Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
бакалавр
(ступінь вищої освіти)

на тему РОЗРОБКА ПРОГРАМНОГО ЗАСТОСУНКУ ВИКОНАННЯ ГРОШОВИХ РОЗРАХУНКОВИХ ОПЕРАЦІЙ
DEVELOPMENT OF SOFTWARE APPLICATION FOR CASH SETTLEMENT OPERATIONS

Виконав: студент 4 курсу, групи КНТ-118
Спеціальності 121 Інженерія програмного забезпечення
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

Петелін Д.Д.
(прізвище та ініціали)

Керівник Гладкова О.М.
(прізвище та ініціали)

Рецензент Зеленьова І.Я.
(прізвище та ініціали)

2022

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ФКНТ
Кафедра програмних засобів
Ступінь бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
_____С.О. Субботін
"____" _____20___року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Петеліна Данила Дмитровича
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка програмного застосунку виконання грошових розрахункових операцій. Development of Software Application for Cash Settlement Operations.
керівник проєкту (роботи) Гладкова Ольга Миколаївна, к.т.н., ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від "27" квітня 2022 року № 102
2. Строк подання студентом проєкту (роботи) червень 2022 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне Завдання
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Розробка архітектури програми. 3. Основні рішення щодо реалізації компонентів системи. 4. Інструкції по використанню вебзастосунку.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	Гладкова О.М., доцент		
Нормоконтроль	Дейнега Л.Ю., ст. викладач		

7. Дата видачі завдання “04” квітня 2022 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2-3 тижні	Розділ 1
3	Розробка архітектури програми	4 тиждень	Розділ 2
4	Розробка програми	5-6 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми	7 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	8 тиждень	Додатки
7	Захист роботи	9 тиждень	

Студент(ка) _____ Петелін Д.Д.
(підпис) (прізвище та ініціали)

Керівник проекту (роботи) _____ Гладкова О.М.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи бакалавра: 87 с., 4 табл., 6 рис., 2 дод., 12 джерел.

РОЗРОБКА ПРОГРАМНОГО ЗАСТОСУНКУ ВИКОНАННЯ ГРОШОВИХ РОЗРАХУНКОВИХ ОПЕРАЦІЙ, JAVA, JAVA EE8, INTELLIJ IDEA.

Об’єкт дослідження – програмні засоби для розрахунку грошових операцій.

Предмет дослідження – програмний реєстратор розрахункових операцій. Система виписки електронних касових чеків, яку використовують продавці.

Мета роботи – створення програмного застосунку для виконання розрахункових операцій, виконання запитів до сервера(запит стану сервера, запит доступних об’єктів і т.д.) та виписки електронних касових чеків.

Матеріали, методи та технічні засоби: функціональне програмування, мови програмування Java, бібліотеки Java EE8, персональний комп’ютер з процессором Intel Core i5 під управлінням операційної системи Windows.

Результати. Створено максимально оптимізований програмний застосунок на основі бібліотеки Java EE8 та серверного REST API, за допомогою якого користувач може відсилати запити до сервера, з метою отримати інформацію, що до стану сервера, переліку доступних об’єктів, операторів (касирів) для суб’єкта господарювання і т.д., а також формувати та виписувати фіскальні чеки.

Висновки. Розроблено застосунок для формування та виписки фіскальних чеків.

Галузь використання – програмний продукт для підприємств.

ABSTRACT

Explanatory note to the bachelor's thesis: 87 p., 4 tables, 6 fig., 2 appendix, 12 sources.

DEVELOPMENT OF SOFTWARE APPLICATION FOR MONEY SETTLEMENT OPERATIONS, JAVA, JAVA EE8, IntelliJ IDEA.

The object of research - software for calculating cash transactions.

The subject of research - software registrar of settlement operations. Electronic cashier's checkout system used by the seller.

The purpose of the work is to create a software application for settlement operations, execution of requests to the server (request for server status, request for available objects, etc.) and issuance of electronic cash receipts.

Materials, methods and technical means: functional programming, Java programming languages, Java EE8 libraries, personal computer with Intel Core i5 processor running Windows operating system.

Results. The most optimized software application based on Java EE8 library and server REST API, with which the user can send requests to the server in order to obtain information about the server status, list of available objects, operators (cashiers) for the business entity etc., as well as to generate and issue fiscal checks.

Conclusions. An application for the formation and issuance of fiscal checks has been developed.

Field of use - a software product for businesses.

ЗМІСТ

С.

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

ОС – операційна система;

ПРРО – програмний реєстратор розрахункових операцій, система виписки електронних касових чеків;

Сервер – фіскальний сервер реєстрації розрахункових операцій;

IDE – (Integrated Development Environment) інтегроване середовище розробки;

REST – Representational State Transfer.

ВСТУП

1 січня 2022 року стало для підприємців вирішальним.З цього дня було внесено зміни до Закону України «Про застосування реєстраторів розрахункових операцій [1] у сфері громадського харчування, торгівлі та послуг які вступили в силу. Це означає, що це ФОП другої-четвертої повинні будуть запровадити використання РРО. Простіше касовий апарат.

Вартість такого касового апарату складає приблизно 20 тисяч гривень, але проблему можна вирішити за допомогою програмного РРО [2], котрі можна використовувати як на персональних комп'ютерах, так і на мобільних пристроях на базі android чи IOS.

Програмний РРО – це застосунок, програма чи сервіс, яку можна встановити на доступні електронні гаджети та імітувати роботу касового апарату. Також ПРРО можна інтегрувати з програмними продуктами підприємства, такими як: 1С, Парус та інші. Вартість такого застосунку набагато менша , ніж традиційного касового апарату, не потребує покупки допоміжного обладнання, швидко налагоджується(реєструється) та простий у використанні.

Історія касових апаратів починається з 1883 року і до цього дня вони складають конкретні механізмами, що проводять операції з розрахунків та друкують чеки для покупців. Масове використання програмних реєстраторів приносить глобальні зміни у сфері підприємництва, порівнянна з винаходом холодильника чи контейнерних перевезень. Ці послуги дозволяють малому бізнесу вийти з тіні, а владі навести лад у сфері малого підприємництва.

З часом ми побачимо результати від такого нововведення.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної області

Для спочатку розглянемо як саме процюють розрахункові реєстратори. Програмні та апаратні РРО мають виконувати одну й ту саму функцію – реєструвати розрахункову операцію у податковій, а саме факт передачі грошей клієнту за отриманий товар чи послугу. Процедура виконується в декілька кроків:

– РРО, створивши чек, надсилає його на фіскальний сервер Державної податкової служби України;

– сервер встановлює йому номер та відправляє назад;

– покупець може отримати чек у роздрукованому вигляді або отримати його електронну версію.

Чек, виданий за допомогою РРО може перевірити будь-хто на сайті ДПС України.

У випадку якщо не має зв’язку із сервером ДПС, у реєстратора є 2000 номерів запасу, які самостійно надає чекам. Після того, як зв'язок відновлено дані про виконані операції відправляються на сервер усі разом.

Є декілька кілька типів ПРРО, які в основному відрізняються за місцем їх розташування:

– стаціонарний встановлюється на певною адресу та призначений для точок роздрібного продажу або в установах, які не виконують віїзд на об’єкт для надання послуг;

– пересувний можна зареєструвати на мобільну господарську одиницю (наприклад автомобіль) та використовується по всій країні, пристрій такого плану можна зареєструвати, наприклад для пересувного кафе;

– каса самообслуговування – це такий же стаціонарний РРО, який передбачає присутність в операції продавця, зробити розрахунок покупець може самостійно, такі каси були обладнані в супермаркетах;

– реєстратор для онлайн-торгівлі використовується за адресою, зазначеною в заяві про реєстрацію та призначений для проведення онлайн-операцій з сайтів.

Програмний РРО можна встановити на будь-який гаджет, що має доступ до мережі та ОС, включаючи Android.

1.2 Огляд існуючих методів вирішення завдання

Пропуную розглянути послуги, що виконують ПРРО. Це будуть платні та безкоштовні розробки, які використовують представники бізнесу в Україні.

1.2.1 Платіжний хмарний сервіс “Fondy”

Нема потреби у завантаженні програми – все виконується в браузері, клієнт може використовувати можливості сервіса з свого кабінету. Але присутній один недолік — сервіс доступний лише для тих клієнтів, які оплачують за допомогою Fondy (рис. 1.1). Якщо ви підключете та будете використовувати сервіс, то з вас не будуь зніматися кошти, але за проведення оплати з вас буде знято стандартну комісію [3].

З допомогою ПРО цієї компанії ви маєте можливість швидко фіскалізувати продажі та подавати онлайн-звіти до податкової служби. Даний сервіс розробник обслуговує самостійно, тим часом клієнт має достум до всіх інструментів сервісу.

Сервіс має дуже просту реєстрацію - крім усім звичайного заповнення форми ви маєте можливість зареєструватися в один клік через облікові записи Google, Facebook або Linkedin.

З переваг — інтеграція ПРРО з іншими можливостями Fondy, у тому числі понад 50 плагінів. Сервіс надає можливість проводити виплати масово, створювати запрошення для клієнтів з інших країн, за допомогою конструктора формувати платіжні кнопки. Також присутні різні інструменти

для моніторингу фінансових подій. Сервіс підтримує різні карти та платіжні системи, зокрема Apple Pay.

Усі операції максимально автоматизовані, а звіти доступні на панелі керування будь-коли.

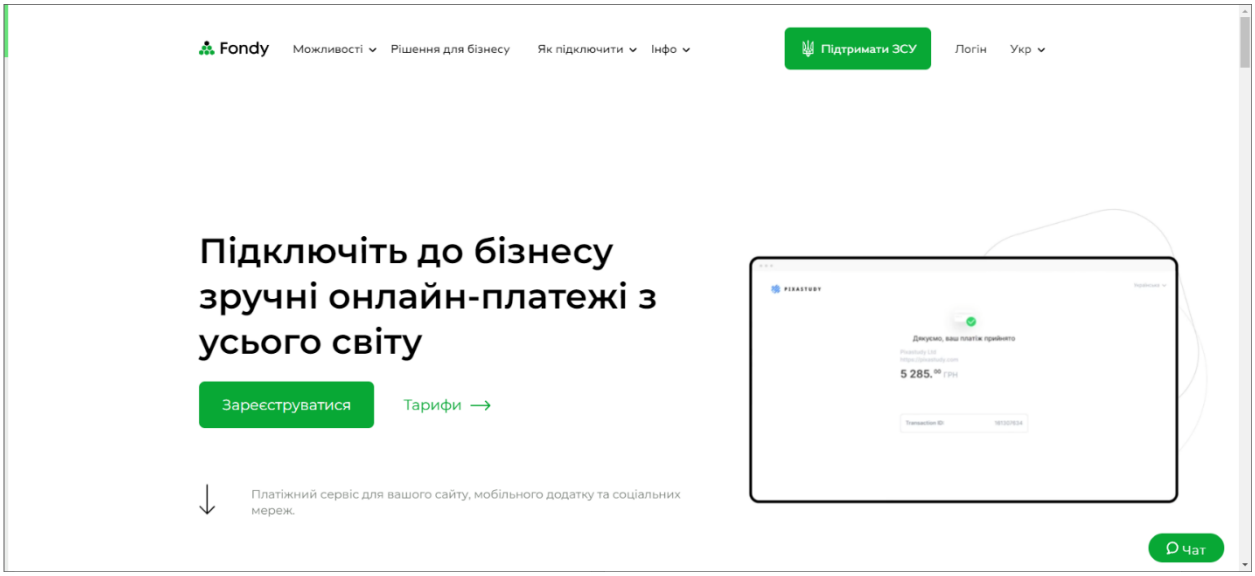


Рисунок 1.1 – Платіжний хмарний сервіс “Fondy” [3]

1.2.2 «Сота каса» та Cashälot

Група компанії «Інтелектуальний сервіс» , яка складає велику команду спеціалістів з бухгалтерського обліку, яка вже 25 років займається створенням програмного забезпечення для електронної роботи з фінансовими документами створила ці два продутки.

Дані сервіси дозволяють:

– реєструвати будь-яку кількість касирів та кас;

– автоматично формувати чеки та звіти та фіскалізувати розрахунки;

– відправляти клієнтам чеки різними зручними способами – через месенджер, на пошту або SMS;

– прийняття оплати у різних форматах – готівкою, кредитом, карткою, використовуючи подарунковий сертифікат.

"Сота каса" [4] - це хмарний сервіс, що замінює касовий аппарат (рис. 1.2). Він доступний як для фізичних, так і для юридичних осіб, та може працювати на різних пристроях. Він не потребує встановлення на конкретний гаджет – користувач має можливість зайти до свого кабінету з будь-якого браузера, просто ввівши свої дані(логін та пароль) на сайті.

Рисунок 1.2 – Платіжний хмарний сервіс "Сота каса" [4]

Cashälot [5] – це десктопний застосунок, який встановлюється на комп'ютер (рис. 1.3). Якщо порівнювати два програмні продукти, то хмарний сервіс "Сота каса" залежить від з'єднання з мережею інтернет, а Cashälot має можливість автономно працювати дозволені йому 36 годин поспіль, при цьому дані підприємця зберігаються на його пристрої.

Програма купується на зазначену кількість кас, ціна буде залежати від форми реєстрації підприємницької діяльності. Для прикладу, якщо програма купується на 3 каси для юридичної особи, то вартість за ці 3 каси складатиме 4290 гривень.

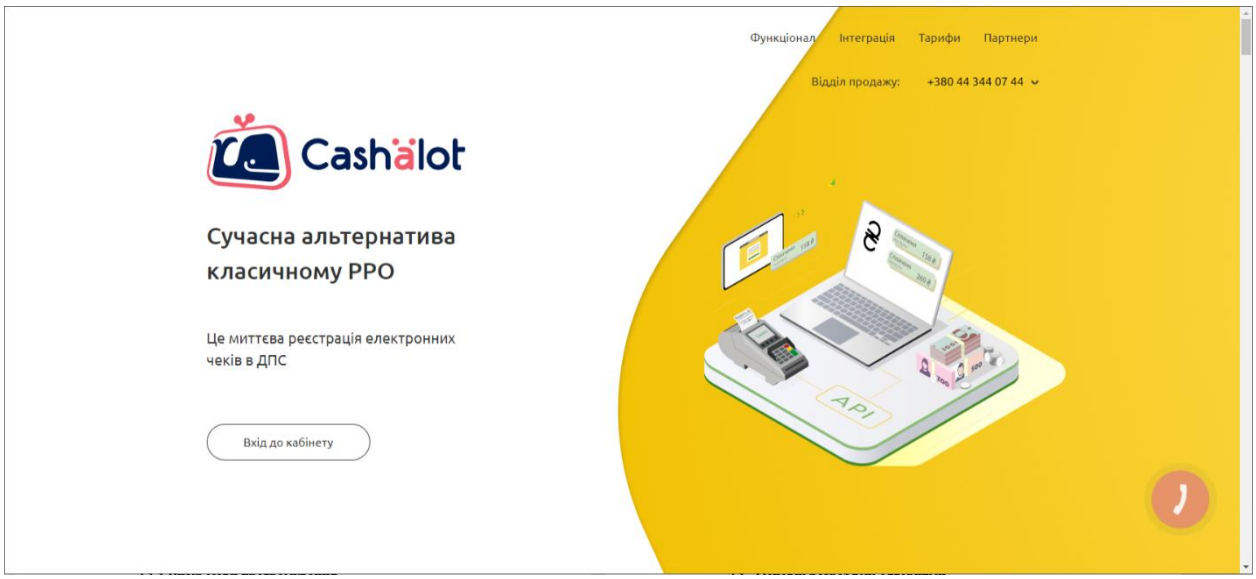


Рисунок 1.3 - Платіжний хмарний сервіс Cashälot [5]

1.2.3 Checkbox

Checkbox каса (рис. 1.4) розпочала свою роботу у жовтні 2020 року. Користувачі можуть одержати продукт із безкоштовним встановленням на 1 місяць тестування. На завершення цього періоду вартість такої послуги складатиме 149 гривень на місяць за касу[6].

