КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет радіофізики, електроніки та комп'ютерних систем
Кафедра комп'ютерної інженерії

Локалізація українського податкового законодавства в Microsoft Dynamics 365 Business Central

Кваліфікаційна робота бакалавра
студента 4 року навчання
Спеціальність: 123 «Комп'ютерна інженерія»
Дмитра КУРИНОГО

Науковий керівник
к.т.н. **Юрій КОБА**,
доцент кафедри комп'ютерної інженерії

Рецензент:
к.ф.-м.н., доцент
кафедри нанофізики конденсованих середовищ
Інституту високих технологій
Ірина ГАВРИЛЬЧЕНКО

До захисту допускаю:
Завідувач кафедрою Юрій БОЙКО

Ухвалено на засіданні кафедри " 16 " червня 2022 р., протокол № 18

Київ 2022

РЕФЕРАТ

Звіт бакалаврської дипломної роботи за об’ємом складає 147 сторінок, містить 18 рисунків, 8 таблиць, 5 додатків, використано 11 інформаційних джерел.

ПОДАТКОВА НАКЛАДНА, ЛОКАЛІЗАЦІЯ, MICROSOFT DYNAMICS 365 BUSINESS CENTRAL, АВТОТЕСТИ, ЗВІТИ ІМПОРТУ ТА ЕКСПОРТУ.

Об’єктом дослідження стала наявна реалізація ПДВ у системі Microsoft Dynamics 365 Business Central.

Метою роботи є створення структури документів податкових накладних, реалізація їх обліку, створення автоматичних тестів та звітів для імпорту/експорту податкових накладних.

Методи розроблення: розробка програмного продукту на основі вимог українського податкового законодавства. Інструменти розроблення: безкоштовне, вільно поширюване середовище розробки Visual Studio Code, мова програмування AL.

Результати роботи: виконано загальний огляд реалізації функціональності ПДВ у системі MS Dynamics 365 BC. Розроблено структуру документів податкових накладних, реалізовано їх облік, розроблено автотести для перевірки коректності обліку на чистій базі або після оновлення продукту, розроблено звіти для імпорту та експорту податкових накладних у формат XML-файлів для подальшого завантаження в систему електронного документообігу МЕДОК.

В подальшому рішення можна розвивати для створення повноцінної локалізації продукту Microsoft Dynamics 365 Business Central не тільки в розділі ПДВ, а в цілому.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧКИ

ПН – податкова накладна;

BC – Microsoft Dynamics 365 Business Central;

ERP – Enterprise Resource Planning;

MS – Microsoft;

ОС – операційна система;

УКТ ЗЕД – українська класифікація товарів зовнішньоекономічної діяльності;

ДКПП – державний класифікатор продуктів і послуг;

ЄРПН – єдиний реєстр податкових накладних;

ПДВ – податок на додану вартість;

ГК – головна книга;

VS Code – Visual Studio Code;

БД – база даних;

Автотест – автоматичний тест;

PK – Primary Key.

ВСТУП

Сьогодні складно уявити людину, яка не користується комп'ютером у повсякденному житті. Чи то для роботи, чи то вдома для навчання, чи в комп'ютерному клубі для відпочинку, кожен день людина полегшує собі життя, використовуючи комп'ютер.

З розвитком комп'ютерних технологій зростає їх інтеграція у повсякденне життя людини, окрім того також зростає їх використання у корпоративних цілях. Адже використання комп'ютера в робочих цілях значно може полегшити робочий процес.

В теперішній час складно оцінити всю залежність та значущість для бізнесу та приватного користування електронних пристроїв, обмін інформацією за допомогою їх. Одна маленька помилка в робочому процесі людини може коштувати дуже великих грошей компанії, тому варто звертати увагу на безпечну та стабільну роботу компанії.

Для вирішення цих задач на сучасних підприємствах впроваджуються комплексні корпоративні інформаційні системи (КІС), важливою складовою яких є ERP системи. ERP системи становлять основу КІС, яка охоплює всі бізнес процеси компанії: планування коштів, обліковий процес, рух товарів та компонентів; придбання та продаж готової продукції та товарів, розрахунок їх вартості та вартості пов'язаних з ними послуг; швидкий доступ до актуальної інформації, щодо стану компанії, кількісті товару на складі, обігу грошей, актуальні замовлення, фінансові операції та інші. Інформація, яка накопичується і обробляється ERP системою, сприяє грамотному управлінню компанією, прибутковому розвитку бізнесу, результативному контролю всіх процесів топ-менеджментом компанії.

1. АКТУАЛЬНІСТЬ РОБІТ ПО ЛОКАЛІЗАЦІЇ ПОДАТКОВОГО ЗАКОНОДАВСТВА В MICROSOFT DYNAMICS 365 BUSINESS CENTRAL

Більшість ERP-систем створюються у загальному вигляді, аби покривати універсальну більшість потреб, часто не враховуючі локальні особливості ведення бізнесу в тій чи іншій країні. Тому виникає потреба у створенні так званих локалізацій.

Локалізація - процес адаптації програмного продукту для використання у певній країні. Наприклад, переклад інтерфейсу користувача, підтримка національних вимог та стандартів, підтримка податкового законодавства тощо. В рамках локалізації ERP-системи відбувається тонке налаштування системи для ведення бізнесу в цільовій країні.

Усім компаніями, які планують вести бізнес у тій чи іншій країні, потрібно дотримуватися законів та правил ведення документації, бізнесу країни, в якій вони працюють. Одним з масштабних розділів у сфері ведення бізнесу є податкова сфера.

В Україні для визначення податкових зобов’язань використовуються податкові накладні. Податкова накладна – це документ, яким користуються платники ПДВ для обліку податкового зобов’язання та податкового кредиту. Для того, щоб покупцю отримати право на податковий кредит, податкову накладну необхідно зареєструвати у єдиному реєстрі податкових накладних. Ці податкові накладні повинні бути створені в момент появи податкового зобов’язання. Зазвичай це визначається правилом першої події:

- Дата отримання передоплати за будь-який товар (ресурс, послугу, роботу тощо),
- Дата фактичного отримання цього товару (ресурсу, послуги).

Судячи з того яка подія виникла раніше, на ту дату і потрібно створювати податкову накладну.

В базовій функціональності системи BC не передбачені податкові накладні, хоча звісно інструментарій для податкової вартості присутній.

В BC можна створити ПДВ бізнес групу, ПДВ товарну групу, виставити, на який рахунок буде нараховуватися ПДВ, і вказати відсоток. У рахунку-фактурі або будь-якому іншому замовленні (продаж/купівлю) автоматично будуть обраховуватися суми з ПДВ та без ПДВ, орієнтуючись на ПДВ бізнес групу, ПДВ товарну групу, які закріплені за певним товаром, або ж клієнтом.

Для функціонування бізнесу в Україні цього не достатньо. Звичайно податкові накладні можна створювати вручну за допомогою паперу та ручки, проте витрат часу на дорогу до податкової і заповнення цих накладних потрібно дуже багато. Окрім того з’являється людський фактор. В умовах величезних компаній це робити неможливо, тому потрібно максимально автоматизувати цей процес. У практичній частині цієї роботи буде створений функціонал податкових накладних, їх облік, вивантаження для імпорту у програму електронного обігу документів МЕДОК.

2. MICROSOFT DYNAMICS 365 BUSINESS CENTRAL ЯК СИСТЕМА ТА ЗАСОБИ РОЗРОБКИ ЇЇ ЛОКАЛІЗАЦІЇ

Однією з передових сучасних ERP-систем є Microsoft Dynamics 365 Business Central, розробник корпорація Microsoft.

Ця ERP-система створена для малого та середнього бізнесу, які переросли потреби бізнес-програм початкового рівня. Компанії, які розвиваються, часто переростають свої початкові бухгалтерські програми та застарілі облікові системи, які вже не можуть підтримувати ту дедалі більшу кількість операцій -транзакцій, кількість записів, не мають інтеграції з іншими бізнес-програмами або ж мають обмежену систему звітності. Інколи деякі компанії мають складну або ж нестандарту (часом модернізовану) систему логістики, систему поповнення запасів. Всі ці задачі може покрити ERP-система MS Dynamics 365 BC. [1]

2.1. Загальна характеристика ERP Microsoft Dynamics 365 Business Central

Microsoft Dynamics 365 Business Central працює у середовищі ОС Windows. Особливості цієї системи характерні для більшості сучасних ERP систем:

- спільна база даних,
- підтримка декількох бізнес доменів,
- інтегровані та модульні функції,
- доступність даних в режимі реального часу,
- посадові ролі та авторизація,
- великий рівень безпеки.

Всі ці особливості дають змогу пов’язати бізнесу фінанси, продажі, послуги, логістику, складський облік в одну інформаційну систему.

Це у свій час полегшує топ-менеджменту процес аналізу роботи як компанії в цілому, так і окремого структурного підрозділу або ж відділу.

Як показано на рис. 1, MS Dynamics 365 BC автоматично об'єднує системи та процеси, дозволяючи користувачам керувати фінансами, продажами, обслуговуванням та операціями. Для налаштування процесу продажу або покупки можна використовувати широкий спектр документів купівлі-продажу, який відповідає потребам компанії. У BC можна керувати замовленнями на продаж/купівлю, загальними замовленнями на продаж/купівлю та процесами замовлень на продаж/купівлю. Також існує можливість створити рахунок-фактуру без процесу замовлення.[2][3]

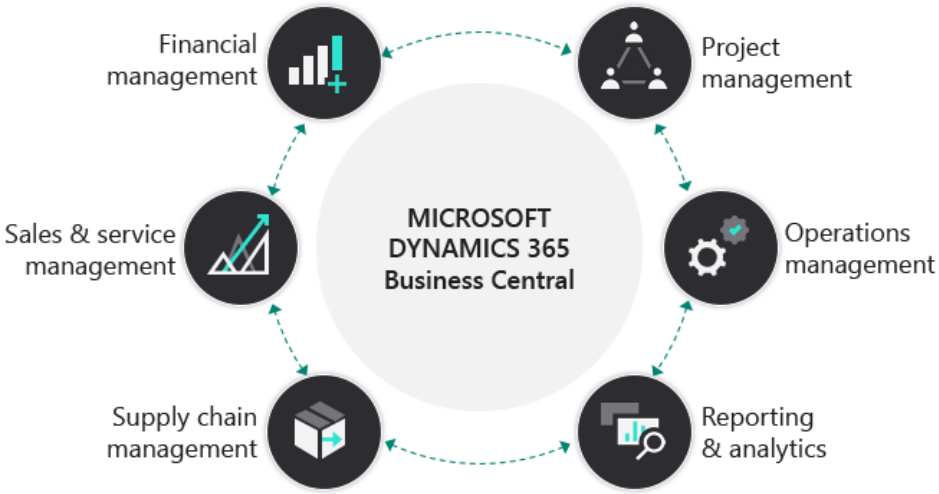


Рисунок 1 – Основні модулі системи BC

MS Dynamics 365 BC має доволі гнучкі варіанти розгортання системи:

1) Хмарне рішення (розміщується на ресурсах Microsoft)
2) Встановлення на приватних серверах

3) Гібридна версія (поєднання хмарного та локального розгортання)

Більшість нових користувачів MS BC обирають варіант хмарного застосунку. Основними перевагами, через які користувачі обирають хмарні застосунки, є: швидке та безперебійне впровадження системи, без додаткових витрат на внутрішні фізичні та IT-ресурси; не потрібно підтримувати роботу серверу, автоматичне оновлення системи двічі на рік, високий рівень безпеки даних, інтеграція з великою кількістю онлайн ресурсів Microsoft. [4]

До основних переваг використання приватного рішення відносять: збереження повної власності над інформацією, можливість ефективного та своєчасного оновлення системи, масштабованість, безперебійна робота з базовим інтернет-підключенням.

Проте вибір моделі розгортання системи потрібно розглядати для кожного окремого випадку, адже все залежить від великої кількості параметрів, як фінансових так і фізичних можливостей клієнта. Все ж, наявність вибору між цими реалізаціями дає змогу обрати підходящу модель системи.

2.2. Загальна характеристика основних модулів системи Microsoft Dynamics 365 Business Central

Dynamics 365 Business Central пропонується у двох варіантах ліцензії: основна та преміум.

До основної версії відносять такі модулі: фінансовий, продажі і покупки, управління складом, модуль взаємовідносин з клієнтами, управління людськими ресурсами, проекти.

До преміумної версії відносяться усі модулі основної версії, а також модулі обслуговування та виробництва.

Далі представлений короткий огляд основних елементів модулів, які використовуються напряму чи побічно при реалізації і роботі податкових накладних.

2.2.1. Модуль продаж і покупок. Замовлення на продаж

Замовлення на продаж, як правило, використовується якщо потрібно зареєструвати факт продажу товару, послуги. Також в замовленні на продаж доступний процес часткового відвантаження замовлення, наприклад, оскільки повна кількість недоступна відразу. Крім того, можна використовувати замовлення на продаж, продаючи товари і доставляючи їх безпосередньо від свого постачальника клієнту, як пряме відправлення. В замовленні на продаж можна використовувати функції розрахунку доставки замовлення, щоб повідомляти клієнтам певні дати доставки. [5]

В загальному випадку замовлення на продаж поділяється на три частини: заголовок документа, рядки документа та вікна з додатковими статистичними або пов’язаними даними (рис. 2).

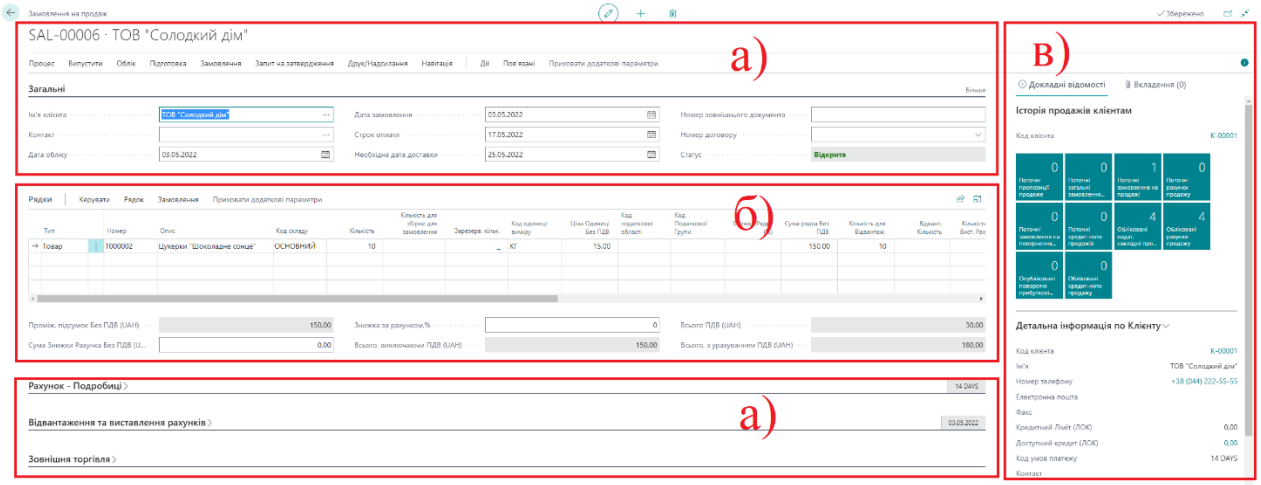


Рисунок 2 – Замовлення на продаж: а – заголовок; б – рядки; в – додаткові дані

Процес продажу також можна розпочати і не з замовлення на продаж, наприклад, спочатку створити пропозицію продажу, яку можна перетворити на рахунок-фактуру або замовлення на продаж, коли вже є домовленість про продаж. Після того, як клієнт підтвердить угоду, можна надіслати підтвердження замовлення, щоб зафіксувати своє зобов'язання доставити продукцію згідно з домовленістю.

З замовлення на продаж можна потрапити на розроблену сторінку-список ПН на продаж за допомогою функціонального меню «Податкові накладні», де будуть відображені ПН пов'язані з поточним документом.

Замовлення на покупку має так ж саму структуру, як і замовлення на продаж, окрім як робота ведеться з постачальником, а не клієнтом.

2.2.2. Модуль фінансів. Податок на додану вартість

Податок на додану вартість (ПДВ) — це податок з операцій, який сплачує кінцевий споживач, у тому числі підприємства.

У системі ВС функціонал ПДВ представлений у вигляді облікових груп ПДВ. Ці групи визначають як саме будуть проводитися і обраховуватися операції з сумами ПДВ. [6]

У таблиці 1 наведено описи двох облікових груп, які приймають участь у проводках ПДВ, та місця їх присвоєння.

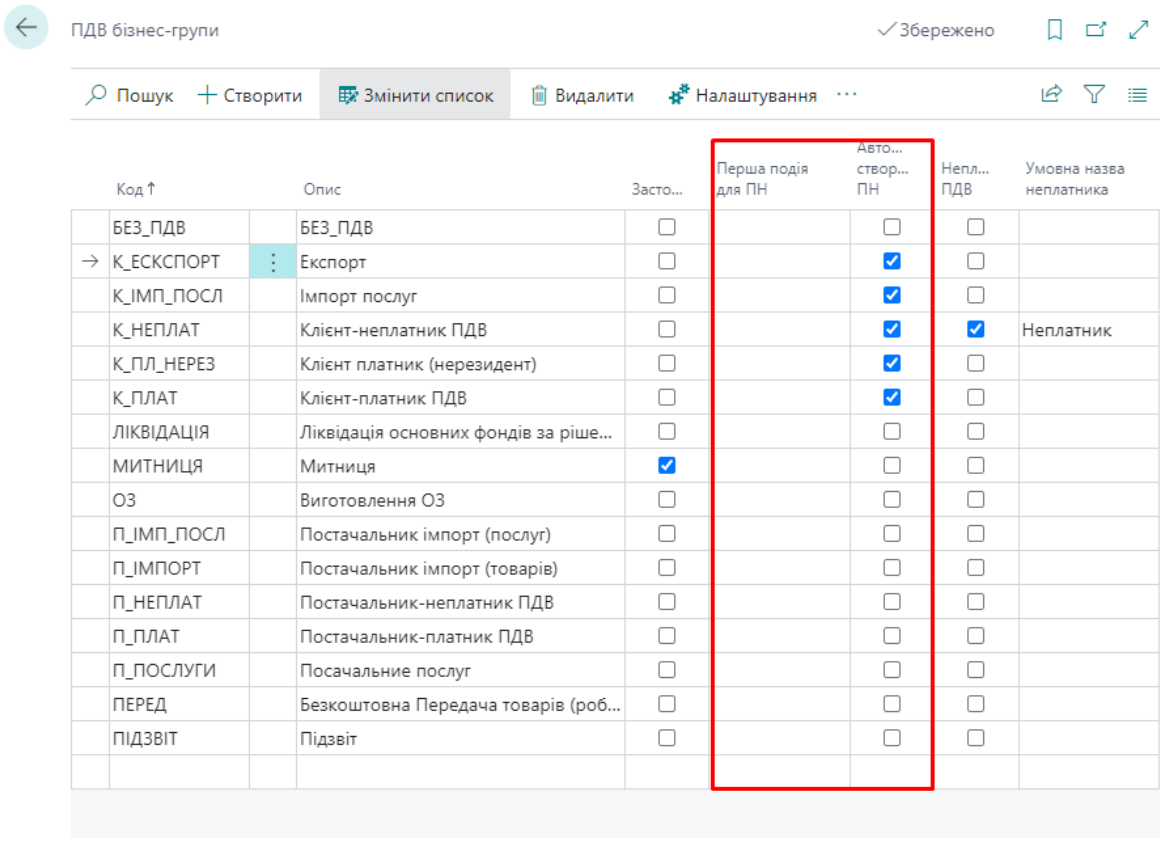
Таблиця 1 – Облікові групи ПДВ

Група обліку ПДВ	**Опис**	**Присвоюється**
ПДВ бізнес-група	Визначає обчислення та облік ПДВ відповідно до місцезнаходження (країни чи регіону) клієнта або постачальника, який приймає участь у процедурі обліку	Клієнтам, постачальникам, рахункам ГК
ПДВ товарна група	Визначає обчислення та облік ПДВ відповідно до типу товару чи ресурсу, що купується чи продається	Товарам, ресурсам, основним рахункам, витратам на товари

При налаштуванні ПДВ бізнес-групи потрібно продумати, скільки груп потрібно, виходячи з кількох факторів, зокрема:

- Місцеве законодавство
- Торгівля як всередині країни, так і за кордоном

Окрім стандартних полів ПДВ бізнес-груп («Код», «Опис», «Застосування не вимагається») в локалізації присутні поля для реалізації першої події (рис. 3).



Код	Опис	Засто...	Перша подія для ПН	Авто... створ... ПН	Непл... ПДВ	Умовна назва неплатника
БЕЗ_ПДВ	БЕЗ_ПДВ	☐		☐	☐	
К_ЕСКСПОРТ	Експорт	☐		☑	☐	
К_ІМП_ПОСЛ	Імпорт послуг	☐		☑	☐	
К_НЕПЛАТ	Клієнт-неплатник ПДВ	☐		☑	☑	Неплатник
К_ПЛ_НЕРЕЗ	Клієнт платник (нерезидент)	☐		☑	☐	
К_ПЛАТ	Клієнт-платник ПДВ	☐		☑	☐	
ЛІКВІДАЦІЯ	Ліквідація основних фондів за ріше...	☐		☐	☐	
МИТНИЦЯ	Митниця	☑		☐	☐	
ОЗ	Виготовлення ОЗ	☐		☐	☐	
П_ІМП_ПОСЛ	Постачальник імпорт (послуг)	☐		☐	☐	
П_ІМПОРТ	Постачальник імпорт (товарів)	☐		☐	☐	
П_НЕПЛАТ	Постачальник-неплатник ПДВ	☐		☐	☐	
П_ПЛАТ	Постачальник-платник ПДВ	☐		☐	☐	
П_ПОСЛУГИ	Посачальние послуг	☐		☐	☐	
ПЕРЕД	Безкоштовна Передача товарів (роб...	☐		☐	☐	
ПІДЗВІТ	Підзвіт	☐		☐	☐	

Рисунок 3 – Сторінка ПДВ бізнес-груп

При налаштуванні ПДВ товарної групи потрібно вирішити, скільки груп потрібно для податкових зобов’язань за товари та ресурси. Наприклад, ПДВ товарні групи можна налаштувати для групування товарів у продовольчі товари, які вимагають 20% ПДВ, і сільськогосподарські товари, які вимагають 14% ПДВ.

Зазвичай для ПДВ товарних груп заповнюється код та опис (рис. 4).

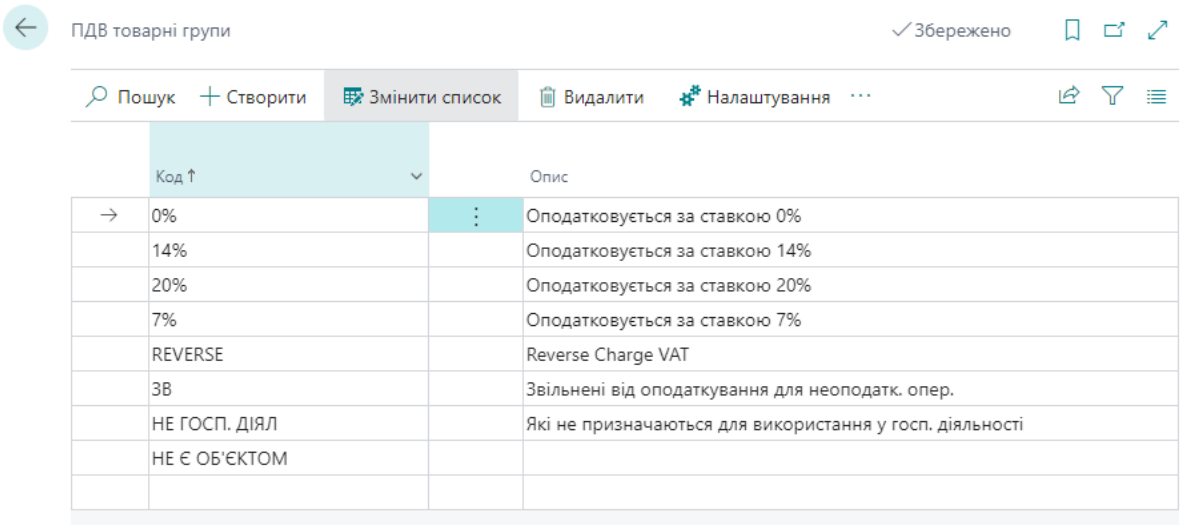


Рисунок 4 – Сторінка ПДВ товарних груп

Замість того, щоб вручну призначати ПДВ товарну групу або ПДВ бізнес-групу відповідним рахункам, можна налаштувати ПДВ товарну групу за замовчуванням для загальних груп товарів та ПДВ бізнес-групу за замовчуванням для загальних бізнес-груп.

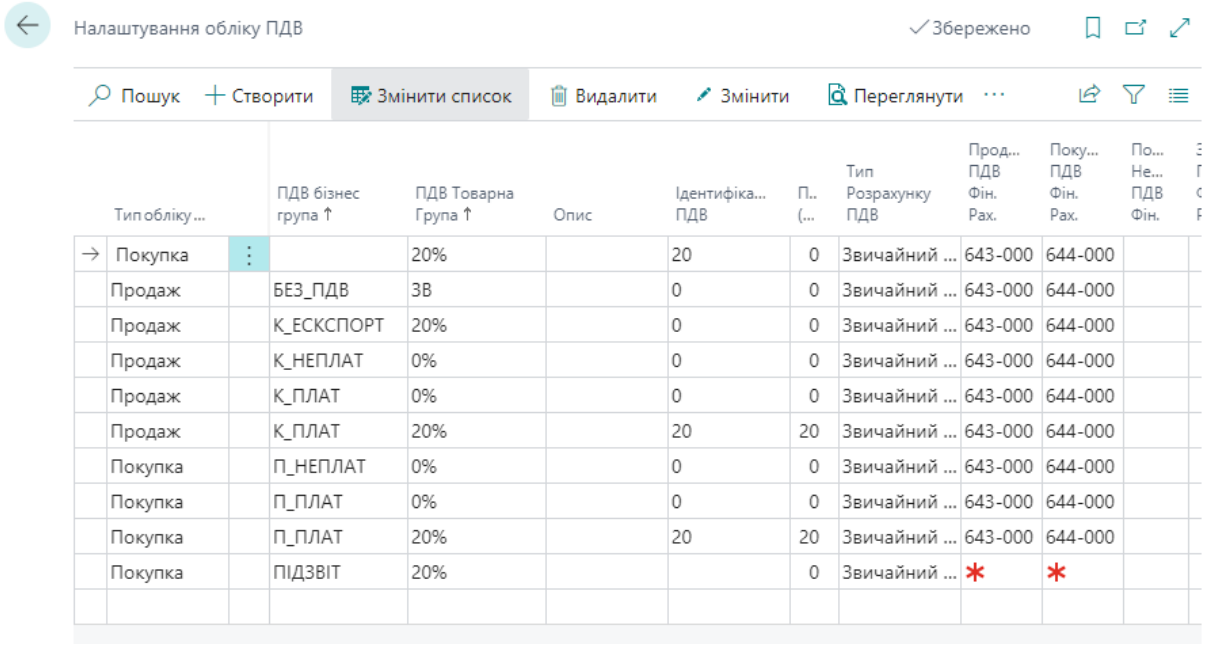
Після цього BC автоматично вставляє відповідний код як ПДВ товарну групу, коли відповідну загальну групу товарів призначають для товару, ресурсу або основного рахунку. Аналогічно і з ПДВ бізнес-групою, коли відповідну загальну бізнес-групу призначають клієнту, постачальнику або рахунку головної книги.

Якщо потрібно, можна змінити ПДВ бізнес-групу та ПДВ товарну групу для окремих записів.

Зі сторінок «ПДВ товарні групи» і «ПДВ бізнес-групи» доступний перегляд ПН, в яких використовується відповідна ПДВ товарна група або ПДВ бізнес-група.

Після створення ПДВ бізнес-груп та ПДВ товарних груп потрібно провести налаштування обліку ПДВ для коректного вирахування сум і переведення їх на потрібні рахунки під час процедури обліку. Для цього використовують сторінку «Налаштування обліку ПДВ».

Сторінка «Налаштування обліку ПДВ» - це матриця, яка поєднує ПДВ бізнес-групи і ПДВ товарні групи (рис. 5). Кожна комбінація визначає рахунки, які використовуватимуться для обліку ПДВ з продажу чи придбання. Також тут вказується і відсоток ПДВ.



Тип обліку...	ПДВ бізнес група	ПДВ Товарна Група	Опис	Ідентифіка... ПДВ	П... (...	Тип Розрахунку ПДВ	Прод... ПДВ Фін. Рах.	Поку... ПДВ Фін. Рах.	По... Не... ПДВ Фін.
Покупка		20%		20	0	Звичайний ...	643-000	644-000	
Продаж	БЕЗ_ПДВ	3В		0	0	Звичайний ...	643-000	644-000	
Продаж	К_ЕСКСПОРТ	20%		0	0	Звичайний ...	643-000	644-000	
Продаж	К_НЕПЛАТ	0%		0	0	Звичайний ...	643-000	644-000	
Продаж	К_ПЛАТ	0%		0	0	Звичайний ...	643-000	644-000	
Продаж	К_ПЛАТ	20%		20	20	Звичайний ...	643-000	644-000	
Покупка	П_НЕПЛАТ	0%		0	0	Звичайний ...	643-000	644-000	
Покупка	П_ПЛАТ	0%		0	0	Звичайний ...	643-000	644-000	
Покупка	П_ПЛАТ	20%		20	20	Звичайний ...	643-000	644-000	
Покупка	ПІДЗВІТ	20%			0	Звичайний ...	*	*	

Рисунок 5 – Сторінка «Налаштування обліку ПДВ»

2.3. Засоби розробки локалізації системи

2.3.1. Visual Studio Code

Для розробки в системі BC використовують середу розробки Visual Studio Code, яка працює в середовищі ОС Windows. Це доволі гнучка середа, яку можна доволі зручно налаштувати, так як просто встановленої середи розробки не достатньо для початку розробки. [7]

В VS Code додатково потрібно встановити розширення з мовою програмування AL. AL – це мова програмування, яка використовується для

маніпулювання даними (наприклад, отримання, вставлення та зміна записів) у базі даних BC та керування виконанням різних об’єктів програми, таких як сторінки, звіти або блоки коду. З допомогою цієї мови можна створювати бізнес-правила, щоб гарантувати, що дані, які зберігаються в базі даних, мають зміст і узгоджуються з тим, як клієнти ведуть бізнес. Майже кожен об’єкт у BC містить тригери, до яких можна додати AL-код. [8]

Також для зручності в процесі розробки можна встановити додаткові необов’язкові розширення, які представлені у таблиці 2.

Таблиця 2 – Корисні розширення VS Code для розробки

Назва	**Опис**
AL Object Designer	Розширення, через яке можна зручно оглядати код стандартних об'єктів ядра системи BC
AL Variable Quickfix	Розширення для перейменування змінних згідно рекомендацій MS в розробці. В процесі роботи не завжди зручно використовувати рекомендовані MS назви змінних, тому аби не марнувати час на подібні виправлення зручно використовувати це розширення
Fix AL File Names	Розширення для перейменування файлів згідно рекомендацій MS в розробці. В процесі роботи не завжди зручно використовувати рекомендовані MS назви файлів, тому аби не марнувати час на подібні виправлення зручно використовувати це розширення
NAB AL Tools	Розширення для роботи з файлами перекладів. Це розширення при оновленні файлу, який містить усі назви полів, об’єктів; тексти помилок, повідомлень, оновлює файли з перекладами, та помічає місця в яких не вистачає перекладів або переклади змінилися і

Назва	Опис
	вимагають повторного перегляду. Також це розширення зручно відображає різні переклади для одного і того ж тексту оригінальною мовою.
WakaTime	Розширення для відслідковування часу, який було витрачено на певний проект, певну мову програмування (наприклад, файли перекладу мають XML формат, а файли підключення до екземпляру БД – JSON формат)

Для створення проекту, який потім завантажується на сервер з BC як окреме розширення, потрібно запустити рядок швидких команд за домогою комбінації клавіш «Ctrl + Shift + P» і вибрати команду «AL: GO!». Після чого створиться проект розширення BC (з унікальним id проекту) в який потрібно додати код наявний у додатках, вказати параметри підключення та підвантажити код стандартних об’єктів ядра. Далі завантажити проект на сервер і використовувати.

