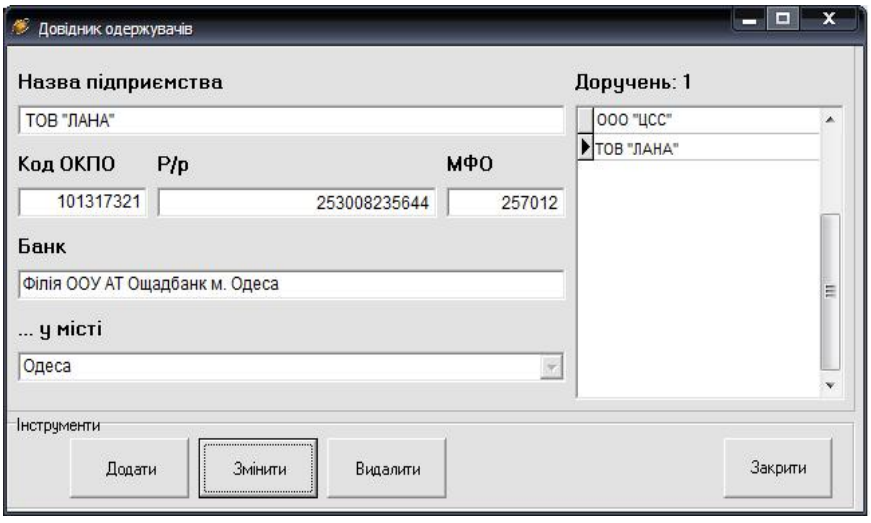


Рисунок 4.12 – Довідник одержувачів

За допомогою Довідника Типових платежів можливо створити нове платіжне доручення і використовувати це типовий платіж неодноразово, змінюючи лише деякі поля, наприклад номер платіжного доручення та дату. Використання Довідника типових платежів доцільно при формуванні платіжних документів, які часто використовуються при здійсненні розрахунків, наприклад розрахунків з бюджетом, оплата податкових зобов'язань, оплата послуг зв'яку, і т.п.

У цьому ж діалоговому вікні можна відрити Довідник платників і Довідник одержувачів, щоб оперативно додати нову інформацію і негайно використовувати її при складанні нового типу платежу (рис.4.13).

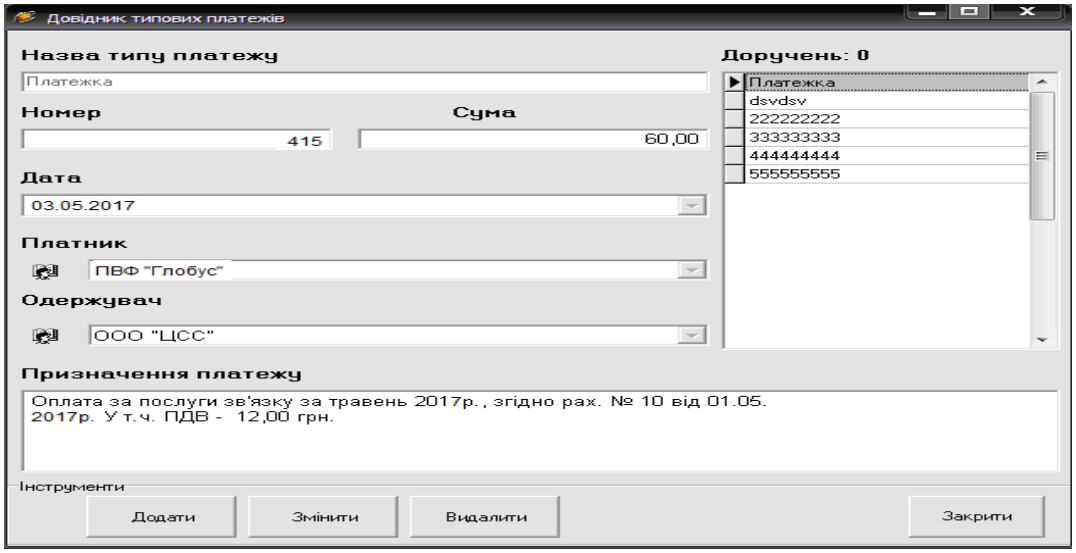


Рисунок 4.13 – Довідник Типових платежів

У всіх Довідниках при виборі вже доданої записи над таблицею зведеної інформації відображається кількість створених платіжних документів для цього запису.

При видаленні будь-якого запису з бази даних програма вимагає підтвердження на видалення, що запобігає випадковому видаленю записів з бази даних. Вікно підтвердження представлено на рис. 4.14.

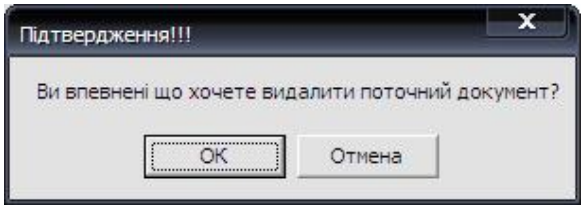


Рисунок 4.14 – Діалогове вікно підтвердження на видалення

Кожне нове вікно програми відкривається в модальному режимі, тобто користувач не може перейти до попереднього вікна, не закривши нове. Така реалізація інтерфейсу системи передбачає функцію захисту системи від втрати даних, а також їх неузгодженості.

Одним з функціональних можливостей розробленої системи є реалізований в ній Конструктор шаблонів.

Конструктор шаблонів реалізує функції необхідні для оперативної зміни шаблону документа при внесенні нових поправок в законодавство, або в наслідок, затвердження нових форм фінансових документів.

Даний модуль дозволяє внести зміни в шаблони документів не здійснюючи зміни в самому програмному коді системи.

Таким чином, розроблена комп'ютерна система обліку первинних бухгалтерських документів є закінченою функціональною одиницею, яка оперативно настроюється при необхідності внесення поправок.

Звернутися до Конструктора шаблонів можна за допомогою Головного меню системи. Треба обрати розділ Головного меню Настройки, обрати Конструктор шаблонів і далі обрати один із шаблонів: Платіжне доручення або Квітанція-повідомлення (рис. 4.15).

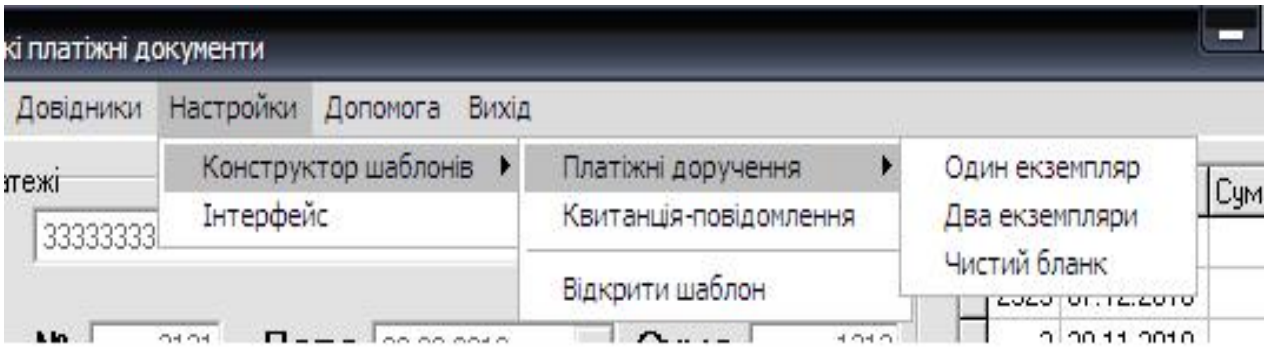


Рисунок 4.15 – Конструктор шаблонів

За допомогою Конструктора шаблонів можливе формування (створення), як чистого бланка для подальшого створення нового шаблону, так і можливе редагування вже існуючого шаблону з урахуванням вимог, що пред'являються.