Сервіс орієнтований на 3 категорії підприємців:

– малий бізнес;

– онлайн-магазини та послуги;

– великі мережі роздрібної торгівлі.

Одним із засновників компанії є Дмитро Дубілет, він же співзасновник Монобанку. Отже можна зробити висновок, що каса Checkbox буде мати схожий рівень надання послуг.

Можливості сервісу:

– автоматично закрити зміни та згадка про зміни які не закриті;

– зміна кількості кас, касирів та торгових точок: під час реєстрації дані відправляються до ДПС автоматично, завдяки чому користувачу не потрібно ходити до податкової;

– створення звіту з вибраної каси обравши будь-який діапазоні дат;

– зручний розділ «Відповіді на запитання», що дає змогу розпочати роботу з сервісом, включає в себе інструкції та правила роботи з касою;

– на панелі керування клієнта та сайті відображаються важливі повідомлення, наприклад, якщо сервери ДПС недоступні, і орієнтовний час їх увімкнення;

– сервіс має партнерів, які займаються продажем касового обладнання та пропонують варіанти купівлі потрібних пристроїв (наприклад, принтера для друку чеків).

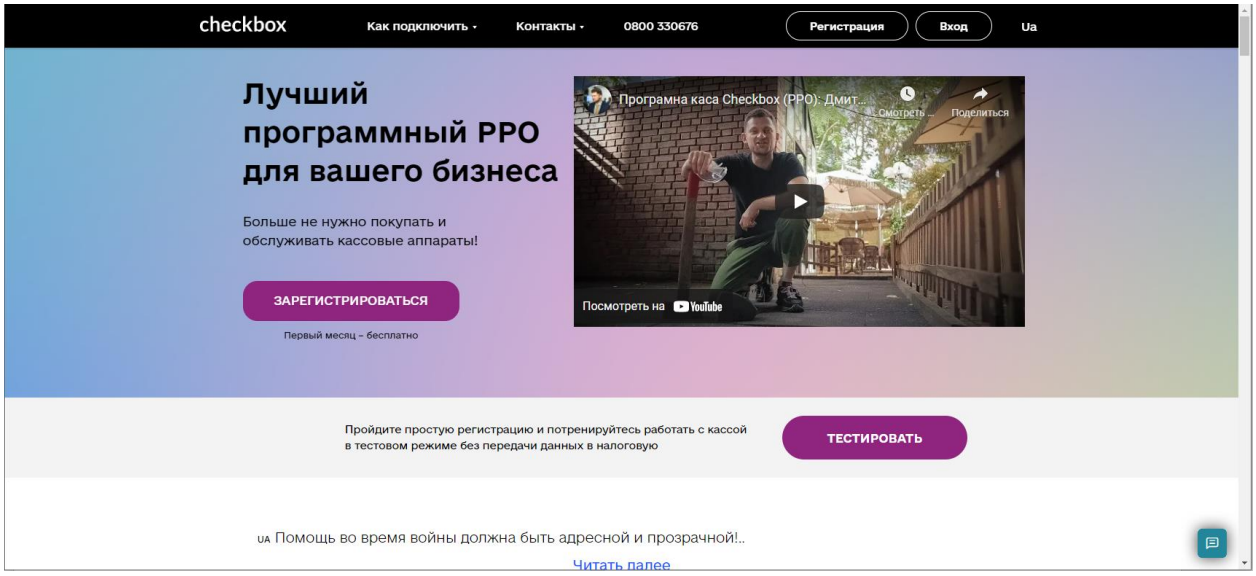


Рисунок 1.4 – Програмна каса Checkbox [6]

1.2.4 Веб сервіс Вчасно-каса[6]

Веб-сервісом «Вчасно Каса» (рис. 1.5) можна використовувати на компьютері або планшеті, зайшовши до особистого кабінету або встановивши мобільний додадок.

Ці продукти дозволяють:

– почти вести свій бізнес підприємцю-початківцю;

– створювати будь-яку кількість чеків та відправляти до месенджерів або на пошту;

– інтегрувати сервіс із онлайн-магазином;

– працювати без Інтернету.

Мета проекту полягає не тільки у допомозі підприємцям з фіскалізацією операцій, а й у їхньому навчанні.

Зайшовши на сайт ві одразу побачите інструкцію з формування чеків, також на головній сторінці присутнє посилання на цілий відеокурс «Все про кас та чеках». Такж на сайт має досить великий розділ, який включає в себе відповіді на часті запитання та поради з використання.

Загалом сервіс досить простий і спрямований на те, щоб ФОП почали використовувати ПРРО у своїй роботі.

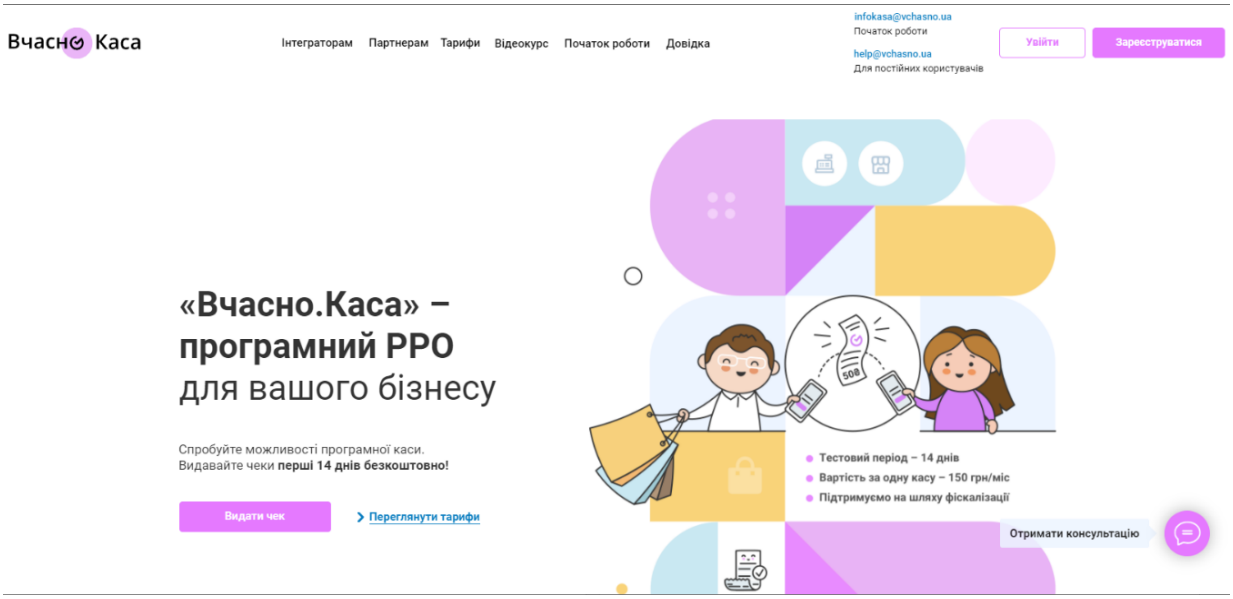


Рисунок 1.5 - Веб-сервісом «Вчасно Каса»

1.3 Порівняння функціоналу існуючих рішень

Проаналізувавши існуючі рішення реалізації застосунків та вебсайтів Програмного РРО, було виділено основні переваги та недоліки усіх функціональних особливостей існуючого програмного забезпечення (табл. 1.1).

Таблиця 1.1 – Порівняльна характеристика найпоширеніших програмних продуктів

Функціональні можливості	Назва програмних продуктів					
	“Fondy”	«Сота каса»	Cashälot	Checkbox	Вчасно-каса	Новий продукт
Зручний дизайн	+	+	-	+	+	+
Легкочитаємий текст	+	+	-	+	+	+
Пошук за ключовою фразою	+	+	+	+	+	+
Відсутність реклами	-	-	-	-	-	+
Адаптивність до різних екранів пристроїв	+	-	+	+	+	+
Посилання на джерело	+	-	-	+	+	+

1.4 Технічне завдання

Програмний продукт буде реалізований як серверний застосунок і матиме свою архітектуру усі данні будуть отримуватись/відправлятись у форматах XML та JSON, залежно від запиту.

Програмний продукт призначений для виконання інформаційних запитів до сервера, а саме:

– запит стану сервера;

– запит доступних об’єктів;

– запит переліку операторів(касирів) для суб’єкта господарювання;

– запит стану ПРРО;

– запит чека розширений;

– запит Z-звіту розширений;

– запит переліку змін за період;

– запит переліку документів зміни;

– запит підсумків останньої зміни;

– запит відомостей про документ за локальним номером.

Також програма генерує код для офлайн сесії та виконує накладання цифрового підпису для відправки документів.

1.5 Висновок до розділу 1

В розділі було проаналізовано існуючі аналоги та зведено таблицю порівняння недоліків та переваг.

Таким чином новий програмний продукт має поєднати в собі функціонал, який є у існуючих альтернативах та стати повноцінним інструментом.

2 ВИБІР ІНСТРУМЕНТАРІЮ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Вибір мови програмування

Для розробки серверного застосунку було обрано мову програмування JAVA версії 1.8 [7].

Java — суворо типізована об'єктно-орієнтована мова програмування загального призначення, розроблена компанією Sun Microsystems (надалі придбаною компанією Oracle).

Програми написані на Java транслюються в байт-код Java, який виконується Java Virtual Machine - це програма, яка виконує обробку байтового коду та передає інструкції обладнання як інтерпретатор.

Повна незалежність байт-коду від операційної системи та обладнання є перевагою такого способу виконання програм, що дає можливість виконання програм на Java на будь-якому пристрої, на якому встановлена відповідна віртуальна машина.

Java має ще одну важливу особливість технології, це гнучка система безпеки, у рамках якої виконання програми повністю керується віртуальною машиною.

Усі операції, що перевищують встановлені повноваження програми (наприклад, спроба несанкціонованого доступу до даних або конект з іншим пристроєм), будуть негайно перервані.

Часто до вад концепції VM відносять зниження продуктивності. Деякі удосконалення дещо збільшили швидкість виконання Java-програм:

– використання технології трансляції байт-коду в машинний код безпосередньо під час роботи програми (JIT-технологія) з можливістю збереження версій класу в машинному коді,

– широке використання платформно-орієнтованого коду (native-код) у бібліотеках за замовчуванням;

– апаратні засоби, котрі забезпечують більш швидку обробку байт-коду (технологія Jazelle, що підтримується деякими процесорами архітектури ARM).

2.2 Вибір фреймворку

Для розробки серверного застосунку було обрано фреймворк JavaEE [8].

Java EE в основному використовується у високопродуктивних проектах, які потребують надійності, масшабованості, гнучкості та є промисловою технологією.

Sun пропонує безкоштовний комплект розробки, SDK, що дає можливість підприємствам створювати свої системи, при цьому не витрачаючи багато коштів, цей фактор сприяє популярності Java EE. Цей пакет містить сервер програм GlassFish з ліцензією для розробки.

Сервер програм J2EE (так званий контейнер J2EE) - це реалізація системи відповідно до специфікації J2EE, що забезпечує роботу модулів з логікою конкретної програми.

Сервер включає в себе як мінімум перелік сервісів:

– EJB-контейнер, котрий підтримує автоматичну синхронізацію Java-об'єктів з базою даних (кероване збереження контейнера, керовану стійкість);

– JMS - сервіс який виконує функцію доставки повідомлень між компонентами та серверами;

– керування ресурсами (доступність до файлової системи, СУБД, поштового сервера тощо);

– безпеки та захисту даних;

– підтримка транзакцій (зокрема двофазних і розподілених).

– веб-сервер та сервлет-сервер;

– підтримка веб-сервісів;

– JSF.

Розробники J2EE-програм також пишуть свої програми відповідно до специфікацій J2EE, що забезпечує їх роботу всередині таких серверів.

Технологію J2EE слід використовувати для створення великих проектів, таких як, організація складних веб-порталів та надання онлайн послуг, особливо якщо є необхідність у забезпеченні безперебійного функціонування багатьох тисяч користувачів.

Також, алтернативою Java EE е фреймворк Spring.

Spring Framework [9] (коротше Spring) - надає комплексну модель програмування та конфігурації для сучасних корпоративних програм на базі Java - на будь-якій платформі розгортування.

Ключовим елементом Spring є інфраструктурна підтримка на рівні програми: Spring зосереджується на «підготовці» корпоративних додатків, щоб команди могли зосередитися на бізнес-логіці на рівні програми без зайвих зв’язків із конкретними середовищами розгортання. Незважаючи на те, що Spring не забезпечував будь-якої конкретної моделі програмування, він став широко поширеним в Java-спільноті. Головним чином як альтернатива і заміна моделі Enterprise JavaBeans.

Spring надає Java-розробникам більшу свободу у проектуванні; крім того, користувачу добре документовані та легкі у використанні різні засоби вирішення проблем, що виникають під час створення програм корпоративного масштабу.

Тим часом, особливості ядра Spring застосовні в будь-якому Java-додатку, і існує безліч розширень та удосконалень для побудови веб-програм на Java Enterprise платформі.

Саме тому Spring має велику популярність серед розробників і визнається як стратегічно важливий фреймворк.

Проаналізувавши основні переваги та недоліки цих фреймворків (табл. 2.1) вирішено, що Java EE найліпше підходить для даного серверного застосунку.

Таблиця 2.1 – Порівняльна характеристика найпоширеніших фреймворків та бібліотек

Критерії	Java EE	Spring
Архітектура	Заснований на тривимірній архітектурній структурі, тобто на логічних рівнях, клієнтських рівнях та рівнях уявлення.	Він заснований на багаторівневій архітектурі, яка включає безліч модулів. Ці модулі виконані поверх його основного контейнера.
Мова	Він використовує об'єктно-орієнтовану мову високого рівня, яка має певний стиль та синтаксис.	Він не має певної моделі програмування.
Інтерфейс	Зазвичай він має графічний інтерфейс користувача, створений з API-інтерфейсів Project Swing або Abstract Window Toolkit.	Синтаксис скрізь однаковий незалежно від IDE або компілятора
Впровадження залежностей	Використовує впровадження залежностей	Використовує впровадження залежностей
Структура	Можлива веб, або не веб-основа	На основі майже 20 модулів
Швидкість	Непогана швидкість	Швідкість менша, ніж у Java EE

2.3 Вибір середовища розробки

В якості середовища для розробки було обрано Intellij idea.

IntelliJ IDEA [10] — це інтегроване середовище розробки (IDE) для мов JVM, розроблене для максимальної продуктивності розробників. Він виконує рутинні й повторювані завдання за вас, забезпечуючи розумне завершення коду, статичний аналіз коду та рефакторинг, а також дозволяє зосередитися на яскравій стороні розробки програмного забезпечення, роблячи це не тільки продуктивним, але й приємним..

Перша версія почала працювати в січні 2001 року та швидко набула популярності як перше середовище для Java з широким набором інтегрованих інструментів для рефакторингу, які дозволяли програмістам швидко реорганізовувати вихідні тексти програм. Дизайн середовища орієнтований на продуктивність роботи програмістів, дозволяючи сконцентруватися на функціональних завданнях, тоді як IntelliJ IDEA бере на себе виконання рутинних операцій.

Починаючи з шостої версії продукту IntelliJ IDEA надає інтегрований інструментарій для розробки графічного інтерфейсу користувача. Серед інших можливостей середовище добре сумісне з багатьма популярними вільними інструментами розробників, такими як CVS, Subversion, Apache Ant, Maven і JUnit. У лютому 2007 року розробники IntelliJ анонсували ранню версію плагіна для підтримки програмування мовою Ruby.