2.3.2. Інструментарій тестування в системі Microsoft Dynamics 365 Business Central

Зазвичай по закінченню роботи розробника, потрібно провести тестування новоствореної або ж вже існуючої модифікації в системі BC. Спочатку розробник тестує виконане завдання по тестовому прикладу бізнес-консультанта. Після того, як тестовий приклад у розробника був вдалим, виконана розробка переходить на тест до бізнес-консультанта, який тестує модифікацію всебічно. Але такий підхід розробки-тестування не є оптимальним впродовж всього проекту, адже проекти можуть тривати декілька років. Якщо проект буде тривалим – буде виконано велику кількість розробок/модифікацій, вийде декілька оновлень системи від MS.

В цьому випадку потрібно перетестувувати усю роботу системи, що потребує великої кількості часу і зусиль. Тому MS радить писати так звані автотести, які будуть заміняти ручне тестування певних розробок. В найкращому випадку автотестами можна покрити 95% доробок системи.

Автотести – це спеціальні тестові функції, які повторюють ті дії які потрібно провести для тестування модифікації (наприклад, створення документа і заповнення його певним чином; натискання на певну кнопку сторінки і перевірка результату її роботи). Вони знаходяться в об’єктах типу «Codeunit» і пишуться розробниками. З цього випливає, що час розробки зростає майже вдвічі (час на написання автотесту, зазвичай, займає приблизно стільки ж часу скільки і розробка модифікації).

Проте в довготривалій перспективі вони виграють дуже велику кількість часу. Наприклад, після проведення чергової розробки, яка потенціально могла зачепити роботу системи, консультант був би вимушений вручну перевіряти окрім як нову розробку ще й функціонал, який міг бути поламаним. Така сама ситуація і з оновленням системи, з виходом його могла змінитися стандартна робота об’єктів і порушитися якась логіка кастомізаційного рішення. У випадку написання автотестів є змога у напівавтоматичному режимі перевірити усі минулі розробки.

Автотести пишуться таким чином, щоб покрити однією тестовою функцією одну невелику задачу. Наприклад, це може бути перевірка правильності заповнення полів, які залежать від заповнення іншого поля; або ж це може бути перевірка облікового процесу, чи перевірка на отримання помилки при певних вхідних даних (негативно-позитивний автотест). [9]

Структура автотесту нескладна і складається з трьох частин:

- GIVEN – частина в якій створюються тестові вхідні дані (наприклад, документ для перевірки процедури обліку)
- WHEN – частина в якій відбувається ключова дія, яка потребує перевірки (наприклад, запуск процедури обліку)

- THEN – частина в якій перевіряється результат роботи системи, порівнюються дані, які створені в системі, з даними, які ми очікували побачити

Автотести повинні бути написані таким чином, аби навіть на повністю чистій базі вони відпрацьовували, тобто в частині «GIVEN» мають бути ініціалізовані усі потрібні таблиці налаштувань і створені усі необхідні допоміжні дані.

Усі автотести незалежні один від одного. Тобто, якщо якийсь тест був невдалим, користувач побачить яка була помилка, проте наступні тести будуть виконані. Є змога вказати, чи потрібно по закінченню тесту відкатувати усі зміни у таблицях, або ж їх потрібно зберегти.

Для виклику і запуску автотестів використовується сторінка «Тестові інструменти» (рис. 6).

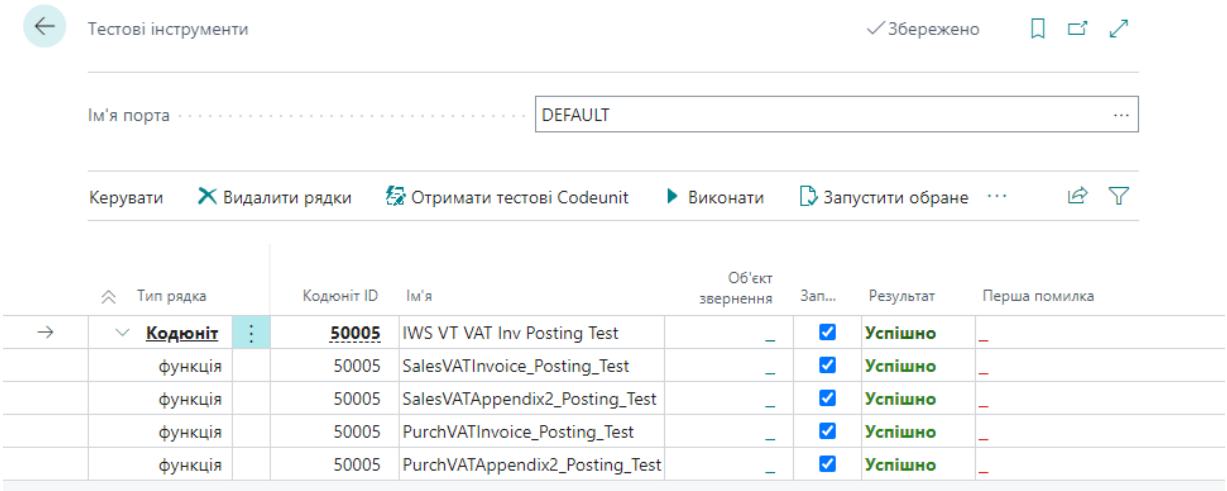


Рисунок 6 – Сторінка «Тестові інструменти»

На сторінці можна вибрати потрібні код’юніти, які містять потрібні функції. Запустити як один або декілька код’юнітів, так і окрему функцію або ж функції. Можна побачити історію результатів певної функції, експортувати результати, а також, за потреби, переглянути стек викликів.

3. РОЗРОБКА УКРАЇНСЬКОЇ ЛОКАЛІЗАЦІЇ ПОДАТКОВОГО ЗАКОНОДАВСТВА В МОДУЛІ «ФІНАНСИ» ТА НАПИСАННЯ АВТОМАТИЧНИХ ТЕСТІВ

3.1. Розробка податкових накладних в рамках локалізації модуля «Фінанси»

Реалізація податкових накладних українського податкового законодавства в системі ВС потребує наступних кроків:

1) Розробка таблиці заголовку податкового документа
2) Розробка таблиці рядків податкового документа
3) Розробка сторінок для відображення податкової накладної (сторінка-частина для відображення рядків документа, сторінка для відображення конкретної ПН, сторінка-список для відображення всіх ПН)
4) Розробка аналогічних таблиць і сторінок для облікованої ПН
5) Розробка процедури обліку ПН, створення відповідних фінансових операцій

3.1.1. Розробка таблиць заголовку і рядків податкової накладної

Розроблено таблицю заголовку ПН на продаж. У таблиці 3 наведений перелік основних полів таблиці, короткий опис полів та функціоналу при валідації поля, за його наявності.

Таблиця 3 – Перелік основних полів таблиці заголовку податкової накладної

ID	Назва поля	Тип	Опис поля/функціоналу
1	Тип податкової накладної	Enum PK	Два варіанти: «Податкова накладна», «Додаток 2»
2	Номер документа	Code, 20 PK	При заповненні поля «Номер документа», очищується поле «Номерна серія».
3	Номерна серія	Code, 20	За номерною серією автоматично буде присвоюватися номер полю «Номер документа» під час створення податкової накладної
4	Зовнішній номер документа	Code, 20	Зовнішній номер документа
5	Дата складання	Date	По суті дата створення
6	Дата реєстрації	Date	Дату реєстрації можна вводити вручну, проте зазвичай податкова накладна реєструється в системі електронного документообігу і звідти отримується дата реєстрації. Обраховується поле «Кількість днів між реєстрацією та випискою»
7	Код клієнта	Code, 20	Підтягуються ярлики аналітики, заповнюється «Назва клієнта», «ІПН», «Реєстраційний номер», «ПДВ бізнес-група», у всіх рядках документа змінюється інформація про клієнта.
8	Назва клієнта	Text, 100	Назва клієнта
9	Оригінальна накладна для Додатку 2	Code, 20	Якщо «Тип ПН» Додаток 2, то вказується номер ПН до якої створювався Додаток 2

ID	Назва поля	Тип	Опис поля/функціоналу
10	ПДВ бізнес-група	Code, 20	У пов’язаних рядках документа змінюється відповідне поле «ПДВ бізнес-група»
11	Тип створення	Enum	Два варіанти: «Створено вручну», «Імпорт»
12	Статус	Enum	Три варіанти: «Відкрито», «Випущено», «Обліковано»
13	Дата створення документа	Date	Дата створення документа
14	Час створення документа	Time	Час створення документа
15	Код користувача	Code, 50	Користувач, що створив документ
16	Всього база ПДВ	Decimal	Це поле FlowField, поле яке автоматично розраховується при отриманні запису з таблиці (фізично не зберігає значення в таблиці). Дорівнює сумі значень (на які нараховується ПДВ) рядків нашого документа
17	Всього сума ПДВ	Decimal	Розраховується як сума значень ПДВ рядків поточного документа
18	Всього сума з ПДВ	Decimal	Сума рядків поточного документа, включаючи ПДВ
19	Реєстрація ПН в ЄРПН	Boolean	Визначає чи зареєстрована податкова накладна в єдиному реєстрі податкових накладних
22	Початкова дата періоду ПДВ	Data	У пов’язаних рядках документа змінюється відповідне поле «Початкова дата періоду ПДВ»

ID	Назва поля	Тип	Опис поля/функціоналу
23	Статус включення у ПДВ період	Enum	Три варіанти: « », «Включено в період ПДВ», «Змінений період ПДВ»
24	Кількість днів між реєстрацією та випискою	Integer	Кількість днів між реєстрацією та випискою
25	ІПН	Code, 20	Індивідуальний податковий номер
27	Відкоригована початкова дата періоду ПДВ	Date	Відкоригована початкова дата періоду ПДВ
29	Дата обліку	Date	Дата, яка буде використовуватися при обліку.
30	Код ЄДРПОУ	Code, 20	Код ЄДРПОУ
31	Тип документа	Enum	Два варіанти: «PNE», «RKE»
32	Кількість разів експортовано	Integer	При експорті документа за допомогою звіта експорту, розробленого в розділі роботи 3.3., збільшується на 1.
33	Джерело податкового номеру	Enum	Шість варіантів: «», «1», «2», «3», «4», «5»
41	Ярлик аналітики 1	Code, 20	Ярлики аналітики використовуються для наскрізного аналізу у всій системі
42	Ярлик аналітики 2	Code, 20	Ярлик аналітики 2
43	Код набору аналітиків	Integer	Створюється новий набір аналітиків, записується код нового набору до документу
46	Тип документа джерела	Enum	П’ять варіантів: «», «Платіж», «Рахунок», «Кредит нота», «Повернення коштів»

ID	Назва поля	Тип	Опис поля/функціоналу
50	Номер документа джерела	Code, 20	Номер документа джерела

Також розроблено допоміжні функції, які потім будуть використовуватися записом таблиці «Заголовок ПН на продаж» (дії пов’язані зі вставленням, видаленням або зміною запису поточної таблиці):

- При видаленні запису заголовка ПН відбувається перевірка, чи існують пов’язані заголовки з типом ПН «Додаток 2», якщо так – помилка і заборона видалення. Перевірка чи зареєстрований ПН в ЄРПН, якщо так – помилка і заборона видалення. При проходженні попередніх перевірок, знайти і видалити усі пов’язані рядки документа.
- При спробі створити запис в таблиці відбувається перевірка поля «Номерна серія» для ПН у відповідній таблиці налаштувань, документу присвоюється автоматично номер, заповнюються поля ким створений документ та час його створення (в тому числі дата складання).
- Реалізований функціонал створення ПН з типом «Додатку 2» до поточної ПН. При створенні ПН «Додаток 2» частина полів поточної ПН переноситься до новоствореного документа, частина полів заповнюється новими значеннями (наприклад, «Дата складання», «Тип ПН», «Дата реєстрації», «Ким створений документ», «Час створення», «Статус» і т.п.). Створюються рядки документа «Додаток 2», які є копією рядків поточної ПН.
- Функціонал валідації ярликів аналітики в системі (оновлення комбінації ярликів, створення нової комбінації ярликів аналітики).

Лістинг коду таблиці «Заголовок ПН» приведено у Додатку А.

Розроблено таблицю рядків ПН на продаж. У таблиці 4 наведений перелік основних полів таблиці, короткий опис полів та функціоналу при валідації поля, за його наявності.

Таблиця 4 – Перелік основних полів таблиці рядків податкової накладної

ID	Назва поля	Тип	Опис поля/функціоналу
1	Тип податкової накладної	Enum PK	Два варіанти: «Податкова накладна», «Додаток 2» При валідації поля заповнюється «Код рядка декларації», який бере своє значення з таблиці налаштувань ПДВ.
2	Номер документа	Code, 20 PK	За цим полем і полем «Тип ПН» утворюється зв’язок заголовка і рядків
3	Номер рядка	Integer PK	Номер рядка
4	Тип	Enum	Доступно 4 опції: « », «Рахунок ГК», «Товар», «Основний засіб»
5	Номер	Code, 20	В залежності від поля «Тип» підвантажується список чи то товару, чи то рахунків ГК, чи то основних засобів
6	Опис	Text, 250	Опис чи то товару, чи то рахунка ГК, чи то основного засобу
7	Одиниця виміру	Code, 10	При валідації також заповнюється «Опис одиниці виміру»
8	Опис одиниці виміру	Text, 50	Опис одиниці виміру
9	ПДВ товарна група	Code, 20	ПДВ товарна група

ID	Назва поля	Тип	Опис поля/функціоналу
10	Фінансовий рахунок витрат	Code, 20	Фінансовий рахунок витрат
11	Кількість	Decimal	При зміні кількості відбувається перерахунок усіх числових значень рядка
12	Ціна без ПДВ	Decimal	Ціна без ПДВ
13	База оподаткування	Decimal	База оподаткування
14	Сума ПДВ	Decimal	Сума ПДВ
15	Сума з ПДВ	Decimal	Сума з ПДВ
16	Статус	Enum	Три варіанти: «Відкрито», «Випущено», «Обліковано»
18	Код ставки ПДВ	Enum	Шість опцій: « », «20», «7», «901», «902», «903»
22	Код ДКПП	Code, 20	Державний класифікатор продуктів і послуг
23	Код УКТ ЗЕД	Code, 20	Українська класифікація товарів зовнішньоекономічної діяльності
31	Код рядка декларації	Integer	Код рядка декларації
41	Ярлик аналітики 1	Code, 20	Ярлики аналітики використовуються для наскрізного аналізу у всій системі
42	Ярлик аналітики 2	Code, 20	Ярлик аналітики 2
43	Код набору аналітиків	Integer	Створюється новий набір аналітиків, записується код нового набору до документу

Розроблено додатковий функціонал пов’язаний з записами таблиці «Рядки ПН»:

- При спробі створити запис в таблиці відбувається перевірка чи заповнені поля «Номер клієнта» та «Дата складання заголовку документа». Відбувається ініціалізація запису, заповнюється інформація про клієнта, «Початкова дата періоду ПДВ».
- Реалізовано заповнення деяких полів з таблиці налаштування ПДВ.
- Реалізовано розрахунок числових значень рядка, при зміні певних числових значень.
- Реалізовані функції роботи з ярликами аналітиками.

Для ПН на покупку набір полів майже ідентичний (наприклад, відмінним є поле «Код постачальника» замість «Код клієнта»). Таблиці облікованих документів ПН мають ті ж самі поля, що і відповідні їм таблиці необлікованих даних. Таблиці облікованих даних не містять ніякого спеціального функціоналу при валідації полів новими значеннями. Це зумовлено тим, що обліковані таблиці заповнюються тільки під час процедури обліку, зазвичай функцією TransferFields, яка копіює значення полів які мають однаковий тип і ID поля в цільовій таблиці і таблиці джерела.

3.1.2. Створення сторінок для відображення даних таблиці податкових накладних

Для відображення даних таблиць заголовку і рядків ПН було розроблено три види сторінок для кожного типу ПН (необлікована ПН на продаж, облікована ПН на продаж, необлікована ПН на покупку, облікована ПН на покупку): сторінка-частина для відображення рядків документа (рис. 7); сторінка-список для відображення всіх ПН, яка є нередагованою тому, що всі зміни відбуваються на

сторінці, яка відображає поточну ПН (рис. 8); сторінка для відображення конкретної ПН (рис. 9). Усі сторінки мають схожий вигляд і функціонал, незалежно від типу ПН, тому для зручності продемонстровано на прикладі ПН на продаж. Лістинг коду сторінки «Список ПН на продаж» приведено у Додатку Б.

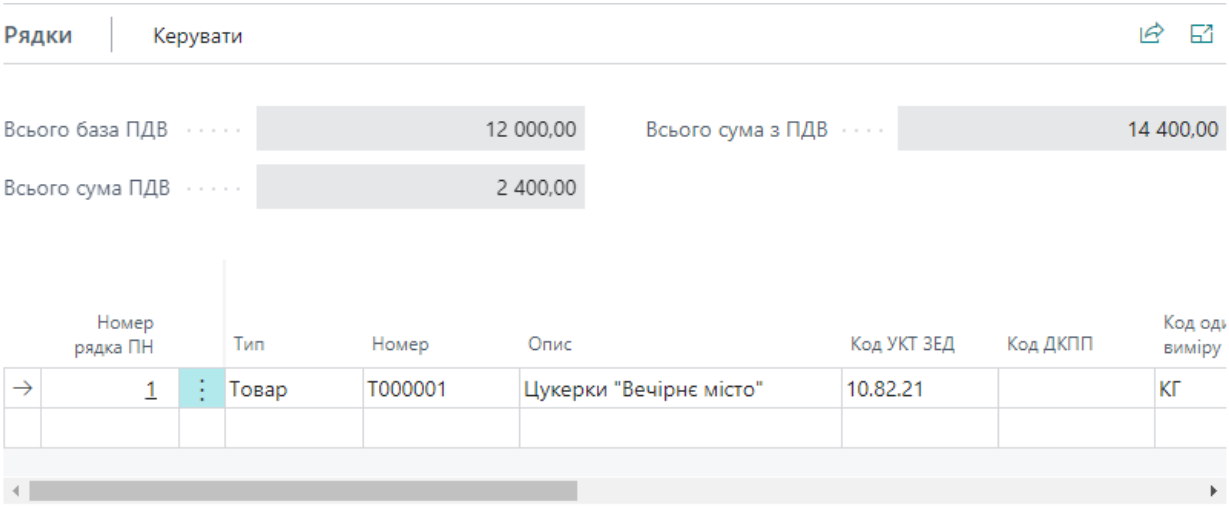


Рисунок 7 – Сторінка-частина для відображення таблиці рядків ПН

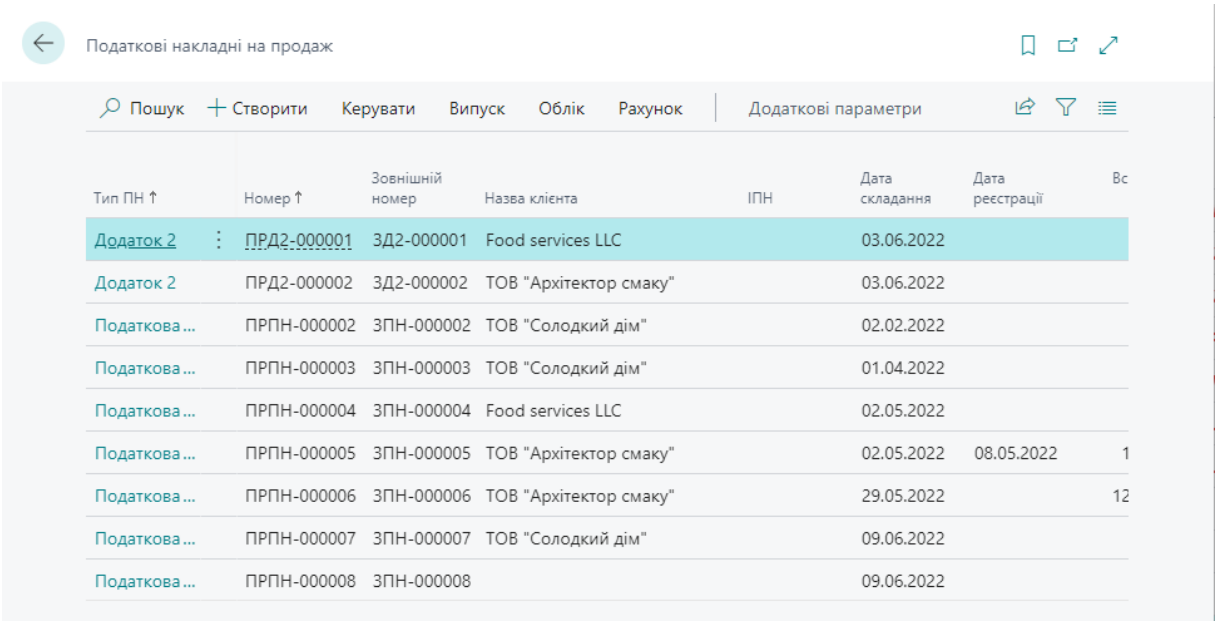


Рисунок 8 – Сторінка-список ПН

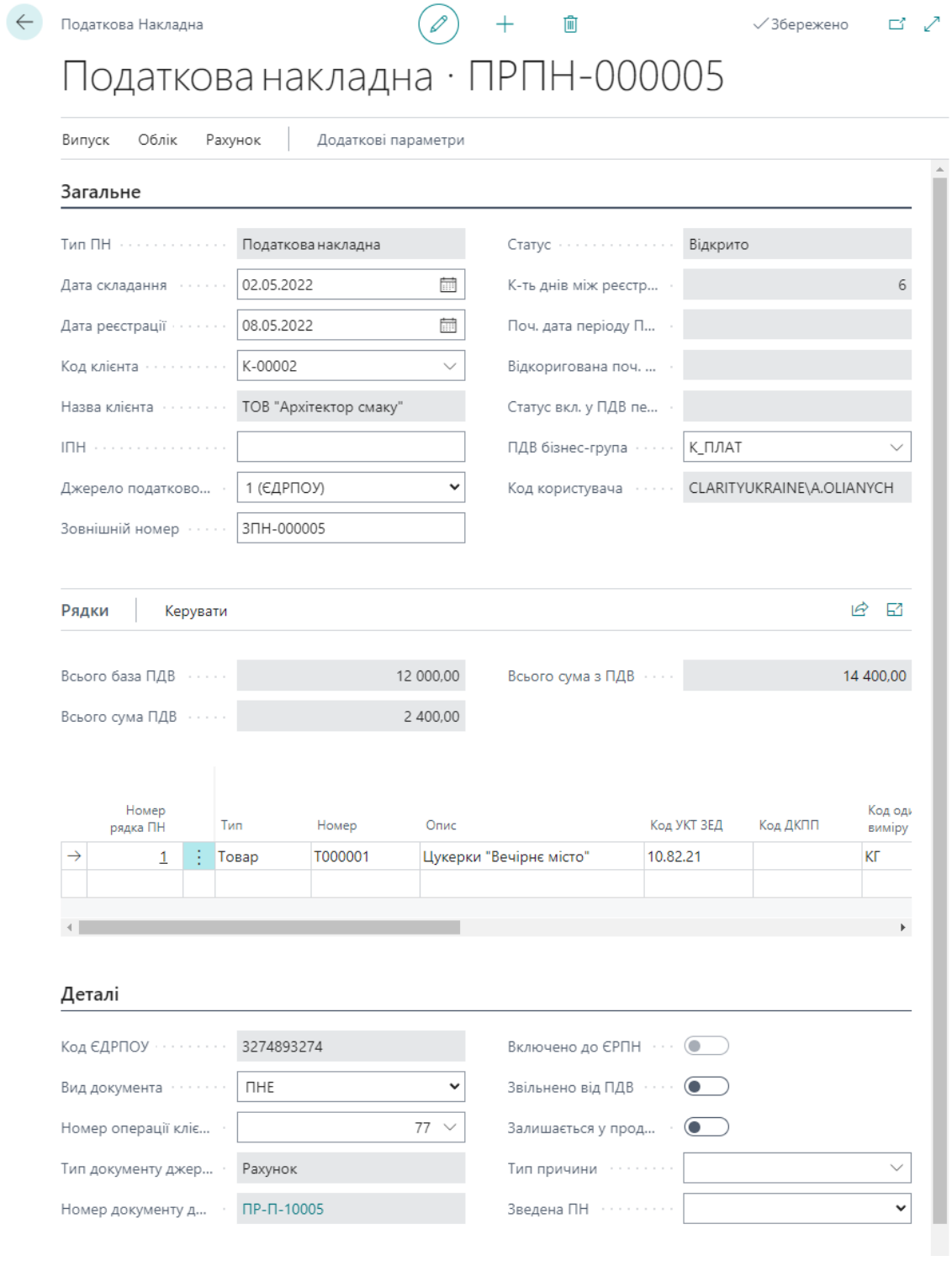


Рисунок 9 – Сторінка для відображення конкретної ПН

Також розроблено функціональне меню у верхніх частинах сторінок. У таблиці 5 приведено основні функції і їх короткий опис.

Таблиця 5 – Перелік та опис функцій меню сторінок ПН

Назва	**Опис**
Змінити	Відкриває документ, всі поля доступні для редагування (якщо виконуються усі умови доступності редагування, або ж такі умови відсутні)
Переглянути	Відкриває документ, всі поля недоступні для редагування
Видалити	Видаляє документ, після усіх вдалих перевірок, які наявні в коді для процедури видалення цього типу документа
Випустити	Змінює поле «Статус» на опцію «Випущено». Поля стають недоступними для редагування (навіть якщо відкрити документ за допомогою кнопки «Змінити»). Ця опція обирається, коли документ готовий до процедури обліку
Відкрити	Змінює поле «Статус» на опцію «Відкрито». Поля стають доступними для редагування. Ця опція обирається за замовчуванням, а також коли документ потрібно змінити перед процедурою обліку
Облік	Запускає процедуру обліку поточного документа
Попередній облік	Запускає процедуру попереднього обліку, використовується для перевірки значень, які будуть отримані після обліку документа. Після попереднього перегляду усі транзакції будуть відмінені
Пакетний облік	Запускає облік, де можна виставити фільтри для обліку одразу певної кількості документів, які підпадають під фільтри
Аналітики	Відкриває сторінку аналітиків, для перегляду усіх пов’язаних аналітиків з даним документом.
Переформувати рядки ПН (доступно тільки	Для випадків «Першої події», коли наявна передплата, у фінансовому журналі створюється відповідний рядок для відповідного клієнта. Запускається процедура обліку цього рядка, під час якої створюється ПН на продаж. Ця ПН на

Назва	Опис
на сторінці поточного ПН)	продаж має тільки один рядок, в якому вказаний не товар, чи рахунок ГК, чи послуга, а те, що це передплата. Далі ця передплата може використовуватися для оформлення замовлення на продаж певних товарів. В цьому випадку, коли створений документ на продаж, при обліку цього документа не відбувається створення податкової накладної (якщо сума облікованих товарів не перебільшує суму передплати) так, як вже наявна в системі передплата. В цьому випадку потрібно зайти на податкову накладну, яка була створена при обліку рядка з фінансового журналу, та натиснути цю кнопку («Переформувати рядки ПН»). В цьому випадку будуть створені рядки в поточній ПН з товарами на суму, яка дорівнює сумі передплати, замість рядка з передплатою.
Змінити ПДВ період	Якщо податкова накладна має визначений період ПДВ, його можна змінити за допомогою цієї кнопки

Сторінки облікованих ПН повністю нередаговані. Меню має додатковий функціонал, представлений у таблиці 6.

Таблиця 6 – Перелік та опис функцій меню сторінок облікованих ПН

Назва	Опис
Скасувати	Відміняє обліковану податкову накладну
Навігатор	Запускає процес пошуку усіх транзакцій у всіх журналах і таблицях системи, які пов’язані з поточним документом
Створити ПН з типом «Додаток 2»	Створює ПН з опцією «Додаток2» в полі «Тип ПН». Цей функціонал використовується якщо у ПН була знайдена помилка, або ж потрібно змінити щось у вже облікованій ПН

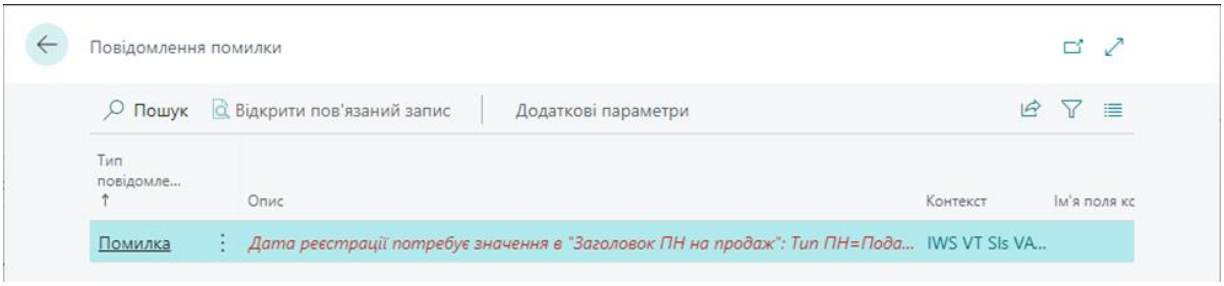
3.1.3. Розробка процесу обліку податкових накладних

Процедура обліку передбачає собою створення усіх потрібних транзакцій в системі, а також облікованої версії ПН. Лістинг коду функціональності «Облік ПН» приведено у Додатку В.

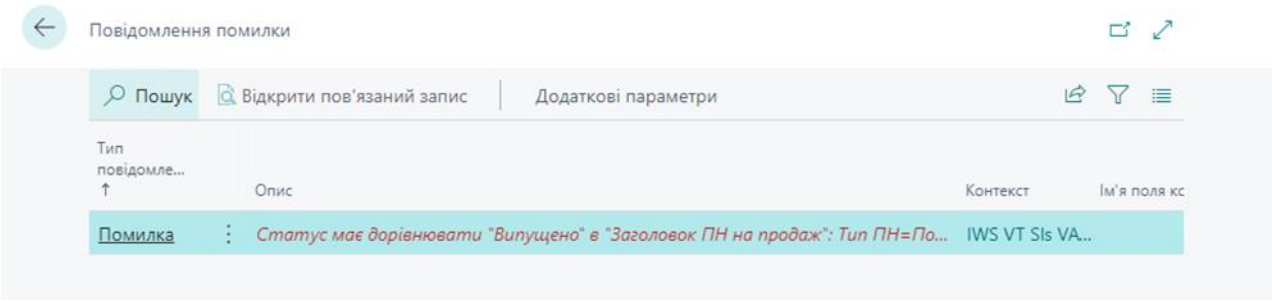
Процедура обліку, яка розроблена має певний перелік перевірок, які треба пройти при обліку, а саме:

- Перевірка, чи заповнена «Дати складання»,
- Перевірка, чи заповнена «Дата реєстрації»,
- Перевірка поля «Статус», що документ випущено,
- Якщо це Додаток 2, перевіряється, чи обліковано вже документ, до якого він є додатком,
- Вираховується, чи дозволено обліковувати податкову накладну в цей період. Для цього перевіряється, чи закритий період ПДВ, в якому намагається провестись облік документа,
- Перевіряється, чи існують рядки (чи є що обліковувати),
- Інші менш значні перевірки.

При наявності будь-якої помилки при обліку і попередньому обліку система сповістить про це (рис. 10).