На рис 4.16 відображен шаблон платіжного документа в Конструкторі шаблонів «Квитанція-повідомлення». Відображені всі поля шаблону, та їх розташування в документі. А на рис. 4.17 відображен шаблон Платіжного до-

кумента «Платіжне доручення» з зазначенними обов’язковими полями та їх розташуванням.

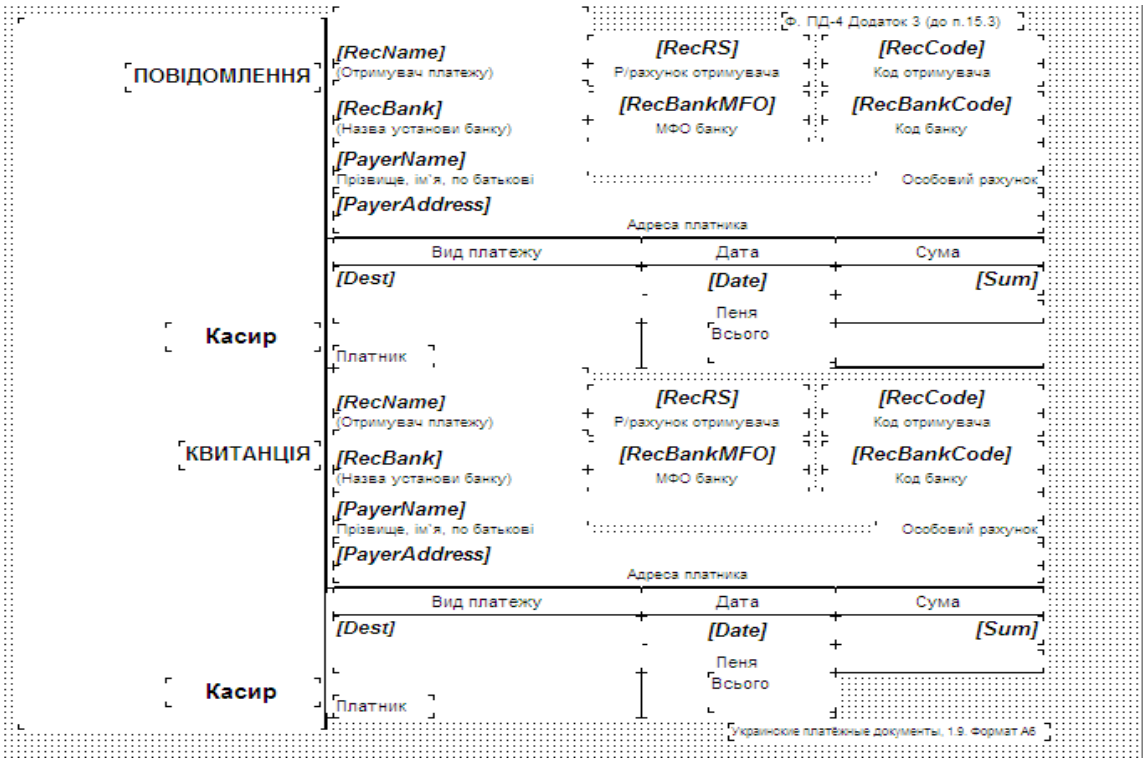


Рисунок 4.16 – Шаблон документа «Квитанція-повідомлення»

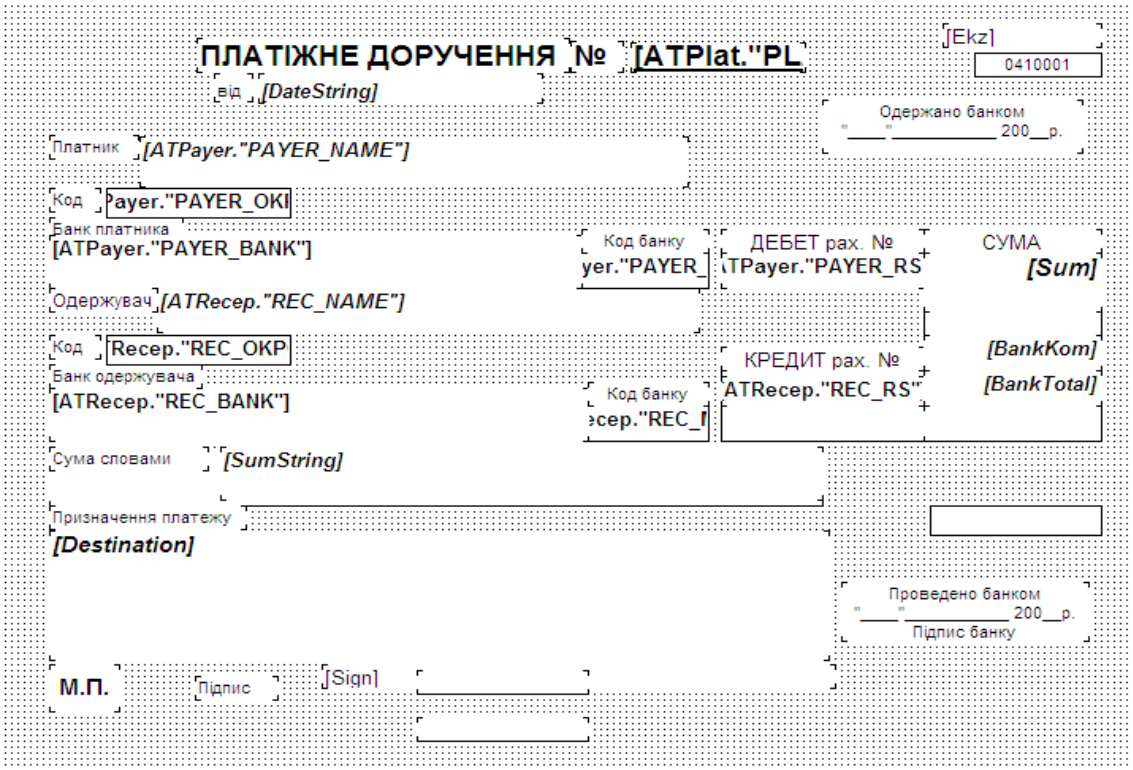


Рисунок 4.17 –Шаблон документа «Платіжне доручення»

4.4 Основні керуючі програмні модулі

Розроблена комп'ютерна система обліку первинних бухгалтерських документів дозволяє здійснювати різні вибірки платіжних документів, за вибором різних критеріїв.

Підрахунок кількості платіжних документів для конкретного платника або одержувача при виборі його запису в зведеній таблиці даних, здійснюється за допомогою наступного програмного коду:

```
id: = frmPlat.dbEdtCount.Text;
       if (id <> '') then
        begin
          frmPlat.lblCount.Caption: = ''; // чистимо
          DM.qPaymentCount.SQL.Text: = 'SELECT count (*) as ex
FROM payment WHERE id_plat =' + id; // текст запиту
          DM.qPaymentCount.Open;
          if DM.qPaymentCount.IsEmpty then Exit; // якщо набір
порожній - виходимо з процедури
          frmPlat.lblCount.Caption: =
DM.qPaymentCount.FieldByName ( 'ex'). AsString; // додаємо чер-
гове значення поля
          end;
```

Помилки під час введення в текстове поле тільки числових значень здійснюється таким кодом:

```
if not (key in [ '0' .. '9', # 8]) then key: = # 0;
```

Підтвердження на закриття програми:

```
if Application.MessageBox (PChar ( 'Ви впевнені что бажаєте за-
крити програму?'),
      'Підтвердження !!!', MB_OKCANCEL) = id_OK then
    frmStart.Close ();
```

Пошук потрібної інформації за датою, номером і сумі реалізований за допомогою функції:

```
DM.tblPayment.Locate ( 'number', frmFindNum-
ber.edtFindNumber.Text, []
```

При створенні нового платіжного доручення реалізована можливість вибору раніше створеного типу платежу, при цьому його дані повинні автоматично вноситися в усі необхідні поля шаблону документа, а саме:

- номер платежу;
- дата створення;
- сума платежу;
- платник;
- одержувач;
- призначення платежу.