Починаючи з версії 9.0, середовище доступне у двох редакціях: Community Edition і Ultimate Edition. Community Edition являє собою повністю вільну версією, доступною під ліцензією Apache 2.0, в ній реалізовано повну підтримку Java SE, Kotlin, Groovy, Scala, а також інтеграцію з найбільш популярними системами керування версіями. У редакції Ultimate Edition, доступний під комерційною ліцензією, реалізована підтримка Java EE, UML-діаграм, підрахунок покриття коду, а також підтримка інших систем керування версіями, мов та фреймворків.

Конкурентом IntelliJ IDEA є середовище Eclipse.

Eclipse[11] — це інтегроване середовище розробки (IDE), що використовується в комп'ютерному програмуванні. Він містить базову робочу область та розширювану систему плагінів для налаштування середовища. Це друга за популярністю IDE для розробки Java, і до 2016 року вона була найпопулярнішою.

Eclipse написаний здебільшого на Java, його в основному використовують для розробки додатків Java, але він також може використовуватися для розробки додатків на інших мовах програмування за допомогою плагінів, включаючи Ada, ABAP, C, C++, C#, Clojure, COBOL, D, Erlang, Fortran, Groovy, Haskell, JavaScript, Julia, Lasso, Lua, NATURAL, Perl, PHP, Prolog, Python, R, Ruby, Rust, Scala і Scheme . Його також можна використовувати для розробки документів за допомогою LaTeX (за допомогою плагіну TeXlipse) і пакетів для програмного забезпечення Mathematica. Середовища розробки включають інструменти розробки Eclipse Java для Java та Scala, Eclipse CDT для C/C++ та Eclipse PDT для PHP, серед інших.

Самі відомі програми на основі Eclipse — різні «Eclipse IDE» для розробки програмного забезпечення на різних мовах (як приклад, найбільш популярний IDE під Java, який спочатку підтримувався, не покладається на будь-які закриті розширення, використовує стандартний відкритий API для доступу до Eclipse Platform). Проаналізувавши основні переваги та недоліки цих IDE (табл. 2.2) вирішено, що IntelliJ IDEA більш зручна для розробки застосунків мовою Java.

Таблиця 2.2 – Порівняльна характеристика середовищ для розробки Eclipse та IntelliJ IDEA.

Основа порівняння	Eclipse	IntelliJ IDEA
Налагодження	Налагодження виконується довше	Більш зручне налагодження коду

Рефакторинг	Інтелектуальні рефакторинги відсутні	Має інтелектуальні рефакторинги
Доступність	Повністю безкоштовне	Якщо вам потрібне середовище для веб- або enterprise-розробки, то вам потрібно придбати Ultimate edition
Можливість працювати з базами даних	Відсутня	Присутні SQL інструменти для роботи з базами даних

2.4 Висновки до розділу 2

Для розробки даного застосунку обрано найбільш підходящий фреймворк JavaEE, та зручне середовище для розробки Intellij idea. Мова програмування Java. Також було використано бібліотеку eu-sign(ІІТ) [12]. Дана бібліотека використовується для накладання електронного підпису на документи, що забезпечує їх безпеку.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Архітектура системи

Усі команди, що реалізовані у застосунку відправляються на Фіскальний Сервер розрахункових операцій. На рисунку 3.1 представлено взаємодію компонента із фіскальним сервером.

Сервер оброблює Url:

– “<адреса>/doc“ – одержання документів (чеків, Z-звітів тощо);

– “<адреса>/pck“ – одержання пакетів документів (офлайн документи тощо);

– “<адреса>/cmd“ – одержання команд.

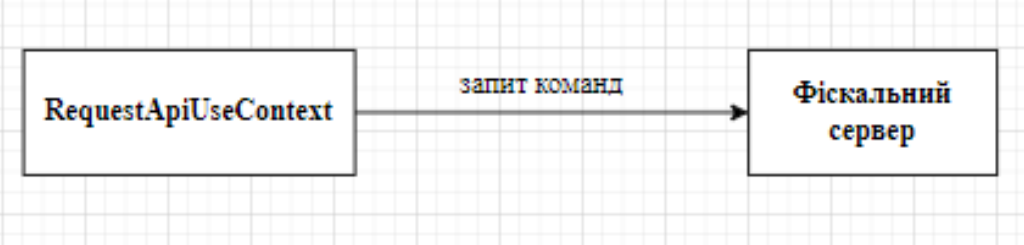


Рисунок 3.1 – Взаємодія компонента RequestApiUseContext з фіскальним сервером

3.2 Основні розроблені функції

Основні розроблені функції застосунку (табл 2.2):

– запит стану сервера;

– запит доступних об’єктів;

– запит переліку операторів(касирів) для суб’єкта господарювання;

– запит стану ПРРО;

– запит чека розширений;

– запит Z-звіту розширений;

– запит переліку змін за період;

– запит переліку документів зміни;

– запит підсумків останньої зміни;

– запит відомостей про документ за локальним номером.

Також застосунок має функцію для формування чеків, у яких виконується розрахунок податків.

Таблиця 3.1 – Опис запитів до сервера.

Мета	**Адреса**	**Тип запиту**	**Результат**	**Зауваження**
Запит стану сервера	http://fs.tax.gov.ua:8609/fs/cmd	POST	Дата і час відображені текстом та мають формат форматі ISO 8601	Запит повинен містити JSON
Запит доступних об’єктів	http://fs.tax.gov.ua:8609/fs/cmd	POST	Користувачу повертається перелік доступних користувачу господарських одиниць і ПРРО	Запит має включати в себе JSON, що засвідчено КЕП користувача
Запит переліку операторів(касирів)	http://fs.tax.gov.ua:8609/fs/cmd	POST	Користувачу повертається перелік усіх операторів, що зареєстровані для суб'єкта господарювання, реєстраційний номер якого (ЄДРПОУ, ДРФО, Картка платника податків) міститься у сертифікаті КЕП, яким засвідчений запит	Запит має включати в себе JSON, що засвідчено КЕП користувача
Запит стану ПРРО	http://fs.tax.gov.ua:8609/fs/cmd	POST	Якщо дані відсутні, користувачу повертається код HTTP відповіді 204 з коментарем “No Content”.	Запит має включати в себе містити JSON, засвідчений

				КЕП користувача
Запит чека розширений	http://fs.tax.gov.ua:8609/fs/cmd	POST	Користувачу повертається чек у форматі Json	Запит має включати в себе JSON

Продовження таблиці 3.1

Мета	**Адреса**	**Тип запиту**	**Результат**	**Зауваження**
Запит Z-звіту розширений	http://fs.tax.gov.ua:8609/fs/cmd	POST	Користувачу повертається Z-звіт у форматі Json	Запит має включати в себе JSON, засвідчений КЕП користувача
Запит переліку змін за період	http://fs.tax.gov.ua:8609/fs/cmd	POST	Користувачу повертається перелік змін за період у форматі Json	Запит має включати в себе JSON, засвідчений КЕП користувача
Запит переліку документів зміни	http://fs.tax.gov.ua:8609/fs/cmd	POST	Користувачу повертається перелік документів зміни за період у форматі Json	Запит має включати в себе JSON, засвідчений КЕП користувача
Запит підсумків останньої зміни	http://fs.tax.gov.ua:8609/fs/cmd	POST	Користувачу повертається підсумки останньої зміни у форматі Json. У разі відсутності даних, повертається код HTTP 204 “No Content”.	Запит має включати в себе JSON, засвідчений КЕП користувача
Запит відомостей про документ за локальним номером	http://fs.tax.gov.ua:8609/fs/cmd	POST	Користувачу повертається Перелік відомостей про документ за локальним номером	Запит має включати в себе JSON, засвідчений КЕП користувача

3.3 Структура програми

Структура проекту складається з таких папок (рис.3.2):

– Conveter – папка, у якій реалізовано класи для конвертації файлів у формат XML;

– Crypto – папка, у якій реалізовано класи для накладання електронного підпису;

– HTTP – папка, у якій реалізовано класи для роботи з веб-сервісами;

– Model – папка, у якій реалізовано класи для опису об’єктів;

– Unit – папка, для виконання J-unit тестів.

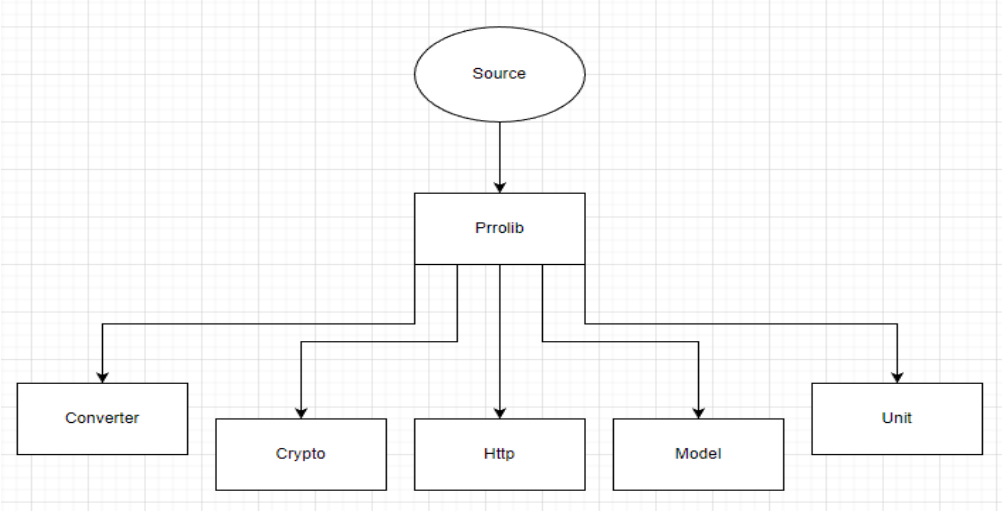


Рисунок 3.2 – Блок-схема папок проекту

3.4 Опис класів програми

Розроблені класи програми та їх опис наведено в таблиці 3.2.

Таблиця 3.2 – опис класів програми.

Назва класу	Опис
Product.java	Клас, що описує товар(назва, ціна, кількість і т.д.)
Documents.java	Клас опису документів

Operator.java	Клас опису оператора(касира)
Operators.java	Клас опису операторів
Shifts.java	Клас опису зміни
ShiftTotals.java	Клас опису підсумків зміни
ShiftTotalsCash.java	Клас опису підсумків зміни

Продовження табліці 3.2

Назва класу	Опис
ShiftTotalsCurrency.java	Клас опису службового внесення/отримання авансу/отримання підкріплення
ShiftTotalsOrderType.java	Клас опису підсумків зміни за типом чека
ShiftTotalPayForm.java	Клас опису підсумку зміни за формою оплати
ShiftTotalsTax.java	Клас опису податків зміни
TaxObjects.java	Клас опису податків
TotalsCurrencyDetails.java	Клас опису підсумків по видам іноземної валюти
TransactionRegistrars	Клас опису реєстрації
RequestApiUseContextTest .java	Кла, у якому виконуються запити до сервера
PdvTest.java	Клас у якому виконуються розрахунки податків

4 ІНСТРУКЦІЇ ПО ВИКОРИСТАННЮ ВЕБЗАСТОСУНКУ

4.1 Визначення вимог до технічних засобів

Вимоги до апаратної частини:

– Процесор, що має частоту 1 ГГц;

– Оперативну пам’ять 512 Мб.

Мінімальний об’єм дискового простору:

– Для 32-розрядної версії простір складає - 4,5 ГБ;

– Для 64-розрядної версії простір складає - 4,5 ГБ.

Сканер, котрий взаємодіє з буфером клавіатури для зчитування штрих-кодів товарів.

Вимоги до операційної системи:

– Windows Vista SP2 і вище;

– постійний доступ інтернету.

Екран: 23.8 (1920x1080) FHD.

Програмне забезпечення:

– Windows 10, 64-х розрядна;

– JDK;

– IntelliJ IDEA.

Мова програмування:

– Java;

– JavaEE.

Мінімальні вимоги до технічних засобів – це будь-який компьютер з можливістю виходу до мережі Інтернет через браузер.

4.2 Інструкція для програміста

Для встановлення програмного продукту та його запуску необхідно мати встановлений JDK(Java Developmant Kit) та середовище для розробки IntelliJ IDEA.

4.3 Інструкція з використання програми для користувача

Встановлення ПРРО зображено на рисунку 4.1 включає в себе наступні кроки:

– переконатись у тому, що ПК відповідає усім необхідним вимогам до системи;

– завантажити на диск дистрибутив;

– запуск файла для встановлення програми setup.exe;

– обрати потрібний шлях для встановлення програми;

– обрати створення ярликів в меню «Пуск» і на Робочому столі;

– натиснути кнопку «Встановити».

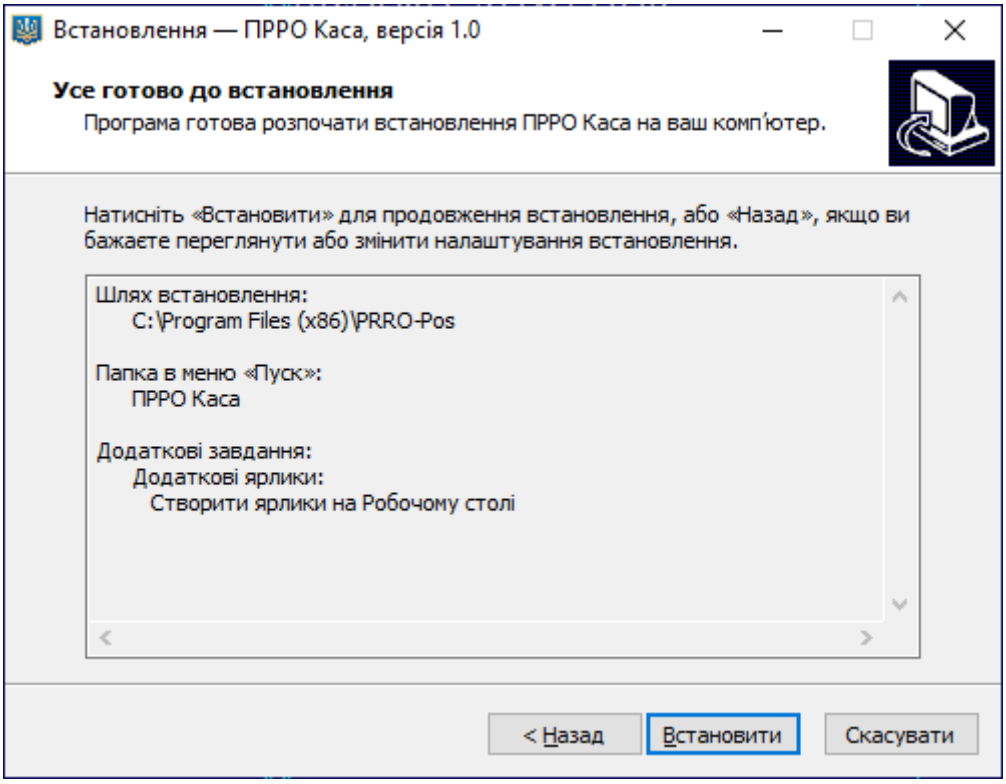


Рисунок 4.1 – Встановлення ПРРО

Після того, як програму встановлено на комп’ютер слід відкрити ПРРО Каса та приступити до роботи. Авторизація в програмі та початок роботи припускає такі кроки:

– реєстрація ПРРО разом з операторами операторами (касирами) в системі здійснюється поданням форм 1-ПРРО і «Повідомлення про надання інформації щодо кваліфікованого сертифіката відкритого ключа (для повідомлень щодо сертифікатів відкритих ключів, які застосовуються в ПРРО)»;

– при вході в програму користувачу слід вказати шлях до каталогу з сертифікатом ЕЦП, що зареєстрований в Електронному кабінеті, і секретним ключем;

– ввести пароль;

– обрати сертифікат користувача;

– обрати господарську одиницю.