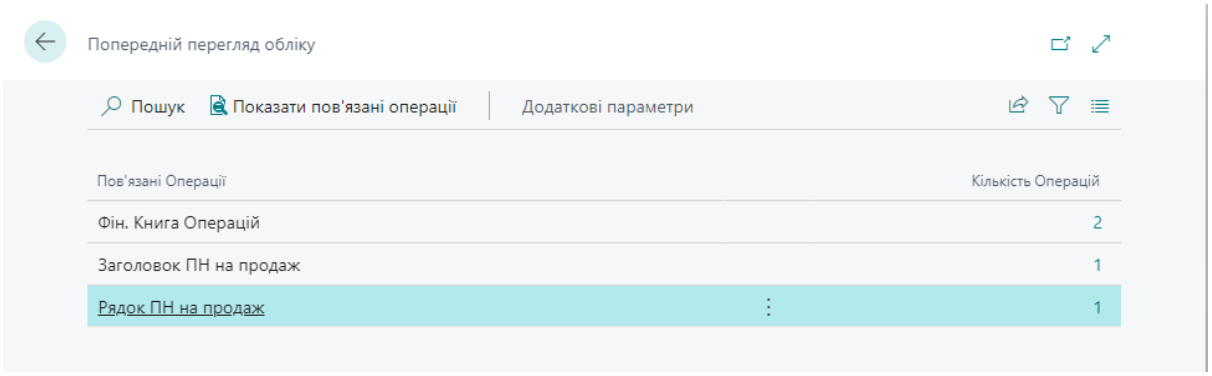
а)



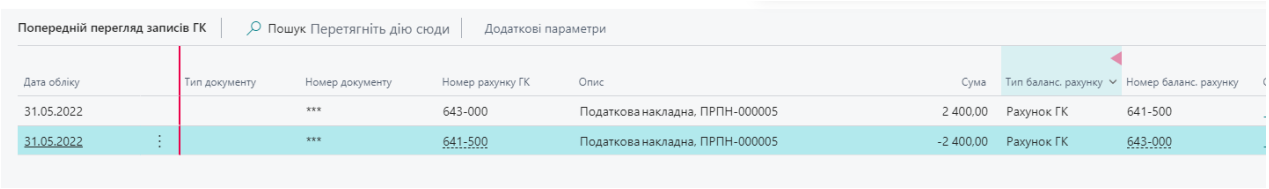
б)

Рисунок 10 – Повідомлення про помилку при попередньому обліку: а) – незаповнене поле «Дата реєстрації»; б) – статус «Відкрито» замість «Випущено»

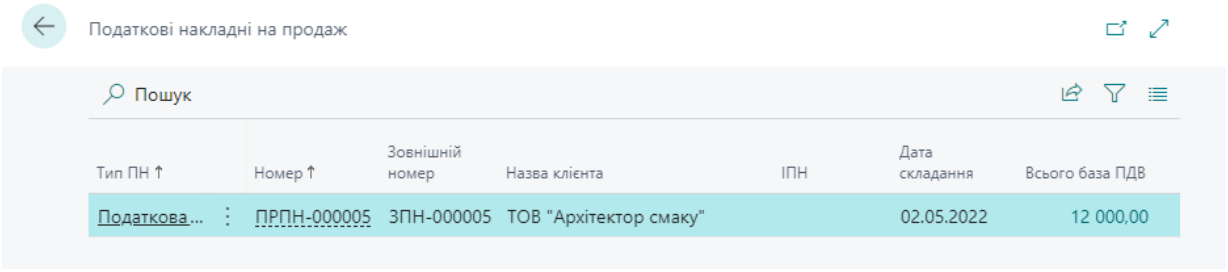
Після проходження усіх перевірок створюється ПН та відповідні транзакції (рис. 11) в системі BC. Їх значення залежать від значень ПН, а також значень таблиць з налаштуванням. Наприклад, з «Налаштування ПДВ» беруться рахунки, які прийматимуть участь у транзакції, в залежності від облікових груп ПДВ у поточному документі.



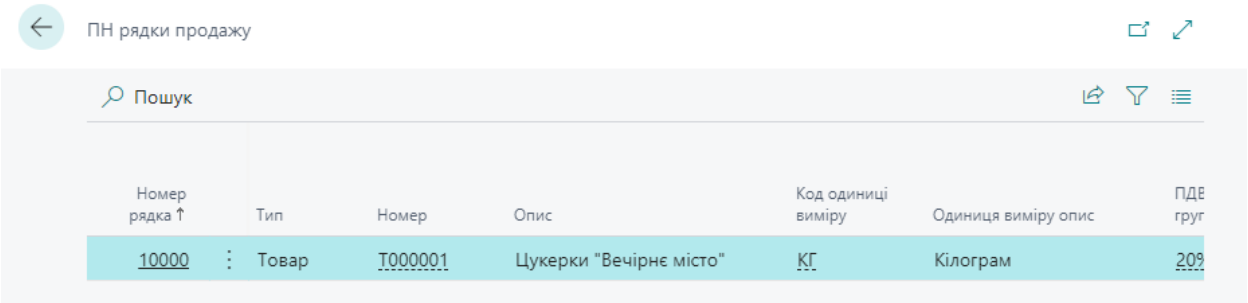
а)



б)



в)



г)

Рисунок 11 – Результати процедури обліку ПН: а) – створені операції під час обліку; б) – створені операції у фінансовій книзі операцій; в) – створений заголовок облікованої ПН; г) – створений рядок облікованої

В результаті розроблено повністю робочий процес обліку з певними перевірками, які не дають змогу провести обліковий процес з помилками у даних, та створенням усіх відповідних операцій та записів.

3.2. Розробка автоматичних тестів

Розроблено чотири автотести для перевірки процесу обліку ПН накладних на продаж і покупку двох типів («Податкова накладна» та «Додаток 2»). Автотести працюють незалежно один від одного та не залежать від завчасно зроблених налаштувань системи. У таблиці 7 наведено перелік автотестів з описом їх принципу роботи.

Лістинг коду розроблених автотестів приведено у Додатку Г.

Таблиця 7 – Перелік розроблених автотестів з принципом їх роботи

Назва	**Принцип роботи**
SalesVATInvoice_Posting_Test Перевірка обліку ПН на продаж з типом «Податкова накладна»	На початку автотеста створюються та заповнюються усі потрібні таблиці налаштувань (наприклад, таблиця «Налаштування ПДВ», «ПДВ Журнал налаштування»). Створюється клієнт, товар, відповідні облікові групи для них. Створюються рахунки ГК (для можливості проведення фінансових транзакцій). Після закінчення усіх налаштувань створюється ПН на продаж, проводиться процедура обліку за допомогою функціоналу обліку,

Назва	Принцип роботи
	який був розроблений раніше. Далі перевіряються значення у фінансовій книзі операцій на відповідність сум, а також рахунків, які приймають участь у транзакції (ці рахунки повинні братися системою з таблиці «Налаштування ПДВ», з матриці значень, опираючись на значення ПДВ бізнес-груп поточного документа)
SalesVATAppendix2_Posting_Test Перевірка обліку ПН на продаж з типом «Додаток 2»	На початку автотеста створюються та заповнюються усі потрібні таблиці налаштувань (наприклад, таблиця «Налаштування ПДВ», «ПДВ Журнал налаштування»). Створюється клієнт, товар, відповідні облікові групи для них. Створюються рахунки ГК (для можливості проведення фінансових транзакцій). Після закінчення усіх налаштувань створюється ПН на продаж, проводиться процедура обліку за допомогою функціоналу обліку, який був розроблений раніше. Для облікованої версії ПН створюється ПН з типом «Додаток 2» і обліковується. Далі перевіряються значення у фінансовій книзі операцій на відповідність сум, а також рахунків, які приймають участь у транзакції (ці рахунки повинні братися системою з таблиці «Налаштування ПДВ», з матриці

Назва	Принцип роботи
	значень, опираючись на значення ПДВ бізнес-груп поточного документа)
PurchVATInvoice_Posting_Test Перевірка обліку ПН на покупку з типом «Податкова накладна»	Принцип роботи аналогічний перевірці обліку ПН на продаж з типом «Податкова накладна», проте використовується ПН на покупку і створюється постачальник замість клієнта.
PurchVATAppendix2_Posting_Test Перевірка обліку ПН на покупку з типом «Додаток 2»	Принцип роботи аналогічний перевірці обліку ПН на продаж з типом «Додаток 2», проте використовується ПН на покупку і створюється постачальник замість клієнта.

Результати розробки можна перевірити на сторінці «Тестові інструменти», вибравши код'юніт 50005, який містить усі вище згадані автотести (рис. 12). Усі автотести були успішно пройдені, а значить розробка процедури обліку працює коректно.

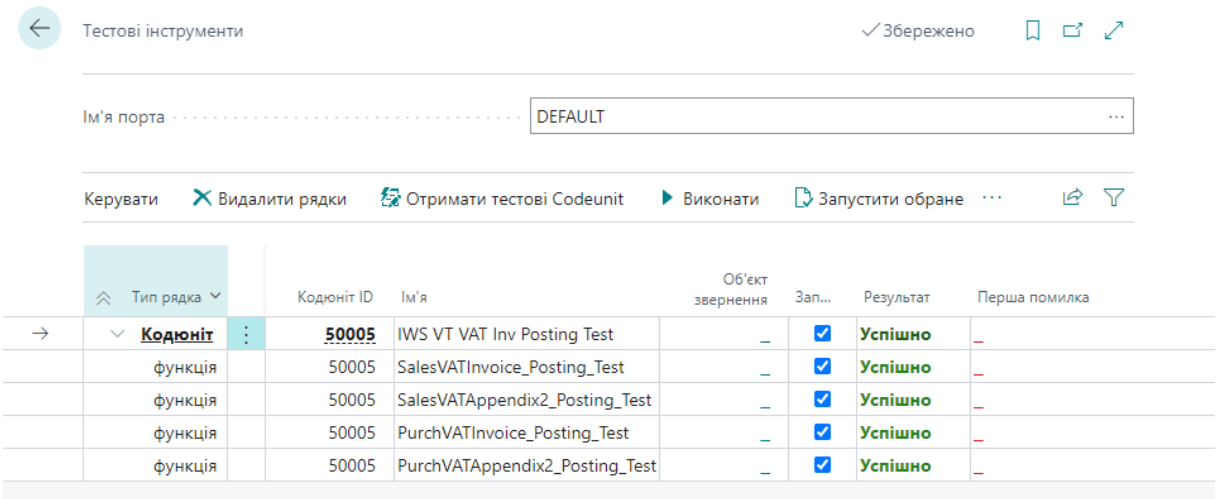


Рисунок 12 – Сторінка «Тестові інструменти» з успішно пройденими автотестами

3.3. Розробка звітів для імпорту та експорту податкових накладних

Розроблено два звіти для роботи з ПН: звіт для експорту ПН у МЕДОК, звіт для імпорту ПН з системи МЕДОК.

Документ який імпортується в МЕДОК або експортується з МЕДОК (після чого імпортується в ВС) має формат файлу XML. У таблиці 8 представлений опис XML-структури ПН для імпорту у МЕДОК. [10][11]

Таблиця 8 – Опис XML-структури ПН для імпорту в МЕДОК

Назва поля	**Зміст**
TIN	ІПН компанії
C_DOC	Тип форми Додаток 2
C_DOC_SUB	Субтип форми Додаток 2
C_DOC_VER	Версія форми Додаток 2
C_REG	Код регіону (податкова інспекція)
C_RAJ	Код адміністративного району (податкова інспекція)
PERIOD_MONTH	Місяць з поля «Дата складання»
PERIOD_YEAR	Рік з поля «Дата складання»
C_STI_ORIG	Код регіону з кодом адміністративного району (податкова інспекція)
C_DOC_STAN	
HTYPR	Зазначається відповідний тип причини
HFILL	Дата виписки ПН
HNUM	Порядковий номер ПН
HNAMESEL	Постачальник (продавець) (найменування)
HNAMEBUY	Отримувач (покупець) (найменування)
HKSEL	ІПН підприємства
HNUM2	Числовий номер філії продавця
HTINSEL	Податковий номер платника податку або серія та/або номер паспорта (постачальник)
HKBUY	ІПН покупця
HFBUY	Код філії покупця
HTINBUY	Податковий номер платника податку або серія та/або номер паспорта (покупець)
HKS	Код ознаки джерела податкового номера (продавця)
HKB	Код ознаки джерела податкового номера (покупця)

Назва поля	Зміст
HBOS	Посадова (уповноважена) особа/фізична особа (законний представник)
HKBOS	Реєстраційний номер облікової картки платника податку
RXXXXG3S	Номенклатура поставки товарів
RXXXXG4	Код товару згідно з УКТ ЗЕД
RXXXXG33	Послуги згідно з ДКПП
RXXXXG4S	Одиниця виміру товару/послуги (умовне позначення (українське)
RXXXXG105_2S	Одиниця виміру товару/послуги (код)
RXXXXG008	Код ставки
RXXXXG009	Код пільги
RXXXXG5	Кількість
RXXXXG6	Ціна постачання одиниці товару\послуги
RXXXXG010	Обсяги постачання (база оподаткування) без урахування податку на додану вартість
RXXXXG11_10	Сума податку на додану вартість
R04G11	Загальна сума коштів, що підлягають сплаті з урахуванням податку на додану вартість
R03G11	Загальна сума податку на додану вартість, у тому числі:
R03G7	Загальна сума податку на додану вартість за основною ставкою

Звіт експорту ПН має зручну сторінку запиту (рис. 13).

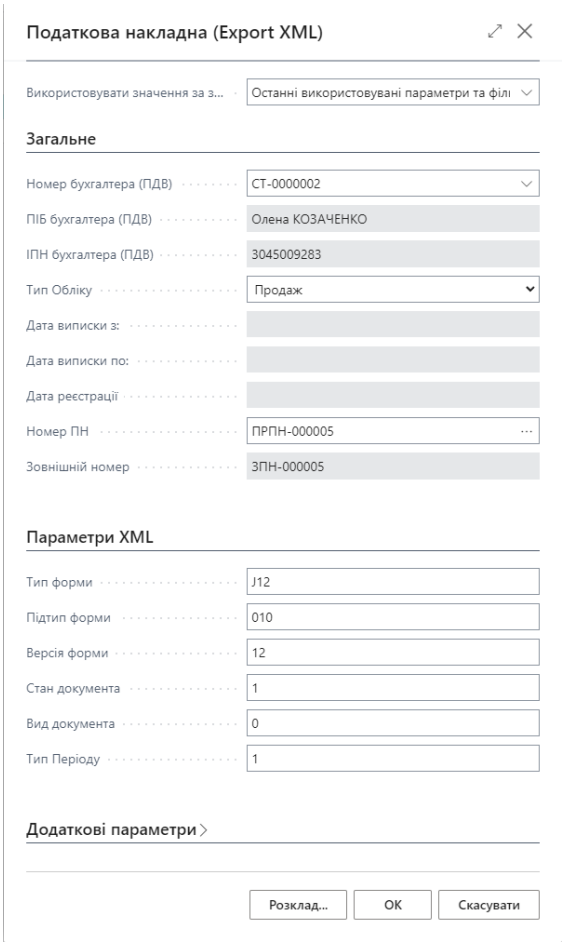


Рисунок 13 – Сторінка запиту звіта експорту ПН

Вказавши усі необхідні фільтри, у звіті спочатку знаходиться документ по фільтрам. Якщо такий документ знайшовся відбувається його перевірка: чи поле «Статус» має опцію «Випущено», чи заповнені усі дати, чи взагалі є що експортувати (можуть бути відсутні рядки ПН), чи заповнені дані в документі для формування правильного XML-файлу. Після усіх потрібних перевірок, формується файл у форматі XML (рис. 14).

```
<?xml version="1.0" encoding="UTF-8"?>
<DECLAR noNamespaceSchemaLocation="J1201010.XSD" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  - <DECLARHEAD>
        <TIN>487329857329</TIN>
        <C_DOC>J12</C_DOC>
        <C_DOC_SUB>010</C_DOC_SUB>
        <C_DOC_VER>12</C_DOC_VER>
        <C_DOC_TYPE>0</C_DOC_TYPE>
        <C_REG>1</C_REG>
        <C_RAJ>1</C_RAJ>
        <PERIOD_MONTH>5</PERIOD_MONTH>
        <PERIOD_TYPE>1</PERIOD_TYPE>
        <PERIOD_YEAR>2022</PERIOD_YEAR>
        <C_STI_ORIG>101</C_STI_ORIG>
        <C_DOC_STAN>1</C_DOC_STAN>
        <D_FILL/>
    </DECLARHEAD>
  - <DECLARBODY>
        <R01G1> </R01G1>
        <HTYPR/>
        <HFILL>02052022</HFILL>
        <HNUM>ЗПН-000005</HNUM>
        <HNAMESEL>CRONUS DEV UA</HNAMESEL>
        <HNAMEBUY>ТОВ "Архітектор смаку"</HNAMEBUY>
        <HKSEL>4128479821</HKSEL>
        <HNUM2/>
        <HTINSEL>487329857329</HTINSEL>
        <HKBUY/>
        <HFBUY/>
        <HTINBUY>3274893274</HTINBUY>
        <HKS>1 (ЄДРПОУ)</HKS>
        <HKB>1 (ЄДРПОУ)</HKB>
        <HBOS>Олена КОЗАЧЕНКО</HBOS>
        <HKBOS>3045009283</HKBOS>
        <RXXXXG3S ROWNUM="1">Цукерки "Вечірнє місто"</RXXXXG3S>
        <RXXXXG4 ROWNUM="1">10.82.21</RXXXXG4>
        <RXXXXG33 ROWNUM="1"/>
        <RXXXXG4S ROWNUM="1">КГ</RXXXXG4S>
        <RXXXXG105_2S ROWNUM="1">КГ</RXXXXG105_2S>
        <RXXXXG008 ROWNUM="1">20</RXXXXG008>
        <RXXXXG009 ROWNUM="1"/>
        <RXXXXG5 ROWNUM="1">100</RXXXXG5>
        <RXXXXG6 ROWNUM="1">120</RXXXXG6>
        <RXXXXG010 ROWNUM="1">12000</RXXXXG010>
        <RXXXXG11_10 ROWNUM="1">2400</RXXXXG11_10>
        <R04G11>14400</R04G11>
        <R03G11>2400</R03G11>
        <R03G7>2400</R03G7>
        <R01G7>12000</R01G7>
    </DECLARBODY>
</DECLAR>
```

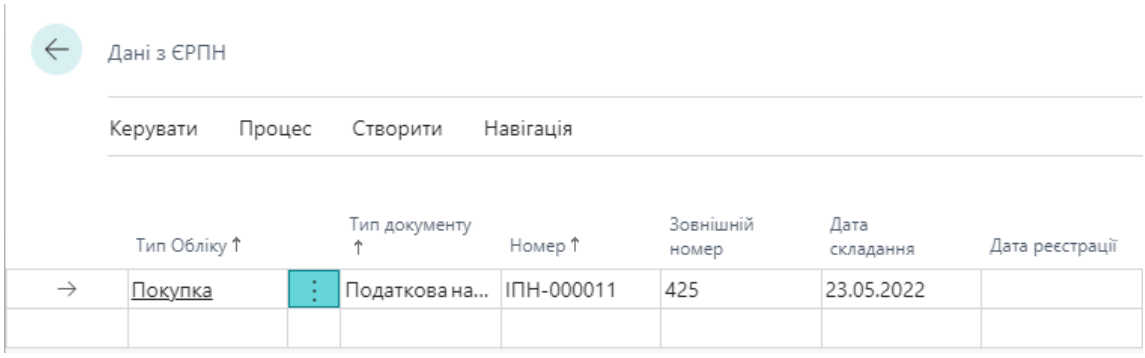

Рисунок 14 – Сформований XML файл для імпорту в МЕДОК

Готовий файл можна завантажувати у МЕДОК для подальшої реєстрації податкової накладної. Лістинг коду звіту «Імпорт ПН» приведено у Додатку Д.

Щодо звіту імпорту, то він доступний зі сторінки «Дані з ЄРПН» у функціональному меню з назвою «Імпортувати XML». Цей звіт працює в зворотному порядку. Спочатку завантажується XML-документ. Далі відбувається перевірка на коректність заповнення цього XML-файлу (у випадку некоректної структури XML-файлу імпорт виконаний не буде). Після усіх необхідний перевірок формується рядок на сторінці «Дані з ЄРПН». Результатом імпорту XML-файла (рис. 15) є створений рядок на сторінці «Дані з ЄРПН» (рис.16).

```
<?xml version="1.0" encoding="UTF-8"?>
<DECLAR noNamespaceSchemaLocation="J1201010.XSD" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <DECLARHEAD>
    <TIN>48732985</TIN>
    <C_DOC>J12</C_DOC>
    <C_DOC_SUB>010</C_DOC_SUB>
    <C_DOC_VER>12</C_DOC_VER>
    <C_DOC_TYPE>0</C_DOC_TYPE>
    <C_REG>1</C_REG>
    <C_RAJ>1</C_RAJ>
    <PERIOD_MONTH>5</PERIOD_MONTH>
    <PERIOD_TYPE>1</PERIOD_TYPE>
    <PERIOD_YEAR>2022</PERIOD_YEAR>
    <C_STI_ORIG>101</C_STI_ORIG>
    <C_DOC_STAN>1</C_DOC_STAN>
    <D_FILL/>
  </DECLARHEAD>
  <DECLARBODY>
    <R01G1> </R01G1>
    <HFILL>23052022</HFILL>
    <HNUM>425</HNUM>
    <HNUM1/>
    <HNAMESEL>ТОВ "Бест Канцелярія"</HNAMESEL>
    <HNAMEBUY>Кронус Україна</HNAMEBUY>
    <HKSEL>436655003421</HKSEL>
    <HNUM2/>
    <HTINSEL>43665500</HTINSEL>
    <HKBUY>487329857356</HKBUY>
    <HFBUY/>
    <HTINBUY>48732985</HTINBUY>
    <HKS>1</HKS>
    <HKB>1</HKB>
    <HBOS>Олена КОЗАЧЕНКО</HBOS>
    <HKBOS>3045009283</HKBOS>
    <RXXXXG3S ROWNUM="1">Папір А4</RXXXXG3S>
    <RXXXXG4 ROWNUM="1">4802</RXXXXG4>
    <RXXXXG33 ROWNUM="1"/>
    <RXXXXG4S ROWNUM="1"/>
    <RXXXXG105_2S ROWNUM="1"/>
    <RXXXXG008 ROWNUM="1">20</RXXXXG008>
    <RXXXXG009 ROWNUM="1"/>
    <RXXXXG5 ROWNUM="1">20</RXXXXG5>
    <RXXXXG6 ROWNUM="1">129</RXXXXG6>
    <RXXXXG010 ROWNUM="1">2580</RXXXXG010>
    <RXXXXG11_10 ROWNUM="1">516</RXXXXG11_10>
    <R04G11>3096</R04G11>
    <R03G11>516</R03G11>
    <R03G7>516</R03G7>
    <R01G7>2580</R01G7>
  </DECLARBODY>
</DECLAR>
```

Рисунок 15 – XML-файл, який використовувався для імпорту в систему BC



а)

Рисунок 16 – Результат імпорту XML-файлу, створений запис у таблиці «Дані з ЄРПН»: а) – перша частина створеного рядка; б) – друга частина створеного рядка

Тип Контрагента	Код Контрагента	Найменування Контрагента	ІПН	Код ЄДРПОУ	Всього база ПДВ	Всього сума ПДВ	Загальна Сума з ПДВ
Постачальник	VEN-000002	ТОВ "Бест Канцелярія"	436655003421	43665500	2 580,00	516,00	3 096,00

б)

Рисунок 16 (продовження) – Результат імпорту XML-файлу, створений запис у таблиці «Дані з ЄРПН»: а) – перша частина створеного рядка; б) – друга частина створеного рядка

Надалі через функціональну кнопку «Створити ПН на Купівлю» (рис. 17) можна сформувати ПН на покупку.

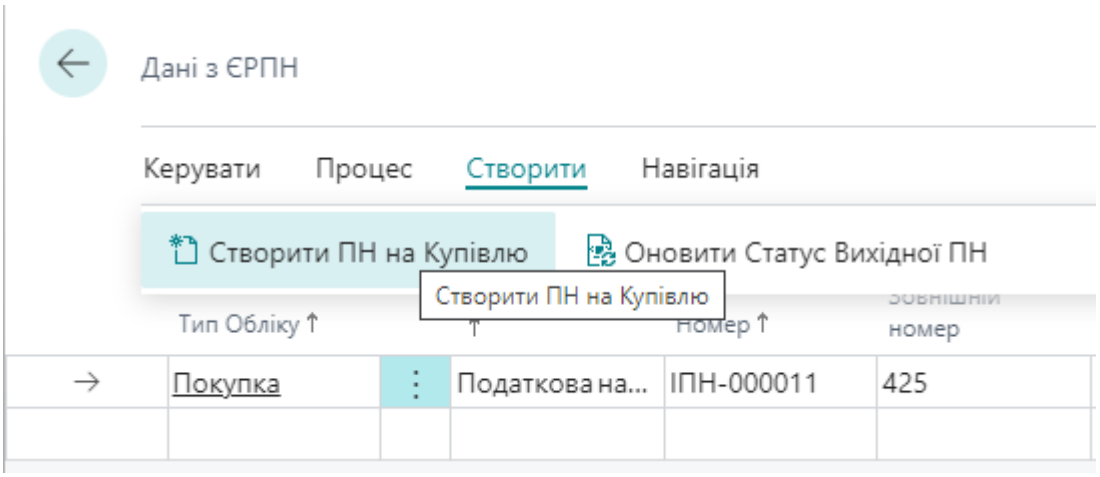


Рисунок 17 – Функціональне меню «Створити ПН на Купівлю»

Попередньо заповнивши дату реєстрації у рядку «Дані з ЄРПН», натискаючи на функціональне меню «Створити ПН на Купівлю», створюється ПН на купівлю (рис. 18).

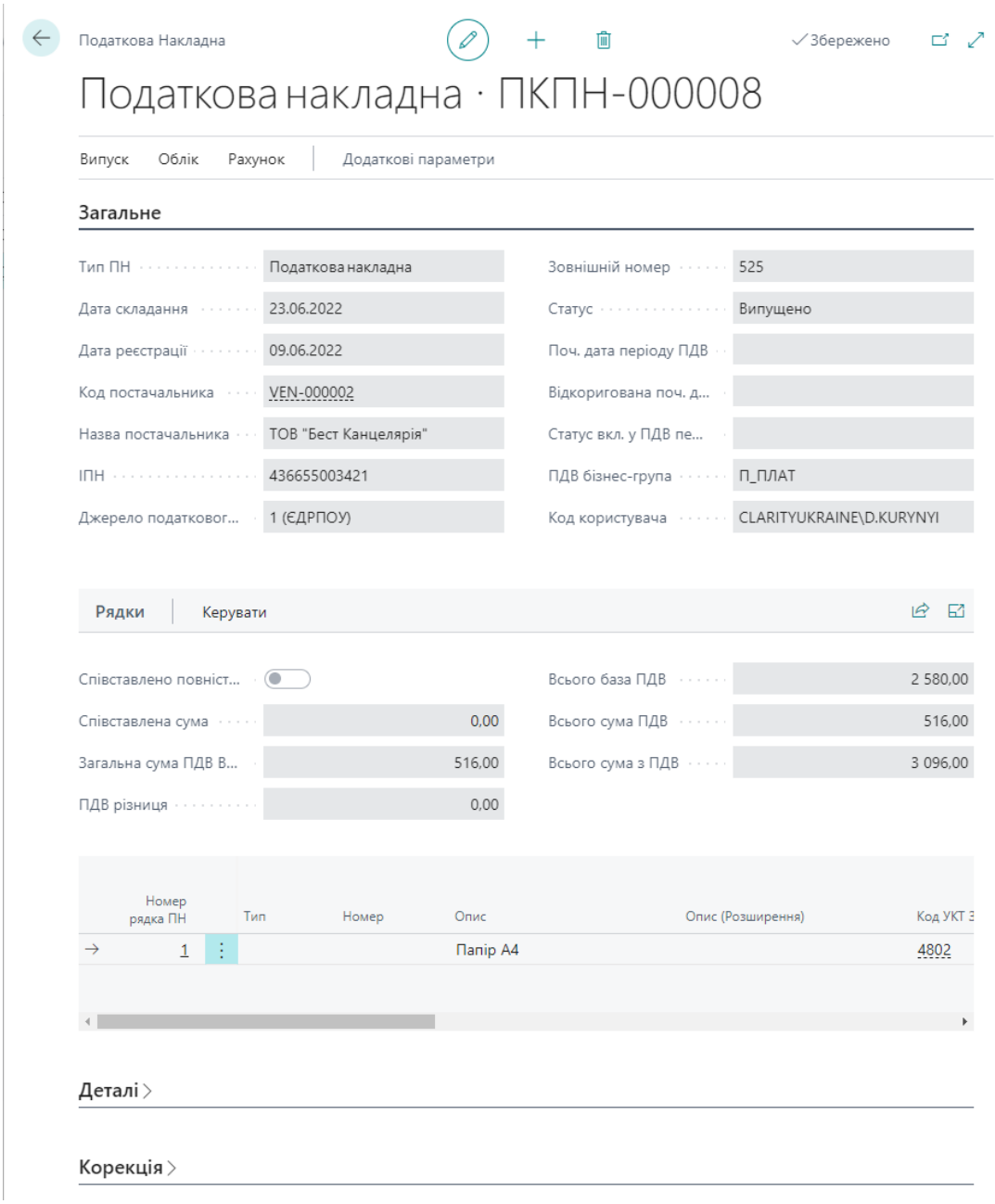


Рисунок 18 – Податкова накладна на купівлю, імпортована з XML-файлу

ВИСНОВКИ

У ході виконання дипломної роботи було:

- Проаналізовано основні модулі системи Microsoft Dynamics 365 Business Central, які мають відношення до формування ПН.
- Описано основний функціонал цих модулів.
- Оглянуто актуальність і доцільність впровадження ERP-систем в компанії.
- Розроблено структури податкових накладних на продаж та купівлю. Впроваджено процедуру обліку податкових накладних.
- Реалізовано зручний імпорт податкових накладних на купівлю, та експорт податкових накладних на продаж.
- Розроблено чотири автотести для перевірки коректної роботи створеного функціоналу після великого оновлення системи або ж якоїсь доробки.