Така автоматична розстановка даних в поля шаблону реалізована за допомогою наступного програмного модуля:

```
if (IntToStr (frmMain.dbLookCBTypePlat.KeyValue) <> '') then
      begin
        DM.qTypePlat.SQL.Text: = 'SELECT * FROM typePlat
   WHERE id =' + IntToStr (frmMain.dbLookCBTypePlat.KeyValue);
   // текст запиту
        DM.qTypePlat.Open;
        if DM.qTypePlat.IsEmpty then Exit; // якщо набір по-
   рожній - виходимо з процедури
        dbEdDate.Text: = DM.qTypePlat.FieldByName ( 'date').
   AsVariant;
        DateTime-
   Picker1.Date:=DM.qTypePlat.FieldByName('date').AsVariant;
        dbEdNumber.Text: = DM.qTypePlat.FieldByName ( 'num-
   ber'). AsVariant;
        dbEdSumma.Text: = DM.qTypePlat.FieldByName (
   'summa'). AsVariant;
        dbCBPlat.KeyValue: = DM.qTypePlat.FieldByName (
   'id_plat'). AsVariant;
        dbCBRecipient.KeyValue: = DM.qTypePlat.FieldByName (
   'id_recipient'). AsVariant;
        dbComment.Text: = DM.qTypePlat.FieldByName (
   'coment'). AsVariant;
        end;
```

Деактивация елементів форми при не натисненні кнопки «Додати» здійснено за допомогою функції:

```
        DM.tblPayment.Cancel;
        DateTimePicker1.Enabled: = false;
```

```
dbLookCBTypePlat.Enabled: = false;
dbEdDate.Enabled: = false;
dbEdNumber.Enabled: = false;
dbEdSumma.Enabled: = false;
dbCBPlat.Enabled: = false;
dbCBRecipient.Enabled: = false;
dbComment.Enabled: = false;
btnAdd.Caption: = 'Додати';
btnEdit.Caption: = 'Изменить';
btnDel.Caption: = 'ВИДАЛИТИ';
```

Деактивация всіх керуючих елементів зв'язку з базою даних при виході з програми:

```
DM.ConnectUkrDocMdb.Connected: = false;
DM.tblPlat.Active:=false;
DM.tblCity.Active:=false;
DM.tblPayment.Active:=false;
DM.tblTypePlat.Active:=false;
DM.tblRecipient.Active:=false;
```

4.5 Результати тестування системи

Результати розробки комп’ютерної системи обліку первинних бухгалтерських документів перевіряються на етапі тестування.

Тестування – це перевірка відповідності програмного забезпечення вимогам, яка здійснюється за допомогою спостереження за його роботою в спеціальних, штучно побудованих ситуаціях [5].

Основних джерел помилок три, перерахуємо їх. Неправильне розуміння задач. Дуже часто люди не розуміють, що їм намагаються сказати інші. Так само і розробники ПЗ, не завжди розуміють, що саме потрібно зробити. Іншим джерелом нерозуміння є відсутність розуміння у самих користувачів і замовників – досить часто вони можуть просити зробити трохи не те, що їм дійсно потрібно. Для запобігання неправильного розуміння завдань програмної системи служить аналіз предметної області. Неправильне рішення задач. Найчастіше, навіть правильно зрозумівши, що саме потрібно зробити, розробники вибирають неправильний підхід до того, як це робити. Можливі рішення можуть забезпечувати лише деякі з необхідних властивостей, вони можуть добре підходити для даної задачі в теорії, але погано працювати на практиці, в конкретних обставинах, в яких повинно буде працювати ПЗ. До-

помогти у виборі правильного рішення може зіставлення альтернативних рішень і ретельний аналіз їх на предмет відповідності всім вимогам, підтримання постійного зв'язку з користувачами та замовниками, надання їм необхідної інформації про обрані рішеннях, демонстрація прототипів, аналіз придатності обираних рішень для роботи в тому контексті, в якому вони будуть використовуватися. Невірні перенесення рішень у код. Маючи правильне рішення вірно зрозумілого завдання, люди здатні зробити досить багато помилок при втіленні цих рішень [5].

Для запуску розробленої комп’ютерної системи обліку первинних бухгалтерських документів натиснемо два рази лівою клавішею мишки по іконці (піктограмі) програми (рис. 4.18):

Рисунок 4.18 – Вигляд іконки (піктограми) створеної системи

Починається завантаження програми, після чого запускається Майстер налаштування програми. У наступному діалоговому вікні вводимо всі необхідні дані (рис. 4.19):

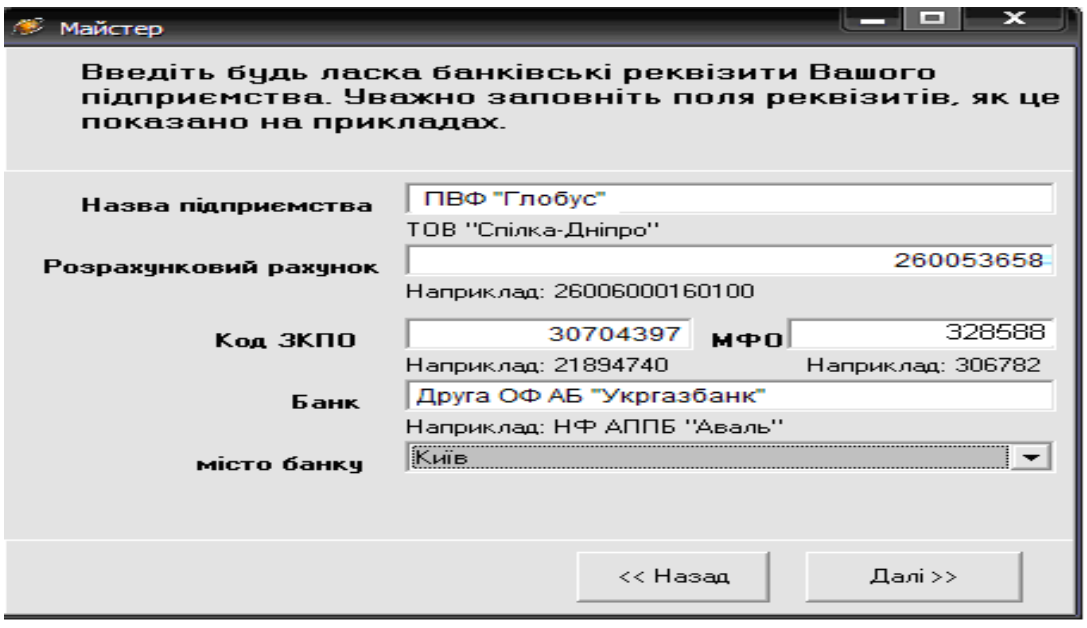


Рисунок 4.19 – Тестове введення даних

Всі дані наведені, при натисканні на кнопці «Далі» перевіряється коректність даних, що вводяться, після чого дані записуються в базу даних і відкривається Головне вікно програми в якому можливо створити платіжний документ під назвою «Платіжне доручення», але перед цим введемо реквізити Платника (рис . 4.20) і Отримувача (рис. 4.21).