Усі господарських одиниць і РРО відображаються відповідно до обраного сертифіката згідно з налаштуванням для даного сертифікату у електронному кабінеті http://cabinet.tax.gov.ua/cashregs/.

Якщо для обраного оператором РРО вже відрито зміну іншим оператором, то цей оператор не зможе працювати з обраним номером РРО

Довідник номенклатури повинен бути заповнений для роботи з РРО.

Неможливо створити чек на товар чи послугу, котрі відсутні в довіднику номенклатури.

Послідовність додавання товарів та послуг в довідник (рис. 4.2):

– за допомогою пункту головного меню «Довідники» - «Номенклатура» слід відкрити довідник номенклатури;

– натиснути кнопку «Додати» в довіднику;

– далі потрібно ввести опис створюваного товару.

Опис товару вводиться за такими рядками:

– внутрішній код товару або послуги, артикул;

– штрих-код – штриховий код товару або послуги;

– код УКТЗЕД – код товарної підкатегорії згідно з УКТ ЗЕД (зазначається у випадках, передбачених чинним законодавством);

– код ДКПП – код послуги згідно з ДКПП (зазначається у випадках, передбачених чинним законодавством);

– акцизна марка – ознака товару, що акцизний податок за акцизною маркою на алкогольні напої;

– найменування – нозва товару або послуги;

– одиниця виміру: код – код одиниці виміру за класифікатором; найменування – одиниця виміру у скороченому варіанті;

– ціна – вартість за одиницю товару або послуги, з урахуванням ПДВ та акцизного податку, якщо РРО працює з цінами, що включають ПДВ та акцизний збір;

– ПДВ – вказується, якщо РРО працює з цінами, що включають ПДВ, а саме літера та ставка;

– акцизний збір – вказується, якщо РРО працює з цінами (тарифами), що мають акцизний збір: літера, ставка.

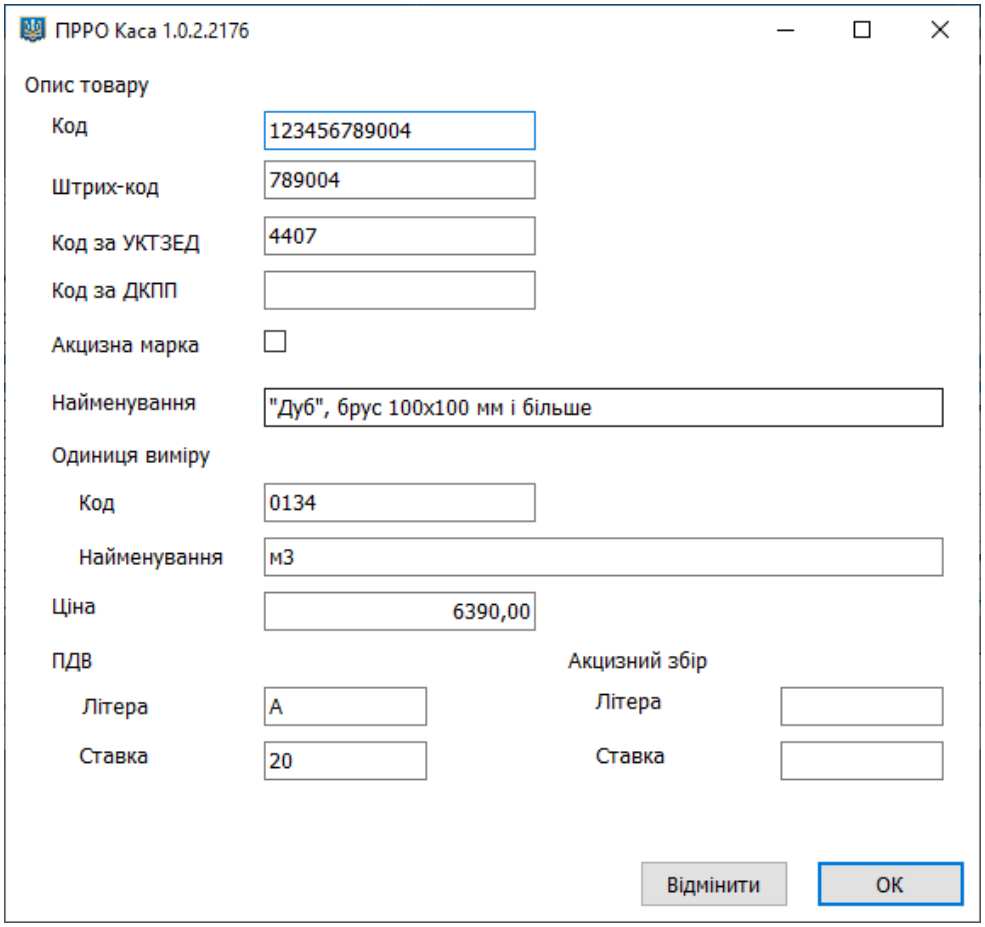


Рисунок 4.2 – Створення/редагування товару

Для підтвердження додавання запису в довідник натиснення кнопки «ОК»

Для коригування запису використовується кнопка «Змінити».

Для видалення запису використовується кнопка «Видалити».

ВИСНОВКИ

Було реалізовано весь функціонал, який був зазначений у цілях проекту.

В ході експлуатації критичних помилок не було виявлено. Застосунок дозволяє швидко и зручно виконувати потрібні користувачу запити та швидко їх обробляти.

Головною перевагою такого застосунку є те, що підприємцю не потрібно витрачати кошти на придбання кассового аппарату. Використання програмного реєстратора є більш дешевим.

Даний продукт можна назвати максимально оптимізованим.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Реєстратори розрахункових операцій [Електронний ресурс]. – Режим доступу: https://business.diia.gov.ua/handbook/finansovij-menedzment/so-take-rro-ta-ak-ce-vikoristovuvati.

2. Програмні реєстратори розрахункових операцій [Електронний ресурс]. – Режим доступу: https://www.bip.net.ua/articles/programni-rro-prro-suchasna-alternatyva-fiskalnym-reyestratoram/.

3. ПРРО Fondy [Electronic resource]. – Access mode: https://docs.fondy.eu/en/.

4. ПРРО “Сота каса” [Електронний ресурс]. – Режим доступу: https://sota-buh.com.ua/account/login.

5. ПРРО Cashalot [Електронний ресурс]. – Режим доступу: https://cashalot.org.ua/.

6. ПРРО Checkbox [Електронний ресурс]. – Режим доступу: https://checkbox.ua/.

7. Java version 1.8 [Electronic resource]. – Access mode: https://www.oracle.com/cis/java/technologies/javase/javase8-archive-downloads.html.

8. “Изучаем Java EE” Энтони Гонсалвес [Электронный ресурс]. – Режим доступа: https://coollib.com/b/516017-entoni-gonsalves-izuchaem-java-ee-7/read.

9. “Pro Spring 4” fourth edition Chris Schafer, Clarence Ho, Rob Harror [Electronic resource]. – Access mode: https://www.oreilly.com/library/view/pro-spring-5/9781484228081/.

10. IntelliJ IDEA [Электронный ресурс]. – Режим доступа: https://www.jetbrains.com/ru-ru/idea/.

11. Eclipce [Electronic resource]. – Access mode: https://www.eclipse.org/.

12. ІІТ захист інформації [Електронний ресурс]. – Режим доступу: https://iit.com.ua/.

ДОДАТОК А

Текст програми

А.1 pom.xml – конфігурація проекту

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.zzs.prrolib</groupId>
    <artifactId>prrolib</artifactId>
    <version>1.0.0</version>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <project.reporting.outputEncoding>UTF-
8</project.reporting.outputEncoding>
    </properties>
    <dependencies>
        <dependency>
            <groupId>com.iit</groupId>
            <artifactId>eu-sign</artifactId>
            <version>1.0.0</version>
        </dependency>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-api</artifactId>
            <version>1.7.32</version>
        </dependency>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-simple</artifactId>
            <version>1.7.32</version>
        </dependency>        <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
            <artifactId>jackson-core</artifactId>
            <version>2.12.5</version>
        </dependency>

        <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
            <artifactId>jackson-annotations</artifactId>
            <version>2.12.5</version>
```

```
    </dependency>       <dependency>
      <groupId>com.fasterxml.jackson.core</groupId>
      <artifactId>jackson-databind</artifactId>
      <version>2.12.5</version>
    </dependency>
    <dependency>
      <groupId>com.fasterxml.jackson.datatype</groupId>
      <artifactId>jackson-datatype-jsr310</artifactId>
      <version>2.12.5</version>
    </dependency>
    <dependency>
      <groupId>com.fasterxml.jackson.datatype</groupId>
      <artifactId>jackson-datatype-jdk8</artifactId>
      <version>2.12.5</version>
    </dependency>
    <dependency>
      <groupId>com.fasterxml.jackson.module</groupId>
      <artifactId>jackson-module-parameter-names</artifactId>
      <version>2.12.5</version>
    </dependency>
    <dependency>
      <groupId>org.junit.jupiter</groupId>
      <artifactId>junit-jupiter</artifactId>
      <version>RELEASE</version>
      <scope>test</scope>
    </dependency>
    <dependency>
      <groupId>org.junit.jupiter</groupId>
      <artifactId>junit-jupiter-engine</artifactId>
      <version>RELEASE</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
  <build>
    <finalName>prrolib</finalName>
    <resources>
      <resource>
        <directory>${basedir}/src/main/resources</directory>
        <excludes>
          <exclude>lib/**</exclude>
          <exclude>test/**</exclude>
          <exclude>prod/**</exclude>
```

```
        </excludes>
    </resource>
</resources>
<plugins>

    <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <configuration>
            <source>8</source>
            <target>8</target>
        </configuration>
    </plugin>
    <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-jar-plugin</artifactId>
        <configuration>
            <archive>
                <manifest>
                    <addClasspath>true</addClasspath>
                    <classpathPrefix>dependency/</classpathPrefix>
                    <mainClass>com.zss.prro.test.TestPrro</mainClass>
                </manifest>
            </archive>
        </configuration>
    </plugin>
    <plugin>
        <artifactId>maven-assembly-plugin</artifactId>
        <configuration>
            <archive>
                <manifest>
                    <mainClass>com.zss.prrolib.ShowWorking</mainClass>
                </manifest>
            </archive>
            <descriptorRefs>
                <descriptorRef>jar-with-dependencies</descriptorRef>
            </descriptorRefs>
        </configuration>
    </plugin>

</plugins>
```

```
    </build>
</project>
```

A.2 Product.java

```
package com.zss.prrolib.model.db;
import java.math.BigDecimal;
import java.math.RoundingMode;
public class Product {
    private String code;
    private String barCode;
    private String uktzed;
    private String name;
    private String unitnm;
    private Double price;
    private Double amount;
    private Boolean hasExciseLabel = false;
    private String exciseLabel;
    private Double prcPdv;
    private Character letterPdv;
    private Double prcExcise;
    private Character letterExcise;
    public Product() {
    }
    public Product(String name, String unitnm, Double price, Double amount,
Double prcPdv, Character letterPdv, Double prcExcise, Character letterExcise) {
        this.name = name;
        this.unitnm = unitnm;
        this.price = price;
        this.amount = amount;
        this.prcPdv = prcPdv;
        this.letterPdv = letterPdv;
        this.prcExcise = prcExcise;
        this.letterExcise = letterExcise;
    }
    public BigDecimal getBDPrice(){
        return
(BigDecimal.valueOf(price).multiply(BigDecimal.valueOf(amount))).setScale(2,
BigDecimal.ROUND_HALF_UP);
```

```
    }
    public BigDecimal getSumExcise(){
        BigDecimal priceExcise;
        priceExcise = getBDPrice()
       .divide(BigDecimal.valueOf(getPrcExcise()).add(BigDecimal.valueOf(100)),
10, RoundingMode.HALF_UP)
                .multiply(BigDecimal.valueOf(getPrcExcise()))
                .setScale(2, BigDecimal.ROUND_HALF_UP);
        return priceExcise;
    }

    public BigDecimal getSumPdv(){
        BigDecimal pricePdv;

        pricePdv = getBDPrice().subtract(getSumExcise())

.divide(BigDecimal.valueOf(getPrcPdv()).add(BigDecimal.valueOf(100)), 10,
RoundingMode.HALF_UP)
                .multiply(BigDecimal.valueOf(getPrcPdv()))
                .setScale(2, BigDecimal.ROUND_HALF_UP);
        return pricePdv;
    }

    public String getCode() {
        return code;
    }

    public void setCode(String code) {
        this.code = code;
    }

    public String getBarCode() {
        return barCode;
    }

    public void setBarCode(String barCode) {
        this.barCode = barCode;
    }

    public String getUktzed() {
        return uktzed;
    }
```

```
public void setUktzed(String uktzed) {
    this.uktzed = uktzed;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}
public String getUnitnm() {
    return unitnm;
}
public void setUnitnm(String unitnm) {
    this.unitnm = unitnm;
}
public Double getPrice() {
    return price;
}
public void setPrice(Double price) {
    this.price = price;
}
public Double getAmount() {
    return amount;
}
public void setAmount(Double amount) {
    this.amount = amount;
}
public Boolean getHasExciseLabel() {
    return hasExciseLabel;
}

public void setHasExciseLabel(Boolean hasExciseLabel) {
    this.hasExciseLabel = hasExciseLabel;
}
public String getExciseLabel() {
    return exciseLabel;
}
public void setExciseLabel(String exciseLabel) {
    this.exciseLabel = exciseLabel;
```

```
}
public Double getPrcPdv() {
    return prcPdv;
}
public void setPrcPdv(Double prcPdv) {
    this.prcPdv = prcPdv;
}
public Character getLetterPdv() {
    return letterPdv;
}
public void setLetterPdv(Character letterPdv) {
    this.letterPdv = letterPdv;
}
public Double getPrcExcise() {
    return prcExcise;
}
public void setPrcExcise(Double prcExcise) {
    this.prcExcise = prcExcise;
}
public Character getLetterExcise() {
    return letterExcise;
}
public void setLetterExcise(Character letterExcise) {
    this.letterExcise = letterExcise;
}
public String getLetters(){
    return new String(new char[] {letterPdv, letterExcise});
}
@Override
public String toString() {
    return "Product{" +
            "code='" + code + '\'' +
            ", barCode='" + barCode + '\'' +
            ", uktzed='" + uktzed + '\'' +
            ", name='" + name + '\'' +
            ", unitnm='" + unitnm + '\'' +
            ", price=" + price +
            ", amount=" + amount +
            ", hasExciseLabel=" + hasExciseLabel +
            ", exciseLabel='" + exciseLabel + '\'' +
            ", prcPdv=" + prcPdv +
            ", letterPdv=" + letterPdv +
```

```
                ", prcExcise=" + prcExcise +
                ", letterExcise=" + letterExcise +
                ", sumPdv=" + getSumPdv() +
                ", sumExcise=" + getSumExcise() +
                ", cost=" + getBDPrice() +
                ", letters='" + getLetters() + '\'' +
                '}';
    }
}
```

A.3 Documents.java

```
package com.zss.prrolib.model;
import com.fasterxml.jackson.annotation.JsonFormat;
import com.fasterxml.jackson.annotation.JsonProperty;
import java.util.Date;
public class Documents {
    /**
     * Фіскальний номер документа
     */
    @JsonProperty("NumFiscal")
    private String numFiscal;
    /**
     * Локальний номер документа
     */
    @JsonProperty("NumLocal")
    private Integer numLocal;
    /**
     * Дата і час операції, зафіксованої документом
     */
    @JsonProperty("DocDateTime")
    @JsonFormat(shape = JsonFormat.Shape.STRING, timezone="Europe/Kiev")
    private Date docDateTime;
    /**
     * Клас документа (“Check”, “ZRep”)
     */
    @JsonProperty("DocClass")
    private  String docClass;
    /**
     * Тип чека (“SaleGoods”, ...)
     */
```