ПЕРЕЛІК ПОСИЛАНЬ ТА ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Навіщо потрібна ERP-система малому бізнесу [Електронний ресурс]

https://inteltech.com.ua/ru/blogs/zachem-nuzhna-erp-sistema-v-malom-biznese

2. Офіційний сайт Microsoft Dynamics 365 Business Central [Електронний ресурс]

https://www.microsoft.com/en-us/download/details.aspx?id=102915

3. Документація по Microsoft Dynamics 365 Business Central [Електронний ресурс]

https://docs.microsoft.com/en-us/dynamics365/business-central/

4. Варіанти розгортань системи BC, їх переваги та недоліки [Електронний ресурс]

https://www.dynamicssquare.com/microsoft-dynamics-365-cloud-vs-on-premise/

5. Курс «Основи MS Dynamics 365 BS» [Електронний ресурс]

https://docs.microsoft.com/en-us/learn/paths/get-started-dynamics-365-business-central/

6. Курс «Налаштування ПДВ в MS Dynamics 365 BS» [Електронний ресурс]

https://docs.microsoft.com/en-us/learn/modules/set-up-vat-dynamics-365-business-central/

7. Офіційний сайт Visual Studio Code [Електронний ресурс]

https://code.visualstudio.com/

8. Rakesh Raul. Microsoft Dynamics NAV 7 Programming Cookbook, – 2013 p., – 312 c. [Посібник]

9. Luc van Vugt. Automated Testing in Microsoft Dynamics 365 Business Central, – 2019 p., – 208 c. [Посібник]
10. Податкова накладна: шпаргалка для заповнення [Електронний ресурс]

https://ibuhgalter.net/articles/686

11. Опис структури XML-файлу податкової накладної для імпорту в МЕДОК [Електронний ресурс]

https://medoc.ua/files/others/032021/ef1b8b2801ec55f56033674c7c4aae16.pdf

ДОДАТКИ

Додаток А

Лістинг коду таблиці «Заголовок ПН».

```
table 21038250 "KDA Sls VAT Invoice Header"
{
    Caption = 'Sales VAT Invoice Header';
    LookupPageId = "KDA Sales VAT Invoice List";
    DrillDownPageId = "KDA Sales VAT Invoice";

    fields
    {
        field(1; "VAT Invoice Type"; Enum "KDA VAT Invoice Type")
        {
            Caption = 'VAT Invoice Type';
            DataClassification = CustomerContent;
            // OptionCaption = 'VAT Invoice,Appendix2';
            // OptionMembers = Invoice,Appendix2;
        }
        field(2; "No."; Code[20])
        {
            Caption = 'No.';
            DataClassification = CustomerContent;
            TableRelation = "KDA Sls VAT Invoice Header"."No." WHERE("VAT
Invoice Type" = FIELD("VAT Invoice Type"));
            ValidateTableRelation = false;

            trigger OnValidate()
            var
                NoSeriesMgt: Codeunit NoSeriesManagement;
            begin
```

```
        if Rec."No." <> xRec."No." then begin
            NoSeriesMgt.TestManual("No. Series");
            "No. Series" := '';
        end;
    end;
}
field(3; "No. Series"; Code[20])
{
    Caption = 'No. Series';
    DataClassification = CustomerContent;
}
field(4; "External Document No."; Code[20])
{
    Caption = 'External Document No.';
    DataClassification = CustomerContent;
}
field(5; "Issue Date"; Date)
{
    Caption = 'Issue Date';
    DataClassification = CustomerContent;

    trigger OnValidate()
    begin
        CalcQtyOfDaysBetwRegAndIssue();
    end;
}
field(6; "Registration Date"; Date)
{
    Caption = 'Registration Date';
    DataClassification = CustomerContent;

    trigger OnValidate()
    begin
        CalcQtyOfDaysBetwRegAndIssue();
    end;
}
field(7; "Customer No."; Code[20])
```

```
            {
                Caption = 'Customer No.';
                DataClassification = CustomerContent;
                TableRelation = Customer;

                trigger OnValidate()
                var
                    Customer: Record Customer;
                    SalesVATInvoiceLine_loc: Record "KDA Sls VAT Invoice Line";
                begin
                    Rec.CreateDim(DATABASE::Customer, Rec."Customer No.", 0, '',
0, '');
                    if "Customer No." <> '' then begin
                        Customer.Get("Customer No.");
                        "Customer Name" := Customer.Name;
                        "Registration No." := Customer."KDA Registration No.";
                        "VAT Registration No." := Customer."VAT Registration
No.";
                        "VAT Bus. Posting Group" := Customer."VAT Bus. Posting
Group";
                        "Tax Number Source" := Customer."KDA Tax Number Source";
                    end
                    else begin
                        "Customer Name" := '';
                        "Registration No." := '';
                        "VAT Registration No." := '';
                        "VAT Bus. Posting Group" := '';
                        "Tax Number Source" := "Tax Number Source"::" ";
                    end;

                    if "Customer No." <> xRec."Customer No." then begin
                        SalesVATInvoiceLine_loc.Reset();
                        SalesVATInvoiceLine_loc.SetRange("VAT Invoice Type",
"VAT Invoice Type");
                        SalesVATInvoiceLine_loc.SetRange("Document No.", "No.");
                        if SalesVATInvoiceLine_loc.FindSet() then
                            repeat
                                SalesVATInvoiceLine_loc.Validate("Customer No.",
"Customer No.");
```

```
                                    SalesVATInvoiceLine_loc.Validate("Customer
Name", "Customer Name");
                                    SalesVATInvoiceLine_loc.Modify();
                                until SalesVATInvoiceLine_loc.Next() = 0;
                            end;
                        end;
                    }
                    field(8; "Customer Name"; Text[100])
                    {
                        Caption = 'Customer Name';
                        DataClassification = CustomerContent;
                        Editable = false;
                    }
                    field(9; "Applied Invoice No."; Code[20])
                    {
                        Caption = 'Applied Invoice No.';
                        DataClassification = CustomerContent;
                        Editable = false;
                        TableRelation = "KDA Sls VAT Invoice Header"."No." WHERE("VAT
Invoice Type" = CONST(Invoice),

"Customer No." = FIELD("Customer No."));
                    }
                    field(10; "VAT Bus. Posting Group"; Code[20])
                    {
                        Caption = 'VAT Bus. Posting Group';
                        DataClassification = CustomerContent;
                        TableRelation = "VAT Business Posting Group";


                        trigger OnValidate()
                        var
                            SalesVATInvoiceLine_loc: Record "KDA Sls VAT Invoice Line";
                            VATPostingSetup_loc: Record "VAT Posting Setup";
                        begin
                            if "VAT Bus. Posting Group" <> xRec."VAT Bus. Posting Group"
then begin
                                SalesVATInvoiceLine_loc.Reset();
                                SalesVATInvoiceLine_loc.SetRange("VAT Invoice Type",
"VAT Invoice Type");
```

```
                        SalesVATInvoiceLine_loc.SetRange("Document No.", "No.");
                        if SalesVATInvoiceLine_loc.FindSet() then
                            repeat
                                if (SalesVATInvoiceLine_loc."VAT Product Posting
Group" <> '') and VATPostingSetup_loc.Get("VAT Bus. Posting Group",
SalesVATInvoiceLine_loc."VAT Product Posting Group") then
                                    case SalesVATInvoiceLine_loc."VAT Invoice
Type" of
                                        SalesVATInvoiceLine_loc."VAT Invoice
Type"::Invoice:

SalesVATInvoiceLine_loc.Validate("VAT Declaration Entry Code",
VATPostingSetup_loc."KDA VAT Decl. Entry Code");
                                        SalesVATInvoiceLine_loc."VAT Invoice
Type"::Appendix2:

SalesVATInvoiceLine_loc.Validate("VAT Declaration Entry Code",
VATPostingSetup_loc."KDA VAT App Decl Entry Code");
                                    end;
                                SalesVATInvoiceLine_loc.Modify();
                            until SalesVATInvoiceLine_loc.Next() = 0;
                    end;
                end;
            }
            field(11; "Creation Type"; Enum "KDA Source Type")
            {
                Caption = 'Creation Type';
                DataClassification = CustomerContent;
                // OptionCaption = 'Manual,Import';
                // OptionMembers = Manual,Import;
                Editable = false;
                Description = 'IW.BC.9860';
            }
            field(12; Status; Enum "KDA Status")
            {
                Caption = 'Status';
                DataClassification = CustomerContent;
                // OptionCaption = 'Open,Released,Posted';
                // OptionMembers = Open,Released,Posted;
                Editable = False;
            }
```

```
                field(13; "System Created Date"; Date)
                {
                    Caption = 'System Created Date';
                    DataClassification = CustomerContent;
                    Editable = false;
                }
                field(14; "System Created Time"; Time)
                {
                    Caption = 'System Created Time';
                    DataClassification = CustomerContent;
                    Editable = false;
                }
                field(15; "User ID"; Code[50])
                {
                    Caption = 'User ID';
                    DataClassification = EndUserIdentifiableInformation;
                    Editable = false;
                }
                field(16; "Total Base Amount"; Decimal)
                {
                    CalcFormula = Sum("KDA Sls VAT Invoice Line"."Base Amount"
WHERE("VAT Invoice Type" = FIELD("VAT Invoice Type"),

"Document No." = FIELD("No.")));
                    Caption = 'Total Base Amount';
                    DecimalPlaces = 2 : 2;
                    Editable = false;
                    FieldClass = FlowField;
                }
                field(17; "Total VAT Amount"; Decimal)
                {
                    CalcFormula = Sum("KDA Sls VAT Invoice Line"."VAT Amount"
WHERE("VAT Invoice Type" = FIELD("VAT Invoice Type"),

"Document No." = FIELD("No.")));
                    Caption = 'Total VAT Amount';
                    DecimalPlaces = 2 : 2;
                    Editable = false;
                    FieldClass = FlowField;
```

```
                }
                field(18; "Total Amount Incl. VAT"; Decimal)
                {
                    CalcFormula = Sum("KDA Sls VAT Invoice Line"."Amount Incl. VAT"
WHERE("VAT Invoice Type" = FIELD("VAT Invoice Type"),

"Document No." = FIELD("No.")));
                    Caption = 'Total Amount Incl. VAT';
                    DecimalPlaces = 2 : 2;
                    Editable = false;
                    FieldClass = FlowField;
                }
                field(19; "Included in ERNN"; Boolean)
                {
                    Caption = 'Included in URTI';
                    DataClassification = CustomerContent;
                    Editable = false;
                }
                field(20; "Kept by seller"; Boolean)
                {
                    Caption = 'Kept by Seller';
                    DataClassification = CustomerContent;
                }
                field(21; "Reason Type"; Code[10])
                {
                    Caption = 'Reason Type';
                    DataClassification = CustomerContent;
                    TableRelation = "Reason Code".Code WHERE("KDA Reason Category" =
CONST(VAT));
                }
                field(22; "Starting Date of VAT Period"; Date)
                {
                    Caption = 'Start Date of VAT Period';
                    DataClassification = CustomerContent;
                    Editable = true;

                    trigger OnValidate()
                    var
```

```
                    SalesVATInvoiceLine_loc: Record "KDA Sls VAT Invoice Line";
                begin
                    if "Starting Date of VAT Period" <> xRec."Starting Date of
VAT Period" then begin
                        SalesVATInvoiceLine_loc.Reset();
                        SalesVATInvoiceLine_loc.SetRange("VAT Invoice Type",
"VAT Invoice Type");
                        SalesVATInvoiceLine_loc.SetRange("Document No.", "No.");
                        if SalesVATInvoiceLine_loc.FindSet() then
                            repeat
                                SalesVATInvoiceLine_loc.Validate("Start date of
VAT Period", "Starting Date of VAT Period");
                                SalesVATInvoiceLine_loc.Modify();
                            until SalesVATInvoiceLine_loc.Next() = 0;
                    end;
                end;
            }
            field(23; "Status of Incl. in VAT period"; Enum "KDA Status of Incl.
in VAT prd.")
            {
                Caption = 'Status of Incl. in VAT Period';
                DataClassification = CustomerContent;
                Editable = false;
                // OptionCaption = ' ,Included in VAT Period,Changed VAT
Period';
                // OptionMembers = " ","Included in VAT Period","Changed VAT
Period";
            }
            field(24; "Qty of Days betw Reg and Issue"; Integer)
            {
                Caption = 'Qty of Days between Reg and Issue';
                DataClassification = CustomerContent;
                Editable = false;
            }
            field(25; "VAT Registration No."; Code[20])
            {
                Caption = 'VAT Registration No.';
                DataClassification = CustomerContent;
                Editable = true;
```

```
            }
            field(27; "Corr. Start Date of VAT Period"; Date)
            {
                Caption = 'Corr. Start Date of VAT Period';
                DataClassification = CustomerContent;
                Editable = true;

                trigger OnValidate()
                var
                    SalesVATInvoiceLine_loc: Record "KDA Sls VAT Invoice Line";
                begin
                    if "Corr. Start Date of VAT Period" <> xRec."Corr. Start
Date of VAT Period" then begin
                        SalesVATInvoiceLine_loc.Reset();
                        SalesVATInvoiceLine_loc.SetRange("VAT Invoice Type",
"VAT Invoice Type");
                        SalesVATInvoiceLine_loc.SetRange("Document No.", "No.");
                        if SalesVATInvoiceLine_loc.FindSet() then
                            repeat
                                SalesVATInvoiceLine_loc.Validate("Corr. Start
Date of VAT Period", "Corr. Start Date of VAT Period");
                                SalesVATInvoiceLine_loc.Modify();
                            until SalesVATInvoiceLine_loc.Next() = 0;
                    end;
                end;
            }
            field(29; "Posting Date"; Date)
            {
                Caption = 'Posting Date';
                DataClassification = CustomerContent;
            }
            field(30; "Registration No."; Code[10])
            {
                Caption = 'Registration No.';
                DataClassification = CustomerContent;
                Editable = false;
            }
            field(31; "Document Type"; Enum "KDA Document Type")
```

```
                {
                    Caption = 'Document Type';
                    DataClassification = CustomerContent;
                    // OptionCaption = 'PNE,RKE';
                    // OptionMembers = PNE,RKE;
                }
                field(32; "No. Exported"; Integer)
                {
                    Caption = 'No. Exported';
                    DataClassification = CustomerContent;
                }
                field(33; "Tax Number Source"; Enum "KDA Tax Number Source")
                {
                    DataClassification = CustomerContent;
                    Caption = 'Tax Number Source';
                    Description = 'IW.BC.13525';
                }
                field(41; "Shortcut Dimension 1 Code"; Code[20])
                {
                    DataClassification = CustomerContent;
                    Caption = 'Shortcut Dimension 1 Code';
                    CaptionClass = '1,2,1';
                    TableRelation = "Dimension Value".Code where("Global Dimension
No." = const(1), Blocked = CONST(false));

                    trigger OnValidate()
                    begin
                        ValidateShortcutDimCode(1, "Shortcut Dimension 1 Code");
                    end;
                }
                field(42; "Shortcut Dimension 2 Code"; Code[20])
                {
                    DataClassification = CustomerContent;
                    Caption = 'Shortcut Dimension 2 Code';
                    CaptionClass = '1,2,2';
                    TableRelation = "Dimension Value".Code where("Global Dimension
No." = const(2), Blocked = CONST(false));
```

```
                trigger OnValidate()
                begin
                    ValidateShortcutDimCode(2, "Shortcut Dimension 2 Code");
                end;
            }
            field(43; "Dimension Set ID"; Integer)
            {
                DataClassification = CustomerContent;
                Caption = 'Dimension Set ID';
                TableRelation = "Dimension Set Entry"."Dimension Set ID";
                Editable = false;

                trigger OnLookup()
                begin
                    ShowDocDim();
                end;

                trigger OnValidate()
                begin
                    DimMgt.UpdateGlobalDimFromDimSetID("Dimension Set ID",
"Shortcut Dimension 1 Code", "Shortcut Dimension 2 Code");
                end;
            }
            field(44; "Cust. Ledger Entry No."; Integer)
            {
                DataClassification = CustomerContent;
                Caption = 'Customer Ledger Entry No.';
                Description = 'IW.BC.9860';
                TableRelation = "Cust. Ledger Entry"."Entry No." where("Customer
No." = field("Customer No."));
            }
            field(46; "Source Document Type"; Enum "KDA Tax Inv. Type")
            {
                DataClassification = CustomerContent;
                Caption = 'Source Document Type';
                Description = 'IW.BC.9860';
                Editable = false;
            }
```

```
            field(47; "Agreement No."; Code[20])
            {
                DataClassification = CustomerContent;
                Caption = 'Agreement No.';
                Description = 'IW.BC.9860';
            }
            field(48; "VAT Exempt"; Boolean)
            {
                DataClassification = CustomerContent;
                Caption = 'VAT Exempt';
                Description = 'IW.BC.9860';
            }
            field(50; "Source Document No."; Code[20])
            {
                DataClassification = CustomerContent;
                Caption = 'Source Document No.';
                Description = 'IW.BC.9860';
                Editable = false;
                TableRelation = if ("Source Document Type" = CONST(Invoice)) 
"Sales Invoice Header"
                else
                if ("Source Document Type" = CONST("Credit Memo")) "Sales 
Cr.Memo Header";
            }
            field(51; "Consolidated VAT Invoice"; Enum "KDA Consolidated VAT 
Invoice")
            {
                DataClassification = CustomerContent;
                Caption = 'Consolidated VAT Invoice';
            }
        }

        keys
        {
            key(Key1; "VAT Invoice Type", "No.")
            {
            }
        }
```

```
        fieldgroups
        {
        }

        var
            SalesVATHeader: Record "KDA Sls VAT Invoice Header";
            DimMgt: Codeunit DimensionManagement;
            NothingToReleaseErr: Label 'There is nothing to release';
            FieldsShouldBeFilledInErrLbl: Label 'One of the fields %1, %2 must
be filled for Line No. %3.', Comment = '%1 - UCGFEA Code, %2 - Goods and Services
Code, %3 - "VAT Invoice Line No."';
            ExtDocNoIsNotEmptyDoYouWantToDeleteErrLbl: Label '%1 isn`t empty. %2
can be already registered in URTI. Do you want to delete %2?', Comment = '%1 -
External Document No., %2 - "VAT Invoice Type"';
            DocIsAlreadyRegisteredinERNNContinueConfLbl: Label '%1 is already
registered in URTI. Continue?', Comment = '%1- "VAT Invoice Type"';
            AppliedAppendix2ErrLbl: Label 'You cannot delete the VAT invoice
because it has applied Appendix 2: %1.', Comment = '%1 -"No."';
            UpdDimLbl: Label 'You may have changed a dimension.\\Do you want to
update the lines?';
            SalesVATInvLine: Record "KDA Sls VAT Invoice Line";

        trigger OnDelete()
        var
            SalesVATInvoiceLine: Record "KDA Sls VAT Invoice Line";
            SalesVATInvHeader: Record "KDA Sls VAT Invoice Header";
        begin
            if "VAT Invoice Type" = "VAT Invoice Type"::Invoice then begin
                SalesVATInvHeader.SetRange("VAT Invoice Type", "VAT Invoice
Type"::Appendix2);
                SalesVATInvHeader.SetRange("Applied Invoice No.", "No.");
                if SalesVATInvHeader.FindFirst() then
                    Error(AppliedAppendix2ErrLbl, SalesVATInvHeader."No.");
            end;

            CheckIncludedInERNN();
            SalesVATInvoiceLine.SetRange("VAT Invoice Type", "VAT Invoice
Type");
            SalesVATInvoiceLine.SetRange("Document No.", "No.");
            if not SalesVATInvoiceLine.IsEmpty then
```

```
        SalesVATInvoiceLine.DeleteAll();
end;

trigger OnInsert()
var
    TaxSetup: Record "Tax Setup";
    NoSeriesMgt: Codeunit NoSeriesManagement;
begin

    TaxSetup.Get();

    case Rec."VAT Invoice Type" of
        Rec."VAT Invoice Type"::Invoice:
            begin
                if Rec."No." = '' then begin
                    TaxSetup.TestField("KDA Sales VAT Invoice Nos.");
                    NoSeriesMgt.InitSeries(
                        TaxSetup."KDA Sales VAT Invoice Nos.",
                        "No. Series",
                        0D,
                        "No.",
                        "No. Series"
                    );
                end;

                TaxSetup.TestField("KDA Sls VAT Ext. Doc. Nos.");
                NoSeriesMgt.InitSeries(
                    TaxSetup."KDA Sls VAT Ext. Doc. Nos.",
                    TaxSetup."KDA Sls VAT Ext. Doc. Nos.",
                    0D,
                    "External Document No.",
                    TaxSetup."KDA Sls VAT Ext. Doc. Nos."
                );
            end;

        Rec."VAT Invoice Type"::Appendix2:
            begin
```

```
                        if Rec."No." = '' then begin
                            TaxSetup.TestField("KDA Sales Appendix 2 Nos.");
                            NoSeriesMgt.InitSeries(
                                TaxSetup."KDA Sales Appendix 2 Nos.",
                                "No. Series",
                                0D,
                                "No.",
                                "No. Series"
                            );
                        end;

                        TaxSetup.TestField("KDA Sls VT App2 Ext Doc Nos");
                        NoSeriesMgt.InitSeries(
                            TaxSetup."KDA Sls VT App2 Ext Doc Nos",
                            TaxSetup."KDA Sls VT App2 Ext Doc Nos",
                            0D,
                            "External Document No.",
                            TaxSetup."KDA Sls VT App2 Ext Doc Nos"
                        );
                    end;
            end;

            "User ID" := CopyStr(UserId, 1, MaxStrLen("User ID"));
            "System Created Date" := Today;
            "System Created Time" := Time;
            if Rec."Issue Date" = 0D then
                Rec."Issue Date" := WorkDate();
        end;

        local procedure CalcQtyOfDaysBetwRegAndIssue()
        begin
            if "Registration Date" > "Issue Date" then
                Evaluate("Qty of Days betw Reg and Issue", Format("Registration
Date" - "Issue Date"))
            else
                "Qty of Days betw Reg and Issue" := 0;
        end;
```

```
          procedure CreateAppendix2FromInvoice(var SalesVATInvoiceAppendix2:
Record "KDA Sls VAT Invoice Header")

          var

              SalesVATInvoiceLineAppendix2: Record "KDA Sls VAT Invoice Line";

              SalesVATInvoiceLine: Record "KDA Sls VAT Invoice Line";

              AppliedNo: Code[20];

          begin

              AppliedNo := "No.";

              TestField("No.");

              TestField("VAT Invoice Type", "VAT Invoice Type"::Invoice);

              SalesVATInvoiceAppendix2.Init();

              SalesVATInvoiceAppendix2.TransferFields(Rec);

              SalesVATInvoiceAppendix2."No." := '';

              SalesVATInvoiceAppendix2."VAT Invoice Type" :=
SalesVATInvoiceAppendix2."VAT Invoice Type"::Appendix2;

              SalesVATInvoiceAppendix2.Validate("Customer No.", "Customer No.");

              SalesVATInvoiceAppendix2."User ID" := CopyStr(UserId, 1,
MaxStrLen(SalesVATInvoiceAppendix2."User ID"));

              SalesVATInvoiceAppendix2.Status :=
SalesVATInvoiceAppendix2.Status::Open;

              SalesVATInvoiceAppendix2."Included in ERNN" := false;

              SalesVATInvoiceAppendix2."External Document No." := '';


              SalesVATInvoiceAppendix2."Issue Date" := Today;

              SalesVATInvoiceAppendix2."VAT Bus. Posting Group" := "VAT Bus.
Posting Group";

              SalesVATInvoiceAppendix2."Document Type" := "Document Type";


              SalesVATInvoiceAppendix2."Applied Invoice No." := AppliedNo;


              SalesVATInvoiceAppendix2."Registration Date" := 0D;

              SalesVATInvoiceAppendix2."Starting Date of VAT Period" := 0D;

              SalesVATInvoiceAppendix2."Corr. Start Date of VAT Period" := 0D;

              SalesVATInvoiceAppendix2."Status of Incl. in VAT period" :=

                  SalesVATInvoiceAppendix2."Status of Incl. in VAT period"::" ";


              SalesVATInvoiceAppendix2."Shortcut Dimension 1 Code" := "Shortcut
Dimension 1 Code";
```

```
            SalesVATInvoiceAppendix2."Shortcut Dimension 1 Code" := "Shortcut
Dimension 1 Code";

            SalesVATInvoiceAppendix2."Dimension Set ID" := "Dimension Set ID";

            SalesVATInvoiceAppendix2."Consolidated VAT Invoice" :=
Rec."Consolidated VAT Invoice";


            SalesVATInvoiceAppendix2.Insert(true);

            SalesVATInvoiceLine.SetRange("VAT Invoice Type", "VAT Invoice
Type");

            SalesVATInvoiceLine.SetRange("Document No.", "No.");

            if SalesVATInvoiceLine.FindSet() then

                repeat

                    SalesVATInvoiceLineAppendix2.Init();

                    SalesVATInvoiceLineAppendix2."VAT Invoice Type" :=
SalesVATInvoiceLineAppendix2."VAT Invoice Type"::Appendix2;

                    SalesVATInvoiceLineAppendix2."Document No." :=
SalesVATInvoiceAppendix2."No.";

                    SalesVATInvoiceLineAppendix2."Line No." :=
SalesVATInvoiceLine."Line No.";

                    SalesVATInvoiceLineAppendix2.Type :=
SalesVATInvoiceLine.Type;

                    SalesVATInvoiceLineAppendix2.Validate("No.",
SalesVATInvoiceLine."No.");

                    SalesVATInvoiceLineAppendix2."VAT Rate Code" :=
SalesVATInvoiceLine."VAT Rate Code";

                    SalesVATInvoiceLineAppendix2."VAT Invoice Line No." :=
SalesVATInvoiceLine."VAT Invoice Line No.";

                    SalesVATInvoiceLineAppendix2."G/L Expense Account" :=
SalesVATInvoiceLine."G/L Expense Account";

                    SalesVATInvoiceLineAppendix2.Quantity := -
SalesVATInvoiceLine.Quantity;

                    SalesVATInvoiceLineAppendix2."Price Excl. VAT" :=
SalesVATInvoiceLine."Price Excl. VAT";

                    SalesVATInvoiceLineAppendix2."Base Amount" := -
SalesVATInvoiceLine."Base Amount";

                    SalesVATInvoiceLineAppendix2."Exemption Code" :=
SalesVATInvoiceLine."Exemption Code";

                    SalesVATInvoiceLineAppendix2."Applied Invoice Line No." :=
SalesVATInvoiceLine."VAT Invoice Line No.";

                    SalesVATInvoiceLineAppendix2."Goods and Services Code" :=
SalesVATInvoiceLine."Goods and Services Code";

                    SalesVATInvoiceLineAppendix2."Amount Incl. VAT" := -
SalesVATInvoiceLine."Amount Incl. VAT";

                    SalesVATInvoiceLineAppendix2."VAT Amount" := -
SalesVATInvoiceLine."VAT Amount";
```

```
                        SalesVATInvoiceLineAppendix2."UCGFEA Code" :=
SalesVATInvoiceLine."UCGFEA Code";

                        SalesVATInvoiceLineAppendix2.Validate("Unit Of Measure
Code", SalesVATInvoiceLine."Unit Of Measure Code");

                        SalesVATInvoiceLineAppendix2."Reason of Correction" :=
SalesVATInvoiceLine."Reason of Correction";

                        SalesVATInvoiceLineAppendix2."Correction Group No." :=
SalesVATInvoiceLine."Correction Group No.";

                        SalesVATInvoiceLineAppendix2.Description :=
SalesVATInvoiceLine.Description;

                        SalesVATInvoiceLineAppendix2."VAT Product Posting Group" :=
SalesVATInvoiceLine."VAT Product Posting Group";

                        SalesVATInvoiceLineAppendix2."Shortcut Dimension 1 Code" :=
SalesVATInvoiceLine."Shortcut Dimension 1 Code";

                        SalesVATInvoiceLineAppendix2."Shortcut Dimension 1 Code" :=
SalesVATInvoiceLine."Shortcut Dimension 1 Code";

                        SalesVATInvoiceLineAppendix2."Dimension Set ID" :=
SalesVATInvoiceLine."Dimension Set ID";

                        if not SalesVATInvoiceLineAppendix2.Insert(true) then

                            SalesVATInvoiceLineAppendix2.Modify();

                    until SalesVATInvoiceLine.Next() = 0;

            end;


            procedure CheckAndReleaseDocument()

            var

                SalesVATInvoiceLine: Record "KDA Sls VAT Invoice Line";

            begin

                TestField("Issue Date");

                TestField("External Document No.");

                TestField("Customer No.");

                SalesVATInvoiceLine.SetRange("VAT Invoice Type", "VAT Invoice
Type");

                SalesVATInvoiceLine.SetRange("Document No.", "No.");

                SalesVATInvoiceLine.SetFilter(Type, '<>%1',
SalesVATInvoiceLine.Type::" ");

                if SalesVATInvoiceLine.FindSet() then

                    repeat

                        SalesVATInvoiceLine.TestField("Unit Of Measure Code");

                        if (SalesVATInvoiceLine."UCGFEA Code" = '') and
(SalesVATInvoiceLine."Goods and Services Code" = '') then

                            Error(FieldsShouldBeFilledInErrLbl,
SalesVATInvoiceLine.FieldCaption("UCGFEA Code"),
```

```
SalesVATInvoiceLine.FieldCaption("Goods and Services Code"),
SalesVATInvoiceLine."VAT Invoice Line No.");
                until SalesVATInvoiceLine.Next() = 0
            else
                Error(NothingToReleaseErr);
            "User ID" := CopyStr(UserId, 1, MaxStrLen("User ID"));
            if Status = Status::Open then begin
                Status := Status::Released;
                SalesVATInvoiceLine.SetRange(Type);
                if SalesVATInvoiceLine.FindSet() then
                    repeat
                        SalesVATInvoiceLine.Status :=
SalesVATInvoiceLine.Status::Released;
                        SalesVATInvoiceLine.Modify();
                    until SalesVATInvoiceLine.Next() = 0;
            end;
            Modify();
        end;

        procedure OpenDocument()
        var
            SalesVATInvoiceLine: Record "KDA Sls VAT Invoice Line";
        begin
            "User ID" := CopyStr(UserId, 1, MaxStrLen("User ID"));
            if Status = Status::Released then begin
                Status := Status::Open;
                SalesVATInvoiceLine.SetRange("Document No.", "No.");
                SalesVATInvoiceLine.SetRange("VAT Invoice Type", "VAT Invoice
Type");
                if SalesVATInvoiceLine.FindSet() then
                    repeat
                        SalesVATInvoiceLine.Status :=
SalesVATInvoiceLine.Status::Open;
                        SalesVATInvoiceLine.Modify();
                    until SalesVATInvoiceLine.Next() = 0;
            end;
            Modify();
        end;
```

```
        procedure Navigate();
        var
            NavigateForm: Page Navigate;
        begin
            NavigateForm.SetDoc("Posting Date", "No.");
            NavigateForm.RUN();
        end;

        [BusinessEvent(false)]
        procedure OnPostVATInv(var SalesVATInvoiceHeader: Record "KDA Sls VAT
Invoice Header")
        begin
        end;

        procedure SalesVATInvoiceLinesExist(var SalesVATInvoiceLine: Record "KDA
Sls VAT Invoice Line"): Boolean
        begin
            SalesVATInvoiceLine.SetRange("VAT Invoice Type", "VAT Invoice
Type");
            SalesVATInvoiceLine.SetRange("Document No.", "No.");
            exit(not SalesVATInvoiceLine.IsEmpty);
        end;

        procedure CreateNewRecord()
        begin
            case "VAT Invoice Type" of
                "VAT Invoice Type"::Invoice:
                    "Document Type" := "Document Type"::PNE;
                "VAT Invoice Type"::Appendix2:
                    "Document Type" := "Document Type"::RKE;
            end;
        end;

        procedure CheckExternalDocNo()
        begin
            if "External Document No." <> '' then
                if GuiAllowed then begin
                    if not Confirm(ExtDocNoIsNotEmptyDoYouWantToDeleteErrLbl,
false, FieldCaption("External Document No."), "VAT Invoice Type") then
```

```
                        Error('');
                end
                else
                    Error('');
        end;

        procedure CheckIncludedInERNN()
        begin
            if "Included in ERNN" then
                if GuiAllowed then begin
                    if not Confirm(DocIsAlreadyRegisteredinERNNContinueConfLbl,
false, "VAT Invoice Type") then
                        Error('');
                end
                else
                    Error('');
        end;

        procedure SetNoFieldVisibility(): Boolean
        var
            NoSeries: Record "No. Series";
            NoSeriesRelationship: Record "No. Series Relationship";
            TaxSetup: Record "Tax Setup";
            NoSeriesMgt: Codeunit NoSeriesManagement;
            NoSeriesCode: Code[20];
            SeriesDate: Date;
            Text003Lbl: Label 'It is not possible to assign numbers
automatically. If you want the program to assign numbers automatically, please
activate %1 in %2 %3.', Comment = '%1=Default Nos. setting,%2=No. Series table
caption,%3=No. Series Code';
        begin
            if "No." = '' then begin
                TaxSetup.Get();

                if ("VAT Invoice Type" = "VAT Invoice Type"::Invoice) then begin
                    NoSeries.Get(TaxSetup."KDA Sales VAT Invoice Nos.");
                    if not NoSeries."Default Nos." then
                        Error(
                          Text003Lbl,
```

```
                            NoSeries.FieldCaption("Default Nos."),
NoSeries.TableCaption, NoSeries.Code);
                    if "No. Series" <> '' then begin
                        NoSeriesCode := TaxSetup."KDA Sales VAT Invoice Nos.";

                        NoSeries.Reset();
                        NoSeriesRelationship.SetRange(Code, NoSeriesCode);
                        if NoSeriesRelationship.FindSet() then
                            repeat
                                NoSeries.Code := NoSeriesRelationship."Series
Code";
                                NoSeries.Mark := true;
                            until NoSeriesRelationship.Next() = 0;
                        if NoSeries.Get(NoSeriesCode) then
                            NoSeries.Mark := true;
                        NoSeries.MarkedOnly := true;

                        NoSeries.Code := "No. Series";
                        if NoSeries.IsEmpty then
                            NoSeries.Get(TaxSetup."KDA Sales VAT Invoice Nos.");
                    end;
                    "No." := NoSeriesMgt.DoGetNextNo(NoSeries.Code, WorkDate(),
false, true);
                    "No. Series" := NoSeries.Code;
                end;
                if ("VAT Invoice Type" = "VAT Invoice Type"::Appendix2) then
begin
                    NoSeries.Get(TaxSetup."KDA Sales Appendix 2 Nos.");
                    if not NoSeries."Default Nos." then
                        Error(
                          Text003Lbl,
                          NoSeries.FieldCaption("Default Nos."),
NoSeries.TableCaption, NoSeries.Code);
                    if "No. Series" <> '' then begin
                        NoSeriesCode := TaxSetup."KDA Sales Appendix 2 Nos.";

                        NoSeries.Reset();
                        NoSeriesRelationship.SetRange(Code, NoSeriesCode);
                        if NoSeriesRelationship.FindSet() then
```

```
                                repeat
                                    NoSeries.Code := NoSeriesRelationship."Series
Code";
                                    NoSeries.Mark := true;
                                until NoSeriesRelationship.Next() = 0;
                            if NoSeries.Get(NoSeriesCode) then
                                NoSeries.Mark := true;
                            NoSeries.MarkedOnly := true;

                            NoSeries.Code := "No. Series";
                            if NoSeries.IsEmpty then
                                NoSeries.Get(TaxSetup."KDA Sales Appendix 2 Nos.");
                        end;
                        "No." := NoSeriesMgt.DoGetNextNo(NoSeries.Code, WorkDate(),
false, true);
                        "No. Series" := NoSeries.Code;
                    end;
                end;
                if not NoSeries.Get("No. Series") then
                    exit(true);

                SeriesDate := WorkDate();
                NoSeriesRelationship.SetRange(Code, "No. Series");
                if not NoSeriesRelationship.IsEmpty then
                    exit(true);

                if NoSeries."Manual Nos." or (NoSeries."Default Nos." = false) then
                    exit(true);

                exit(NoSeriesMgt.DoGetNextNo("No. Series", SeriesDate, false, true)
= '');
            end;

            procedure ValidateShortcutDimCode(FieldNumber: Integer; var
ShortcutDimCode: Code[20])
            var
                OldDimSetID: Integer;
            begin
                OldDimSetID := "Dimension Set ID";
```

```
            DimMgt.ValidateShortcutDimValues(FieldNumber, ShortcutDimCode,
"Dimension Set ID");
            if "No." <> '' then
                Modify();
            if OldDimSetID <> "Dimension Set ID" then begin
                Modify();
                if VATSalesLinesExist then
                    UpdateAllLineDim("Dimension Set ID", OldDimSetID);
            end;
        end;

        procedure ShowDocDim()
        var
            OldDimSetID: Integer;
        begin
            OldDimSetID := "Dimension Set ID";
            "Dimension Set ID" := DimMgt.EditDimensionSet(
                "Dimension Set ID",
                StrSubstNo('%1 %2', "VAT Invoice Type", "No."),
                "Shortcut Dimension 1 Code",
                "Shortcut Dimension 2 Code"
            );
            if OldDimSetID <> "Dimension Set ID" then begin
                Modify();
                if VATSalesLinesExist then
                    UpdateAllLineDim("Dimension Set ID", OldDimSetID);
            end;
        end;

        procedure CreateDim(Type1: Integer; No1: Code[20]; Type2: Integer; No2:
Code[20]; Type3: Integer; No3: Code[20])
        var
            SourceCodeSetup: Record "Source Code Setup";
            TableID: array[10] of Integer;
            No: array[10] of Code[20];
            IsHandled: Boolean;
            OldDimSetID: Integer;
        begin
```

```
            IsHandled := false;
            if IsHandled then
                exit;

            SourceCodeSetup.Get();
            TableID[1] := Type1;
            No[1] := No1;
            TableID[2] := Type2;
            No[2] := No2;
            TableID[3] := Type3;
            No[3] := No3;

            "Shortcut Dimension 1 Code" := '';
            "Shortcut Dimension 2 Code" := '';
            OldDimSetID := "Dimension Set ID";
            GetSalesVATHeader();
            "Dimension Set ID" :=
              DimMgt.GetRecDefaultDimID(
                Rec, CurrFieldNo, TableID, No, SourceCodeSetup."KDA Close VAT
Period",
                "Shortcut Dimension 1 Code", "Shortcut Dimension 2 Code", 0, 0);
            DimMgt.UpdateGlobalDimFromDimSetID("Dimension Set ID", "Shortcut
Dimension 1 Code", "Shortcut Dimension 2 Code");

            if (OldDimSetID <> "Dimension Set ID") and VATSalesLinesExist then
begin
                Modify;
                UpdateAllLineDim("Dimension Set ID", OldDimSetID);
            end;
        end;

        procedure GetSalesVATHeader()
        begin
            if ("VAT Invoice Type" <> SalesVATHeader."VAT Invoice Type") or
("No." <> SalesVATHeader."No.") then
                if SalesVATHeader.Get("VAT Invoice Type", "No.") then;
        end;
```

```
        procedure UpdateAllLineDim(NewParentDimSetID: Integer;
OldParentDimSetID: Integer)
        var
            NewDimSetID: Integer;
            ShippedReceivedItemLineDimChangeConfirmed: Boolean;
        begin
            if NewParentDimSetID = OldParentDimSetID then
                exit;
            if GuiAllowed then
                if not Confirm(UpdDimLbl) then
                    exit;

            SalesVATInvLine.Reset();
            SalesVATInvLine.SetRange("VAT Invoice Type", "VAT Invoice Type");
            SalesVATInvLine.SetRange("Document No.", "No.");
            SalesVATInvLine.LockTable();
            if SalesVATInvLine.Find('-') then
                repeat
                    NewDimSetID :=
DimMgt.GetDeltaDimSetID(SalesVATInvLine."Dimension Set ID", NewParentDimSetID,
OldParentDimSetID);
                    if SalesVATInvLine."Dimension Set ID" <> NewDimSetID then
begin
                        SalesVATInvLine."Dimension Set ID" := NewDimSetID;

                        DimMgt.UpdateGlobalDimFromDimSetID(
                          SalesVATInvLine."Dimension Set ID",
SalesVATInvLine."Shortcut Dimension 1 Code", SalesVATInvLine."Shortcut Dimension 2
Code");

                        SalesVATInvLine.Modify();
                    end;
                until SalesVATInvLine.Next() = 0;
        end;

        procedure VATSalesLinesExist(): Boolean
        begin
            SalesVATInvLine.Reset();
            SalesVATInvLine.SetRange("VAT Invoice Type", "VAT Invoice Type");
```

```
    SalesVATInvLine.SetRange("Document No.", "No.");
    exit(not SalesVATInvLine.IsEmpty);
end;

procedure InitNo()
var
    TaxSetup: Record "Tax Setup";
    NoSeriesMgt: Codeunit NoSeriesManagement;
begin
    TaxSetup.Get();
    case Rec."VAT Invoice Type" of
        Rec."VAT Invoice Type"::Invoice:
            begin
                if Rec."No." = '' then begin
                    TaxSetup.TestField("KDA Sales VAT Invoice Nos.");
                    NoSeriesMgt.InitSeries(
                        TaxSetup."KDA Sales VAT Invoice Nos.",
                        "No. Series",
                        0D,
                        "No.",
                        "No. Series"
                    );
                end;

                TaxSetup.TestField("KDA Sls VAT Ext. Doc. Nos.");
                NoSeriesMgt.InitSeries(
                    TaxSetup."KDA Sls VAT Ext. Doc. Nos.",
                    TaxSetup."KDA Sls VAT Ext. Doc. Nos.",
                    0D,
                    "External Document No.",
                    TaxSetup."KDA Sls VAT Ext. Doc. Nos."
                );
            end;

        Rec."VAT Invoice Type"::Appendix2:
            begin
                if Rec."No." = '' then begin
```

```
                    TaxSetup.TestField("KDA Sales Appendix 2 Nos.");
                    NoSeriesMgt.InitSeries(
                        TaxSetup."KDA Sales Appendix 2 Nos.",
                        "No. Series",
                        0D,
                        "No.",
                        "No. Series"
                    );
                end;

                TaxSetup.TestField("KDA Sls VT App2 Ext Doc Nos");
                NoSeriesMgt.InitSeries(
                    TaxSetup."KDA Sls VT App2 Ext Doc Nos",
                    TaxSetup."KDA Sls VT App2 Ext Doc Nos",
                    0D,
                    "External Document No.",
                    TaxSetup."KDA Sls VT App2 Ext Doc Nos"
                );
            end;
    end;
end;

procedure CopyFromSalesHeader(SalesHeader: Record "Sales Header")
begin
    "Issue Date" := SalesHeader."Posting Date";
    "Customer No." := SalesHeader."Sell-to Customer No.";
    "Customer Name" := SalesHeader."Sell-to Customer Name";
    "VAT Registration No." := SalesHeader."VAT Registration No.";
    "Cust. Ledger Entry No." := SalesHeader."Doc. No. Occurrence";
    "Agreement No." := SalesHeader."KDA Agreement No.";
    "VAT Bus. Posting Group" := SalesHeader."VAT Bus. Posting Group";
    "VAT Exempt" := SalesHeader."KDA VAT Exempt";
    "Source Document No." := SalesHeader."No.";
    "Dimension Set ID" := SalesHeader."Dimension Set ID";

    OnAfterCopyFromSalesHeader(Rec, SalesHeader);
end;
```

```
        procedure SourceDocumentNoDrillDown()
        var
            SalesInvoiceHeader: Record "Sales Invoice Header";
            SalesCrMemoHeader: Record "Sales Cr.Memo Header";
        begin
            case "Source Document Type" of
                "Source Document Type"::Invoice:
                    begin
                        SalesInvoiceHeader.Get("Source Document No.");
                        Page.Run(Page::"Posted Sales Invoice",
SalesInvoiceHeader);
                    end;
                "Source Document Type"::"Credit Memo":
                    begin
                        SalesCrMemoHeader.Get("Source Document No.");
                        Page.Run(Page::"Posted Sales Credit Memo",
SalesCrMemoHeader);
                    end;
            end;
        end;


        [IntegrationEvent(false, false)]
        procedure OnAfterCopyFromSalesHeader(var Rec: Record "KDA Sls VAT
Invoice Header"; SalesHeader: Record "Sales Header");
        begin


        end;
    }
```