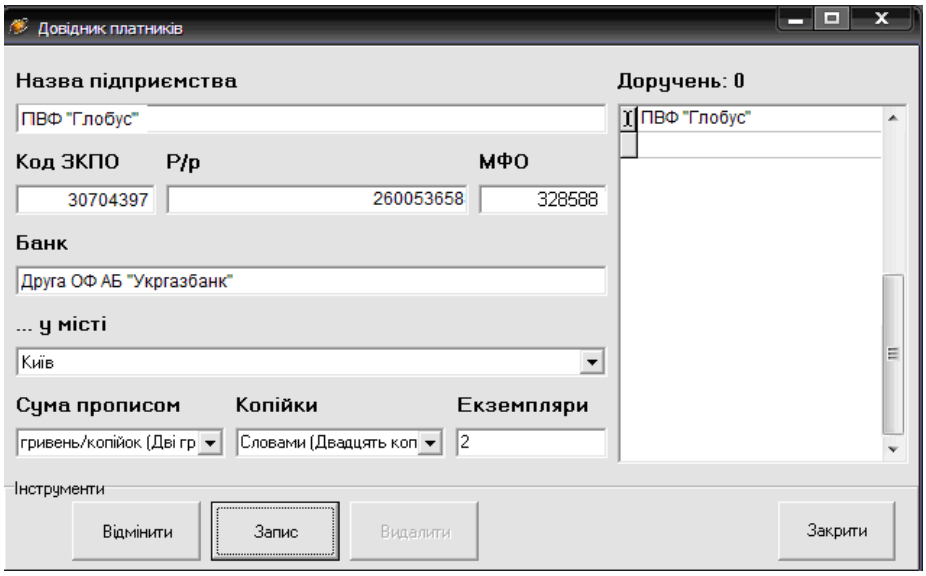


Рисунок 4.20 – Внесення даних Платника

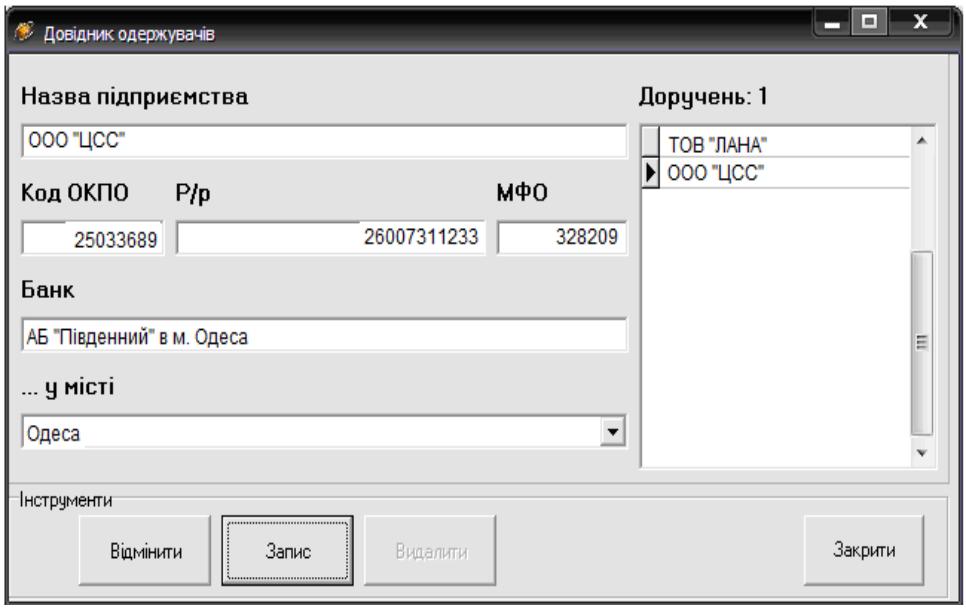
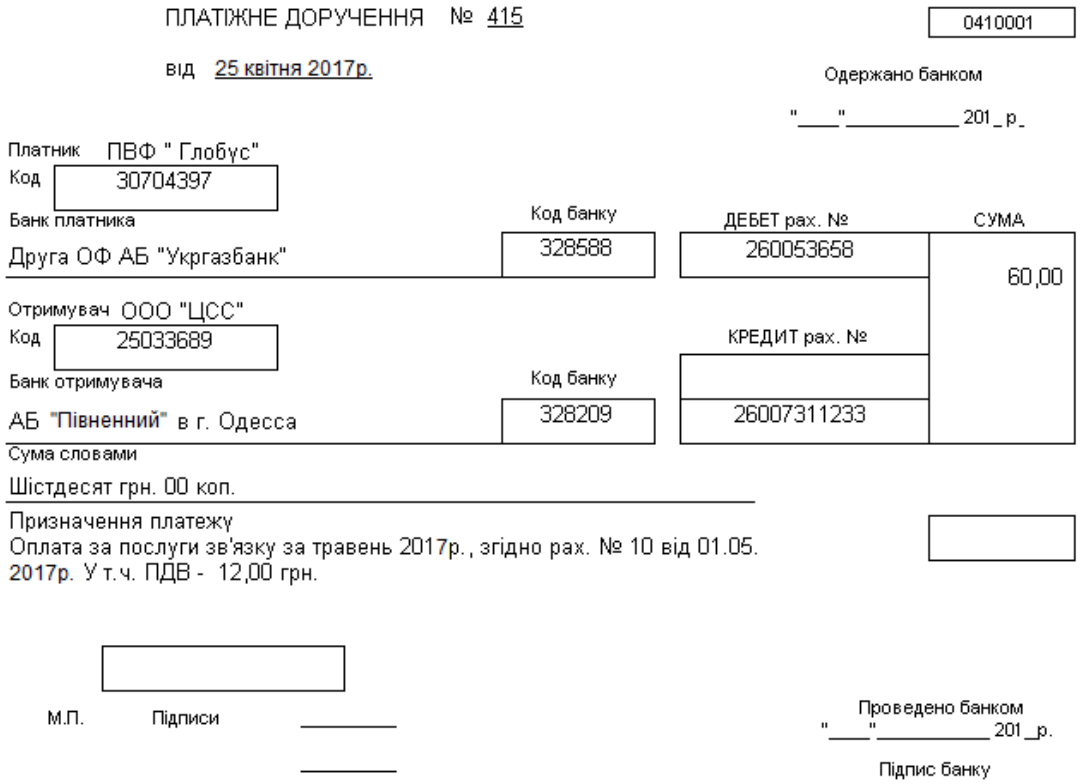


Рисунок 4.21 – Внесення даних Одержувача

Після введення всіх необхідних реквізитів Платника і Одержувача, а так само заповнення полів сума, дата, призначення платежу, формується пла-

тіжний документ – «Платіжне доручення» відповідно до заданого шаблону, сформованому за допомогою Конструктора шаблонів. Сформоване, засобами комп'ютерної системи обліку первинних бухгалтерських документів, Платіжне-доручення має вигляд в версії для друку (рис. 4.22).


ПЛАТІЖНЕ ДОРУЧЕННЯ № 415

0410001

від 25 квітня 2017р.

Одержано банком

"____"_____ 201_р.

Платник ПВФ " Глобус"

Код 30704397

Банк платника

Друга ОФ АБ "Укргазбанк"

Код банку 328588

ДЕБЕТ рах. № 260053658

СУМА 60,00

Отримувач ООО "ЦСС"

Код 25033689

Банк отримувача

АБ "Північний" в г. Одесса

Код банку 328209

КРЕДИТ рах. № 26007311233

Сума словами

Шістдесят грн. 00 коп.