```
@JsonProperty("CheckDocType")
private  String checkDocType;
/**
 * Розширений тип чека (“CheckGoods”, ...)
 */
@JsonProperty("CheckDocSubType")
private  String checkDocSubType;
/**
 * Ознака відкликаного документа
 */
@JsonProperty("Revoked")
private  Boolean Revoked;
/**
 * Ознака сторнованого документа
 */
@JsonProperty("Storned")
private  Boolean storned;
public String getNumFiscal() {
    return numFiscal;
}
public void setNumFiscal(String numFiscal) {
    this.numFiscal = numFiscal;
}
public Integer getNumLocal() {
    return numLocal;
}
public void setNumLocal(Integer numLocal) {
    this.numLocal = numLocal;
}
public Date getDocDateTime() {
    return docDateTime;
}
public void setDocDateTime(Date docDateTime) {
    this.docDateTime = docDateTime;
}
public String getDocClass() {
    return docClass;
}
public void setDocClass(String docClass) {
    this.docClass = docClass;
}
public String getCheckDocType() {
```

```
        return checkDocType;
    }
    public void setCheckDocType(String checkDocType) {
        this.checkDocType = checkDocType;
    }
    public String getCheckDocSubType() {
        return checkDocSubType;
    }
    public void setCheckDocSubType(String checkDocSubType) {
        this.checkDocSubType = checkDocSubType;
    }
    public Boolean getRevoked() {
        return Revoked;
    }
    public void setRevoked(Boolean revoked) {
        Revoked = revoked;
    }
    public Boolean getStorned() {
        return storned;
    }
    public void setStorned(Boolean storned) {
        this.storned = storned;
    }
    @Override
    public String toString() {
        return "Documents{" +
                "numFiscal='" + numFiscal + '\'' +
                ", numLocal=" + numLocal +
                ", docDateTime=" + docDateTime +
                ", docClass='" + docClass + '\'' +
                ", checkDocType='" + checkDocType + '\'' +
                ", checkDocSubType='" + checkDocSubType + '\'' +
                ", Revoked=" + Revoked +
                ", storned=" + storned +
                '}';
    }
}
```

A.4 Operator.java

```
package com.zss.prrolib.model;
import com.fasterxml.jackson.annotation.JsonProperty;
public class Operator {

    @JsonProperty("SubjectKeyId")
    private String subjectKeyId;

    @JsonProperty("RegNum")
    private String regNum;
    /**
     * Старший касир
     */
    @JsonProperty("ChiefCashier")
    private Boolean chiefCashier;
    public String getSubjectKeyId() {
        return subjectKeyId;
    }
    public void setSubjectKeyId(String subjectKeyId) {
        this.subjectKeyId = subjectKeyId;
    }
    public String getRegNum() {
        return regNum;
    }
    public void setRegNum(String regNum) {
        this.regNum = regNum;
    }
    public Boolean getChiefCashier() {
        return chiefCashier;
    }
    public void setChiefCashier(Boolean chiefCashier) {
        this.chiefCashier = chiefCashier;
    }
    @Override
    public String toString() {
        return "Operator{" +
                "subjectKeyId='" + subjectKeyId + '\'' +
                ", regNum='" + regNum + '\'' +
                ", chiefCashier=" + chiefCashier +
                '}';
    }
}
```

A.5 Operators.java

```
package com.zss.prrolib.model;
import com.fasterxml.jackson.annotation.JsonProperty;
import java.util.List;
import java.util.UUID;
public class Operators {
    /**
     * Унікальний ідентифікатор відповіді
     */
    @JsonProperty("UID")
    private UUID uid;
    /**
     * Реєстраційний номер суб'єкта господарювання
(ЄДРПОУ,ДРФО,Картка платника податків)
     */
    @JsonProperty("Tin")
    private String tin;
    /**
     * Перелік операторів
     */
    @JsonProperty("Operators")
    List<Operator> operators;
    public UUID getUid() {
        return uid;
    }
    public String getTin() {
        return tin;
    }
    public List<Operator> getOperators() {
        return operators;
    }
    @Override
    public String toString() {
        return "Operators{" +
                "uid=" + uid +
                ", tin='" + tin + '\'' +
                ", operators=" + operators +
                '}';
    }
}
```

A.6 Shifts.java

```
package com.zss.prrolib.model;
import com.fasterxml.jackson.annotation.JsonFormat;
import com.fasterxml.jackson.annotation.JsonProperty;
import com.zss.prrolib.util.DiffUtils;
import java.util.Date;
public class Shifts {
    /**
     * Ідентифікатор зміни
     */
    @JsonProperty("ShiftId")
    private Long shiftId;
    /**
     * номер документа “Відкриття зміни”
     */
    @JsonProperty("OpenShiftFiscalNum")
    private String openShiftFiscalNum;
    /**
     * Фіскальний номер документа “Закриття зміни”
     */
    @JsonProperty("CloseShiftFiscalNum")
    private  String closeShiftFiscalNum;

    @JsonProperty("Opened")
    @JsonFormat(shape = JsonFormat.Shape.STRING, timezone="Europe/Kiev")
    private Date opened;
    /**
     * Прізвище та ініціали оператора, котрий відкриває зміну
     */
    @JsonProperty("OpenName")
    private  String openName;
    /**
     * Ідентифікатор ключа суб’єкта сертифікату оператора
     */
    @JsonProperty("OpenSubjectKeyId")
    private  String openSubjectKeyId;
    /**
     * Дата і час закриття зміни
     */
    @JsonProperty("Closed")
    @JsonFormat(shape = JsonFormat.Shape.STRING, pattern = "yyyy-MM-
dd'T'HH:mm:ss", timezone = "Europe/Kiev")
    private  Date closed;
    /**
     * П.І.Б. оператора, що закрив зміну
     */
```

```
@JsonProperty("CloseName")
private  String closeName;
/**
 * Ідентифікатор ключа суб'єкта сертифікату оператора
 */
@JsonProperty("CloseSubjectKeyId")
private  String closeSubjectKeyId;
/**
 * Фіскальний номер Z-звіту
 */
@JsonProperty("ZRepFiscalNum")
private  String zRepFiscalNum;

@JsonProperty("Testing")
private Boolean testing;
public Long getShiftId() {
    return shiftId;
}
public void setShiftId(Long shiftId) {
    this.shiftId = shiftId;
}
public String getOpenShiftFiscalNum() {
    return openShiftFiscalNum;
}
public void setOpenShiftFiscalNum(String openShiftFiscalNum) {
    this.openShiftFiscalNum = openShiftFiscalNum;
}
public String getCloseShiftFiscalNum() {
    return closeShiftFiscalNum;
}
public void setCloseShiftFiscalNum(String closeShiftFiscalNum) {
    this.closeShiftFiscalNum = closeShiftFiscalNum;
}
public Date getOpened() {
    return opened;
}
public void setOpened(Date opened) {
    this.opened = opened;
}
public String getOpenName() {
    return openName;
}
public void setOpenName(String openName) {
    this.openName = openName;
}
```

```
public String getOpenSubjectKeyId() {
    return openSubjectKeyId;
}
public void setOpenSubjectKeyId(String openSubjectKeyId) {
    this.openSubjectKeyId = openSubjectKeyId;
}
public Date getClosed() {
    return closed;
}
public void setClosed(Date closed) {
    this.closed = closed;
}

public String getCloseName() {
    return closeName;
}
public void setCloseName(String closeName) {
    this.closeName = closeName;
}
public String getCloseSubjectKeyId() {
    return closeSubjectKeyId;
}
public void setCloseSubjectKeyId(String closeSubjectKeyId) {
    this.closeSubjectKeyId = closeSubjectKeyId;
}
public String getzRepFiscalNum() {
    return zRepFiscalNum;
}
public void setzRepFiscalNum(String zRepFiscalNum) {
    this.zRepFiscalNum = zRepFiscalNum;
}
public Boolean getTesting() {
    return testing;
}
public void setTesting(Boolean testing) {
    this.testing = testing;
}
@Override
public String toString() {
    return "Shifts{" +
            "shiftId=" + shiftId +
            ", openShiftFiscalNum='" + openShiftFiscalNum + '\'' +
            ", closeShiftFiscalNum='" + closeShiftFiscalNum + '\'' +
            ", opened=" + opened +
            ", openName='" + openName + '\'' +
```

```
                ", openSubjectKeyId='" + openSubjectKeyId + '\'' +
                ", closed=" + closed +
                ", closeName='" + closeName + '\'' +
                ", closeSubjectKeyId='" + closeSubjectKeyId + '\'' +
                ", zRepFiscalNum='" + zRepFiscalNum + '\'' +
                ", testing=" + testing +
                '}';
    }
}
```

A.7 ShiftTotals.java

```
package com.zss.prrolib.model;
import com.fasterxml.jackson.annotation.JsonProperty;
import java.math.BigDecimal;
public class ShiftTotals {
    /**
     * Підсумки реалізації
     */
    @JsonProperty("Real")
    private ShiftTotalsOrderType real;
    /**
     * Підсумки повернення
     */
    @JsonProperty("Ret")
    private ShiftTotalsOrderType ret;
    /**
     * Підсумки видачі готівки
     */
    @JsonProperty("Cash")
    private  ShiftTotalsCash  cash;
    /**
     * Підсумки операцій з іноземною валютою
     */
    @JsonProperty("Currency")
    private  ShiftTotalsCurrency currency;
    /**
     * Службове внесення/Отримання авансу/Отримання підкріплення
     */
    @JsonProperty("ServiceInput")
    private BigDecimal serviceInput;
    /**
     * Службова видача/Інкасація
     */
```

```
@JsonProperty("ServiceOutput")
private  BigDecimal ServiceOutput;
public ShiftTotalsOrderType getReal() {
    return real;
}
public void setReal(ShiftTotalsOrderType real) {
    this.real = real;
}
public ShiftTotalsOrderType getRet() {
    return ret;
}
public void setRet(ShiftTotalsOrderType ret) {
    this.ret = ret;
}
public ShiftTotalsCash getCash() {
    return cash;
}
public void setCash(ShiftTotalsCash cash) {
    this.cash = cash;
}
public ShiftTotalsCurrency getCurrency() {
    return currency;
}
public void setCurrency(ShiftTotalsCurrency currency) {
    this.currency = currency;
}
public BigDecimal getServiceInput() {
    return serviceInput;
}
public void setServiceInput(BigDecimal serviceInput) {
    this.serviceInput = serviceInput;
}
public BigDecimal getServiceOutput() {
    return ServiceOutput;
}
public void setServiceOutput(BigDecimal serviceOutput) {
    ServiceOutput = serviceOutput;
}
@Override
public String toString() {
    return "ShiftTotals{" +
            "real=" + real +
            ", ret=" + ret +
            ", cash=" + cash +
            ", currency=" + currency +
```

```
                ", serviceInput=" + serviceInput +
                ", ServiceOutput=" + ServiceOutput +
                '}';
    }
}
```

A.8 ShiftTotalsCash.java

```
package com.zss.prrolib.model;
import com.fasterxml.jackson.annotation.JsonProperty;
import java.math.BigDecimal;
public class ShiftTotalsCash {
    /**
     * Загальна сума
     */
    @JsonProperty("Sum")
    public BigDecimal Sum;
    /**
     * Загальна сума комісії
     */
    @JsonProperty("Commission")
    public BigDecimal Commission;
    /**
     * Кількість чеків
     */
    @JsonProperty("OrdersCount")
    public Integer OrdersCount;
    public BigDecimal getSum() {
        return Sum;
    }
    public void setSum(BigDecimal sum) {
        Sum = sum;
    }
    public BigDecimal getCommission() {
        return Commission;
    }
    public void setCommission(BigDecimal commission) {
        Commission = commission;
    }
    public Integer getOrdersCount() {
        return OrdersCount;
    }
    public void setOrdersCount(Integer ordersCount) {
```

```
        OrdersCount = ordersCount;
    }
    @Override
    public String toString() {
        return "ShiftTotalsCash{" +
                "Sum=" + Sum +
                ", Commission=" + Commission +
                ", OrdersCount=" + OrdersCount +
                '}';
    }
}
```

A.9 ShiftTotalsCurrency.java

```
package com.zss.prrolib.model;
import com.fasterxml.jackson.annotation.JsonProperty;
import java.math.BigDecimal;
import java.util.List;
public class ShiftTotalsCurrency {
    /**
     * Отримано авансів національною валютою
     */
    @JsonProperty("TotalInAdvance")
    public BigDecimal totalInAdvance;
    /**
     * Отримано підкріплень національною валютою
     */
    @JsonProperty("TotalInAttach")
    public BigDecimal totalInAttach;
    /**
     * Здано по інкасації національною валютою
     */
    @JsonProperty("TotalSurrCollection")
    public BigDecimal totalSurrCollection;
    /**
     * Отримано комісії конвертації
     */
    @JsonProperty("Commission")
    public BigDecimal commission;
    /**
     * Кількість розрахункових документів за зміну
     */
    @JsonProperty("CalcDocsCnt")
    public Integer calcDocsCnt;
```

```
/**
 * Прийнято національної валюти для переказу
 */
@JsonProperty("AcceptedN")
public BigDecimal acceptedN;
/**
 * Видано національної валюти при виплаті переказу
 */
@JsonProperty("IssuedN")
public BigDecimal issuedN;
/**
 * Отримано комісії в національній валюті при здійсненні переказів
 */
@JsonProperty("CommissionN")
public BigDecimal commissionN;
/**
 * Кількість операцій (документів) переказів або виплат переказів
 */
@JsonProperty("TransfersCnt")
public Integer transfersCnt;
/**
 * Підсумки по видам іноземної валюти
 */
@JsonProperty("TotalsCurrencyDetails")
public List<TotalsCurrencyDetails> Details;

public BigDecimal getTotalInAdvance() {
    return totalInAdvance;
}
public void setTotalInAdvance(BigDecimal totalInAdvance) {
    this.totalInAdvance = totalInAdvance;
}
public BigDecimal getTotalInAttach() {
    return totalInAttach;
}
public void setTotalInAttach(BigDecimal totalInAttach) {
    this.totalInAttach = totalInAttach;
}
public BigDecimal getTotalSurrCollection() {
    return totalSurrCollection;
}
public void setTotalSurrCollection(BigDecimal totalSurrCollection) {
    this.totalSurrCollection = totalSurrCollection;
}
public BigDecimal getCommission() {
```

```
    return commission;
}
public void setCommission(BigDecimal commission) {
    this.commission = commission;
}
public Integer getCalcDocsCnt() {
    return calcDocsCnt;
}
public void setCalcDocsCnt(Integer calcDocsCnt) {
    this.calcDocsCnt = calcDocsCnt;
}
public BigDecimal getAcceptedN() {
    return acceptedN;
}
public void setAcceptedN(BigDecimal acceptedN) {
    this.acceptedN = acceptedN;
}
public BigDecimal getIssuedN() {
    return issuedN;
}
public void setIssuedN(BigDecimal issuedN) {
    this.issuedN = issuedN;
}
public BigDecimal getCommissionN() {
    return commissionN;
}
public void setCommissionN(BigDecimal commissionN) {
    this.commissionN = commissionN;
}
public Integer getTransfersCnt() {
    return transfersCnt;
}
public void setTransfersCnt(Integer transfersCnt) {
    this.transfersCnt = transfersCnt;
}
public List<TotalsCurrencyDetails> getDetails() {
    return Details;
}
public void setDetails(List<TotalsCurrencyDetails> details) {
    Details = details;
}
@Override
public String toString() {
    return "ShiftTotalsCurrency{" +
            "totalInAdvance=" + totalInAdvance +
```