Додаток Б

Лістинг коду сторінки «Список ПН».

```
page 21038250 "KDA Sales VAT Invoice List"
{

    ApplicationArea = All;
    Caption = 'Sales VAT Invoices';
    CardPageID = "KDA Sales VAT Invoice";
    Editable = false;
    PromotedActionCategories = 'New,Process,Report,Release,Posting,Invoice';
    PageType = List;
    SourceTable = "KDA Sls VAT Invoice Header";
    SourceTableView = SORTING("No.", "VAT Invoice Type")
                      WHERE(Status = FILTER(<> Posted));
    UsageCategory = Lists;
    layout
    {
        area(content)
        {
            repeater(Group)
            {
                field("VAT Invoice Type"; Rec."VAT Invoice Type")
                {
                    ApplicationArea = All;
                    ToolTip = 'Specifies the VAT Invoice Type.';
                }
                field("No."; Rec."No.")
                {
                    ApplicationArea = All;
                    ToolTip = 'Specifies the No.';
                }
                field("External Document No."; Rec."External Document No.")
                {
                    ApplicationArea = All;
```

```
        ToolTip = 'Specifies the External Document No.';
    }
    field("Customer No."; Rec."Customer No.")
    {
        ApplicationArea = All;
        Visible = false;
        ToolTip = 'Specifies the Customer No.';
    }
    field("Customer Name"; Rec."Customer Name")
    {
        ApplicationArea = All;
        ToolTip = 'Specifies the Customer Name.';
    }
    field("VAT Registration No."; Rec."VAT Registration No.")
    {
        ApplicationArea = All;
        ToolTip = 'Specifies the VAT Registration No.';
    }
    field("Issue Date"; Rec."Issue Date")
    {
        ApplicationArea = All;
        ToolTip = 'Specifies the Issue Date.';
    }
    field("Registration Date"; Rec."Registration Date")
    {
        ApplicationArea = All;
        ToolTip = 'Specifies the Registration Date.';
    }
    field("Total Base Amount"; Rec."Total Base Amount")
    {
        ApplicationArea = All;
        DecimalPlaces = 2 : 2;
        ToolTip = 'Specifies the Total Base Amount.';
    }
    field("Total VAT Amount"; Rec."Total VAT Amount")
    {
        ApplicationArea = All;
```

```
                    DecimalPlaces = 2 : 2;
                    ToolTip = 'Specifies the Total VAT Amount.';
                }
                field("Total Amount Incl. VAT"; Rec."Total Amount Incl.
VAT")
                {
                    ApplicationArea = All;
                    DecimalPlaces = 2 : 2;
                    ToolTip = 'Specifies the Total Amount Incl. VAT.';
                }
                field(Status; Rec.Status)
                {
                    ApplicationArea = All;
                    ToolTip = 'Specifies the Status.';
                }
                field("Source Document Type"; Rec."Source Document Type")
                {
                    ApplicationArea = All;
                    ToolTip = 'Specifies the Source Document Type.';
                    Description = 'IW.BC.14107';
                }
                field("Source Document No."; Rec."Source Document No.")
                {
                    ApplicationArea = All;
                    ToolTip = 'Specifies the Source Document No.';
                    Description = 'IW.BC.14107';
                }
                field("Included in ERNN"; Rec."Included in ERNN")
                {
                    ApplicationArea = All;
                    ToolTip = 'Specifies the Included in ERNN.';
                }
                field("User ID"; Rec."User ID")
                {
                    ApplicationArea = All;
                    ToolTip = 'Specifies the User ID.';
                }
```

```
                field("Creation Type"; Rec."Creation Type")
                {
                    ApplicationArea = All;
                    ToolTip = 'Specifies the Creation Type.';
                }
                field("VAT Bus. Posting Group"; Rec."VAT Bus. Posting
Group")
                {
                    ApplicationArea = All;
                    Visible = false;
                    ToolTip = 'Specifies the VAT Bus. Posting Group.';
                }
            }
        }
    }

    actions
    {
        area(Processing)
        {
            action(Release)
            {
                Caption = 'Release';
                ApplicationArea = All;
                Image = ReleaseDoc;
                Promoted = true;
                PromotedCategory = Category4;
                PromotedOnly = true;
                ToolTip = 'Release';

                trigger OnAction()
                var
                    SalesVATInvoiceHeader: Record "KDA Sls VAT Invoice
Header";
                begin
                    SalesVATInvoiceHeader.Copy(Rec);
                    CurrPage.SetSelectionFilter(SalesVATInvoiceHeader);
                    if SalesVATInvoiceHeader.FindSet() then
```

```
                repeat
                    SalesVATInvoiceHeader.CheckAndReleaseDocument();
                until SalesVATInvoiceHeader.Next() = 0;
            CurrPage.Update();
        end;
    }
    action(Open)
    {
        Caption = 'Reopen';
        ApplicationArea = All;
        Image = ReOpen;
        Promoted = true;
        PromotedCategory = Category4;
        PromotedOnly = true;
        ToolTip = 'Reopen';

        trigger OnAction()
        var
            SalesVATInvoiceHeader: Record "KDA Sls VAT Invoice
Header";
        begin
            SalesVATInvoiceHeader.Copy(Rec);
            CurrPage.SetSelectionFilter(SalesVATInvoiceHeader);
            if SalesVATInvoiceHeader.FindSet() then
                repeat
                    SalesVATInvoiceHeader.OpenDocument();
                until SalesVATInvoiceHeader.Next() = 0;
            CurrPage.Update();
        end;
    }
    action(Post)
    {
        Caption = 'Post';
        ApplicationArea = All;
        Image = PostDocument;
        Promoted = true;
        PromotedIsBig = true;
```

```
                PromotedCategory = Category5;
                PromotedOnly = true;
                ToolTip = 'Post';

                trigger OnAction()
                begin
                    CODEUNIT.Run(CODEUNIT::"KDA Sls. VAT Doc.-Post Y/N",
Rec);
                end;
            }
            action(PreviewPosting)
            {
                Caption = 'Preview Posting';
                ApplicationArea = All;
                Image = PreviewChecks;
                Promoted = true;
                PromotedIsBig = true;
                PromotedCategory = Category5;
                PromotedOnly = true;
                ToolTip = 'Preview Posting';

                trigger OnAction()
                var
                    SalesVATDocPostYesNo: Codeunit "KDA Sls. VAT Doc.-Post
Y/N";
                begin
                    SalesVATDocPostYesNo.SetPreviewMode(true);
                    SalesVATDocPostYesNo.Preview(Rec);
                end;
            }
            action(PostBatch)
            {
                Caption = 'Post Batch';
                ApplicationArea = All;
                Ellipsis = true;
                Image = PostBatch;
                Promoted = true;
                PromotedIsBig = true;
```

```
                PromotedCategory = Category5;
                PromotedOnly = true;
                ToolTip = 'Post Batch';

                trigger OnAction()
                var
                    VATInvoiceHeader: Record "KDA Sls VAT Invoice Header";
                    BatchPostVATInvoices: Report "KDA Btch Post Sls VAT
Inv.";
                begin
                    VATInvoiceHeader.COPYFILTERS(Rec);
                    BatchPostVATInvoices.SETTABLEVIEW(VATInvoiceHeader);
                    BatchPostVATInvoices.RUN();
                end;
            }
            // IW.BC.13741 >
            action(Dimensions)
            {
                AccessByPermission = TableData Dimension = R;
                ApplicationArea = Dimensions;
                Caption = 'Dimensions';
                Enabled = Rec."No." <> '';
                Image = Dimensions;
                Promoted = true;
                PromotedCategory = Category6;
                PromotedIsBig = true;
                ShortCutKey = 'Alt+D';
                ToolTip = 'View or edit dimensions, such as area, project,
or department, that you can assign to sales and purchase documents to
distribute costs and analyze transaction history.';

                trigger OnAction()
                begin
                    Rec.ShowDocDim;
                    CurrPage.SaveRecord;
                end;
            }
        }
```

```
        }
    }
```

Додаток В

Лістинг коду функціональності «Облік ПН».

```
codeunit 21038251 "KDA Post Sales VAT Invoices"
{

    TableNo = "KDA Sls VAT Invoice Header";

    trigger OnRun()
    begin
        IsDocumentPosted := true;
        Code(Rec);
    end;

    var
        TaxSetup: Record "Tax Setup";
        GLSetup: Record "General Ledger Setup";
        VATInvoiceHeader: Record "KDA Sls VAT Invoice Header";
        VATPeriod: Record "KDA VAT Period UA";
        DetermineperiodoftaxInvRep: Report "KDA Determ.PeriodOfTax Inv.";
        VATManagement: Codeunit "KDA VAT Management";
        BatchMode: Boolean;
        IsDocumentPosted: Boolean;
        PreviewMode: Boolean;
        DateFrom: Date;
        VATCorrCoef: Decimal;
        PostedStatus: Enum "KDA Status";
        Text000Lbl: Label '%1 %2 was not successfully posted.', Comment =
'%1 - "VAT Invoice Type", %2 - "No."';
        Text001Lbl: Label '%1 %2 was successfully posted.', Comment = '%1 -
"VAT Invoice Type", %2 - "No."';
        AppInvPostedErr: Label 'Applied Invoice No. must be posted.';
        VATPeriodClosedErr: Label 'Required VAT Period is already closed.';
        Text002Lbl: Label 'VAT Invoice %1 will be blocked.', Comment = '%1 -
"No."';
        Text003Lbl: Label 'You can''t post VAT Invoice %1 as it doesn''t
belong to current VAT Period. You can use the "Change VAT Period" function
```

```
and then post the VAT Invoice through "Close Period" function.', Comment =
'%1 - "No."';

        Text004Lbl: Label 'You can''t post VAT Invoice %1 due to expiration
of registration of the document.', Comment = '%1 - "No."';


    local procedure "Code"(_VATInvoiceHeader: Record "KDA Sls VAT Invoice
Header")

    var

        AppliedSalesInvHdr: Record "KDA Sls VAT Invoice Header";

        RegistrationDateIsChecked: Boolean;

    begin

        _VATInvoiceHeader.TestField("Issue Date");

        OnBeforeTestFieldRegistrationDate(RegistrationDateIsChecked);

        if not RegistrationDateIsChecked then

            _VATInvoiceHeader.TestField("Registration Date");

        _VATInvoiceHeader.TestField("Status",
_VATInvoiceHeader.Status::Released);


        if _VATInvoiceHeader."VAT Invoice Type" = _VATInvoiceHeader."VAT
Invoice Type"::Appendix2 then

            if AppliedSalesInvHdr.Get(_VATInvoiceHeader."VAT Invoice
Type"::Invoice, _VATInvoiceHeader."Applied Invoice No.") then

                if AppliedSalesInvHdr.Status <>
AppliedSalesInvHdr.Status::Posted then

                    Error(AppInvPostedErr);


        DateFrom := CalcDate('<CM+1D-1M>', _VATInvoiceHeader."Issue Date");

        DetermineperiodoftaxInvRep.UseRequestPage(false);


        VATPeriod.Reset();

        VATPeriod.SetFilter("Starting Date", '<=%1',
_VATInvoiceHeader."Issue Date");

        VATPeriod.SetFilter("Ending Date", '>=%1', _VATInvoiceHeader."Issue
Date");

        VATPeriod.FindFirst();

        DetermineperiodoftaxInv(VATPeriod, _VATInvoiceHeader, DateFrom);


        VATPeriodCheck(_VATInvoiceHeader);


        VATInvoiceHeader.Get(_VATInvoiceHeader."VAT Invoice Type",
_VATInvoiceHeader."No.");
```

```
        GLSetup.Get();
        TaxSetup.GET();
        if VATInvoiceHeader."Registration Date" <> 0D then
            if (VATInvoiceHeader."Issue Date" -
VATInvoiceHeader."Registration Date")
                < TaxSetup."KDA Days btw reg and issue"
            then
                if (VATInvoiceHeader."Issue Date" < VATPeriod."Starting
Date") or (VATInvoiceHeader."Issue Date" > VATPeriod."Ending Date") then
                    if not BatchMode then
                        Error(Text003Lbl, VATInvoiceHeader."No.")
                    else begin
                        IsDocumentPosted := false;
                        exit;
                    end;
        if VATInvoiceHeader."Registration Date" <> 0D then
              (VATInvoiceHeader."Issue Date" -
VATInvoiceHeader."Registration Date")
                  < TaxSetup."KDA Days btw reg and issue"
            then
                if (VATInvoiceHeader."Issue Date" >= VATPeriod."Starting
Date") and (VATInvoiceHeader."Issue Date" <= VATPeriod."Ending Date") and
                   (VATInvoiceHeader."Starting Date of VAT Period" = 0D)
                   and (VATInvoiceHeader."Corr. Start Date of VAT Period" =
0D)
                then
                    if not BatchMode then
                        Error(Text004Lbl, VATInvoiceHeader."No.")
                    else begin
                        IsDocumentPosted := false;
                        exit;
                    end;

        if (VATInvoiceHeader."Status of Incl. in VAT period" <>
VATInvoiceHeader."Status of Incl. in VAT period"::" ") and
            ((VATInvoiceHeader."Starting Date of VAT Period" <> 0D)
            or (VATInvoiceHeader."Corr. Start Date of VAT Period" <> 0D))
then begin
            PostedStatus := PostedStatus::Posted;
```

```
            GLSetup.Get();
            VATCorrCoef := 1;

            VATManagement.SetPreviewMode(PreviewMode);
            VATManagement.SetBatchMode(BatchMode);

            VATInvoiceHeader.Modify();
            if VATInvoiceHeader."Corr. Start Date of VAT Period" <> 0D then
begin
                VATInvoiceHeader."Posting Date" := CalcDate('<CM>',
VATInvoiceHeader."Corr. Start Date of VAT Period");
                VATInvoiceHeader.Modify();
                VATManagement.PostVATInvHeader(VATInvoiceHeader.RecordId,
VATPeriod, false, VATCorrCoef, PostedStatus, VATInvoiceHeader."Posting
Date");
            end else
                VATManagement.PostVATInvHeader(VATInvoiceHeader.RecordId,
VATPeriod, false, VATCorrCoef, PostedStatus, VATPeriod."Ending Date");

            VATInvoiceHeader.GET(VATInvoiceHeader."VAT Invoice Type",
VATInvoiceHeader."No.");

            if (not PreviewMode) and (VATInvoiceHeader."Corr. Start Date of
VAT Period" = 0D) then begin
                VATInvoiceHeader."Posting Date" := VATPeriod."Ending Date";
                VATInvoiceHeader.Modify();
            end;

            if (not PreviewMode) and (not BatchMode) then
                if GuiAllowed then
                    Message(Text001Lbl, VATInvoiceHeader."VAT Invoice Type",
VATInvoiceHeader."No.");
        end
        else
            if (not PreviewMode) and (not BatchMode) then
                if GuiAllowed then
                    Message(Text000Lbl, VATInvoiceHeader."VAT Invoice Type",
VATInvoiceHeader."No.");

    end;
```

```
    local procedure VATPeriodCheck(VATInvoiceHeader: Record "KDA Sls VAT
Invoice Header")

    var

        VATPeriod_loc: record "KDA VAT Period UA";

    begin

        if VATInvoiceHeader."Status of Incl. in VAT period" =
VATInvoiceHeader."Status of Incl. in VAT period"::"Included in VAT period"
then

            VATPeriod_loc.Get(VATInvoiceHeader."Starting Date of VAT
Period");

        if VATInvoiceHeader."Status of Incl. in VAT period" =
VATInvoiceHeader."Status of Incl. in VAT period"::"Changed VAT period" then

            VATPeriod_loc.Get(VATInvoiceHeader."Corr. Start Date of VAT
Period");


        if VATPeriod_loc.Closed then

            Error(VATPeriodClosedErr);

    end;


    procedure SetPreviewMode(_PreviewMode: Boolean)

    begin

        PreviewMode := _PreviewMode;

    end;


    procedure SetBatchMode(_BatchMode: Boolean)

    begin

        BatchMode := _BatchMode;

    end;


    procedure SetPreviewForCloseMode(_PreviewForCloseMode: Boolean)

    begin

    end;


    local procedure Determineperiodoftax Inv(_VATPeriod: Record "KDA VAT
Period UA"; var _VATInvoiceHeader: Record "KDA Sls VAT Invoice Header";
_DateFrom: Date)

    var

        GLSetup_loc: Record "General Ledger Setup";

        NextSetupDate: Date;

        SetupDate: Date;
```

```
    begin
        GLSetup_loc.Get();


        TaxSetup.GET();
        TaxSetup.TESTFIELD("KDA VAT Inv. Registr. Term");


        SetupDate := DMY2Date(
                            TaxSetup."KDA VAT Inv. Registr. Term",
                            Date2DMY(_DateFrom, 2),
                            Date2DMY(_DateFrom, 3)
                            );
        NextSetupDate := CalcDate('<+1M>', SetupDate);


        begin
            if VATInvoiceHeader."Registration Date" <> 0D then
                IF (_VATInvoiceHeader."Issue Date" -
_VATInvoiceHeader."Registration Date") >= TaxSetup."KDA Days btw reg and
issue" THEN
                    if _VATInvoiceHeader."Status of Incl. in VAT period" <>
_VATInvoiceHeader."Status of Incl. in VAT period"::"Changed VAT period" then
begin
                        Error(Text002Lbl, _VATInvoiceHeader."No.");
                        _VATInvoiceHeader."Status of Incl. in VAT period" :=
_VATInvoiceHeader."Status of Incl. in VAT period"::" ";
                        _VATInvoiceHeader.Modify();
                    end;


            case true of
                (_VATInvoiceHeader."Issue Date" <= SetupDate) and
(_VATInvoiceHeader."Posting Date" > _VATPeriod."Ending Date"):
                    exit;


                (SetupDate < _VATInvoiceHeader."Issue Date") and
(_VATInvoiceHeader."Issue Date" <= _VATPeriod."Ending Date") and
(_VATInvoiceHeader."Posting Date" > NextSetupDate):
                    exit;


                (_VATPeriod."Starting Date" <= _VATInvoiceHeader."Issue
Date") and (_VATInvoiceHeader."Issue Date" <= SetupDate) and
(_VATInvoiceHeader."Posting Date" <= _VATPeriod."Ending Date"):
                    begin
```

```
                        _VATInvoiceHeader.Validate("Starting Date of VAT
Period", VATPeriod."Starting Date");


                        if _VATInvoiceHeader."Status of Incl. in VAT period"
<> _VATInvoiceHeader."Status of Incl. in VAT period"::"Changed VAT period"
then

                            _VATInvoiceHeader."Status of Incl. in VAT
period" := _VATInvoiceHeader."Status of Incl. in VAT period"::"Included in
VAT period";


                        _VATInvoiceHeader.Modify();

                    end;


                (SetupDate < _VATInvoiceHeader."Issue Date") and
(_VATInvoiceHeader."Issue Date" <= _VATPeriod."Ending Date") and
(_VATInvoiceHeader."Posting Date" <= NextSetupDate):

                    begin

                        _VATInvoiceHeader.Validate("Starting Date of VAT
Period", _VATPeriod."Starting Date");


                        if _VATInvoiceHeader."Status of Incl. in VAT period"
<> _VATInvoiceHeader."Status of Incl. in VAT period"::"Changed VAT period"
then

                            _VATInvoiceHeader."Status of Incl. in VAT
period" := _VATInvoiceHeader."Status of Incl. in VAT period"::"Included in
VAT period";


                        _VATInvoiceHeader.Modify();

                    end;

            end;

            if _VATInvoiceHeader."Status of Incl. in VAT period" <>
_VATInvoiceHeader."Status of Incl. in VAT period"::"Changed VAT period" then
begin

                _VATInvoiceHeader."Corr. Start Date of VAT Period" := 0D;

                _VATInvoiceHeader.Modify();

            end;

        end;


    end;


    procedure GetIsDocumentPosted(): Boolean

    begin

        exit(IsDocumentPosted);
```

```
        end;


        [IntegrationEvent(false, false)]
        local procedure OnBeforeTestFieldRegistrationDate(var
  RegistrationDateIsChecked: Boolean)
        begin
end;
```