Призначення платежу

Оплата за послуги зв'язку за травень 2017р., згідно рах. № 10 від 01.05. 2017р. У т.ч. ПДВ - 12,00 грн.

М.П. Підписи _____

_____

Проведено банком

"____"_____ 201_р.

Підпис банку

Рисунок 4.22 – Підготовлене платіжне доручення для друку

Розроблена комп'ютерна система обліку первинних бухгалтерських документів пройшла тестування, всі основні модулі програми працюють коректно і виконують всі необхідні функції пов'язані з підготовкою та обліком первинних бухгалтерських платіжних документів.

ВИСНОВКИ

У дипломній роботі проведено дослідження основних особливостей складання, обліку, формування первинних бухгалтерських і фінансових документів. Визначено вимоги до змісту і складанню первинних бухгалтерських документів, а так само до програмного забезпечення, що реалізує функції обліку первинних бухгалтерських платіжних документів для надання в банківські установи України та здійснення платежів.

Розроблена комп'ютерна система обліку первинних бухгалтерських документів призначена для підготовки і друку платіжних доручень в банк, що відповідають вимогам національного банку України, і включає в себе наступні модулі:

- ведення бази платіжних доручень по декількох підприємствах;
- фільтрація журналу платіжних доручень по будь-яким параметрам;
- автоматична нумерація платіжного доручення при формуванні нового;
- автоматична підстановка банківських реквізитів при виборі платника або одержувача в ході формування нового доручення;
- пошук платника або одержувача за кодом ОКПО або найменуванням (рекурсивний пошук);
- формування суми прописом;
- конструктор форми платіжного доручення;
- можливість друку порожнього бланка;
- формування та друк реєстру платіжних доручень.

У процесі виконання дипломної роботи вивчена предметна область, сформульовані вимоги до системи, проведено проектування бази даних, розроблена архітектура системи і інтерфейс користувачів системи. Створена програмна реалізація. Компютерна система обліку первинних платіжних документів є цілком закінченим функціонуючим додатком, що допускає здійснення змін чи доповнення функціонального розширення.

ПЕРЕЛІК ПОСИЛАТЬ

1. Ф.Ф. Бутинця Теорія бухгалтерського обліку. Підручник. – Житомир: ЖТІ, 2000. – 640 с.
2. В.П. Завгородній Автоматизація бухгалтерського обліку, контролю, аналізу та аудиту. – К.: «А.С.К.», 1998. – 450 с.
3. Н.М. Ткаченко. Бухгалтерський облік на підприємствах України з різними формами власності. – К.: «А.С.К.», 2003. – 780 с.
4. О.М.Ананьєв, В.М. Білик, Я. А. Гончарук. Інформаційні системи і технології в комерційній діяльності. Навчальний посібник . – Львів: Новий Світ-2000, 2006. – 584 с.
5. Брауде Эрик Дж. Технология разработки программного обеспечения. – М.: Computer Science, 2004. –655 с.
6. Орлов С.А. Технологии разработки программного обеспечения: уч. пособие. – 2-е изд. – СПб.: Питер, 2003. – 480 с.
7. В.М. Петров. Информационные системы: уч. пособие. – СПб.: Питер, 2002. – 588 с.
8. Карпова Т.С. Базы данных: модели, разработка, реализация. – СПб.: Питер, 2001. – 304 с.
9. Алексеев Е. Р., Чеснокова О. В., Кучер Т. В. Free Pascal и Lazarus: Учебник по программированию. – М.: Издательство «ДМК-пресс», 2010. – 442 с.
10. Грофф Дж. Р., Вайнберг П.Н., Оппелъ Э. Дж. SQL. Полное руководство. – М.: Издательство «Вильямс», 2015. – 959 с.
11. Скляр А. Я. Введение в InterBase. – М.: Издательство «Горячая Линия – Телеком», 2002. – 520 с.
12. Мансуров К.Т. Основы программирования в среде Lazarus. – М.: Издательство «Вильямс», 2010. – 772с.
13. Алексеев Е.Р., Чеснокова О.В., Кучер Т.В. Самоучитель по программированию на Free Pascal и Lazarus. – Донецк.: ДонНТУ, Технопарк ДонНТУ УНИТЕХ, 2009. – 503 с.
14. И.О. Одинцов. Профессиональное программирование. Системный подход. – 2-е изд., перераб. и доп. – СПб.: БХВ-Петербург, 2004. – 624 с.