```
                ", totalInAttach=" + totalInAttach +
                ", totalSurrCollection=" + totalSurrCollection +
                ", commission=" + commission +
                ", calcDocsCnt=" + calcDocsCnt +
                ", acceptedN=" + acceptedN +
                ", issuedN=" + issuedN +
                ", commissionN=" + commissionN +
                ", transfersCnt=" + transfersCnt +
                ", Details=" + Details +
                '}';
    }
}
```

A.10 ShiftTotalsOrderType.java

```
package com.zss.prrolib.model;
import com.fasterxml.jackson.annotation.JsonProperty;
import java.math.BigDecimal;
import java.util.List;
public class ShiftTotalsOrderType {
    /**
     * Загальна сума
     */
    @JsonProperty("Sum")
    private BigDecimal sum;
    /**
     * Загальна сума коштів, виданих клієнту ломбарда
     */
    @JsonProperty("PwnSumIssued")
    private BigDecimal pwnSumIssued;
    /**
     * Загальна сума коштів, одержаних від клієнта ломбарда
     */
    @JsonProperty("PwnSumReceived")
    private BigDecimal pwnSumReceived;
    /**
     * Заокруглення (наприклад, 0.71)
     */
    @JsonProperty("RndSum")
    private BigDecimal rndSum;
    /**
     * Загальна сума без заокруглення (наприклад, 1000.71)
     */
    @JsonProperty("NoRndSum")
```

```
private BigDecimal noRndSum;
/**
 * Загальна сума переказів коштів
 */
@JsonProperty("TotalCurrencySum")
private BigDecimal totalCurrencySum;
/**
 * Загальна сума комісії від переказів коштів
 */
@JsonProperty("TotalCurrencyCommission")
private BigDecimal totalCurrencyCommission;
/**
 * Кількість чеків
 */
@JsonProperty("OrdersCount")
private Integer ordersCount;
/**
 * Кількість операції переказу коштів
 */
@JsonProperty("TotalCurrencyCost")
private Integer totalCurrencyCost;
/**
 * Підсумки по формам оплати
 */
@JsonProperty("PayForm")
public List<ShiftTotalsPayForm> payForm;
/**
 * Податки/збори
 */
@JsonProperty("Tax")
public List<ShiftTotalsTax> tax;
public BigDecimal getSum() {
    return sum;
}
public void setSum(BigDecimal sum) {
    this.sum = sum;
}
public BigDecimal getPwnSumIssued() {
    return pwnSumIssued;
}
public void setPwnSumIssued(BigDecimal pwnSumIssued) {
    this.pwnSumIssued = pwnSumIssued;
}
public BigDecimal getPwnSumReceived() {
    return pwnSumReceived;
```

```
    }
    public void setPwnSumReceived(BigDecimal pwnSumReceived) {
        this.pwnSumReceived = pwnSumReceived;
    }
    public BigDecimal getRndSum() {
        return rndSum;
    }
    public void setRndSum(BigDecimal rndSum) {
        this.rndSum = rndSum;
    }
    public BigDecimal getNoRndSum() {
        return noRndSum;
    }
    public void setNoRndSum(BigDecimal noRndSum) {
        this.noRndSum = noRndSum;
    }
    public BigDecimal getTotalCurrencySum() {
        return totalCurrencySum;
    }
    public void setTotalCurrencySum(BigDecimal totalCurrencySum) {
        this.totalCurrencySum = totalCurrencySum;
    }
    public BigDecimal getTotalCurrencyCommission() {
        return totalCurrencyCommission;
    }
    public void setTotalCurrencyCommission(BigDecimal
totalCurrencyCommission) {
        this.totalCurrencyCommission = totalCurrencyCommission;
    }
    public Integer getOrdersCount() {
        return ordersCount;
    }
    public void setOrdersCount(Integer ordersCount) {
        this.ordersCount = ordersCount;
    }
    public Integer getTotalCurrencyCost() {
        return totalCurrencyCost;
    }
    public void setTotalCurrencyCost(Integer totalCurrencyCost) {
        this.totalCurrencyCost = totalCurrencyCost;
    }
    public List<ShiftTotalsPayForm> getPayForm() {
        return payForm;
    }
    public void setPayForm(List<ShiftTotalsPayForm> payForm) {
```

```
        this.payForm = payForm;
    }
    public List<ShiftTotalsTax> getTax() {
        return tax;
    }
    public void setTax(List<ShiftTotalsTax> tax) {
        this.tax = tax;
    }
    @Override
    public String toString() {
        return "ShiftTotalsOrderType{" +
                "sum=" + sum +
                ", pwnSumIssued=" + pwnSumIssued +
                ", pwnSumReceived=" + pwnSumReceived +
                ", rndSum=" + rndSum +
                ", noRndSum=" + noRndSum +
                ", totalCurrencySum=" + totalCurrencySum +
                ", totalCurrencyCommission=" + totalCurrencyCommission +
                ", ordersCount=" + ordersCount +
                ", totalCurrencyCost=" + totalCurrencyCost +
                ", payForm=" + payForm +
                ", tax=" + tax +
                '}';
    }
}
```

A.11 ShiftTotalPayForm.java

```
package com.zss.prrolib.model;
import com.fasterxml.jackson.annotation.JsonProperty;
import java.math.BigDecimal;
public class ShiftTotalsPayForm {
    /**
     * Код форми оплати
     */
    @JsonProperty("PayFormCode")
    private Integer payFormCode;
    /**
     * Найменування форми оплати
     */
    @JsonProperty("PayFormName")
    private String payFormName;
    /**
     * Сума оплати
```

```
     */
    @JsonProperty("Sum")
    private BigDecimal sum;
    public Integer getPayFormCode() {
        return payFormCode;
    }
    public void setPayFormCode(Integer payFormCode) {
        this.payFormCode = payFormCode;
    }
    public String getPayFormName() {
        return payFormName;
    }
    public void setPayFormName(String payFormName) {
        this.payFormName = payFormName;
    }
    public BigDecimal getSum() {
        return sum;
    }
    public void setSum(BigDecimal sum) {
        this.sum = sum;
    }
    @Override
    public String toString() {
        return "ShiftTotalsPayForm{" +
                "payFormCode=" + payFormCode +
                ", payFormName='" + payFormName + '\'' +
                ", sum=" + sum +
                '}';
    }
}
```

A.12 TaxObjects.java

```
package com.zss.prrolib.model;
import com.fasterxml.jackson.annotation.JsonProperty;
import java.util.List;
public class TaxObjects {
    /**
     *Ідентифікатор запису ГО
     */
    @JsonProperty("Entity")
    private Long entity;// Ідентифікатор запису ГО
    /**
     *Ознака ФОП – платника єдиного податку
```

```
     */
    @JsonProperty("SingleTax")
    private Boolean singleTax; // Ознака ФОП – платника єдиного податку

    /**
     *Найменування ГО
     */
    @JsonProperty("Name")
    private String name; // Найменування ГО
    /**
     *Адреса ГО
     */
    @JsonProperty("Address")
    private String address; // Адреса ГО
    /**
     *Код ЄДРПОУ/ДРФО платника податків
     */
    @JsonProperty("Tin")
    private String tin; // Код ЄДРПОУ/ДРФО платника податків
    /**
     *Податковий номер платника ПДВ
     */
    @JsonProperty("Ipn")
    private String ipn; // Податковий номер платника ПДВ
    /**
     *Найменування суб'єкта господарювання
     */
    @JsonProperty("OrgName")
    private String orgName; // Найменування суб'єкта господарювання
    /**
     *Перелік ПРРО, зареєстрованих для ГО
     */
    @JsonProperty("TransactionsRegistrars")
    private List<TransactionsRegistrars> transactionsRegistrars; // Перелік ПРРО,
зареєстрованих для ГО
    @JsonProperty("ChiefCashier")
    private String chiefCashier;
    @JsonProperty("TaxObjGuid")
    private String taxObjGuid;
    @JsonProperty("TaxObjId")
    private String taxObjId;

    public Long getEntity() {
        return entity;
    }
```

```
    public void setEntity(Long entity) {
        this.entity = entity;
    }

    public Boolean getSingleTax() {
        return singleTax;
    }
    public void setSingleTax(Boolean singleTax) {
        this.singleTax = singleTax;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getAddress() {
        return address;
    }
    public void setAddress(String address) {
        this.address = address;
    }
    public String getTin() {
        return tin;
    }
    public void setTin(String tin) {
        this.tin = tin;
    }
    public String getIpn() {
        return ipn;
    }
    public void setIpn(String ipn) {
        this.ipn = ipn;
    }
    public String getOrgName() {
        return orgName;
    }
    public void setOrgName(String orgName) {
        this.orgName = orgName;
    }
    public List<TransactionsRegistrars> getTransactionsRegistrars() {
        return transactionsRegistrars;
    }
    public void setTransactionsRegistrars(List<TransactionsRegistrars>
transactionsRegistrars) {
```

```
        this.transactionsRegistrars = transactionsRegistrars;
    }

    public String getChiefCashier() {
        return chiefCashier;
    }
    public void setChiefCashier(String chiefCashier) {
        this.chiefCashier = chiefCashier;
    }
    public String getTaxObjGuid() {
        return taxObjGuid;
    }
    public void setTaxObjGuid(String taxObjGuid) {
        this.taxObjGuid = taxObjGuid;
    }
    public String getTaxObjId() {
        return taxObjId;
    }
    public void setTaxObjId(String taxObjId) {
        this.taxObjId = taxObjId;
    }
    @Override
    public String toString() {
        return "TaxObjects{" +
                "entity=" + entity +
                ", singleTax=" + singleTax +
                ", name='" + name + '\'' +
                ", address='" + address + '\'' +
                ", tin='" + tin + '\'' +
                ", ipn='" + ipn + '\'' +
                ", orgName='" + orgName + '\'' +
                ", transactionsRegistrars=" + transactionsRegistrars +
                ", chiefCashier='" + chiefCashier + '\'' +
                ", taxObjGuid='" + taxObjGuid + '\'' +
                ", taxObjId='" + taxObjId + '\'' +
                '}';
    }
}
```

A.13 TotalsCurrencyDetails.java

```
package com.zss.prrolib.model;
import com.fasterxml.jackson.annotation.JsonProperty;
import java.math.BigDecimal;
```

```
public class TotalsCurrencyDetails {
    /**
     * Код валюти
     */
    @JsonProperty("ValCd")
    private Integer valCd;

    /**
     * Символьний код валюти
     */
    @JsonProperty("ValSymCd")
    private String valSymCd;

    /**
     * Загальна сума придбаної іноземної валюти
     */
    @JsonProperty("BuyValI")
    private BigDecimal buyValI;

    /**
     * Загальна сума проданої іноземної валюти
     */
    @JsonProperty("SellValI")
    private BigDecimal sellValI;

    /**
     * Загальна сума придбаної національної валюти
     */
    @JsonProperty("BuyValN")
    private BigDecimal buyValN;

    /**
     * Загальна сума проданої національної валюти
     */
    @JsonProperty("SellValN")
    private BigDecimal sellValN;

    /**
     * Загальна сума поверненої клієнтами іноземної валюти за операціями
«сторно»
     */
    @JsonProperty("StorBuyValI")
    private BigDecimal storBuyValI;

    /**
```

```
    * Загальна сума виданої клієнтам національної валюти за операціями
«сторно»
     */
    @JsonProperty("StorSellValI")
    private BigDecimal storSellValI;
    /**
     * Загальна сума поверненої клієнтами національної валюти за операціями
«сторно»
     */
    @JsonProperty("StorBuyValN")
    private BigDecimal storBuyValN;
    /**
     * Загальна сума виданої клієнтам національної валюти за операціями
«сторно»
     */
    @JsonProperty("StorSellValN")
    private BigDecimal storSellValN;
    /**
     * Загальна сума прийнятої іноземної валюти за операціями конвертації
     */
    @JsonProperty("CInValI")
    private BigDecimal cInValI;
    /**
     * Загальна сума виданої іноземної валюти за операціями конвертації
     */
    @JsonProperty("COutValI")
    private BigDecimal cOutValI;
    /**
     * Загальна сума комісії за операціями конвертації
     */
    @JsonProperty("Commission")
    private BigDecimal Commission;
    /**
     * Отримано авансів
     */
    @JsonProperty("InAdvance")
    private BigDecimal InAdvance;
    /**
     * Отримано підкріплень
     */
    @JsonProperty("InAttach")
    private BigDecimal InAttach;
    /**
     * Здано по інкасації
     */
```

```
@JsonProperty("SurrCollection")
private BigDecimal SurrCollection;
/**
 * Видано іноземної валюти по сторно конвертації
 */
@JsonProperty("StorCInValI")
private BigDecimal StorCInValI;
/**
 * Повернуто іноземної валюти по сторно конвертації
 */
@JsonProperty("StorCOutValI")
private BigDecimal StorCOutValI;
/**
 * Повернуто суму комісії з сторно конвертації
 */
@JsonProperty("StorCommission")
private BigDecimal StorCommission;
public Integer getValCd() {
    return valCd;
}
public void setValCd(Integer valCd) {
    this.valCd = valCd;
}
public String getValSymCd() {
    return valSymCd;
}
public void setValSymCd(String valSymCd) {
    this.valSymCd = valSymCd;
}
public BigDecimal getBuyValI() {
    return buyValI;
}
public void setBuyValI(BigDecimal buyValI) {
    this.buyValI = buyValI;
}
public BigDecimal getSellValI() {
    return sellValI;
}
public void setSellValI(BigDecimal sellValI) {
    this.sellValI = sellValI;
}
public BigDecimal getBuyValN() {
    return buyValN;
}
```

```
public void setBuyValN(BigDecimal buyValN) {
    this.buyValN = buyValN;
}
public BigDecimal getSellValN() {
    return sellValN;
}
public void setSellValN(BigDecimal sellValN) {
    this.sellValN = sellValN;
}
public BigDecimal getStorBuyValI() {
    return storBuyValI;
}
public void setStorBuyValI(BigDecimal storBuyValI) {
    this.storBuyValI = storBuyValI;
}
public BigDecimal getStorSellValI() {
    return storSellValI;
}
public void setStorSellValI(BigDecimal storSellValI) {
    this.storSellValI = storSellValI;
}
public BigDecimal getStorBuyValN() {
    return storBuyValN;
}
public void setStorBuyValN(BigDecimal storBuyValN) {
    this.storBuyValN = storBuyValN;
}
public BigDecimal getStorSellValN() {
    return storSellValN;
}
public void setStorSellValN(BigDecimal storSellValN) {
    this.storSellValN = storSellValN;
}
public BigDecimal getcInValI() {
    return cInValI;
}
public void setcInValI(BigDecimal cInValI) {
    this.cInValI = cInValI;
}
public BigDecimal getcOutValI() {
    return cOutValI;
}
public void setcOutValI(BigDecimal cOutValI) {
    this.cOutValI = cOutValI;
}
```

```
public BigDecimal getCommission() {
    return Commission;
}
public void setCommission(BigDecimal commission) {
    Commission = commission;
}
public BigDecimal getInAdvance() {
    return InAdvance;
}
public void setInAdvance(BigDecimal inAdvance) {
    InAdvance = inAdvance;
}
public BigDecimal getInAttach() {
    return InAttach;
}
public void setInAttach(BigDecimal inAttach) {
    InAttach = inAttach;
}
public BigDecimal getSurrCollection() {
    return SurrCollection;
}
public void setSurrCollection(BigDecimal surrCollection) {
    SurrCollection = surrCollection;
}
public BigDecimal getStorCInValI() {
    return StorCInValI;
}
public void setStorCInValI(BigDecimal storCInValI) {
    StorCInValI = storCInValI;
}
public BigDecimal getStorCOutValI() {
    return StorCOutValI;
}
public void setStorCOutValI(BigDecimal storCOutValI) {
    StorCOutValI = storCOutValI;
}
public BigDecimal getStorCommission() {
    return StorCommission;
}
public void setStorCommission(BigDecimal storCommission) {
    StorCommission = storCommission;
}
@Override
public String toString() {
    return "TotalsCurrencyDetails{" +
```