Додаток Г

Лістинг коду автотестів для перевірки функціональності обліку ПН.

```
codeunit 50000 "KDA VAT Inv Posting Test"
{
    Description = 'VAT Inv Posting Test';
    Subtype = Test;

    [Test]
    [TransactionModel(TransactionModel::AutoRollback)]
[HandlerFunctions('VATDocPostYesNo_ConfirmHandler,VATDocPostYesNo_MessageHandler')
]
    procedure SalesVATInvoice_Posting_Test()
    // [FEATURE] [VAT Sales Invoice]
    // [SCENARIO] Create and post VAT Sales Invoice
    // [PREREQ] Prerequisites
    // [COMMENT] Comment
    var
        SalesVATInvHeader: Record "KDA Sls VAT Invoice Header";
        SalesVATInvLine: Record "KDA Sls VAT Invoice Line";
        VATDocPostYesNo: Codeunit "KDA Sls. VAT Doc.-Post Y/N";
        VATPostingSetup2: Record "VAT Posting Setup";
        GLEntry: Record "G/L Entry";
    begin
        // [GIVEN] Filled VAT Sales Invoice
        CreateSalesVATInvoice(SalesVATInvHeader);

        // [WHEN] Posting VAT Sales Invoice
        VATDocPostYesNo.Run(SalesVATInvHeader);

        // [THEN] Appropriate entries are created
        GLEntry.SetRange(Description, Format(SalesVATInvHeader."VAT Invoice Type")
+ ', ' + SalesVATInvHeader."No.");
        if GLEntry.FindSet() then begin
```

```
            SalesVATInvLine.SetRange("VAT Invoice Type", SalesVATInvHeader."VAT
Invoice Type");

            SalesVATInvLine.SetRange("Document No.", SalesVATInvHeader."No.");

            if SalesVATInvLine.FindFirst() then;


            VATPostingSetup2.SetRange("VAT Bus. Posting Group",
SalesVATInvHeader."VAT Bus. Posting Group");

            VATPostingSetup2.SetRange("VAT Prod. Posting Group",
SalesVATInvLine."VAT Product Posting Group");

            VATPostingSetup2.SetRange("KDA VAT Posting Type", "KDA VAT Posting
Type"::Sale);

            if VATPostingSetup2.FindFirst() then;


            repeat

                if GLEntry.Amount > 0 then begin

                    AssertEquals(GLEntry.Amount, SalesVATInvLine."VAT Amount");

                    AssertEquals(GLEntry."G/L Account No.",
VATPostingSetup2."Sales VAT Account");

                    AssertEquals(GLEntry."Bal. Account No.", VATPostingSetup2."KDA
Settlement Account");

                end

                else begin

                    AssertEquals(GLEntry.Amount, -SalesVATInvLine."VAT Amount");

                    AssertEquals(GLEntry."G/L Account No.", VATPostingSetup2."KDA
Settlement Account");

                    AssertEquals(GLEntry."Bal. Account No.",
VATPostingSetup2."Sales VAT Account");

                end;

            until GLEntry.Next() = 0;

        end;

    end;


    [Test]

    [TransactionModel(TransactionModel::AutoRollback)]

[HandlerFunctions('VATDocPostYesNo_ConfirmHandler,VATDocPostYesNo_MessageHandler,P
ostedSalesDoc_PageHandler')]

    procedure SalesVATAppendix2_Posting_Test()

    // [FEATURE] [VAT Sales Appendix2]

    // [SCENARIO] Create and post VAT Sales Appendix2
```

```
    // [PREREQ] Prerequisites
    // [COMMENT] Comment
    var
        SalesVATInvHeader: Record "KDA Sls VAT Invoice Header";
        SalesVATInvHeader2: Record "KDA Sls VAT Invoice Header";
        SalesVATInvLine2: Record "KDA Sls VAT Invoice Line";
        VATDocPostYesNo: Codeunit "KDA Sls. VAT Doc.-Post Y/N";
        SalesVATInvTestPage: TestPage "KDA Sales VAT Invoice";
        VATPostingSetup2: Record "VAT Posting Setup";
        GLEntry: Record "G/L Entry";
    begin
        // [GIVEN] Filled VAT Sales Appendix2
        CreateSalesVATInvoice(SalesVATInvHeader);
        VATDocPostYesNo.Run(SalesVATInvHeader);

        SalesVATInvoice_PageHandler(SalesVATInvTestPage, SalesVATInvHeader);

        SalesVATInvHeader2.SetRange("Applied Invoice No.",
SalesVATInvHeader."No.");
        if SalesVATInvHeader2.FindLast() then;
        SalesVATInvHeader2.Validate("Registration Date", WorkDate());
        SalesVATInvHeader2.Validate("Issue Date", WorkDate());
        SalesVATInvHeader2.Validate(Status, SalesVATInvHeader.Status::Released);

        SalesVATInvLine2.SetRange(SalesVATInvLine2."VAT Invoice Type",
SalesVATInvHeader2."VAT Invoice Type");
        SalesVATInvLine2.SetRange("Document No.", SalesVATInvHeader2."No.");
        if SalesVATInvLine2.FindFirst() then
            SalesVATInvLine2.Validate("Price Excl. VAT", 200);

        // [WHEN] Posting VAT Sales Appendix2
        VATDocPostYesNo.Run(SalesVATInvHeader2);

        // [THEN] Appropriate entries are created
        GLEntry.SetRange(Description, Format(SalesVATInvHeader2."VAT Invoice
Type") + ', ' + SalesVATInvHeader2."No.");
        if GLEntry.FindSet() then begin
            VATPostingSetup2.SetRange("VAT Bus. Posting Group",
SalesVATInvHeader2."VAT Bus. Posting Group");
```

```
            VATPostingSetup2.SetRange("VAT Prod. Posting Group",
SalesVATInvLine2."VAT Product Posting Group");

            VATPostingSetup2.SetRange("KDA VAT Posting Type", "KDA VAT Posting
Type"::Sale);

            if VATPostingSetup2.FindFirst() then;

            repeat

                if GLEntry.Amount < 0 then begin

                    AssertEquals(GLEntry.Amount, SalesVATInvLine2."VAT Amount");

                    AssertEquals(GLEntry."G/L Account No.",
VATPostingSetup2."Sales VAT Account");

                    AssertEquals(GLEntry."Bal. Account No.", VATPostingSetup2."KDA
Settlement Account");

                end

                else begin

                    AssertEquals(GLEntry.Amount, -SalesVATInvLine2."VAT Amount");

                    AssertEquals(GLEntry."G/L Account No.", VATPostingSetup2."KDA
Settlement Account");

                    AssertEquals(GLEntry."Bal. Account No.",
VATPostingSetup2."Sales VAT Account");

                end;

            until GLEntry.Next() = 0;

        end;

    end;


    [Test]

    [TransactionModel(TransactionModel::AutoRollback)]

[HandlerFunctions('VATDocPostYesNo_ConfirmHandler,VATDocPostYesNo_MessageHandler')
]

    procedure PurchVATInvoice_Posting_Test()

    // [FEATURE] [VAT Purchase Invoice]

    // [SCENARIO] Create and post VAT Purchase Invoice

    // [PREREQ] Prerequisites

    // [COMMENT] Comment

    var

        PurchVATInvHeader: Record "KDA Purch VAT Inv Header";

        PurchVATInvLine: Record "KDA Purch VAT Invoice Line";

        VATDocPostYesNo: Codeunit "KDA Prch VAT Doc.-Post Y/N";

        VATPostingSetup2: Record "VAT Posting Setup";

        GLEntry: Record "G/L Entry";
```

```
    begin
        // [GIVEN] Filled VAT Purchase Invoice
        CreatePurchVATInvoice(PurchVATInvHeader);

        // [WHEN] Posting VAT Purchase Invoice
        VATDocPostYesNo.Run(PurchVATInvHeader);

        // [THEN] Appropriate entries are created
        GLEntry.SetRange(Description, Format(PurchVATInvHeader."VAT Invoice Type")
+ ', ' + PurchVATInvHeader."No.");
        if GLEntry.FindSet() then begin

            PurchVATInvLine.SetRange("VAT Invoice Type", PurchVATInvHeader."VAT
Invoice Type");
            PurchVATInvLine.SetRange("Document No.", PurchVATInvHeader."No.");
            if PurchVATInvLine.FindFirst() then;

            VATPostingSetup2.SetRange("VAT Bus. Posting Group",
PurchVATInvHeader."VAT Bus. Posting Group");
            VATPostingSetup2.SetRange("VAT Prod. Posting Group",
PurchVATInvLine."VAT Product Posting Group");
            VATPostingSetup2.SetRange("KDA VAT Posting Type", "KDA VAT Posting
Type"::Purchase);
            if VATPostingSetup2.FindFirst() then;

            repeat
                if GLEntry.Amount > 0 then begin
                    AssertEquals(GLEntry.Amount, PurchVATInvLine."VAT Amount");
                    AssertEquals(GLEntry."G/L Account No.", VATPostingSetup2."KDA
Settlement Account");
                    AssertEquals(GLEntry."Bal. Account No.",
VATPostingSetup2."Purchase VAT Account");
                end
                else begin
                    AssertEquals(GLEntry.Amount, -PurchVATInvLine."VAT Amount");
                    AssertEquals(GLEntry."G/L Account No.",
VATPostingSetup2."Purchase VAT Account");
                    AssertEquals(GLEntry."Bal. Account No.", VATPostingSetup2."KDA
Settlement Account");
                end;
            until GLEntry.Next() = 0;
```

```
        end;
    end;

    [Test]
    [TransactionModel(TransactionModel::AutoRollback)]
[HandlerFunctions('VATDocPostYesNo_ConfirmHandler,VATDocPostYesNo_MessageHandler,P
ostedPurchDoc_PageHandler')]
    procedure PurchVATAppendix2_Posting_Test()
    // [FEATURE] [VAT Purchase Appendix2]
    // [SCENARIO] Create and post VAT Purchase Appendix2
    // [PREREQ] Prerequisites
    // [COMMENT] Comment
    var
        PurchVATInvHeader: Record "KDA Purch VAT Inv Header";
        PurchVATInvHeader2: Record "KDA Purch VAT Inv Header";
        PurchVATInvLine2: Record "KDA Purch VAT Invoice Line";
        VATDocPostYesNo: Codeunit "KDA Prch VAT Doc.-Post Y/N";
        PurchVATInvTestPage: TestPage "KDA Purch. VAT Invoice";
        VATPostingSetup2: Record "VAT Posting Setup";
        GLEntry: Record "G/L Entry";
    begin
        // [GIVEN] Filled VAT Purchase Appendix2
        CreatePurchVATInvoice(PurchVATInvHeader);
        VATDocPostYesNo.Run(PurchVATInvHeader);

        PurchVATInvoice_PageHandler(PurchVATInvTestPage, PurchVATInvHeader);

        PurchVATInvHeader2.SetRange("Applied Invoice No.",
PurchVATInvHeader."No.");
        if PurchVATInvHeader2.FindLast() then;
        PurchVATInvHeader2.Validate("Registration Date", WorkDate());
        PurchVATInvHeader2.Validate("Issue Date", WorkDate());
        PurchVATInvHeader2.Validate(Status, PurchVATInvHeader.Status::Released);

        PurchVATInvLine2.SetRange(PurchVATInvLine2."VAT Invoice Type",
PurchVATInvHeader2."VAT Invoice Type");
        PurchVATInvLine2.SetRange("Document No.", PurchVATInvHeader2."No.");
        if PurchVATInvLine2.FindFirst() then
```

```
            PurchVATInvLine2.Validate("Price Excl. VAT", 200);


        // [WHEN] Posting VAT Purchase Appendix2

        VATDocPostYesNo.Run(PurchVATInvHeader2);


        // [THEN] Appropriate entries are created

        GLEntry.SetRange(Description, Format(PurchVATInvHeader2."VAT Invoice
Type") + ', ' + PurchVATInvHeader2."No.");

        if GLEntry.FindSet() then begin

            VATPostingSetup2.SetRange("VAT Bus. Posting Group",
PurchVATInvHeader2."VAT Bus. Posting Group");

            VATPostingSetup2.SetRange("VAT Prod. Posting Group",
PurchVATInvLine2."VAT Product Posting Group");

            VATPostingSetup2.SetRange("KDA VAT Posting Type", "KDA VAT Posting
Type"::Purchase);

            if VATPostingSetup2.FindFirst() then;

            repeat

                if GLEntry.Amount < 0 then begin

                    AssertEquals(GLEntry.Amount, PurchVATInvLine2."VAT Amount");

                    AssertEquals(GLEntry."G/L Account No.", VATPostingSetup2."KDA
Settlement Account");

                    AssertEquals(GLEntry."Bal. Account No.",
VATPostingSetup2."Purchase VAT Account");

                end

                else begin

                    AssertEquals(GLEntry.Amount, -PurchVATInvLine2."VAT Amount");

                    AssertEquals(GLEntry."G/L Account No.",
VATPostingSetup2."Purchase VAT Account");

                    AssertEquals(GLEntry."Bal. Account No.", VATPostingSetup2."KDA
Settlement Account");

                end;

            until GLEntry.Next() = 0;

        end;

    end;


    [PageHandler]

    procedure PostedSalesDoc_PageHandler(var Page: TestPage "KDA Sales VAT
Invoice")

    begin

        Page.Close();

    end;
```

```
    procedure SalesVATInvoice_PageHandler(var SalesVATInvTP: TestPage "KDA Sales
VAT Invoice"; SlsVATInvHdr: Record "KDA Sls VAT Invoice Header")

    begin

        SalesVATInvTP.OpenView();

        SalesVATInvTP.GoToRecord(SlsVATInvHdr);

        SalesVATInvTP.CreateVATAppendix2.Invoke();

        SalesVATInvTP.Trap();

    end;


    [PageHandler]

    procedure PostedPurchDoc_PageHandler(var Page: TestPage "KDA Purch. VAT
Invoice")

    begin

        Page.Close();

    end;


    procedure PurchVATInvoice_PageHandler(var PurchVATInvTP: TestPage "KDA Purch.
VAT Invoice"; PrchVATInvHdr: Record "KDA Purch VAT Inv Header")

    begin

        PurchVATInvTP.OpenView();

        PurchVATInvTP.GoToRecord(PrchVATInvHdr);

        PurchVATInvTP.CreateVATAppendix2.Invoke();

        PurchVATInvTP.Trap();

    end;


    [ConfirmHandler]

    procedure VATDocPostYesNo_ConfirmHandler(Question: Text[1024]; VAR Reply:
Boolean)

    begin

        Reply := true;

    end;


    [MessageHandler]

    procedure VATDocPostYesNo_MessageHandler(MessageText: Text[1024])

    begin


    end;
```

```
    procedure InitializeVATSetupForSales(VATBusPostGroup: Code[20];
VATProdPostGroup: Code[20]; GLAcc1: Code[20]; GLAcc2: Code[20]; GLAcc3: Code[20])

    var

        VATPostSetup: Record "VAT Posting Setup";

    begin

        // to do if empty

        VATPostSetup.Init();

        VATPostSetup.Validate("VAT Bus. Posting Group", VATBusPostGroup);

        VATPostSetup.Validate("VAT Prod. Posting Group", VATProdPostGroup);

        VATPostSetup.Validate("KDA VAT Posting Type", VATPostSetup."KDA VAT
Posting Type"::Sale);

        VATPostSetup.Validate("KDA G/L Expense Account", GLAcc1);

        VATPostSetup.Validate("KDA Settlement Account", GLAcc2);

        VATPostSetup.Validate("KDA Tax Deductable", true);

        VATPostSetup.Validate("VAT %", 21);

        VATPostSetup.Validate("Sales VAT Account", GLAcc3);

        VATPostSetup.Insert(true);

    end;


    procedure InitializeVATSetupForPurchases(VATBusPostGroup: Code[20];
VATProdPostGroup: Code[20]; GLAcc1: Code[20]; GLAcc2: Code[20]; GLAcc3: Code[20])

    var

        VATPostSetup: Record "VAT Posting Setup";

    begin

        // to do if empty

        VATPostSetup.Init();

        VATPostSetup.Validate("VAT Bus. Posting Group", VATBusPostGroup);

        VATPostSetup.Validate("VAT Prod. Posting Group", VATProdPostGroup);

        VATPostSetup.Validate("KDA VAT Posting Type", VATPostSetup."KDA VAT
Posting Type"::Purchase);

        VATPostSetup.Validate("KDA G/L Expense Account", GLAcc1);

        VATPostSetup.Validate("KDA Settlement Account", GLAcc2);

        VATPostSetup.Validate("KDA Tax Deductable", true);

        VATPostSetup.Validate("VAT %", 21);

        VATPostSetup.Validate("Purchase VAT Account", GLAcc3);

        VATPostSetup.Insert(true);

    end;
```

```
    procedure CreateSalesVATInvHeader(var SalesVATInvHeader: Record "KDA Sls VAT
Invoice Header"): Code[20]
    begin
        SalesVATInvHeader.Init();
        TextGen += 1;
        SalesVATInvHeader."No." := Format(TextGen);
        SalesVATInvHeader.Insert(true);
        exit(SalesVATInvHeader."No.");
    end;


    procedure CreateSalesVATInvLine(var SalesVATInvLine: Record "KDA Sls VAT
Invoice Line"; SalesVATInvHeader: Record "KDA Sls VAT Invoice Header"): Integer
    begin
        SalesVATInvLine.Init();
        SalesVATInvLine."VAT Invoice Type" := SalesVATInvHeader."VAT Invoice
Type";
        SalesVATInvLine."Document No." := SalesVATInvHeader."No.";
        LastLineNo += 10000;
        SalesVATInvLine."Line No." := LastLineNo;
        SalesVATInvLine.Insert(true);
        exit(SalesVATInvLine."Line No.");
    end;


    procedure FillSalesVATInvLine(var SalesVATInvLine: Record "KDA Sls VAT Invoice
Line"; GLAcc: Record "G/L Account"; Item: Record Item)
    begin
        SalesVATInvLine.Validate(Type, SalesVATInvLine.Type::Item);
        SalesVATInvLine.Validate("No.", Item."No.");
        SalesVATInvLine.Validate(Quantity, 5);
        SalesVATInvLine.Validate("Price Excl. VAT", 100);
        SalesVATInvLine.Modify(true);
    end;


    procedure CreateCustomer(var Customer: Record Customer)
    begin
        Customer.Init();
        TextGen += 1;
        Customer."No." := Format(TextGen);
        Customer.Insert(true);
```

```
    end;


    procedure CreateVendor(var Vendor: Record Vendor)
    begin
        Vendor.Init();
        TextGen += 1;
        Vendor."No." := Format(TextGen);
        Vendor.Insert(true);
    end;


    procedure CreatePurchVATInvHeader(var PurchVATInvHeader: Record "KDA Purch VAT
Inv Header"): Code[20]
    begin
        PurchVATInvHeader.Init();
        TextGen += 1;
        PurchVATInvHeader."No." := Format(TextGen);
        PurchVATInvHeader.Insert(true);
        exit(PurchVATInvHeader."No.");
    end;


    procedure CreatePurchVATInvLine(var PurchVATInvLine: Record "KDA Purch VAT
Invoice Line"; PurchVATInvHeader: Record "KDA Purch VAT Inv Header"): Integer
    begin
        PurchVATInvLine.Init();
        PurchVATInvLine."VAT Invoice Type" := PurchVATInvHeader."VAT Invoice
Type";
        PurchVATInvLine."Document No." := PurchVATInvHeader."No.";
        LastLineNo += 10000;
        PurchVATInvLine."Line No." := LastLineNo;
        PurchVATInvLine.Insert(true);
        exit(PurchVATInvLine."Line No.");
    end;


    procedure FillPurchVATInvLine(var PurchVATInvLine: Record "KDA Purch VAT
Invoice Line"; GLAcc: Record "G/L Account"; Item: Record Item)
    begin
        PurchVATInvLine.Validate(Type, PurchVATInvLine.Type::Item);
        PurchVATInvLine.Validate("No.", Item."No.");
        PurchVATInvLine.Validate(Quantity, 5);
```

```
    PurchVATInvLine.Validate("Price Excl. VAT", 100);
    PurchVATInvLine.Modify(true);
end;

procedure CreateReasonCode(var ReasonCode: Record "Reason Code")
begin
    ReasonCode.Init();
    TextGen += 1;
    ReasonCode.Code := Format(TextGen);
    ReasonCode.Insert(true);
end;

procedure CreateUnitOfMeasure(var UnitOfMeasure: Record "Unit of Measure")
begin
    UnitOfMeasure.Init();
    UnitOfMeasure.Validate(Code, 'БУТЛЬ_ТЕСТ');
    UnitOfMeasure.Insert(true);
end;

procedure CreateGLAccount(var GLAcc: Record "G/L Account"): Code[20]
begin
    GLAcc.Init();
    TextGen += 1;
    GLAcc."No." := Format(TextGen);
    GLAcc.Insert(true);
    exit(GLAcc."No.");
end;

procedure CreateItem(var Item: Record Item): Code[20]
begin
    Item.Init();
    Item."No." := 'Вода_ТЕСТ';
    Item.Insert(true);
    exit(Item."No.");
end;
```

```
    procedure CreateClassOfGoodsFEA(var CreateClassOfGoodsFEA: Record "KDA
Classific. of Goods FEA"): Code[20]

    begin

        CreateClassOfGoodsFEA.Init();

        CreateClassOfGoodsFEA."Code" := '220101';

        CreateClassOfGoodsFEA.Insert(true);

        exit(CreateClassOfGoodsFEA."Code");

    end;


    procedure CreateGenBusPostGroup(var GenBusPostGroup: Record "Gen. Business
Posting Group"): Code[20]

    begin

        GenBusPostGroup.Init();

        GenBusPostGroup."Code" := 'КЛІЄНТ_ВІТА';

        GenBusPostGroup.Insert(true);

        exit(GenBusPostGroup."Code");

    end;


    procedure CreateVATProdPostGroup(var VATProdPostGroup: Record "VAT Product
Posting Group"): Code[20]

    begin

        VATProdPostGroup.Init();

        VATProdPostGroup."Code" := '21%';

        VATProdPostGroup.Insert(true);

        exit(VATProdPostGroup."Code");

    end;


    procedure CreateGenProdPostGroup(var GenProdPostGroup: Record "Gen. Product
Posting Group"): Code[20]

    begin

        GenProdPostGroup.Init();

        GenProdPostGroup."Code" := 'ТОВАРИ_ТЕСТ';

        GenProdPostGroup.Insert(true);

        exit(GenProdPostGroup."Code");

    end;


    procedure CreateVATBusPostGroup(var VATBusPostGroup: Record "VAT Business
Posting Group"): Code[20]

    begin
```

```
        VATBusPostGroup.Init();
        VATBusPostGroup."Code" := 'К_ПДАТА';
        VATBusPostGroup.Insert(true);
        exit(VATBusPostGroup."Code");
    end;

    procedure CreateSalesVATInvoice(var SlsVATInv: Record "KDA Sls VAT Invoice
Header"): Code[20]
    var
        TaxSetup: Record "Tax Setup";
        NoSeries: Record "No. Series";
        VATPeriod: Record "KDA VAT Period UA";
        GenJnlTemplate: Record "Gen. Journal Template";
        GenJnlBatch: Record "Gen. Journal Batch";
        SourceCode: Record "Source Code";
        VATJnlSetup: Record "KDA VAT Journal Setup";
        Customer: Record Customer;
        GLAcc1: Record "G/L Account";
        GLAcc2: Record "G/L Account";
        GLAcc3: Record "G/L Account";
        GenBusPostGroup: Record "Gen. Business Posting Group";
        GenProdPostGroup: Record "Gen. Product Posting Group";
        Item: Record Item;
        ClassOfGoodsFEA: Record "KDA Classific. of Goods FEA";
        ReasonCode: Record "Reason Code";
        UnitOfMeasure: Record "Unit of Measure";
        VATBusPostGroup: Record "VAT Business Posting Group";
        VATProdPostGroup: Record "VAT Product Posting Group";
        SlsVATInvLine: Record "KDA Sls VAT Invoice Line";
    begin
        NoSeries.FindFirst();
        TaxSetup.Get();
        TaxSetup."KDA Sls VAT Ext. Doc. Nos." := NoSeries.Code;
        TaxSetup."KDA Sales VAT Invoice Nos." := NoSeries.Code;
        TaxSetup."KDA Sales Appendix 2 Nos." := NoSeries.Code;
        TaxSetup."KDA Sls VT App2 Ext Doc Nos" := NoSeries.Code;
        TaxSetup."KDA VAT Inv. Registr. Term" := 20;
```

```
        Evaluate(TaxSetup."KDA VAT Period", '1M');
        TaxSetup."KDA Days btw reg and issue" := 365;
        TaxSetup.Modify(true);

        VATPeriod.Init();
        VATPeriod."Starting Date" := CALCDATE('<-CM>', WorkDate());
        VATPeriod."Ending Date" := CALCDATE('<CM>', WorkDate());
        VATPeriod.Name := 'TestMonth';
        VATPeriod.Insert();

        SourceCode.Init();
        SourceCode.Code := 'ADJADDCURR';
        SourceCode.Description := 'Adjust Add. Reporting Currency';
        SourceCode.Insert();

        GenJnlTemplate.Init();
        GenJnlTemplate.Name := 'НДС';
        GenJnlTemplate.Description := 'НДС';
        GenJnlTemplate.Type := GenJnlTemplate.Type::General;
        GenJnlTemplate."Bal. Account Type" := GenJnlTemplate."Bal. Account
Type"::"G/L Account";
        GenJnlTemplate."Source Code" := SourceCode.Code;
        GenJnlTemplate."Force Doc. Balance" := true;
        GenJnlTemplate."Copy VAT Setup to Jnl. Lines" := true;
        GenJnlTemplate.Insert();

        GenJnlBatch.Init();
        GenJnlBatch."Journal Template Name" := GenJnlTemplate.Name;
        GenJnlBatch.Name := 'СТАНДАРТ';
        GenJnlBatch.Description := 'Журнал за замовчуванням';
        GenJnlBatch."Bal. Account Type" := GenJnlBatch."Bal. Account Type"::"G/L
Account";
        GenJnlBatch."Copy VAT Setup to Jnl. Lines" := true;
        GenJnlBatch."Template Type" := GenJnlBatch."Template Type"::General;
        GenJnlBatch.Insert();

        VATJnlSetup.Init();
        VATJnlSetup."User ID" := '';
```

```
        VATJnlSetup."Sales VAT Invoice Template" := GenJnlTemplate.Name;
        VATJnlSetup."Sales VAT Invoice Batch" := GenJnlBatch.Name;
        VATJnlSetup."Sales App 2 Template" := GenJnlTemplate.Name;
        VATJnlSetup."Sales App 2 Batch" := GenJnlBatch.Name;
        VATJnlSetup.Insert();

        CreateUnitOfMeasure(UnitOfMeasure);

        CreateClassOfGoodsFEA(ClassOfGoodsFEA);
        ClassOfGoodsFEA.Validate(Description, 'тест_води');
        ClassOfGoodsFEA.Modify(true);

        CreateVATProdPostGroup(VATProdPostGroup);
        VATProdPostGroup.Validate(Description, 'Оподатковується за ставкою 21%');
        VATProdPostGroup.Modify(true);

        CreateGenProdPostGroup(GenProdPostGroup);
        GenProdPostGroup.Validate(Description, 'Товари_тест');
        GenProdPostGroup.Validate("Def. VAT Prod. Posting Group",
VATProdPostGroup.Code);
        GenProdPostGroup.Modify(true);

        CreateVATBusPostGroup(VATBusPostGroup);
        VATBusPostGroup.Validate(Description, 'VATBusPostGroup123');
        VATBusPostGroup.Modify(true);

        CreateItem(Item);
        Item.Validate(Description, 'Водичка');
        Item.Validate(Type, Item.Type::Inventory);
        Item.Validate("Base Unit of Measure", UnitOfMeasure.Code);
        Item.Validate("KDA UCGFEA Code", ClassOfGoodsFEA.Code);
        Item.Validate("Gen. Prod. Posting Group", GenProdPostGroup.Code);
        Item.Validate("VAT Prod. Posting Group", VATProdPostGroup.Code);
        Item.Modify(true);

        ReasonCode.DeleteAll();
        CreateReasonCode(ReasonCode);
```

```
        ReasonCode.Validate("KDA Reason Category", ReasonCode."KDA Reason
Category"::VAT);

        ReasonCode.Validate("KDA VAT Reason of Corr.", ReasonCode."KDA VAT Reason
of Corr."::"Item Change");

        ReasonCode.Modify(true);


        CreateGenBusPostGroup(GenBusPostGroup);

        GenBusPostGroup.Validate(Description, 'Вітчизняні клієнти123');

        GenBusPostGroup.Validate("Def. VAT Bus. Posting Group",
VATBusPostGroup.Code);

        GenBusPostGroup.Modify(true);


        CreateCustomer(Customer);

        Customer.Validate(Name, 'Ромашка ТОВ');

        Customer.Validate(Address, 'вул. Лесі Українки, 128-г');

        Customer.Validate("KDA Registration No.", '12345777');

        Customer.Validate("VAT Registration No.", '384899509');

        Customer.Validate("Gen. Bus. Posting Group", GenBusPostGroup.Code);

        Customer.Validate("VAT Bus. Posting Group", VATBusPostGroup.Code);

        Customer.Validate("KDA Tax Number Source", Customer."KDA Tax Number
Source"::"1");

        Customer.Modify(true);


        CreateGLAccount(GLAcc1);

        CreateGLAccount(GLAcc2);

        CreateGLAccount(GLAcc3);

        InitializeVATSetupForSales(VATBusPostGroup.Code, VATProdPostGroup.Code,
GLAcc1."No.", GLAcc2."No.", GLAcc3."No.");


        CreateSalesVATInvHeader(SlsVATInv);

        SlsVATInv.Validate("No.");

        SlsVATInv.Validate("VAT Invoice Type", SlsVATInv."VAT Invoice
Type"::Invoice);

        SlsVATInv.Validate("Issue Date", WorkDate());

        SlsVATInv.Validate("Registration Date", WorkDate());

        SlsVATInv.Validate("Customer No.", Customer."No.");

        SlsVATInv.Validate("Document Type", SlsVATInv."Document Type"::PNE);

        SlsVATInv.Validate(Status, SlsVATInv.Status::Released);

        SlsVATInv.Modify(true);
```

```
        CreateSalesVATInvLine(SlsVATInvLine, SlsVATInv);

        FillSalesVATInvLine(SlsVATInvLine, GLAcc1, Item);
    end;

    procedure CreatePurchVATInvoice(var PrchVATInv: Record "KDA Purch VAT Inv
Header"): Code[20]
    var
        TaxSetup: Record "Tax Setup";
        NoSeries: Record "No. Series";
        VATPeriod: Record "KDA VAT Period UA";
        GenJnlTemplate: Record "Gen. Journal Template";
        GenJnlBatch: Record "Gen. Journal Batch";
        SourceCode: Record "Source Code";
        VATJnlSetup: Record "KDA VAT Journal Setup";
        Vendor: Record Vendor;
        GLAcc1: Record "G/L Account";
        GLAcc2: Record "G/L Account";
        GLAcc3: Record "G/L Account";
        GenBusPostGroup: Record "Gen. Business Posting Group";
        GenProdPostGroup: Record "Gen. Product Posting Group";
        Item: Record Item;
        ClassOfGoodsFEA: Record "KDA Classific. of Goods FEA";
        ReasonCode: Record "Reason Code";
        UnitOfMeasure: Record "Unit of Measure";
        VATBusPostGroup: Record "VAT Business Posting Group";
        VATProdPostGroup: Record "VAT Product Posting Group";
        PrchVATInvLine: Record "KDA Purch VAT Invoice Line";
    begin
        NoSeries.FindFirst();
        TaxSetup.Get();
        // TaxSetup."KDA Sls VAT Ext. Doc. Nos." := NoSeries.Code;
        TaxSetup."KDA Purch. VAT Invoice Nos." := NoSeries.Code;
        TaxSetup."KDA Purch. Appendix 2 Nos." := NoSeries.Code;
        // TaxSetup."KDA Sls VT App2 Ext Doc Nos" := NoSeries.Code;
        TaxSetup."KDA VAT Inv. Registr. Term" := 20;
```

```
        Evaluate(TaxSetup."KDA VAT Period", '1M');
        TaxSetup."KDA Days btw reg and issue" := 365;
        TaxSetup.Modify(true);

        VATPeriod.Init();
        VATPeriod."Starting Date" := CALCDATE('<-CM>', WorkDate());
        VATPeriod."Ending Date" := CALCDATE('<CM>', WorkDate());
        VATPeriod.Name := 'TestMonth';
        VATPeriod.Insert();

        SourceCode.Init();
        SourceCode.Code := 'ADJADDCURR';
        SourceCode.Description := 'Adjust Add. Reporting Currency';
        SourceCode.Insert();

        GenJnlTemplate.Init();
        GenJnlTemplate.Name := 'НДС';
        GenJnlTemplate.Description := 'НДС';
        GenJnlTemplate.Type := GenJnlTemplate.Type::General;
        GenJnlTemplate."Bal. Account Type" := GenJnlTemplate."Bal. Account
Type"::"G/L Account";
        GenJnlTemplate."Source Code" := SourceCode.Code;
        GenJnlTemplate."Force Doc. Balance" := true;
        GenJnlTemplate."Copy VAT Setup to Jnl. Lines" := true;
        GenJnlTemplate.Insert();

        GenJnlBatch.Init();
        GenJnlBatch."Journal Template Name" := GenJnlTemplate.Name;
        GenJnlBatch.Name := 'СТАНДАРТ';
        GenJnlBatch.Description := 'Журнал за замовчуванням';
        GenJnlBatch."Bal. Account Type" := GenJnlBatch."Bal. Account Type"::"G/L
Account";
        GenJnlBatch."Copy VAT Setup to Jnl. Lines" := true;
        GenJnlBatch."Template Type" := GenJnlBatch."Template Type"::General;
        GenJnlBatch.Insert();

        VATJnlSetup.Init();
        VATJnlSetup."User ID" := '';
```

```
        VATJnlSetup."Purchases VAT Invoice Template" := GenJnlTemplate.Name;
        VATJnlSetup."Purchases VAT Invoice Batch" := GenJnlBatch.Name;
        VATJnlSetup."Purchases App 2 Template" := GenJnlTemplate.Name;
        VATJnlSetup."Purchases App 2 Batch" := GenJnlBatch.Name;
        VATJnlSetup.Insert();

        CreateUnitOfMeasure(UnitOfMeasure);

        CreateClassOfGoodsFEA(ClassOfGoodsFEA);
        ClassOfGoodsFEA.Validate(Description, 'тест_води');
        ClassOfGoodsFEA.Modify(true);

        CreateVATProdPostGroup(VATProdPostGroup);
        VATProdPostGroup.Validate(Description, 'Оподатковується за ставкою 21%');
        VATProdPostGroup.Modify(true);

        CreateGenProdPostGroup(GenProdPostGroup);
        GenProdPostGroup.Validate(Description, 'Товари_тест');
        GenProdPostGroup.Validate("Def. VAT Prod. Posting Group",
VATProdPostGroup.Code);
        GenProdPostGroup.Modify(true);

        CreateVATBusPostGroup(VATBusPostGroup);
        VATBusPostGroup.Validate(Description, 'VATBusPostGroup123');
        VATBusPostGroup.Validate("KDA Does Not Req. Applic.", true);
        VATBusPostGroup.Modify(true);

        CreateItem(Item);
        Item.Validate(Description, 'Водичка');
        Item.Validate(Type, Item.Type::Inventory);
        Item.Validate("Base Unit of Measure", UnitOfMeasure.Code);
        Item.Validate("KDA UCGFEA Code", ClassOfGoodsFEA.Code);
        Item.Validate("Gen. Prod. Posting Group", GenProdPostGroup.Code);
        Item.Validate("VAT Prod. Posting Group", VATProdPostGroup.Code);
        Item.Modify(true);

        ReasonCode.DeleteAll();
```

```
        CreateReasonCode(ReasonCode);

        ReasonCode.Validate("KDA Reason Category", ReasonCode."KDA Reason
Category"::VAT);

        ReasonCode.Validate("KDA VAT Reason of Corr.", ReasonCode."KDA VAT Reason
of Corr."::"Item Change");

        ReasonCode.Modify(true);


        CreateGenBusPostGroup(GenBusPostGroup);

        GenBusPostGroup.Validate(Description, 'Вітчизняні клієнти123');

        GenBusPostGroup.Validate("Def. VAT Bus. Posting Group",
VATBusPostGroup.Code);

        GenBusPostGroup.Modify(true);


        CreateVendor(Vendor);

        Vendor.Validate(Name, 'Ромашка ТОВ');

        Vendor.Validate(Address, 'вул. Лесі Українки, 128-г');

        Vendor.Validate("KDA Registration No.", '12345777');

        Vendor.Validate("VAT Registration No.", '384899509');

        Vendor.Validate("Gen. Bus. Posting Group", GenBusPostGroup.Code);

        Vendor.Validate("VAT Bus. Posting Group", VATBusPostGroup.Code);

        Vendor.Validate("KDA Tax Number Source", Vendor."KDA Tax Number
Source"::"1");

        Vendor.Modify(true);


        CreateGLAccount(GLAcc1);

        CreateGLAccount(GLAcc2);

        CreateGLAccount(GLAcc3);

        InitializeVATSetupForPurchases(VATBusPostGroup.Code,
VATProdPostGroup.Code, GLAcc1."No.", GLAcc2."No.", GLAcc3."No.");


        CreatePurchVATInvHeader(PrchVATInv);

        PrchVATInv.Validate("No.");

        PrchVATInv.Validate("VAT Invoice Type", PrchVATInv."VAT Invoice
Type"::Invoice);

        PrchVATInv.Validate("Issue Date", WorkDate());

        PrchVATInv.Validate("Registration Date", WorkDate());

        PrchVATInv.Validate("Vendor No.", Vendor."No.");

        PrchVATInv.Validate("Document Type", PrchVATInv."Document Type"::RKE);

        PrchVATInv.Validate(Status, PrchVATInv.Status::Released);
```

```
    PrchVATInv.Modify(true);

    CreatePurchVATInvLine(PrchVATInvLine, PrchVATInv);

    FillPurchVATInvLine(PrchVATInvLine, GLAcc1, Item);
end;

procedure AssertTrue(Value: Boolean)
begin
    if not Value then
        Error(AssetError, Value, true);
end;

procedure AssertFalse(Value: Boolean)
begin
    if Value then
        Error(AssetError, Value, False);
end;

procedure AssertEquals(Value: Boolean; Expected: Boolean)
begin
    if Value <> Expected then
        Error(AsserEqualsError, Value, Expected);
end;

procedure AssertEquals(Value: Date; Expected: Date)
begin
    if Value <> Expected then
        Error(AsserEqualsError, Value, Expected);
end;

procedure AssertEquals(Value: Code[2048]; Expected: Code[2048])
begin
    if Value <> Expected then
        Error(AsserEqualsError, Value, Expected);
end;
```

```
    procedure AssertEquals(Value: Text; Expected: Text)
    begin
        if Value <> Expected then
            Error(AsserEqualsError, Value, Expected);
    end;

    procedure AssertEquals(Value: Decimal; Expected: Decimal)
    begin
        if Value <> Expected then
            Error(AsserEqualsError, Value, Expected);
    end;

    procedure AssertEquals(Value: Integer; Expected: Integer)
    begin
        if Value <> Expected then
            Error(AsserEqualsError, Value, Expected);
    end;

    var
        AsserEqualsError: Label 'Value %1 was expected to be %2';
        AssetError: Label 'Value %1 was expected to be %2';
        TextGen: Integer;
        LastLineNo: Integer;
}
```