Д О Д А Т К И

ДОДАТОК А ОСНОВНІ КОДИ ПРОГРАМНИХ МОДУЛІВ

```
program UkrDoc;
uses
  Forms,
  untWizard1 in 'untWizard1.pas' {frmWizard1},
  untWizard2 in 'untWizard2.pas' {frmWizard2},
  untDM in 'untDM.pas' {DM: TDataModule},
  untWizard in 'untWizard.pas' {frmWizard},
  untWizardEnd in 'untWizardEnd.pas' {frmWizardEnd},
  untMain in 'untMain.pas' {frmMain},
  untStart in 'untStart.pas' {frmStart},
  untFindNumber in 'untFindNumber.pas' {frmFindNumber},
  untFindDate in 'untFindDate.pas' {frmFindDate},
  untFindSumma in 'untFindSumma.pas' {frmFindSumma},
  untPlat in 'untPlat.pas' {frmPlat},
  untRecipient in 'untRecipient.pas' {frmRecipient},
  untTypePlat in 'untTypePlat.pas' {frmTypePlat};
{$ R * .res}
begin
  Application.Initialize;
  Application.CreateForm (TfrmStart, frmStart);
  Application.CreateForm (TDM, DM);
  Application.CreateForm (TfrmWizard, frmWizard);
  Application.CreateForm (TfrmWizard1, frmWizard1);
  Application.CreateForm (TfrmWizard2, frmWizard2);
  Application.CreateForm (TfrmWizardEnd, frmWizardEnd);
  Application.CreateForm (TfrmMain, frmMain);
  Application.CreateForm (TfrmFindNumber, frmFindNumber);
  Application.CreateForm (TfrmFindDate, frmFindDate);
  Application.CreateForm (TfrmFindSumma, frmFindSumma);
  Application.CreateForm (TfrmPlat, frmPlat);
  Application.CreateForm (TfrmRecipient, frmRecipient);
  Application.CreateForm (TfrmTypePlat, frmTypePlat);
  Application.Run;
end.


unit untDM;

interface
uses
  SysUtils, Classes, DB, ADODB;
type
  TDM = class (TDataModule)
    ConnectUkrDocMdb: TADOConnection;
    tblPlat: TADOTable;
    tblPlatname: TWideStringField;
    tblPlatdeposit: TIntegerField;
    tblPlatzkpo: TIntegerField;
    tblPlatmfo: TIntegerField;
    tblPlatbank: TWideStringField;
    tblPlatcity: TIntegerField;
```

```
    dsPlat: TDataSource;
    tblCity: TADOTable;
    dsCity: TDataSource;
    tblCityid: TAutoIncField;
    tblCityname: TWideStringField;
    tblPlatlang_sum: TWideStringField;
    tblPlatkop: TWideStringField;
    tblPlatekz: TWideStringField;
    tblPayment: TADOTable;
    dsPayment: TDataSource;
    tblPlatid: TAutoIncField;
    tblRecipient: TADOTable;
    dsRecipient: TDataSource;
    tblRecipientid: TAutoIncField;
    tblRecipientname: TWideStringField;
    tblRecipientOKPO: TIntegerField;
    tblRecipientdeposit: TIntegerField;
    tblRecipientMFO: TIntegerField;
    tblRecipientbank: TWideStringField;
    tblRecipientcity: TIntegerField;
    qTypePlat: TADOQuery;
    qPaymentCount: TADOQuery;
    dsTypePlat: TDataSource;
    tblTypePlat: TADOTable;
    tblTypePlatid: TAutoIncField;
    tblTypePlatname: TWideStringField;
    tblTypePlatnumber: TIntegerField;
    tblTypePlatsumma: TBCDField;
    tblTypePlatdate: TDateTimeField;
    tblTypePlatid_plat: TIntegerField;
    tblTypePlatid_recipient: TIntegerField;
    tblTypePlatcoment: TMemoField;
    tblPaymentid: TAutoIncField;
    tblPaymentnumber: TIntegerField;
    tblPaymentsumma: TBCDField;
    tblPaymentdate: TDateTimeField;
    tblPaymentid_plat: TIntegerField;
    tblPaymentid_recipient: TIntegerField;
    tblPaymentcomment: TMemoField;
    tblPaymentid_type_plat: TIntegerField;
    procedure dsPaymentDataChange (Sender: TObject; Field: TField);
    procedure dsPlatDataChange (Sender: TObject; Field: TField);
    procedure dsRecipientDataChange (Sender: TObject; Field: TField);
    procedure dsTypePlatDataChange (Sender: TObject; Field: TField);
  private
    {Private declarations}
  public
    {Public declarations}
  end;
var
  DM: TDM;
implementation
```

```
      uses untWizard1, untWizard2, untMain, untPlat, untRecipient, unt-
TypePlat;
      {$ R * .dfm}
      procedure TDM.dsPaymentDataChange (Sender: TObject; Field: TField);
      begin
          if frmMain.dbEdDate.Text = '' then
            begin
            end
              else
          frmMain.DateTimePicker1.Date:=StrToDate(frmMain.dbEdDate.Text);
      end;
      procedure TDM.dsPlatDataChange (Sender: TObject; Field: TField);
      var
        id: String;
      begin
            id: = frmPlat.dbEdtCount.Text;
             if (id <> '') then
              begin
                frmPlat.lblCount.Caption: = ''; // чистимо
                DM.qPaymentCount.SQL.Text: = 'SELECT count (*) as ex FROM pay-
ment WHERE id_plat =' + id; // текст запиту
                DM.qPaymentCount.Open;
                if DM.qPaymentCount.IsEmpty then Exit; // якщо набір порожній
- виходимо з процедури
                frmPlat.lblCount.Caption: = DM.qPaymentCount.FieldByName (
'ex'). AsString; // додаємо чергове значення поля
                end;
      end;

      procedure TDM.dsRecipientDataChange (Sender: TObject; Field: TField);
      var
        id: String;
      begin
            id: = frmRecipient.dbEdtCount.Text;
             if (id <> '') then
              begin
                frmRecipient.lblCount.Caption: = ''; // чистимо
                DM.qPaymentCount.SQL.Text: = 'SELECT count (*) as ex FROM pay-
ment WHERE id_recipient =' + id; // текст запиту
                DM.qPaymentCount.Open;
                if DM.qPaymentCount.IsEmpty then Exit; // якщо набір порожній
- виходимо з процедури
                frmRecipient.lblCount.Caption: = DM.qPaymentCount.FieldByName
( 'ex'). AsString; // додаємо чергове значення поля
                end;
      end;
      procedure TDM.dsTypePlatDataChange (Sender: TObject; Field: TField);
      var
        id: String;
      begin
        if frmTypePlat.dbEdDate.Text = '' then
            begin
            end
```

```
             else

frmTypePlat.DateTimePicker1.Date:=StrToDate(frmTypePlat.dbEdDate.Text);
            id: = frmTypePlat.dbEdtCount.Text;
             if (id <> '') then
              begin
                frmTypePlat.lblCount.Caption: = ''; // чистимо
                DM.qPaymentCount.SQL.Text: = 'SELECT count (*) as ex FROM pay-
ment WHERE id_type_plat =' + id; // текст запиту
                DM.qPaymentCount.Open;
                if DM.qPaymentCount.IsEmpty then Exit; // якщо набір порожній
- виходимо з процедури
                frmTypePlat.lblCount.Caption: = DM.qPaymentCount.FieldByName (
'ex'). AsString; // додаємо чергове значення поля
                end;
      end;
      end.

      unit untStart;
      interface
      uses
        Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
        Dialogs, ExtCtrls, Gauges, StdCtrls, jpeg;
      type
        TfrmStart = class (TForm)
          Image1: TImage;
          Label1: TLabel;
          Label3: TLabel;
          Gauge1: TGauge;
          Timer1: TTimer;
          procedure Timer1Timer (Sender: TObject);
          procedure FormCreate (Sender: TObject);
          procedure FormClose (Sender: TObject; var Action: TCloseAction);
        private
          {Private declarations}
        public
          {Public declarations}
          procedure LoadApplicationPart1;
          procedure LoadApplicationPart2;
         end;
      var
        frmStart: TfrmStart;
        gp: Integer;
      implementation
      uses untMain, untWizard, untDM, untPlat, untRecipient, untTypePlat;
      {$ R * .dfm}
      procedure TfrmStart.LoadApplicationPart1;
      begin
        try
             DM.ConnectUkrDocMdb.Connected: = true;
             DM.tblPlat.Active:=true;
             DM.tblCity.Active:=true;
```

```
            DM.tblPayment.Active:=true;