```
                "valCd=" + valCd +
                ", valSymCd='" + valSymCd + '\'' +
                ", buyValI=" + buyValI +
                ", sellValI=" + sellValI +
                ", buyValN=" + buyValN +
                ", sellValN=" + sellValN +
                ", storBuyValI=" + storBuyValI +
                ", storSellValI=" + storSellValI +
                ", storBuyValN=" + storBuyValN +
                ", storSellValN=" + storSellValN +
                ", cInValI=" + cInValI +
                ", cOutValI=" + cOutValI +
                ", Commission=" + Commission +
                ", InAdvance=" + InAdvance +
                ", InAttach=" + InAttach +
                ", SurrCollection=" + SurrCollection +
                ", StorCInValI=" + StorCInValI +
                ", StorCOutValI=" + StorCOutValI +
                ", StorCommission=" + StorCommission +
                '}';
    }
}
```

A.14 TransactionRegistrars.java

```
package com.zss.prrolib.model;
import com.fasterxml.jackson.annotation.JsonProperty;
public class TransactionsRegistrars {
    /**
     * Фіскальний номер ПРРО
     */
    @JsonProperty("NumFiscal")
    private Long numFiscal;
    /**
     * Локальний номер ПРРО
     */
    @JsonProperty("NumLocal")
    private Long numLocal;
    /**
     * Найменування ПРРО
     */
    @JsonProperty("Name")
    private String name;
    /**
```

```
     * Ознака скасованої реєстрації
     */
    @JsonProperty("Closed")
    private Boolean closed;
    TaxObjects taxObjects;
    public TaxObjects getTaxObjects() {
        return taxObjects;
    }
    public void setTaxObjects(TaxObjects taxObjects) {
        this.taxObjects = taxObjects;
    }
    public Long getNumFiscal() {
        return numFiscal;
    }
    public void setNumFiscal(Long numFiscal) {
        this.numFiscal = numFiscal;
    }
    public Long getNumLocal() {
        return numLocal;
    }
    public void setNumLocal(Long numLocal) {
        this.numLocal = numLocal;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public Boolean getClosed() {
        return closed;
    }
    public void setClosed(Boolean closed) {
        this.closed = closed;
    }
    @Override
    public String toString() {
        return "TransactionsRegistrars{" +
                "numFiscal=" + numFiscal +
                ", numLocal=" + numLocal +
                ", name='" + name + '\'' +
                ", closed=" + closed +
                '}';
    }
}
```

A.15 RequestApiUseContextTest.java

```
package com.zss.prrolib.http;
import com.zss.prrolib.crypto.Signer;
import com.zss.prrolib.crypto.SignerUseContext;
import com.zss.prrolib.model.Operator;
import com.zss.prrolib.model.Operators;
import com.zss.prrolib.model.request.RequestCheck;
import com.zss.prrolib.model.response.*;
import com.zss.prrolib.model.xml.enums.DocumentRequestType;
import com.zss.prrolib.util.DiffUtils;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.Test;
import java.text.SimpleDateFormat;
import java.time.ZoneId;
import java.time.ZonedDateTime;
import java.util.*;
import static org.junit.jupiter.api.Assertions.*;
class RequestApiUseContextTest {
    private static String fileKey = "key004/Key-6.pfx";
    private static String password = "zs753014";
    private static byte[] byteKey;
    private static Long numFiscal = 4000149016l;
    @BeforeAll
    static void initLib() {
//      загрузка библиотеки ЭЦП
        try {
            byteKey = DiffUtils.readBytesFromFileNio(fileKey);
            SignerUseContext.setPathOsplm(System.getProperty("user.dir"));
            SignerUseContext.initLib();
        } catch (Exception e) {
            System.out.println("Ошибка загрузки библиотеки ЭЦП");
            e.printStackTrace();
        }
    }
    @Test
    void jsonServerState() {
        Object o = null;
        try {
            String json = RequestApiUseContext.jsonServerState();
            o = RequestApiUseContext.signAndRequestJson(1, byteKey, password,
false, json, ResponseServerState.class);
        } catch (Exception e) {
```

```
            e.printStackTrace();
        }
    }
    @Test
    void jsonObjects() {
        Object o = null;
        try {
            String json = RequestApiUseContext.jsonObjects();
            System.out.println(json);
            ResponseObjects ro = (ResponseObjects) RequestApiUseContext
                    .signAndRequestJson(1, byteKey, password, false, json,
ResponseObjects.class);
        } catch (Exception e) {
            e.printStackTrace();
        }

    }
    @Test
    void jsonOperators() {
        Object o = null;
        try {
            String json = RequestApiUseContext.jsonOperators();
            o =  RequestApiUseContext.signAndRequestJson(1, byteKey, password,
false,  json, Operators.class);

            Operators o1 = (Operators) o;
            System.out.println(o1.getUid());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    @Test
    void jsonTransactionsRegistrarState() {
        ResponseTransactionsRegistrarState o = null;
        try {
            String json =
RequestApiUseContext.jsonTransactionsRegistrarState(numFiscal, null, null,
true);
            o =  (ResponseTransactionsRegistrarState)RequestApiUseContext
                    .signAndRequestJson(1, byteKey, password, false,  json,
ResponseTransactionsRegistrarState.class);
            System.out.println(o.getResponseErrorMessage());
            System.out.println(o.getShiftState() + " " +
o.getTaxObject().getTransactionsRegistrars().get(0).getNumFiscal());
        } catch (Exception e) {
```

```
            e.printStackTrace();
        }
    }
    @Test
    void jsonCheckExt() {
        ResponseCheckExt o = null;
        try {
            String json = RequestApiUseContext.jsonCheckExt(numFiscal,
"293541931", DocumentRequestType.OriginalXml);
            o = (ResponseCheckExt) RequestApiUseContext.signAndRequestJson(1,
byteKey, password, false, json, ResponseCheckExt.class);
            DiffUtils.writeBytesToFileNio(o.getDecodeData().getBytes(), "doc.xml");
            DiffUtils.writeBytesToFileNio(Base64.getDecoder().decode(o.getData()),
"doc1.xml");
            System.out.println(o.getDecodeData());
        } catch (Exception e){
            e.printStackTrace();
        }
    }
    @Test
    void jsonZRepExt() {
        ResponseZRepExt o = null;
        try{
            String json = RequestApiUseContext.jsonZRepExt(numFiscal,
"259022089", DocumentRequestType.Visualization);
            o = (ResponseZRepExt) RequestApiUseContext.signAndRequestJson(1,
byteKey, password, false, json, ResponseZRepExt.class);
            System.out.println(o.getDecodeData());
        } catch (Exception e){
            e.printStackTrace();
        }
    }

    @Test
    void jsonShifts() {
        SimpleDateFormat sdfUTC = new SimpleDateFormat("yyyy-MM-
dd'T'HH:mm:ssZ");
        sdfUTC.setTimeZone(TimeZone.getTimeZone("UTC"));
        GregorianCalendar calendar = new GregorianCalendar(2021,
Calendar.OCTOBER, 01, 0, 0);
        Date d1 = calendar.getTime();
        calendar = new GregorianCalendar(2022, Calendar.DECEMBER, 31, 23, 0);
        Date d2 = calendar.getTime();
        System.out.println(sdfUTC.format(d1) + " " + sdfUTC.format(d2));
        ZonedDateTime zdt1 = ZonedDateTime.of(2021, 10, 01, 0, 0, 0, 0,
```

```
ZoneId.of("Europe/Kiev"));
        ZonedDateTime zdt2 = ZonedDateTime.of(2022, 12, 31, 23, 0, 0, 0,
ZoneId.of("Europe/Kiev"));
        System.out.println(zdt1 + " " + zdt2);
        assertEquals(Date.from(zdt1.toInstant()), d1);
        assertEquals(Date.from(zdt2.toInstant()), d2);
        Object o = null;
        try{
            String json = RequestApiUseContext.jsonShifts(numFiscal,
Date.from(zdt1.toInstant()), Date.from(zdt2.toInstant()));
            o = RequestApiUseContext.signAndRequestJson(1, byteKey, password,
false, json, ResponseShifts.class);
        } catch (Exception e){
            e.printStackTrace();
        }
    }
    @Test
    void jsonDocuments() {
        Object o = null;
        try {
            String json = RequestApiUseContext.jsonDocuments(numFiscal, 6790313l,
null);
            o = RequestApiUseContext.signAndRequestJson(1, byteKey, password,
false, json, ResponseDocuments.class);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    @Test
    void jsonLastShiftTotals() {
        Object o = null;
        try {
            String json = RequestApiUseContext.jsonLastShiftTotals(numFiscal);
            o = RequestApiUseContext.signAndRequestJson(1, byteKey, password,
false, json, ResponseLastShiftTotals.class);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    @Test
    void jsonDocumentInfoByLocalNum() {
        Object o = null;
        try {
            String json =
RequestApiUseContext.jsonDocumentInfoByLocalNum(numFiscal, 122);
```

```
            o = RequestApiUseContext
                    .signAndRequestJson(1, byteKey, password, false, json,
ResponseDocumentInfoByLocalNum.class);
        } catch (Exception e){
            e.printStackTrace();
        }
    }
}
```

A.16 PdvTest.java

```
        package com.zss.prrolib.math;
import com.zss.prrolib.util.DiffUtils;
import org.junit.jupiter.api.Test;
import java.io.DataInput;
import java.math.BigDecimal;
import java.math.BigInteger;
import java.math.RoundingMode;
public class PdvTest {
    private static Double priceStatic = 99.99d;
    private static  Double prcPDV = 20.00d;
    private static  Double  prcExcise =5.00d;
    @Test
    public void testCalculateSumPDV() {
        BigDecimal price = new BigDecimal(priceStatic);
        price.setScale(2, RoundingMode.HALF_UP);
        price = price.setScale(2, RoundingMode.HALF_UP);
        DiffUtils.p(price);
        BigDecimal prcPDV = new BigDecimal(PdvTest.prcPDV);
        BigDecimal prcExcise = new BigDecimal(PdvTest.prcExcise);
        BigDecimal prcAll = prcPDV.add(prcExcise);
        BigDecimal prc1;
        BigDecimal tmpBigDecimal =  prcAll.divide(new BigDecimal(100), 2,
BigDecimal.ROUND_HALF_UP).add(new BigDecimal(1.00));
        DiffUtils.p(prcPDV, prcExcise, tmpBigDecimal);
        BigDecimal x = price.divide(tmpBigDecimal, 2,
BigDecimal.ROUND_HALF_UP);
        prc1 = x.divide(new BigDecimal(100.00), 7,
BigDecimal.ROUND_HALF_UP);
        BigDecimal sumPdv = prc1.multiply(prcPDV).setScale(2,
BigDecimal.ROUND_HALF_UP);
        BigDecimal sumExcise = prc1.multiply(prcExcise).setScale(2,
```

```
BigDecimal.ROUND_HALF_UP);
        DiffUtils.p(x, prc1, sumPdv, sumExcise);