Додаток Д

Лістинг коду звіту «Імпорт ПН».

```
report 21038253 "KDA VAT Appdx2 XML Exp."
{
    ApplicationArea = All;
    Caption = 'VAT Appendix2 XML Export';
    ProcessingOnly = true;
    UsageCategory = ReportsAndAnalysis;
    dataset
    {
    }

    requestpage
    {
        SaveValues = true;

        layout
        {
            area(content)
            {
                group(General)
                {
                    Caption = 'General';
                    field(VATAccountantNoField; VATAccountantNo)
                    {
                        ApplicationArea = All;
                        Caption = 'VAT Accountant No.';
                        ToolTip = 'Specifies the VAT Accountant No.';
                        TableRelation = Employee."No.";

                        trigger OnValidate()
                        begin
```

```
                        GetAccountantInfo(VATAccountantNo, VATAccountantName,
AccountantTaxID);
                    end;
                }
                field(VATAccountantNameField; VATAccountantName)
                {
                    ApplicationArea = All;
                    Caption = 'VAT Accountant Name';
                    ToolTip = 'Specifies the VAT Accountant Name';
                    Editable = false;
                }
                field(AccountantTaxIDField; AccountantTaxID)
                {
                    ApplicationArea = All;
                    Caption = 'Accountant Tax ID';
                    ToolTip = 'Specifies the Accountant Tax ID';
                    Editable = false;
                }
                field(VATPostingTypeField; VATPostingType)
                {
                    ApplicationArea = All;
                    Caption = 'Posting Type';
                    ToolTip = 'Specifies the Posting Type';
                    // OptionCaption = 'Purchase,Sale';

                    trigger OnValidate()
                    begin
                        VATAppendixNo := '';
                        ExternalDocumentNo := '';
                    end;
                }
                field(VATInvoiceStartingDateField; VATInvoiceStartingDate)
                {
                    ApplicationArea = All;
                    Caption = 'Issue Date From:';
                    Editable = VatStartingEndingDatesEditable;
                    ToolTip = 'Specifies the Issue Date From';
```

```
                    trigger OnValidate()
                    begin
                        if VATInvoiceStartingDate <> 0D then begin
                            CheckEditable('StartingEndingDates');
                            VATInvoiceEndingDate := CALCDATE('<CM>',
VATInvoiceStartingDate)
                        end
                        else begin
                            VATInvoiceEndingDate := 0D;
                            CheckEditable('default');
                        end;
                    end;
                }
                field(VATInvoiceEndingDateField; VATInvoiceEndingDate)
                {
                    ApplicationArea = All;
                    Caption = 'Issue Date To:';
                    ToolTip = 'Specifies the Issue Date To';
                    Editable = VatStartingEndingDatesEditable;

                    trigger OnValidate()
                    begin
                        if VATInvoiceEndingDate <> 0D then
                            CheckEditable('StartingEndingDates')
                        else
                            CheckEditable('default');
                    end;
                }
                field(RegistrationDateField; RegistrationDate)
                {
                    ApplicationArea = All;
                    Caption = 'Registration Date';
                    Editable = RegistrationDateEditable;
                    ToolTip = 'Specifies the Registration Date';

                    trigger OnValidate()
```

```
                        begin
                            if RegistrationDate <> 0D then
                                CheckEditable('RegistrationDate')
                            else
                                CheckEditable('default');
                        end;
                    }
                    field(VATAppendixNoField; VATAppendixNo)
                    {
                        ApplicationArea = All;
                        Caption = 'VAT Appendix2 No.';
                        ToolTip = 'Specifies the VAT Appendix2 No.';
                        Editable = VatAppendixNoEditable;

                        trigger OnLookup(var Text: Text): Boolean
                        var
                            SalesVATInvoiceHeader: Record "KDA Sls VAT Invoice
Header";
                            PurchaseVATInvoiceHeader: Record "KDA Purch VAT Inv
Header";
                            SalesVATInvoiceList: Page "KDA Sls VAT Inv List
Lookup";
                            PurchaseVATInvoiceList: Page "KDA Prch VAT Inv Lst
Lookup";
                        begin
                            case VATPostingType of
                                VATPostingType::Purchase:
                                    begin
                                        PurchaseVATInvoiceHeader.SetRange("VAT
Invoice Type", PurchaseVATInvoiceHeader."VAT Invoice Type"::Appendix2);
                                        PurchaseVATInvoiceHeader.SetFilter(Status,
'%1|%2', PurchaseVATInvoiceHeader.Status::Released,
PurchaseVATInvoiceHeader.Status::Posted);
                                        PurchaseVATInvoiceHeader.SetFilter("No.",
StrSubstNo(TxtVarLbl, VATAppendixNo));
                                        PurchaseVATInvoiceList.LookupMode(true);

PurchaseVATInvoiceList.SetTableView(PurchaseVATInvoiceHeader);

PurchaseVATInvoiceList.SetRecord(PurchaseVATInvoiceHeader);
                                        if ACTION::LookupOK =
PurchaseVATInvoiceList.RunModal() then begin
```

```
PurchaseVATInvoiceList.GetRecord(PurchaseVATInvoiceHeader);
                                            VATAppendixNo :=
PurchaseVATInvoiceHeader."No.";
                                            ExternalDocumentNo :=
PurchaseVATInvoiceHeader."External Document No.";
                                        end;
                                    end;
                                VATPostingType::Sale:
                                    begin
                                        SalesVATInvoiceHeader.SetRange("VAT
Invoice Type", SalesVATInvoiceHeader."VAT Invoice Type"::Appendix2);
                                        SalesVATInvoiceHeader.SetFilter(Status,
'%1|%2', PurchaseVATInvoiceHeader.Status::Released,
PurchaseVATInvoiceHeader.Status::Posted);
                                        SalesVATInvoiceHeader.SetFilter("No.",
StrSubstNo(TxtVarLbl, VATAppendixNo));
                                        SalesVATInvoiceList.LookupMode(true);

SalesVATInvoiceList.SetTableView(SalesVATInvoiceHeader);

SalesVATInvoiceList.SetRecord(SalesVATInvoiceHeader);
                                        if ACTION::LookupOK =
SalesVATInvoiceList.RunModal() then begin

SalesVATInvoiceList.GetRecord(SalesVATInvoiceHeader);
                                            VATAppendixNo :=
SalesVATInvoiceHeader."No.";
                                            ExternalDocumentNo :=
SalesVATInvoiceHeader."External Document No.";
                                        end;
                                    end;
                            end;
                            if VATAppendixNo <> '' then
                                CheckEditable('VATAppendixNo')
                            else
                                CheckEditable('default');
                        end;


                        trigger OnValidate()
                        var
                            SalesVATInvoiceHeader: Record "KDA Sls VAT Invoice
Header";
```

```
                PurchaseVATInvoiceHeader: Record "KDA Purch VAT Inv
Header";
            begin
                if VATAppendixNo = '' then
                    ExternalDocumentNo := '';

                case VATPostingType of
                    VATPostingType::Purchase:
                        begin
                            PurchaseVATInvoiceHeader.SetRange("VAT
Invoice Type", PurchaseVATInvoiceHeader."VAT Invoice Type"::Invoice);
                            PurchaseVATInvoiceHeader.SetRange("No.",
VATAppendixNo);
                            if PurchaseVATInvoiceHeader.FindFirst()
then
                                if PurchaseVATInvoiceHeader.Status =
PurchaseVATInvoiceHeader.Status::Open then
                                    Error(OpenStatusErrorLbl,
VATAppendixNo)
                                else begin
                                    VATAppendixNo :=
PurchaseVATInvoiceHeader."No.";
                                    ExternalDocumentNo :=
PurchaseVATInvoiceHeader."External Document No.";
                                end;
                        end;
                    VATPostingType::Sale:
                        begin
                            SalesVATInvoiceHeader.SetRange("VAT
Invoice Type", SalesVATInvoiceHeader."VAT Invoice Type"::Invoice);
                            SalesVATInvoiceHeader.SetRange("No.",
VATAppendixNo);
                            if SalesVATInvoiceHeader.FindFirst() then
                                if SalesVATInvoiceHeader.Status =
SalesVATInvoiceHeader.Status::Open then
                                    Error(OpenStatusErrorLbl,
VATAppendixNo)
                                else begin
                                    VATAppendixNo :=
SalesVATInvoiceHeader."No.";
                                    ExternalDocumentNo :=
SalesVATInvoiceHeader."External Document No.";
                                end;
```

```
                    end;
                end;
                if VATAppendixNo <> '' then
                    CheckEditable('VATAppendixNo')
                else
                    CheckEditable('default');
            end;
        }
        field(ExternalDocumentNoField; ExternalDocumentNo)
        {
            ApplicationArea = All;
            Caption = 'External Document No.';
            ToolTip = 'Specifies the External Document No.';
            Editable = false;
        }
    }
    group(XMLParameters)
    {
        Caption = 'XML Parameters';
        field(FormTypeField; FormType)
        {
            ApplicationArea = All;
            Caption = 'Form Type';
            ToolTip = 'Specifies the Form Type';
        }
        field(FormSubTypeField; FormSubType)
        {
            ApplicationArea = All;
            Caption = 'Form SubType';
            ToolTip = 'Specifies the Form SubType';
        }
        field(FormVersionField; FormVersion)
        {
            ApplicationArea = All;
            Caption = 'Form Version';
            ToolTip = 'Specifies the Form Version';
        }
```

```
            field(DocumentStateField; DocumentState)
            {
                ApplicationArea = All;
                Caption = 'Document State';
                ToolTip = 'Specifies the Document State';
            }
            field(DocumentTypeField; DocumentType)
            {
                ApplicationArea = All;
                Caption = 'Document Type';
                ToolTip = 'Specifies the Document Type';
            }
            field(PeriodTypeField; PeriodType)
            {
                ApplicationArea = All;
                Caption = 'Period Type';
                ToolTip = 'Specifies the Period Type';
            }
        }
    }
}

actions
{
}

trigger OnOpenPage()
var
    Employee: Record Employee;
begin
    TaxSetup.Get();
    FormType := TaxSetup."KDA VAT Appndx2 Form Type";
    FormSubType := TaxSetup."KDA VAT Appndx2 Frm SubType";
    FormVersion := TaxSetup."KDA VAT Appndx2 Frm Version";

    CompanyInformation.Get();
    if CompanyInformation."KDA VAT Accountant No." <> '' then begin
```

```
                VATAccountantNo := CompanyInformation."KDA VAT Accountant No.";

                GetAccountantInfo(VATAccountantNo, VATAccountantName,
AccountantTaxID);
            end;

            VATPostingType := VATPostingType::Sale;

            DocumentState := '1';
            DocumentType := '0';
            PeriodType := '1';

            if VATAppendixNo <> '' then
                CheckEditable('VATAppendixNo')
            else
                if (VATInvoiceStartingDate <> 0D) or (VATInvoiceEndingDate <> 0D)
then
                    CheckEditable('StartingEndingDates')
                else
                    if RegistrationDate <> 0D then
                        CheckEditable('RegistrationDate')
                    else
                        CheckEditable('default');
        end;
    }

    labels
    {
    }

    trigger OnPreReport()
    var
        SalesVATInvoiceHeader: Record "KDA Sls VAT Invoice Header";
        SalesVATInvoiceHeader2: Record "KDA Sls VAT Invoice Header";
        PurchaseVATInvoiceHeader: Record "KDA Purch VAT Inv Header";
        PurchaseVATInvoiceHeader2: Record "KDA Purch VAT Inv Header";
        DocumentCounter: Integer;
    begin
```

```
        if (VATAppendixNo = '') and (VATInvoiceStartingDate = 0D) and
(VATInvoiceEndingDate = 0D) and (RegistrationDate = 0D) then

            Error(ExportMsgErrorLbl, VatInvoiceNoCaptionLbl,
VatInvoiceStartingDateCaptionLbl, VatInvoiceEndingDateCaptionLbl,
RegistrationDateCaptionLbl);


        CompanyInformation.Get();

        DataCompression.CreateZipArchive();

        case VATPostingType of

            VATPostingType::Sale:

                begin

                    SalesVATInvoiceHeader.SetRange("VAT Invoice Type",
SalesVATInvoiceHeader."VAT Invoice Type"::Appendix2);

                    if VATAppendixNo <> '' then

                        SalesVATInvoiceHeader.SetRange("No.", VATAppendixNo)

                    else

                        if VATInvoiceStartingDate <> 0D then begin

                            SalesVATInvoiceHeader.SetFilter(Status, '%1|%2',
SalesVATInvoiceHeader.Status::Released, SalesVATInvoiceHeader.Status::Posted);

                            SalesVATInvoiceHeader.SetRange("Issue Date",
VATInvoiceStartingDate, VATInvoiceEndingDate);

                        end

                        else

                            if RegistrationDate <> 0D then begin

                                SalesVATInvoiceHeader.SetFilter(Status, '%1|%2',
SalesVATInvoiceHeader.Status::Released, SalesVATInvoiceHeader.Status::Posted);

                                SalesVATInvoiceHeader.SetRange("Registration
Date", RegistrationDate);

                            end;


                    SalesVATInvoiceHeader2.CopyFilters(SalesVATInvoiceHeader);

                    SalesVATInvoiceHeader2.SetFilter("No. Exported", '>=1');

                    SalesVATInvoiceHeader.SetRange("No. Exported", 0);


                    FirstIteration := SalesVATInvoiceHeader.FindSet();

                    SecondIteration := SalesVATInvoiceHeader2.FindSet();


                    if FirstIteration then

                        repeat

                            DocumentCounter += 1;
```

```
                            CreateSalesXML(SalesVATInvoiceHeader, DocumentCounter,
SalesVATInvoiceHeader.Count);
                            SalesVATInvoiceHeader."No. Exported" += 1;
                            SalesVATInvoiceHeader.Modify();
                        until SalesVATInvoiceHeader.Next() = 0;
                    if SecondIteration then begin
                        repeat
                            if ExportedVatAppendixes = '' then
                                ExportedVatAppendixes +=
SalesVATInvoiceHeader2."No."
                            else
                                ExportedVatAppendixes += ', ' +
SalesVATInvoiceHeader2."No.";
                        until SalesVATInvoiceHeader2.Next() = 0;
                        if GuiAllowed then begin
                            if Dialog.Confirm(ExportedVatInvoicesConfirmLbl, true,
ExportedVatAppendixes) then begin
                                SalesVATInvoiceHeader2.FindSet();
                                repeat
                                    DocumentCounter += 1;
                                    CreateSalesXML(SalesVATInvoiceHeader2,
DocumentCounter, SalesVATInvoiceHeader2.Count);
                                    SalesVATInvoiceHeader2."No. Exported" += 1;
                                    SalesVATInvoiceHeader2.Modify();
                                until SalesVATInvoiceHeader2.Next() = 0;
                            end;
                        end
                        else begin
                            SalesVATInvoiceHeader2.FindSet();
                            repeat
                                DocumentCounter += 1;
                                CreateSalesXML(SalesVATInvoiceHeader2,
DocumentCounter, SalesVATInvoiceHeader2.Count);
                                SalesVATInvoiceHeader2."No. Exported" += 1;
                                SalesVATInvoiceHeader2.Modify();
                            until SalesVATInvoiceHeader2.Next() = 0;
                        end;
                    end;
                    if not (FirstIteration) and not (SecondIteration) then
```

```
                        Error(ErrorNothingToExportLbl);
                end;
            VATPostingType::Purchase:
                begin
                    PurchaseVATInvoiceHeader.SetRange("VAT Invoice Type",
PurchaseVATInvoiceHeader."VAT Invoice Type"::Appendix2);
                    if VATAppendixNo <> '' then
                        PurchaseVATInvoiceHeader.SetRange("No.", VATAppendixNo)
                    else
                        if VATInvoiceStartingDate <> 0D then begin
                            PurchaseVATInvoiceHeader.SetFilter(Status, '%1|%2',
PurchaseVATInvoiceHeader.Status::Released,
PurchaseVATInvoiceHeader.Status::Posted);
                            PurchaseVATInvoiceHeader.SetRange("Issue Date",
VATInvoiceStartingDate, VATInvoiceEndingDate);
                        end
                        else
                            if RegistrationDate <> 0D then begin
                                PurchaseVATInvoiceHeader.SetFilter(Status,
'%1|%2', PurchaseVATInvoiceHeader.Status::Released,
PurchaseVATInvoiceHeader.Status::Posted);
                                PurchaseVATInvoiceHeader.SetRange("Registration
Date", RegistrationDate);
                            end;

PurchaseVATInvoiceHeader2.CopyFilters(PurchaseVATInvoiceHeader);
                    PurchaseVATInvoiceHeader2.SetFilter("No. Exported", '>=1');
                    PurchaseVATInvoiceHeader.SetRange("No. Exported", 0);

                    FirstIteration := PurchaseVATInvoiceHeader.FindSet();
                    SecondIteration := PurchaseVATInvoiceHeader2.FindSet();

                    if FirstIteration then
                        repeat
                            DocumentCounter += 1;
                            CreatePurchaseXML(PurchaseVATInvoiceHeader,
DocumentCounter, PurchaseVATInvoiceHeader.Count);
                            PurchaseVATInvoiceHeader."No. Exported" += 1;
                            PurchaseVATInvoiceHeader.Modify();
                        until PurchaseVATInvoiceHeader.Next() = 0;
```

```
                    if SecondIteration then begin
                        repeat
                            if ExportedVatAppendixes = '' then
                                ExportedVatAppendixes +=
PurchaseVATInvoiceHeader2."No."
                            else
                                ExportedVatAppendixes += ', ' +
PurchaseVATInvoiceHeader2."No.";
                        until PurchaseVATInvoiceHeader2.Next() = 0;
                        if GuiAllowed then begin
                            if Dialog.Confirm(ExportedVatInvoicesConfirmLbl, true,
ExportedVatAppendixes) then begin
                                PurchaseVATInvoiceHeader2.FindSet();
                                repeat
                                    DocumentCounter += 1;
                                    CreatePurchaseXML(PurchaseVATInvoiceHeader2,
DocumentCounter, PurchaseVATInvoiceHeader2.Count);
                                    PurchaseVATInvoiceHeader2."No. Exported" += 1;
                                    PurchaseVATInvoiceHeader2.Modify();
                                until PurchaseVATInvoiceHeader2.Next() = 0;
                            end;
                        end
                        else begin
                            PurchaseVATInvoiceHeader2.FindSet();
                            repeat
                                DocumentCounter += 1;
                                CreatePurchaseXML(PurchaseVATInvoiceHeader2,
DocumentCounter, PurchaseVATInvoiceHeader2.Count);
                                PurchaseVATInvoiceHeader2."No. Exported" += 1;
                                PurchaseVATInvoiceHeader2.Modify();
                            until PurchaseVATInvoiceHeader2.Next() = 0;
                        end;
                    end;
                    if not (FirstIteration) and not (SecondIteration) then
                        Error(ErrorNothingToExportLbl);
                end;
            end;
        end;
```

```
trigger OnPostReport()
var
    TempBlob: Codeunit "Temp Blob";
    IStreamZipLoc: InStream;
    OStreamZipLoc: OutStream;
    ToFileLoc: Text;
begin
    if IsZip then begin
        TempBlob.CreateOutStream(OStreamZipLoc);
        DataCompression.SaveZipArchive(OStreamZipLoc);
        TempBlob.CreateInStream(IStreamZipLoc);
        if VATPostingType = VATPostingType::Purchase then
            ToFileLoc := 'Purchase_VAT_Appendix2_'
        else
            ToFileLoc := 'Sale_VAT_Appendix2_';
        ToFileLoc += Format(Today()) + '.zip';
        DownloadFromStream(IStreamZipLoc, 'Save Archive', '', '', ToFileLoc);
    end;
end;

var
    TaxSetup: Record "Tax Setup";
    CompanyInformation: Record "Company Information";
    FileManagement: Codeunit "File Management";
    DataCompression: Codeunit "Data Compression";
    VATAccountantNo: Code[20];
    VATAccountantName: Text[62];
    AccountantTaxID: Code[12];
    VATPostingType: Enum "KDA VAT Posting Type";
    VATAppendixNo: Code[20];
    ExternalDocumentNo: Code[20];
    VATInvoiceStartingDate: Date;
    VATInvoiceEndingDate: Date;
    RegistrationDate: Date;
    FormType: Code[10];
    FormSubType: Code[10];
    FormVersion: Code[10];
```

```
DocumentState: Code[10];
DocumentType: Code[10];
PeriodType: Code[10];
IsZip: Boolean;
XmlSchemaInstPathLbl: Label 'http://www.w3.org/2001/XMLSchema-instance';
XsiTxt: Label 'xsi';
TxtVarLbl: Label '*%1*', Comment = '%1 - "VAT Appendix No."';
NoNamespaceSchemaLocationTxt: Label 'noNamespaceSchemaLocation';
HERPN0Txt: Label 'HERPN0';
HERPNTxt: Label 'HERPN';
J1201010XSDTxt: Label 'J1201010.XSD';
DeclarTxt: Label 'DECLAR';
DeclarHeadTxt: Label 'DECLARHEAD';
DeclarBodyTxt: Label 'DECLARBODY';
TinTxt: Label 'TIN';
CDocTxt: Label 'C_DOC';
CDocSubTxt: Label 'C_DOC_SUB';
CDocVerTxt: Label 'C_DOC_VER';
CDocTypeTxt: Label 'C_DOC_TYPE';
CRegTxt: Label 'C_REG';
CRajTxt: Label 'C_RAJ';
PeriodMonthTxt: Label 'PERIOD_MONTH';
PeriodTypeTxt: Label 'PERIOD_TYPE';
PeriodYearTxt: Label 'PERIOD_YEAR';
CStiOrigTxt: Label 'C_STI_ORIG';
CDocStanTxt: Label 'C_DOC_STAN';
DFillTxt: Label 'D_FILL';
R03G10STxt: Label 'R03G10S';
HOrig1Txt: Label 'HORIG1';
HTyprTxt: Label 'HTYPR';
HFillTxt: Label 'HFILL';
HNumTxt: Label 'HNUM';
HNum1Txt: Label 'HNUM1';
HNameSelTxt: Label 'HNAMESEL';
HNameBuyTxt: Label 'HNAMEBUY';
HKSelTxt: Label 'HKSEL';
HNum2Txt: Label 'HNUM2';
```

```
HTinSelTxt: Label 'HTINSEL';
HKBuyTxt: Label 'HKBUY';
HFBuyTxt: Label 'HFBUY';
HTinBuyTxt: Label 'HTINBUY';
HKSLbl: Label 'HKS';
HKBLbl: Label 'HKB';
HBosTxt: Label 'HBOS';
HKBosTxt: Label 'HKBOS';
RowNumTxt: Label 'ROWNUM';
RXXXXG3STxt: Label 'RXXXXG3S';
RXXXXG4Txt: Label 'RXXXXG4';
RXXXXG32Txt: Label 'RXXXXG32';
RXXXXG33Txt: Label 'RXXXXG33';
RXXXXG4STxt: Label 'RXXXXG4S';
RXXXXG105_2STxt: Label 'RXXXXG105_2S';
RXXXXG5Txt: Label 'RXXXXG5';
RXXXXG6Txt: Label 'RXXXXG6';
RXXXXG008Txt: Label 'RXXXXG008';
RXXXXG009Txt: Label 'RXXXXG009';
RXXXXG010Txt: Label 'RXXXXG010';
RXXXXG11_10Txt: Label 'RXXXXG11_10';
R01G9Txt: Label 'R01G9';
NoVATTxt: Label 'Без ПДВ';
FolderPath: Text[250];
HPodFillTxt: Label 'HPODFILL';
HPodNumTxt: Label 'HPODNUM';
HPodNum1Txt: Label 'HPODNUM1';
HPodNum2Txt: Label 'HPODNUM2';
RXXXXG001Txt: Label 'RXXXXG001';
RXXXXG21Txt: Label 'RXXXXG21';
RXXXXG22Txt: Label 'RXXXXG22';
RXXXXG7Txt: Label 'RXXXXG7';
RXXXXG8Txt: Label 'RXXXXG8';
R001G03Txt: Label 'R001G03';
R02G9Txt: Label 'R02G9';
R02G111Txt: Label 'R02G111';
R01G111Txt: Label 'R01G111';
```

```
        R006G03Txt: Label 'R006G03';

        R007G03Txt: Label 'R007G03';

        R01G11Txt: Label 'R01G11';

        R01G14Txt: Label 'R01G14';

        R03G14Txt: Label 'R03G14';

        R01G1Txt: Label 'R01G1', Locked = true;

        ExportMsgErrorLbl: Label 'Nothing to export. Please, specify one of the
parameters %1, %2, %3 or %4.', Comment = '%1 - "VAT Invoice No.", %2 - "Starting
Date", %3 - "Ending Date", %4 - Registration Date';

        OpenStatusErrorLbl: Label 'Status must not be "Open" for VAT Appendix2
%1.', Comment = '%1 - VAT Appendix No.';

        ErrorNothingToExportLbl: Label 'Nothing to export.';

        [InDataSet]

        VatAppendixNoEditable: Boolean;

        [InDataSet]

        VatStartingEndingDatesEditable: Boolean;

        [InDataSet]

        RegistrationDateEditable: Boolean;

        ExportedVatAppendixes: Text;

        ExportedVatInvoicesConfirmLbl: Label 'VAT Appendix2 %1 has been already
exported. Do you want to re-export?', Comment = '%1 - Exported VAT Appendixes';

        FirstIteration: Boolean;

        SecondIteration: Boolean;

        VatInvoiceNoCaptionLbl: Label 'VAT Appendix2 No.';

        VatInvoiceStartingDateCaptionLbl: Label 'VAT Invoice Starting Date';

        VatInvoiceEndingDateCaptionLbl: Label 'VAT Invoice Ending Date';

        RegistrationDateCaptionLbl: Label 'Registration Date';



    local procedure CreateSalesXML(SalesVATInvoiceHeader: Record "KDA Sls VAT
Invoice Header"; DocumentCounter: Integer; SalesCount: Integer)

    var

        TempXMLBuffer: Record "XML Buffer" temporary;

        SalesVATInvoiceLine: Record "KDA Sls VAT Invoice Line";