            DM.tblTypePlat.Active:=true;
            DM.tblRecipient.Active:=true;
          except
            on e: Exception do
            begin
               MessageDlg ('Немає доступу до бази даних, перевірте ее наяв-
ність !!!',
                           mtInformation, [mbOk], 0);
               DM.ConnectUkrDocMdb.Connected: = false;
               Close ();
               end;
               end;
               LoadApplicationPart2;
      end;
      procedure TfrmStart.LoadApplicationPart2;
      begin
        if FileExists ( 'ukrdoc.ini') then
          frmMain.Visible: = true
        else
            frmWizard.Visible: = true;
      end;
      procedure TfrmStart.Timer1Timer (Sender: TObject);
      begin

        if gp = 100 then begin
          Timer1.Enabled: = false;
          Gauge1.Visible: = false;
          Sleep (500);
          Visible: = false;
          LoadApplicationPart1;
        end else begin
          gp: = gp + 1;
          Gauge1.Progress: = gp
        end;
      end;
      procedure TfrmStart.FormCreate (Sender: TObject);
      begin
          gp: = 0;
      end;
      procedure TfrmStart.FormClose (Sender: TObject; var Action: TCloseAc-
tion);
      begin
             DM.ConnectUkrDocMdb.Connected: = false;
             DM.tblPlat.Active:=false;
             DM.tblCity.Active:=false;
             DM.tblPayment.Active:=false;
             DM.tblTypePlat.Active:=false;
             DM.tblRecipient.Active:=false;
      end;
      end.
```

```
unit untMain;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, Menus, Buttons, StdCtrls, DBCtrls, ExtCtrls, ComCtrls, Mask,
  Grids, DBGrids;
type
  TfrmMain = class (TForm)
    MainMenu1: TMainMenu;
    mDoc: TMenuItem;
    mDov: TMenuItem;
    mProperty: TMenuItem;
    mHelp: TMenuItem;
    mDocPlat: TMenuItem;
    mDocKvit: TMenuItem;
    mDovPlat: TMenuItem;
    mDovPol: TMenuItem;
    mDovTypePlat: TMenuItem;
    mPropertyShablon: TMenuItem;
    mPropertyShablonPlat: TMenuItem;
    mPropertyShablonKvit: TMenuItem;
    mPropertyShablonOpen: TMenuItem;
    mPropertyShablonPlatEx1: TMenuItem;
    mPropertyShablonPlatEx2: TMenuItem;
    mPropertyShablonPlatClear: TMenuItem;
    mHelpPO: TMenuItem;
    mHelpAvtor: TMenuItem;
    mPropertyView: TMenuItem;
    N1: TMenuItem;
    N2: TMenuItem;
    pnlPayment: TPanel;
    grpToolbox: TGroupBox;
    grpPlat: TGroupBox;
    btnPlat: TBitBtn;
    grpRecipient: TGroupBox;
    btnRecipient: TBitBtn;
    grpComment: TGroupBox;
    dbComment: TDBMemo;
    DBGrid1: TDBGrid;
    grpTypePlat: TGroupBox;
    btnTypePlat: TBitBtn;
    lblNumber: TLabel;
    dbEdNumber: TDBEdit;
    lblDate: TLabel;
    DateTimePicker1: TDateTimePicker;
    dbEdDate: TDBEdit;
    lblSumma: TLabel;
    dbEdSumma: TDBEdit;
    dbCBPlat: TDBLookupComboBox;
    dbCBRecipient: TDBLookupComboBox;
    mExit: TMenuItem;
    btnAdd: TBitBtn;
```

```
btnEdit: TBitBtn;
btnDel: TBitBtn;
pmGrid: TPopupMenu;
fNumber: TMenuItem;
fDate: TMenuItem;
fSumma: TMenuItem;
separator: TMenuItem;
View: TMenuItem;
dbLookCBTypePlat: TDBLookupComboBox;
pnlKvitPov: TPanel;
lblPlat: TLabel;
lblAdressPlat: TLabel;
lblAcctPlat: TLabel;
lblTypePlat: TLabel;
lblDatePlat: TLabel;
lblSummaPlat: TLabel;
lblEx: TLabel;
lblRecipient: TLabel;
lblAcct: TLabel;
lblBank: TLabel;
lblMFO: TLabel;
lblIdBank: TLabel;
lblIdRecipient: TLabel;
edtPlat: TEdit;
edtAccatPlat: TEdit;
edtAdressPlat: TEdit;
edtTypePlat: TEdit;
edtSummaPlat: TEdit;
DateTimePicker2: TDateTimePicker;
btnViewReport: TButton;
edtRecipient: TEdit;
edtAcct: TEdit;
edtBank: TEdit;
edtMFO: TEdit;
edtIdRecipient: TEdit;
edtIdBank: TEdit;
procedure FormClose (Sender: TObject; var Action: TCloseAction);
procedure mExitClick (Sender: TObject);
procedure FormCreate (Sender: TObject);
procedure DateTimePicker1Change (Sender: TObject);
procedure btnAddClick (Sender: TObject);
procedure btnEditClick (Sender: TObject);
procedure btnDelClick (Sender: TObject);
procedure fNumberClick (Sender: TObject);
procedure fDateClick (Sender: TObject);
procedure fSummaClick (Sender: TObject);
procedure mDocPlatClick (Sender: TObject);
procedure mDocKvitClick (Sender: TObject);
procedure edtAcctKeyPress (Sender: TObject; var Key: Char);
procedure mDovPlatClick (Sender: TObject);
procedure btnPlatClick (Sender: TObject);
procedure mDovPolClick (Sender: TObject);
procedure btnRecipientClick (Sender: TObject);
```

```
          procedure btnTypePlatClick (Sender: TObject);
          procedure mDovTypePlatClick (Sender: TObject);
          procedure dbLookCBTypePlatClick (Sender: TObject);
        private
          {Private declarations}
        public
          {Public declarations}
        end;
      var
        frmMain: TfrmMain;
      implementation
      uses untDM, untWizard, untStart, untFindNumber, untFindDate, untFind-
Summa,
        untPlat, untRecipient, untTypePlat;
      {$ R * .dfm}
      procedure TfrmMain.FormClose (Sender: TObject; var Action: TCloseAc-
tion);
      begin
          frmStart.Close ();
      end;
      procedure TfrmMain.mExitClick (Sender: TObject);
      begin
          if Application.MessageBox (PChar ( 'Ви впевнені что бажаєте закрити
програму?'),
            'Підтвердження !!!', MB_OKCANCEL) = id_OK then
          frmStart.Close ();
      end;
      procedure TfrmMain.FormCreate (Sender: TObject);
      begin
          DM.tblPayment.Cancel;
          DateTimePicker1.Enabled: = false;
          dbLookCBTypePlat.Enabled: = false;
          dbEdDate.Enabled: = false;
          dbEdNumber.Enabled: = false;
          dbEdSumma.Enabled: = false;
          dbCBPlat.Enabled: = false;
          dbCBRecipient.Enabled: = false;
          dbComment.Enabled: = false;
          btnAdd.Caption: = 'Додати';
          btnEdit.Caption: = 'Изменить';
          btnDel.Caption: = 'ВИДАЛИТИ';
         end;
      procedure TfrmMain.DateTimePicker1Change (Sender: TObject);
      begin
          dbEdDate.Text: = DateToStr (DateTimePicker1.Date);
      end;
      procedure TfrmMain.btnAddClick (Sender: TObject);
      begin
        if (btnAdd.Caption = 'Додати') then
          begin
            DM.tblPayment.Insert;
            DateTimePicker1.Enabled: = true;
            dbLookCBTypePlat.Enabled: = true;
```

```
      dbEdDate.Enabled: = true;
      dbEdNumber.Enabled: = true;
      dbEdSumma.Enabled: = true;
      dbCBPlat.Enabled: = true;
      dbCBRecipient.Enabled: = true;
      dbComment.Enabled: = true;
       btnDel.Enabled: = false;
       btnEdit.Caption: = 'Запис';
       btnAdd.Caption: = 'Відмінити';
      end // cancel
        else
          begin
          DM.tblPayment.Cancel;
          DateTimePicker1.Enabled: = false;
          dbLookCBTypePlat.Enabled: = false;
          dbEdDate.Enabled: = false;
          dbEdNumber.Enabled: = false;
          dbEdSumma.Enabled: = false;
          dbCBPlat.Enabled: = false;
          dbCBRecipient.Enabled: = false;
          dbComment.Enabled: = false;
          btnDel.Enabled: = true;
          btnAdd.Caption: = 'Додати';
          btnEdit.Caption: = 'Изменить';
          btnDel.Caption: = 'ВИДАЛИТИ';
          end;
end;
procedure TfrmMain.btnEditClick (Sender: TObject);
begin
  if (btnEdit.Caption = 'Изменить') then
    begin
      DM.tblPayment.Edit;
      DateTimePicker1.Enabled: = true;
      dbLookCBTypePlat.Enabled: = true;
      dbEdDate.Enabled: = true;
      dbEdNumber.Enabled: = true;