    }
    @Test
    public void testFloatCalculateSumPDV() {
  Float price = priceStatic.floatValue();
        DiffUtils.p(price);
        Float prcAll = prcPDV.floatValue() + prcExcise.floatValue();
        Float prc1;
        DiffUtils.p(prcPDV, prcExcise);
        Float tmp =  prcAll/100f + 1.0f;
        Float x = price/tmp;
        prc1 = x/100.0f;
        Float sumPdv = prc1 * prcPDV.floatValue();
        Float sumExcise = prc1 * prcExcise.floatValue();
        BigDecimal sumBDPdv = new BigDecimal(sumPdv).setScale(2,
BigDecimal.ROUND_HALF_UP);
        BigDecimal sumBDExcise = new BigDecimal(sumExcise).setScale(2,
BigDecimal.ROUND_HALF_UP);
        DiffUtils.p(x, prc1, sumPdv, sumExcise, sumBDPdv, sumBDExcise);
    }
    @Test
    public void testDoubleCalculateSumPDV() {
        Double price = priceStatic;
        DiffUtils.p(price);
        Double prcAll = prcPDV + prcExcise;
        Double prc1;
        DiffUtils.p(prcPDV, prcExcise);
        Double x = price/(1 + prcAll/100);
        prc1 = x/100;
        Double sumPdv = prc1 * prcPDV;
        Double sumExcise = prc1 * prcExcise;
        BigDecimal sumBDPdv = new BigDecimal(sumPdv).setScale(2,
BigDecimal.ROUND_HALF_UP);
        BigDecimal sumBDExcise = new BigDecimal(sumExcise).setScale(2,
BigDecimal.ROUND_HALF_UP);
        DiffUtils.p(x, prc1, sumPdv, sumExcise, sumBDPdv, sumBDExcise);
    }
}
```

ДОДАТОК Б

Слайди презентації

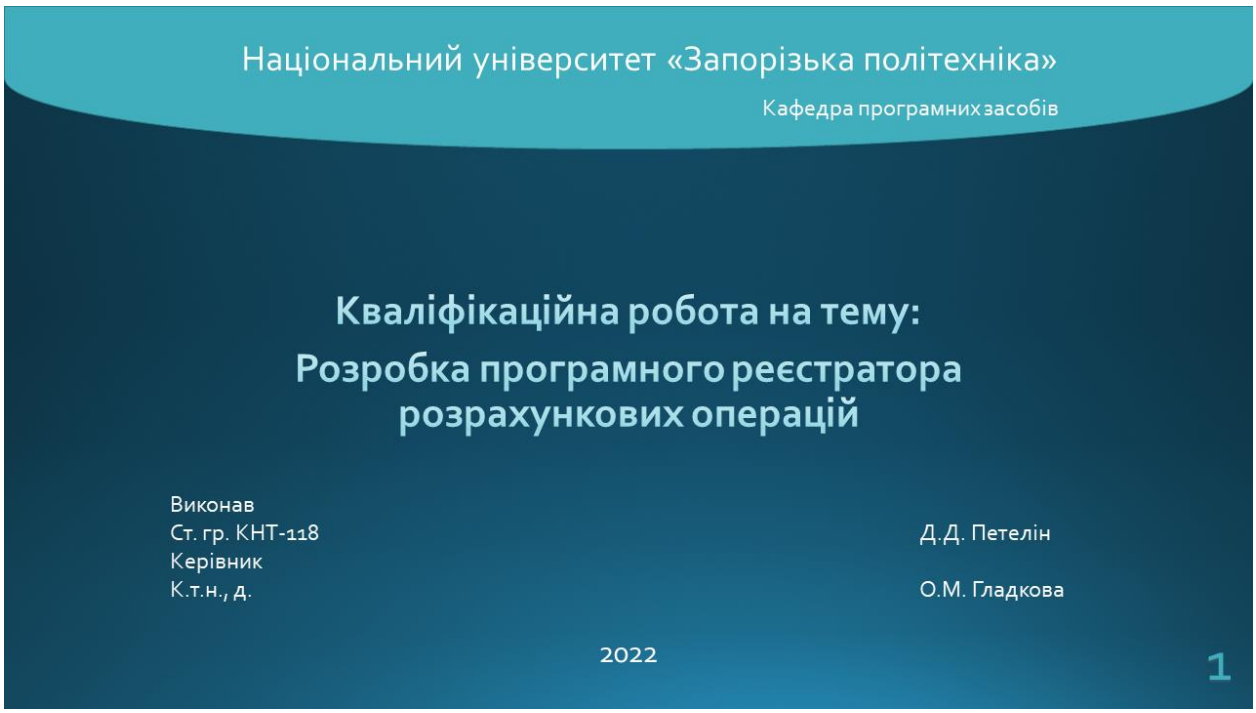


Рисунок Б.1 – Слайд 1

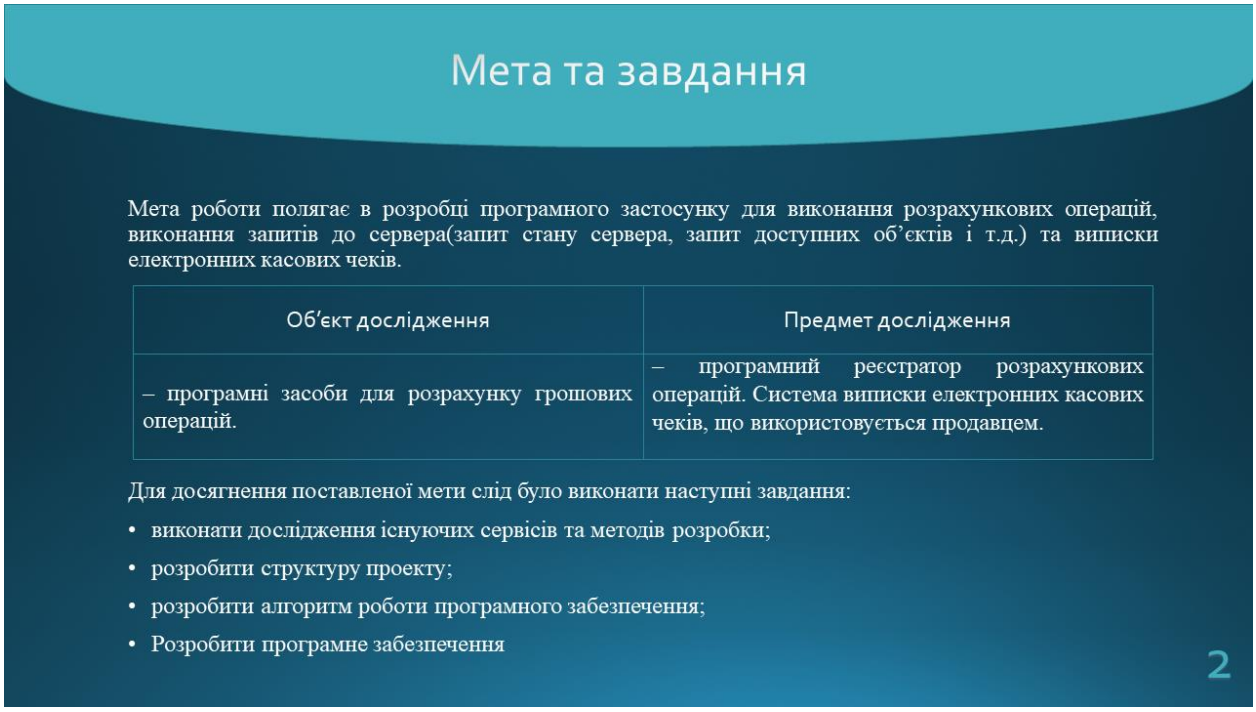


Рисунок Б.2 – Слайд 2

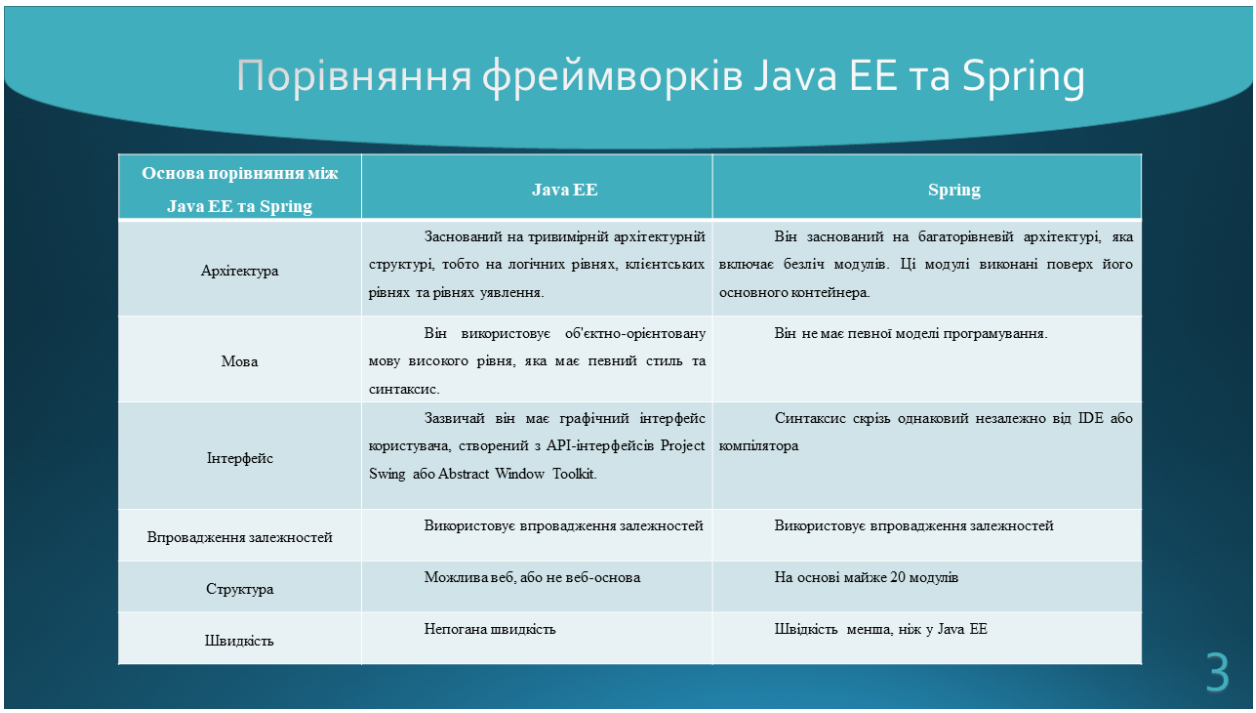


Рисунок Б.3 – Слайд 3

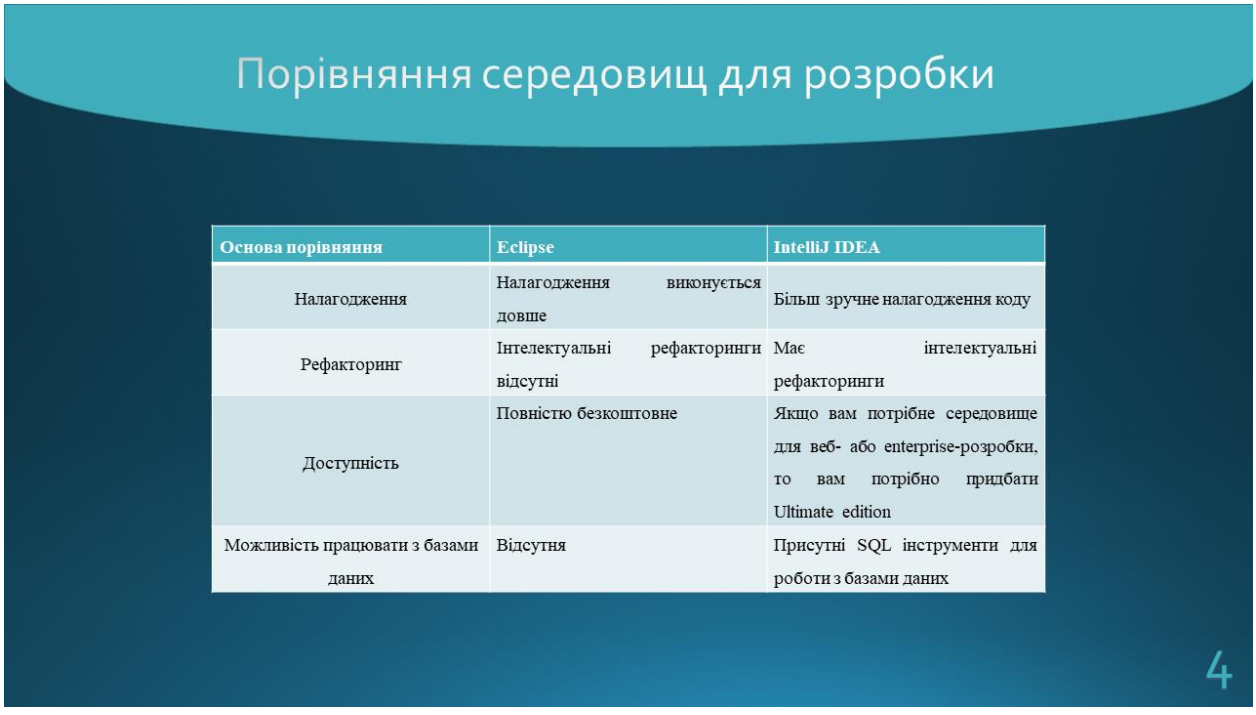


Рисунок Б.4 – Слайд4

Основні функціональні частини програми

Програмний продукт призначений для виконання інформаційних запитів до сервера, а саме:

- запит стану сервера;
- запит доступних об'єктів;
- запит переліку операторів(касирів) для суб'єкта господарювання;
- запит стану ПРРО;
- запит чека розширений;
- запит Z-звіту розширений;
- запит переліку змін за період;
- запит переліку документів зміни;
- запит підсумків останньої зміни;
- запит відомостей про документ за локальним номером.

Також застосунок має функцію для формування чеків, у яких виконується розрахунок податків

5

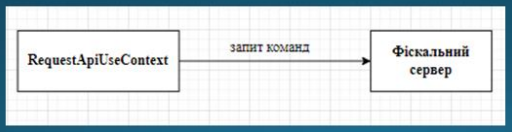
Рисунок Б.5 – Слайд 5

Архітектура системи

Усі команди, що реалізовані у застосунку відправляються на Фіскальний Сервер розрахункових операцій. На рисунку 3.1 представлено взаємодію компонента із фіскальним сервером.

Сервер оброблює Url:

- "<адреса>/doc" – одержання документів (чеків, Z-звітів тощо)
- "<адреса>/pck" – одержання пакетів документів (офлайн документи тощо)
- "<адреса>/cmd" – одержання команд



6

Рисунок Б.6 – Слайд6

Рисунок Б.7 – Слайд 7

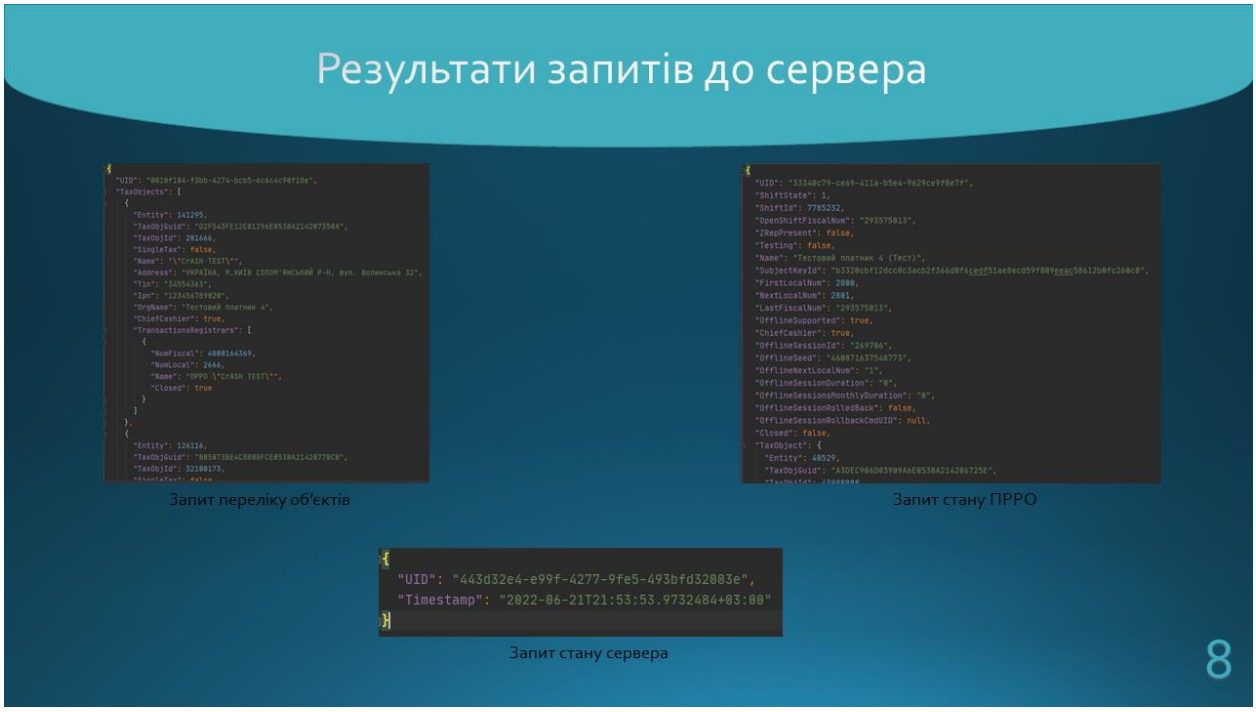


Рисунок Б.8 – Слайд 8

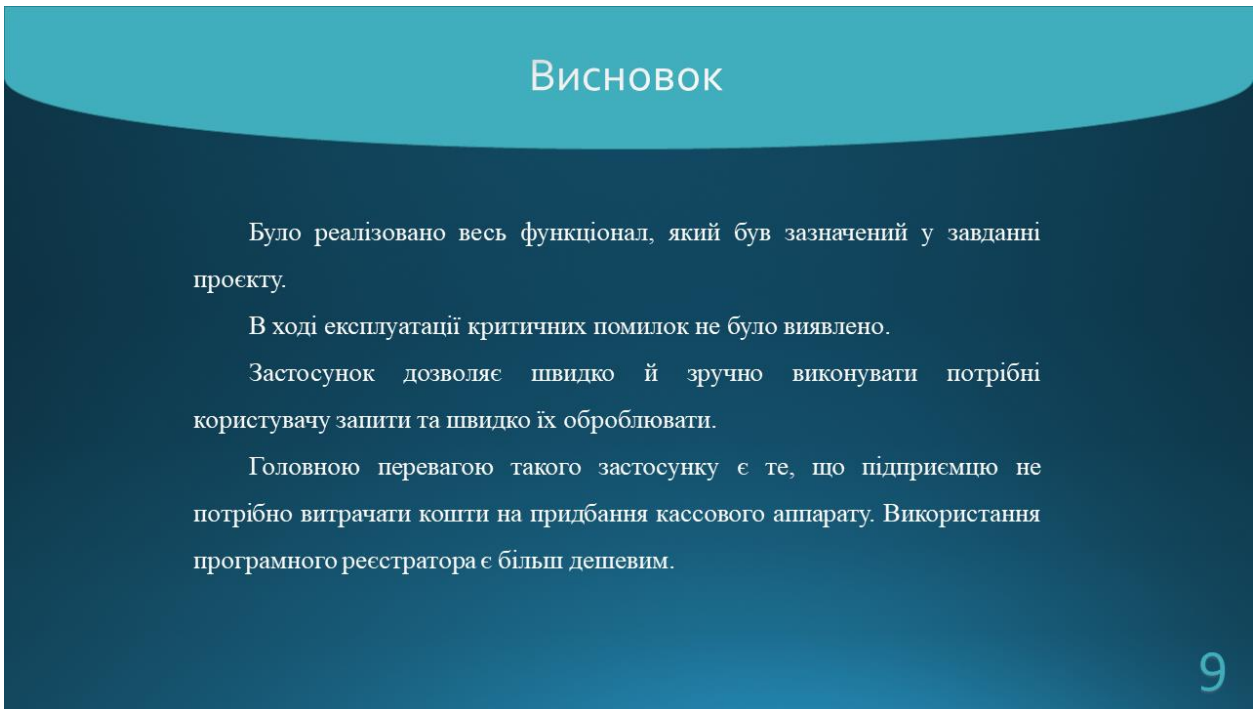


Рисунок Б.9 – Слайд 9