        UnitofMeasure: Record "Unit of Measure";

        SalesVATInvoiceHeaderApplied: Record "KDA Sls VAT Invoice Header";

        ReasonCode: Record "Reason Code";

        LineCounter: Integer;

        VATBusPostGroup: Record "VAT Business Posting Group";
```

```
        NewHNAMEBUY: Text[20];

    begin

        CreateXMLHeader(TempXMLBuffer, SalesVATInvoiceHeader."Issue Date");

        TempXMLBuffer.AddGroupElement(DeclarBodyTxt);

        TempXMLBuffer.AddElement(HERPN0Txt, '');

        TempXMLBuffer.AddElement(HERPNTxt, '');

        TempXMLBuffer.AddElement(R01G1Txt,
Format(SalesVATInvoiceHeader."Consolidated VAT Invoice"));

        if DefineR03G10S(true, SalesVATInvoiceHeader."VAT Invoice Type",
SalesVATInvoiceHeader."No.") then

            TempXMLBuffer.AddElement(R03G10STxt, NoVATTxt);

        if SalesVATInvoiceHeader."Kept by seller" then

            TempXMLBuffer.AddElement(HOrig1Txt, '1');

        TempXMLBuffer.AddElement(HTyprTxt, SalesVATInvoiceHeader."Reason Type");

        TempXMLBuffer.AddElement(HFillTxt, Format(SalesVATInvoiceHeader."Issue
Date", 0, '<Day,2><Month,2><Year4>'));

        TempXMLBuffer.AddElement(HNumTxt, SalesVATInvoiceHeader."External Document
No.");


        if SalesVATInvoiceHeaderApplied.Get(SalesVATInvoiceHeaderApplied."VAT
Invoice Type"::Invoice, SalesVATInvoiceHeader."Applied Invoice No.") then begin

            TempXMLBuffer.AddElement(HPodFillTxt,
Format(SalesVATInvoiceHeaderApplied."Issue Date", 0, '<Day,2><Month,2><Year4>'));

            TempXMLBuffer.AddElement(HPodNumTxt,
SalesVATInvoiceHeaderApplied."External Document No.");

        end;

        TempXMLBuffer.AddElement(HPodNum1Txt, '');

        TempXMLBuffer.AddElement(HPodNum2Txt, '');

        TempXMLBuffer.AddElement(HNameSelTxt, CompanyInformation.Name);

        if VATBusPostGroup.Get(SalesVATInvoiceHeader."VAT Bus. Posting Group")
then begin

            if VATBusPostGroup."KDA VAT Non-payer" then begin

                if VATBusPostGroup."KDA Cond.VAT Non-payer Name" <> '' then

                    NewHNAMEBUY := VATBusPostGroup."KDA Cond.VAT Non-payer Name"

                else

                    NewHNAMEBUY := '';

            end

            else

                NewHNAMEBUY := '';

        end
```

```
        else
            NewHNAMEBUY := '';
        if NewHNAMEBUY <> '' then
            TempXMLBuffer.AddElement(HNameBuyTxt, NewHNAMEBUY)
        else
            TempXMLBuffer.AddElement(HNameBuyTxt, SalesVATInvoiceHeader."Customer
Name");
        TempXMLBuffer.AddElement(HKSelTxt, CompanyInformation."VAT Registration
No.");
        TempXMLBuffer.AddElement(HNum2Txt, '');
        TempXMLBuffer.AddElement(HTinSelTxt, CompanyInformation."Registration
No.");
        TempXMLBuffer.AddElement(HKBuyTxt, SalesVATInvoiceHeader."VAT Registration
No.");
        TempXMLBuffer.AddElement(HFBuyTxt,
GetCustomerBranchCode(SalesVATInvoiceHeader."Customer No."));
        if not VATBusPostGroup."KDA VAT Non-payer" then
            TempXMLBuffer.AddElement(HTinBuyTxt,
SalesVATInvoiceHeader."Registration No.")
        else
            TempXMLBuffer.AddElement(HTinBuyTxt, '');
        TempXMLBuffer.AddElement(HKSLbl, Format(CompanyInformation."KDA Tax Number
Source"));
        TempXMLBuffer.AddElement(HKBLbl, Format(SalesVATInvoiceHeader."Tax Number
Source"));
        TempXMLBuffer.AddElement(HBosTxt, VATAccountantName);
        TempXMLBuffer.AddElement(HKBosTxt, AccountantTaxID);
        SalesVATInvoiceLine.SetRange("VAT Invoice Type",
SalesVATInvoiceHeader."VAT Invoice Type");
        SalesVATInvoiceLine.SetRange("Document No.", SalesVATInvoiceHeader."No.");
        if SalesVATInvoiceLine.FindSet() then
            repeat
                LineCounter += 1;
                AddElementWithRowNumAttr(TempXMLBuffer, LineCounter, RXXXXG001Txt,
Format(SalesVATInvoiceLine."Applied Invoice Line No."));
                AddElementWithRowNumAttr(TempXMLBuffer, LineCounter, RXXXXG21Txt,
SalesVATInvoiceLine."Reason of Correction");
                AddElementWithRowNumAttr(TempXMLBuffer, LineCounter, RXXXXG22Txt,
Format(SalesVATInvoiceLine."Correction Group No."));


                AddElementWithRowNumAttr(TempXMLBuffer, LineCounter, RXXXXG3STxt,
SalesVATInvoiceLine.Description);
```

```
                AddElementWithRowNumAttr(TempXMLBuffer, LineCounter, RXXXXG4Txt,
SalesVATInvoiceLine."UCGFEA Code");

                if SalesVATInvoiceLine."Imported Item" then

                    AddElementWithRowNumAttr(TempXMLBuffer, LineCounter,
RXXXXG32Txt, '1');

                AddElementWithRowNumAttr(TempXMLBuffer, LineCounter, RXXXXG33Txt,
SalesVATInvoiceLine."Goods and Services Code");

                UnitofMeasure.Get(SalesVATInvoiceLine."Unit Of Measure Code");

                AddElementWithRowNumAttr(TempXMLBuffer, LineCounter, RXXXXG4STxt,
UnitofMeasure."KDA Symbol Ukr.");

                AddElementWithRowNumAttr(TempXMLBuffer, LineCounter,
RXXXXG105_2STxt, UnitofMeasure."KDA Code Ukr.");


                if ReasonCode.Get(SalesVATInvoiceLine."Reason of Correction") then
begin

                    if ReasonCode."KDA VAT Reason of Corr." in [ReasonCode."KDA
VAT Reason of Corr."::"Item Change",

                                                                ReasonCode."KDA
VAT Reason of Corr."::"Quantity Change",

                                                                ReasonCode."KDA
VAT Reason of Corr."::" "]

                    then

                        AddElementWithRowNumAttr(TempXMLBuffer, LineCounter,
RXXXXG5Txt, Format(SalesVATInvoiceLine.Quantity, 0, 9))

                    else

                        AddElementWithRowNumAttr(TempXMLBuffer, LineCounter,
RXXXXG8Txt, Format(SalesVATInvoiceLine.Quantity, 0, 9));


                    if ReasonCode."KDA VAT Reason of Corr." in [ReasonCode."KDA
VAT Reason of Corr."::"Item Change",

                                                                ReasonCode."KDA
VAT Reason of Corr."::"Quantity Change"]

                    then

                        AddElementWithRowNumAttr(TempXMLBuffer, LineCounter,
RXXXXG6Txt, Format(SalesVATInvoiceLine."Price Excl. VAT", 0, 9))

                    else

                        AddElementWithRowNumAttr(TempXMLBuffer, LineCounter,
RXXXXG7Txt, Format(SalesVATInvoiceLine."Price Excl. VAT", 0, 9));

                end;


                AddElementWithRowNumAttr(TempXMLBuffer, LineCounter, RXXXXG008Txt,
Format(SalesVATInvoiceLine."VAT Rate Code"));

                AddElementWithRowNumAttr(TempXMLBuffer, LineCounter, RXXXXG009Txt,
SalesVATInvoiceLine."Exemption Code");
```

```
                AddElementWithRowNumAttr(TempXMLBuffer, LineCounter, RXXXXG010Txt,
Format(SalesVATInvoiceLine."Base Amount", 0, 9));

                AddElementWithRowNumAttr(TempXMLBuffer, LineCounter,
RXXXXG11_10Txt, Format(SalesVATInvoiceLine."VAT Amount", 0, 9));

            until SalesVATInvoiceLine.Next() = 0;

        SalesVATInvoiceLine.CalcSums("Amount Incl. VAT", "VAT Amount");

        TempXMLBuffer.AddElement(R001G03Txt, Format(SalesVATInvoiceLine."VAT
Amount", 0, 9));

        SalesVATInvoiceLine.SetRange("VAT Rate Code", SalesVATInvoiceLine."VAT
Rate Code"::"20");

        SalesVATInvoiceLine.CalcSums("VAT Amount", "Base Amount");

        TempXMLBuffer.AddElement(R02G9Txt, Format(SalesVATInvoiceLine."VAT
Amount", 0, 9));

        TempXMLBuffer.AddElement(R01G9Txt, Format(SalesVATInvoiceLine."Base
Amount", 0, 9));

        SalesVATInvoiceLine.SetRange("VAT Rate Code", SalesVATInvoiceLine."VAT
Rate Code"::"7");

        SalesVATInvoiceLine.CalcSums("VAT Amount", "Base Amount");

        TempXMLBuffer.AddElement(R02G111Txt, Format(SalesVATInvoiceLine."VAT
Amount", 0, 9));

        TempXMLBuffer.AddElement(R01G111Txt, Format(SalesVATInvoiceLine."Base
Amount", 0, 9));

        SalesVATInvoiceLine.SetRange("VAT Rate Code", SalesVATInvoiceLine."VAT
Rate Code"::"14");

        SalesVATInvoiceLine.CalcSums("VAT Amount", "Base Amount");

        TempXMLBuffer.AddElement(R01G14Txt, Format(SalesVATInvoiceLine."Base
Amount", 0, 9));

        TempXMLBuffer.AddElement(R03G14Txt, Format(SalesVATInvoiceLine."VAT
Amount", 0, 9));

        SalesVATInvoiceLine.SetRange("VAT Rate Code", SalesVATInvoiceLine."VAT
Rate Code"::"901");

        SalesVATInvoiceLine.CalcSums("Base Amount");

        TempXMLBuffer.AddElement(R006G03Txt, Format(SalesVATInvoiceLine."Base
Amount", 0, 9));

        SalesVATInvoiceLine.SetRange("VAT Rate Code", SalesVATInvoiceLine."VAT
Rate Code"::"902");

        SalesVATInvoiceLine.CalcSums("Base Amount");

        TempXMLBuffer.AddElement(R007G03Txt, Format(SalesVATInvoiceLine."Base
Amount", 0, 9));

        SalesVATInvoiceLine.SetRange("VAT Rate Code", SalesVATInvoiceLine."VAT
Rate Code"::"903");

        SalesVATInvoiceLine.CalcSums("Base Amount");

        TempXMLBuffer.AddLastElement(R01G11Txt, Format(SalesVATInvoiceLine."Base
Amount", 0, 9));
```

```
        if SalesCount = 1 then
            SaveXMLToFile(TempXMLBuffer, SalesVATInvoiceHeader."Issue Date",
DocumentCounter)
        else begin
            IsZip := true;
            SaveXMLToZip(TempXMLBuffer, SalesVATInvoiceHeader."Issue Date",
DocumentCounter);
        end;
        TempXMLBuffer.DeleteAll();
    end;


    local procedure CreatePurchaseXML(PurchaseVATInvoiceHeader: Record "KDA Purch
VAT Inv Header"; DocumentCounter: Integer; PurchCount: Integer)
    var
        TempXMLBuffer: Record "XML Buffer" temporary;
        PurchaseVATInvoiceLine: Record "KDA Purch VAT Invoice Line";
        UnitofMeasure: Record "Unit of Measure";
        PurchaseVATInvoiceHeaderApplied: Record "KDA Purch VAT Inv Header";
        ReasonCode: Record "Reason Code";
        LineCounter: Integer;
    begin
        CreateXMLHeader(TempXMLBuffer, PurchaseVATInvoiceHeader."Issue Date");
        TempXMLBuffer.AddGroupElement(DeclarBodyTxt);
        TempXMLBuffer.AddElement(HERPN0Txt, '');
        TempXMLBuffer.AddElement(HERPNTxt, '');
        TempXMLBuffer.AddElement(R01G1Txt,
Format(PurchaseVATInvoiceHeader."Consolidated VAT Invoice"));
        if DefineR03G10S(false, PurchaseVATInvoiceHeader."VAT Invoice Type",
PurchaseVATInvoiceHeader."No.") then
            TempXMLBuffer.AddElement(R03G10STxt, NoVATTxt);


        TempXMLBuffer.AddElement(HFillTxt, Format(PurchaseVATInvoiceHeader."Issue
Date", 0, '<Day,2><Month,2><Year4>'));
        TempXMLBuffer.AddElement(HNumTxt, PurchaseVATInvoiceHeader."External
Document No.");
        TempXMLBuffer.AddElement(HNum1Txt, '');


        if
PurchaseVATInvoiceHeaderApplied.Get(PurchaseVATInvoiceHeaderApplied."VAT Invoice
Type"::Invoice, PurchaseVATInvoiceHeader."Applied Invoice No.") then begin
```

```
            TempXMLBuffer.AddElement(HPodFillTxt,
Format(PurchaseVATInvoiceHeaderApplied."Issue Date", 0,
'<Day,2><Month,2><Year4>'));

            TempXMLBuffer.AddElement(HPodNumTxt,
PurchaseVATInvoiceHeaderApplied."External Document No.");

        end;

        TempXMLBuffer.AddElement(HPodNum1Txt, '');

        TempXMLBuffer.AddElement(HPodNum2Txt,
GetVendorBranchCode(PurchaseVATInvoiceHeader."Vendor No."));

        TempXMLBuffer.AddElement(HNameSelTxt, PurchaseVATInvoiceHeader."Vendor
Name");

        TempXMLBuffer.AddElement(HNameBuyTxt, CompanyInformation.Name);

        TempXMLBuffer.AddElement(HKSelTxt, PurchaseVATInvoiceHeader."VAT
Registration No.");

        TempXMLBuffer.AddElement(HNum2Txt,
GetVendorBranchCode(PurchaseVATInvoiceHeader."Vendor No."));

        TempXMLBuffer.AddElement(HTinSelTxt,
PurchaseVATInvoiceHeader."Registration No.");

        TempXMLBuffer.AddElement(HKBuyTxt, CompanyInformation."VAT Registration
No.");

        TempXMLBuffer.AddElement(HFBuyTxt, '');

        TempXMLBuffer.AddElement(HTinBuyTxt, CompanyInformation."Registration
No.");

        TempXMLBuffer.AddElement(HKSLbl, Format(PurchaseVATInvoiceHeader."Tax
Number Source"));

        TempXMLBuffer.AddElement(HKBLbl, Format(CompanyInformation."KDA Tax Number
Source"));

        TempXMLBuffer.AddElement(HBosTxt, VATAccountantName);

        TempXMLBuffer.AddElement(HKBosTxt, AccountantTaxID);

        PurchaseVATInvoiceLine.SetRange("VAT Invoice Type",
PurchaseVATInvoiceHeader."VAT Invoice Type");

        PurchaseVATInvoiceLine.SetRange("Document No.",
PurchaseVATInvoiceHeader."No.");

        if PurchaseVATInvoiceLine.FindSet() then

            repeat

                LineCounter += 1;

                AddElementWithRowNumAttr(TempXMLBuffer, LineCounter, RXXXXG001Txt,
Format(PurchaseVATInvoiceLine."Applied Invoice Line No."));

                AddElementWithRowNumAttr(TempXMLBuffer, LineCounter, RXXXXG21Txt,
PurchaseVATInvoiceLine."Reason of Correction");

                AddElementWithRowNumAttr(TempXMLBuffer, LineCounter, RXXXXG22Txt,
Format(PurchaseVATInvoiceLine."Correction Group No."));
```

```
                AddElementWithRowNumAttr(TempXMLBuffer, LineCounter, RXXXXG3STxt,
PurchaseVATInvoiceLine.Description);

                AddElementWithRowNumAttr(TempXMLBuffer, LineCounter, RXXXXG4Txt,
PurchaseVATInvoiceLine."UCGFEA Code");

                if PurchaseVATInvoiceLine."Imported Item" then

                    AddElementWithRowNumAttr(TempXMLBuffer, LineCounter,
RXXXXG32Txt, '1');

                AddElementWithRowNumAttr(TempXMLBuffer, LineCounter, RXXXXG33Txt,
PurchaseVATInvoiceLine."Goods and Services Code");

                UnitofMeasure.Get(PurchaseVATInvoiceLine."Unit Of Measure Code");

                AddElementWithRowNumAttr(TempXMLBuffer, LineCounter, RXXXXG4STxt,
UnitofMeasure."KDA Symbol Ukr.");

                AddElementWithRowNumAttr(TempXMLBuffer, LineCounter,
RXXXXG105_2STxt, UnitofMeasure."KDA Code Ukr.");


                if ReasonCode.Get(PurchaseVATInvoiceLine."Reason of Correction")
then begin

                    if ReasonCode."KDA VAT Reason of Corr." in [ReasonCode."KDA
VAT Reason of Corr."::"Item Change",

                                                                ReasonCode."KDA
VAT Reason of Corr."::"Quantity Change",

                                                                ReasonCode."KDA
VAT Reason of Corr."::" "]

                    then

                        AddElementWithRowNumAttr(TempXMLBuffer, LineCounter,
RXXXXG5Txt, Format(PurchaseVATInvoiceLine.Quantity, 0, 9))

                    else

                        AddElementWithRowNumAttr(TempXMLBuffer, LineCounter,
RXXXXG8Txt, Format(PurchaseVATInvoiceLine.Quantity, 0, 9));


                    if ReasonCode."KDA VAT Reason of Corr." in [ReasonCode."KDA
VAT Reason of Corr."::"Item Change",

                                                                ReasonCode."KDA
VAT Reason of Corr."::"Quantity Change"]

                    then

                        AddElementWithRowNumAttr(TempXMLBuffer, LineCounter,
RXXXXG6Txt, Format(PurchaseVATInvoiceLine."Price Excl. VAT", 0, 9))

                    else

                        AddElementWithRowNumAttr(TempXMLBuffer, LineCounter,
RXXXXG7Txt, Format(PurchaseVATInvoiceLine."Price Excl. VAT", 0, 9));

                end;

                AddElementWithRowNumAttr(TempXMLBuffer, LineCounter, RXXXXG008Txt,
Format(PurchaseVATInvoiceLine."VAT Rate Code", 0, 1));
```

```
                AddElementWithRowNumAttr(TempXMLBuffer, LineCounter, RXXXXG009Txt,
PurchaseVATInvoiceLine."Exemption Code");

                AddElementWithRowNumAttr(TempXMLBuffer, LineCounter, RXXXXG010Txt,
Format(PurchaseVATInvoiceLine."Base Amount", 0, 9));

                AddElementWithRowNumAttr(TempXMLBuffer, LineCounter,
RXXXXG11_10Txt, Format(PurchaseVATInvoiceLine."VAT Amount", 0, 9));

            until PurchaseVATInvoiceLine.Next() = 0;

        PurchaseVATInvoiceLine.CalcSums("Amount Incl. VAT", "VAT Amount");

        TempXMLBuffer.AddElement(R001G03Txt, Format(PurchaseVATInvoiceLine."VAT
Amount", 0, 9));

        PurchaseVATInvoiceLine.SetRange("VAT Rate Code",
PurchaseVATInvoiceLine."VAT Rate Code"::"20");

        PurchaseVATInvoiceLine.CalcSums("VAT Amount", "Base Amount");

        TempXMLBuffer.AddElement(R02G9Txt, Format(PurchaseVATInvoiceLine."VAT
Amount", 0, 9));

        TempXMLBuffer.AddElement(R01G9Txt, Format(PurchaseVATInvoiceLine."Base
Amount", 0, 9));

        PurchaseVATInvoiceLine.SetRange("VAT Rate Code",
PurchaseVATInvoiceLine."VAT Rate Code"::"7");

        PurchaseVATInvoiceLine.CalcSums("VAT Amount", "Base Amount");

        TempXMLBuffer.AddElement(R02G111Txt, Format(PurchaseVATInvoiceLine."VAT
Amount", 0, 9));

        TempXMLBuffer.AddElement(R01G111Txt, Format(PurchaseVATInvoiceLine."Base
Amount", 0, 9));

        PurchaseVATInvoiceLine.SetRange("VAT Rate Code",
PurchaseVATInvoiceLine."VAT Rate Code"::"14");

        PurchaseVATInvoiceLine.CalcSums("VAT Amount", "Base Amount");

        TempXMLBuffer.AddElement(R01G14Txt, Format(PurchaseVATInvoiceLine."Base
Amount", 0, 9));

        TempXMLBuffer.AddElement(R03G14Txt, Format(PurchaseVATInvoiceLine."VAT
Amount", 0, 9));

        PurchaseVATInvoiceLine.SetRange("VAT Rate Code",
PurchaseVATInvoiceLine."VAT Rate Code"::"901");

        PurchaseVATInvoiceLine.CalcSums("Base Amount");

        TempXMLBuffer.AddElement(R006G03Txt, Format(PurchaseVATInvoiceLine."Base
Amount", 0, 9));

        PurchaseVATInvoiceLine.SetRange("VAT Rate Code",
PurchaseVATInvoiceLine."VAT Rate Code"::"902");

        PurchaseVATInvoiceLine.CalcSums("Base Amount");

        TempXMLBuffer.AddElement(R007G03Txt, Format(PurchaseVATInvoiceLine."Base
Amount", 0, 9));

        PurchaseVATInvoiceLine.SetRange("VAT Rate Code",
PurchaseVATInvoiceLine."VAT Rate Code"::"903");

        PurchaseVATInvoiceLine.CalcSums("Base Amount");
```

```
        TempXMLBuffer.AddLastElement(R01G11Txt,
Format(PurchaseVATInvoiceLine."Base Amount", 0, 9));

        if PurchCount = 1 then

            SaveXMLToFile(TempXMLBuffer, PurchaseVATInvoiceHeader."Issue Date",
DocumentCounter)

        else begin

            IsZip := true;

            SaveXMLToZip(TempXMLBuffer, PurchaseVATInvoiceHeader."Issue Date",
DocumentCounter);

        end;

        TempXMLBuffer.DeleteAll();

    end;


    local procedure CreateXMLHeader(var TempXMLBuffer: Record "XML Buffer"
temporary; IssueDate: Date)

    begin

        TempXMLBuffer.CreateRootElement(DeclarTxt);

        TempXMLBuffer.AddNamespace(XsiTxt, XmlSchemaInstPathLbl);

        TempXMLBuffer.AddAttribute(NoNamespaceSchemaLocationTxt, J1201010XSDTxt);

        TempXMLBuffer.AddGroupElement(DeclarHeadTxt);

        TempXMLBuffer.AddElement(TinTxt, CompanyInformation."Registration No.");

        TempXMLBuffer.AddElement(CDocTxt, FormType);

        TempXMLBuffer.AddElement(CDocSubTxt, FormSubType);

        TempXMLBuffer.AddElement(CDocVerTxt, FormVersion);

        TempXMLBuffer.AddElement(CDocTypeTxt, DocumentType);

        TempXMLBuffer.AddElement(CRegTxt, CompanyInformation."KDA Region Code
(Tax)");

        TempXMLBuffer.AddElement(CRajTxt, CompanyInformation."KDA Adm. District
Code Tax");

        TempXMLBuffer.AddElement(PeriodMonthTxt, Format(Date2DMY(IssueDate, 2)));

        TempXMLBuffer.AddElement(PeriodTypeTxt, PeriodType);

        TempXMLBuffer.AddElement(PeriodYearTxt, Format(Date2DMY(IssueDate, 3)));

        TempXMLBuffer.AddElement(CStiOrigTxt, CalcCStiOrig(CompanyInformation."KDA
Region Code (Tax)", CompanyInformation."KDA Adm. District Code Tax"));

        TempXMLBuffer.AddElement(CDocStanTxt, DocumentState);

        TempXMLBuffer.AddLastElement(DFillTxt, Format(RegistrationDate, 0,
'<Day,2><Month,2><Year4>'));

        TempXMLBuffer.GetParent();

    end;
```

```
    local procedure CalcCStiOrig(CReg: Code[10]; CRay: Code[10]): Text[250]
    var
        CRegDec: Decimal;
        CRayDec: Decimal;
    begin
        Evaluate(CRegDec, CReg);
        Evaluate(CRayDec, CRay);
        exit(Format(CRegDec * 100 + CRayDec, 0, 9));
    end;


    local procedure DefineR03G10S(Sales: Boolean; VATInvoiceType: Enum "KDA VAT
Invoice Type"; DocumentNo: Code[20]): Boolean
    var
        SalesVATInvoiceLine: Record "KDA Sls VAT Invoice Line";
        PurchaseVATInvoiceLine: Record "KDA Purch VAT Invoice Line";
        TotalQtyOfLines: Integer;
    begin
        if Sales then begin
            SalesVATInvoiceLine.SetRange("VAT Invoice Type", VATInvoiceType);
            SalesVATInvoiceLine.SetRange("Document No.", DocumentNo);
            TotalQtyOfLines := SalesVATInvoiceLine.Count;
            SalesVATInvoiceLine.SetRange("VAT Rate Code", SalesVATInvoiceLine."VAT
Rate Code"::"903");
            if TotalQtyOfLines = SalesVATInvoiceLine.Count then
                exit(true)
        end else begin
            PurchaseVATInvoiceLine.SetRange("VAT Invoice Type", VATInvoiceType);
            PurchaseVATInvoiceLine.SetRange("Document No.", DocumentNo);
            TotalQtyOfLines := PurchaseVATInvoiceLine.Count;
            PurchaseVATInvoiceLine.SetRange("VAT Rate Code",
PurchaseVATInvoiceLine."VAT Rate Code"::"903");
            if TotalQtyOfLines = PurchaseVATInvoiceLine.Count then
                exit(true)
        end;
        exit(false);
    end;


    local procedure GetCustomerBranchCode(CustomerNo: Code[20]): Code[20]
```

```
    var
        Customer: Record Customer;
    begin
        Customer.Get(CustomerNo);
        exit(Customer."KDA Branch Code");
    end;

    local procedure GetVendorBranchCode(VendorNo: Code[20]): Code[20]
    var
        Vendor: Record Vendor;
    begin
        Vendor.Get(VendorNo);
        exit(Vendor."KDA Branch Code");
    end;

    local procedure AddElementWithRowNumAttr(var TempXMLBuffer: Record "XML
Buffer" temporary; LineNo: Integer; ElementName: Text[250]; ElementValue: Text)
    begin
        TempXMLBuffer.AddGroupElement(ElementName);
        TempXMLBuffer.SetValueWithoutModifying(ElementValue);
        TempXMLBuffer.Modify(true);
        TempXMLBuffer.AddAttribute(RowNumTxt, Format(LineNo));
        TempXMLBuffer.GetParent();
    end;

    local procedure FillString(Str: Text[250]; Length: Integer): Text[50]
    var
        StringLen: Integer;
        i: Integer;
        Result: Text[50];
    begin
        StringLen := StrLen(Str);
        Result := '';
        i := 0;
        while Length - StringLen > i do begin
            i += 1;
            Result[i] := '0';
```

```
        end;
        Result += Str;
        exit(Result);
    end;

    local procedure SaveXMLToFile(var TempXMLBuffer: Record "XML Buffer"
temporary; IssueDate: Date; DocumentCounter: Integer)
    var
        XMLBufferReader: Codeunit 1239;
        TempBlob: Codeunit "Temp Blob";
        InStr: InStream;
        ClientFileName: Text;
    begin
        XMLBufferReader.SaveToTempBlob(TempBlob, TempXMLBuffer);
        TempBlob.CREATEINSTREAM(InStr);
        ClientFileName := FolderPath + GetFileName(IssueDate, DocumentCounter) +
'.xml';
        DownloadFromStream(InStr, '', FolderPath,
FileManagement.GetToFilterText('', ClientFileName), ClientFileName);
    end;

    local procedure SaveXMLToZip(var TempXMLBuffer: Record "XML Buffer" temporary;
IssueDate: Date; DocumentCounter: Integer)
    var
        XMLBufferReader: Codeunit 1239;
        TempBlob: Codeunit "Temp Blob";
        InStr: InStream;
        ClientFileName: Text;
    begin
        XMLBufferReader.SaveToTempBlob(TempBlob, TempXMLBuffer);
        TempBlob.CreateInStream(InStr);
        ClientFileName := FolderPath + GetFileName(IssueDate, DocumentCounter) +
'.xml';
        DataCompression.AddEntry(InStr, ClientFileName);
    end;

    local procedure GetFileName(IssueDate: Date; DocumentCounter: Integer): Text
    begin
        exit(FillString(CompanyInformation."KDA Region Code (Tax)", 2) +
```

```
            FillString(CompanyInformation."KDA Adm. District Code Tax", 2) +
            FillString(CompanyInformation."Registration No.", 10) +
            FillString(FormType, 3) +
            FillString(FormSubType, 3) +
            FillString(FormVersion, 2) +
            FillString(DocumentState, 1) +
            FillString(DocumentType, 2) +
            FillString(PeriodType, 1) +
            FillString(Format(Date2DMY(IssueDate, 2)), 2) +
            FillString(Format(Date2DMY(IssueDate, 3)), 4) +
            FillString(CalcCStiOrig(CompanyInformation."KDA Region Code (Tax)",
CompanyInformation."KDA Adm. District Code Tax"), 4) +
            FillString(Format(DocumentCounter), 1)
            );
    end;

    local procedure CheckEditable(FieldEdit: Text)
    begin
        case FieldEdit of
            'VATAppendixNo':
                begin
                    VatAppendixNoEditable := true;
                    VatStartingEndingDatesEditable := false;
                    RegistrationDateEditable := false;

                    VATInvoiceStartingDate := 0D;
                    VATInvoiceEndingDate := 0D;
                    RegistrationDate := 0D;
                end;
            'StartingEndingDates':
                begin
                    VatAppendixNoEditable := false;
                    VatStartingEndingDatesEditable := true;
                    RegistrationDateEditable := false;

                    VATAppendixNo := '';
                    RegistrationDate := 0D;
```

```
                end;
            'RegistrationDate':
                begin
                    VatAppendixNoEditable := false;
                    VatStartingEndingDatesEditable := false;
                    RegistrationDateEditable := true;

                    VATAppendixNo := '';
                    VATInvoiceStartingDate := 0D;
                    VATInvoiceEndingDate := 0D;
                end;
            else begin
                    VatAppendixNoEditable := true;
                    VatStartingEndingDatesEditable := true;
                    RegistrationDateEditable := true;

                    VATAppendixNo := '';
                    VATInvoiceStartingDate := 0D;
                    VATInvoiceEndingDate := 0D;
                    RegistrationDate := 0D;
                end;
        end;
    end;
    local procedure GetAccountantInfo(AccountantNo: Code[20]; var AccountantName:
Text[62]; var AccountantTaxID: Code[12])
    var
        Employee: Record Employee;
    begin
        AccountantName := '';
        AccountantTaxID := '';

        if Employee.Get(AccountantNo) then begin
            AccountantName := Employee."First Name" + ' ' +
UpperCase(Employee."Last Name");
            AccountantTaxID := Employee."KDA Tax ID";
        end;
    end;
}
```