      dbEdSumma.Enabled: = true;
      dbCBPlat.Enabled: = true;
      dbCBRecipient.Enabled: = true;
      dbComment.Enabled: = true;
       btnDel.Enabled: = false;
       btnEdit.Caption: = 'Запис';
      btnAdd.Caption: = 'Відмінити';
      end // save
        else
          begin
          DM.tblPayment.Post;
          DateTimePicker1.Enabled: = false;
          dbLookCBTypePlat.Enabled: = false;
          dbEdDate.Enabled: = false;
          dbEdNumber.Enabled: = false;
          dbEdSumma.Enabled: = false;
          dbCBPlat.Enabled: = false;
```

```
                dbCBRecipient.Enabled: = false;
                dbComment.Enabled: = false;
                btnDel.Enabled: = true;
                btnAdd.Caption: = 'Додати';
                btnEdit.Caption: = 'Изменить';
                btnDel.Caption: = 'ВИДАЛИТИ';
                end;
      end;
      procedure TfrmMain.btnDelClick (Sender: TObject);
      begin
          if Application.MessageBox (PChar ( 'Ви впевнені что хочете ВИДАЛИТИ
поточний документ?'),
            'Підтвердження !!!', MB_OKCANCEL) = id_OK then
                DM.tblPayment.Delete;
      end;
      procedure TfrmMain.fNumberClick (Sender: TObject);
      begin
          frmFindNumber.ShowModal;
          if (frmFindNumber.edtFindNumber.Text <> '') then
            begin
              if not DM.tblPayment.Locate ( 'number', frmFindNum-
ber.edtFindNumber.Text, []) then
                  ShowMessage ( 'Запис Полтава!');
            end;
      end;
      procedure TfrmMain.fDateClick (Sender: TObject);
      begin
          frmFindDate.ShowModal;
          if (frmFindDate.edtFlag.Text = '1') then
            begin
              if not DM.tblPayment.Locate ( 'date', DateToStr (frmFind-
Date.DateTimePicker1.Date), []) then
                  ShowMessage ( 'Запис Полтава!');
            end;
      end;
      procedure TfrmMain.fSummaClick (Sender: TObject);
      begin
          frmFindSumma.ShowModal;
          if (frmFindSumma.edtFindSumma.Text <> '') then
            begin
              if not DM.tblPayment.Locate ( 'summa', frmFind-
Summa.edtFindSumma.Text, []) then
                  ShowMessage ( 'Запис Полтава!');
            end;
      end;
      procedure TfrmMain.mDocPlatClick (Sender: TObject);
      begin
            pnlKvitPov.Visible: = false;
            mDocPlat.Checked: = true;
            mDocKvit.Checked: = false;
      end;
      procedure TfrmMain.mDocKvitClick (Sender: TObject);
      begin
```

```
            pnlKvitPov.Visible: = true;
            mDocPlat.Checked: = false;
            mDocKvit.Checked: = true;
      end;
      procedure TfrmMain.edtAcctKeyPress (Sender: TObject; var Key: Char);
      begin
        if not (key in [ '0' .. '9', decimalseparator, # 8]) then key: = # 0;
      end;
      procedure TfrmMain.mDovPlatClick (Sender: TObject);
      begin
          frmPlat.ShowModal;
      end;
      procedure TfrmMain.btnPlatClick (Sender: TObject);
      begin
          frmPlat.ShowModal;
      end;
      procedure TfrmMain.mDovPolClick (Sender: TObject);
      begin
          frmRecipient.ShowModal;
      end;
      procedure TfrmMain.btnRecipientClick (Sender: TObject);
      begin
          frmRecipient.ShowModal;
      end;
      procedure TfrmMain.btnTypePlatClick (Sender: TObject);
      begin
          frmTypePlat.ShowModal;
      end;
      procedure TfrmMain.mDovTypePlatClick (Sender: TObject);
      begin
            frmTypePlat.ShowModal;
      end;
      procedure TfrmMain.dbLookCBTypePlatClick (Sender: TObject);
      begin
              if (IntToStr (frmMain.dbLookCBTypePlat.KeyValue) <> '') then
              begin
                DM.qTypePlat.SQL.Text: = 'SELECT * FROM typePlat WHERE id =' +
IntToStr (frmMain.dbLookCBTypePlat.KeyValue); // текст запиту
                DM.qTypePlat.Open;
                if DM.qTypePlat.IsEmpty then Exit; // якщо набір порожній -
виходимо з процедури
                dbEdDate.Text: = DM.qTypePlat.FieldByName ( 'date'). AsVari-
ant;
                DateTime-
Picker1.Date:=DM.qTypePlat.FieldByName('date').AsVariant;
                dbEdNumber.Text: = DM.qTypePlat.FieldByName ( 'number'). As-
Variant;
                dbEdSumma.Text: = DM.qTypePlat.FieldByName ( 'summa'). AsVari-
ant;
                dbCBPlat.KeyValue: = DM.qTypePlat.FieldByName ( 'id_plat').
AsVariant;
                dbCBRecipient.KeyValue: = DM.qTypePlat.FieldByName (
'id_recipient'). AsVariant;
```

```
                dbComment.Text: = DM.qTypePlat.FieldByName ( 'coment'). As-
Variant;
                end;
      end;
      end.

      unit untWizard1;
      interface
      uses
        Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
        Dialogs, StdCtrls, ExtCtrls, Mask, DBCtrls;
      type
        TfrmWizard1 = class (TForm)
          pnlCaption: TPanel;
          lbCap1: TLabel;
          lbCap2: TLabel;
          lbCap3: TLabel;
          pnlInfo: TPanel;
          lbNameOrg: TLabel;
          lbExNameOrg: TLabel;
          lbAcct: TLabel;
          lbExAcct: TLabel;
          lbZKPO: TLabel;
          lbMFO: TLabel;
          lbBank: TLabel;
          lbExZKPO: TLabel;
          lbExMFO: TLabel;
          lbExBank: TLabel;
          lbCity: TLabel;
          pnlTools: TPanel;
          btnNext: TButton;
          dbEdtNameOrg: TDBEdit;
          dbEdtAcct: TDBEdit;
          dbEdtZKPO: TDBEdit;
          dbEdtMFO: TDBEdit;
          dbEdtBank: TDBEdit;
          dbLookCBCity: TDBLookupComboBox;
          btnPrev: TButton;
           procedure btnNextClick (Sender: TObject);
          procedure FormActivate (Sender: TObject);
          procedure dbEdtAcctKeyPress (Sender: TObject; var Key: Char);
          procedure FormClose (Sender: TObject; var Action: TCloseAction);
          procedure btnPrevClick (Sender: TObject);
        private
          {Private declarations}
          procedure WMSysCommand (var Msg: TWMSysCommand); message
WM_SYSCOMMAND;
        public
          {Public declarations}
        end;
      var
        frmWizard1: TfrmWizard1;
```

```
implementation
uses untWizard2, untDM, untWizard, untStart;
{$ R * .dfm}
procedure TfrmWizard1.WMSysCommand (var Msg: TWMSysCommand);
 begin
   if ((Msg.CmdType and $ FFF0) = SC_MOVE) then
   begin
     Msg.Result: = 0;
     Exit;
   end;
   inherited;
 end;
procedure TfrmWizard1.btnNextClick (Sender: TObject);
begin

  if ((dbEdtAcct.EditText <> '') and (dbEdtBank.EditText <> '') and
       (DbEdtMFO.EditText <> '') and (dbEdtNameOrg.EditText <> '') and
       (DbEdtZKPO.EditText <> '') and (dbLookCBCity.Text <> '')) then
        begin
          frmWizard1.Visible: = false;
          frmWizard2.Visible: = true;
        end
   else
    MessageDlg ( 'необходимо заповнити усі поля!', MtInformation,
[mbOk], 0);
 end;
procedure TfrmWizard1.FormActivate (Sender: TObject);
begin
  Left: = 400;
  Top: = 190;
  ClientHeight: = 369;
  ClientWidth: = 449;

end;
procedure TfrmWizard1.dbEdtAcctKeyPress (Sender: TObject; var Key:
Char);
begin
  if not (key in [ '0' .. '9', # 8]) then key: = # 0;
end;
procedure TfrmWizard1.FormClose (Sender: TObject; var Action: TCloseAc-
tion);
begin
  DM.tblPlat.Cancel;
  DM.ConnectUkrDocMdb.Connected: = false;
  frmStart.Close ();
end;
procedure TfrmWizard1.btnPrevClick (Sender: TObject);
begin
  frmWizard1.Visible: = false;
  frmWizard.Visible: = true;
end;
end.
```