Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Комп'ютерно-інформаційних систем і програмної інженерії
(назва факультету)
Програмної інженерії
(повна назва кафедри)

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломної роботи

магістр
(освітній рівень)

на тему: Розробка клієнт-серверного додатку для операційної системи Android з використанням Java технологій

Виконав: студент (ка) **VI** курсу, групи **СПм-62**
напряму підготовки (спеціальності)
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)

	(підпис)	Мельник І.І. (прізвище та ініціали)
Керівник	(підпис)	Цуприк Г.Б. (прізвище та ініціали)
Нормоконтроль	(підпис)	Бойко І.В. (прізвище та ініціали)
Рецензент	(підпис)	(прізвище та ініціали)

м. Тернопіль – 2019

АНОТАЦІЯ

Мельник І.І. Розробка клієнт-серверного додатку для операційної системи Android з використанням Java технологій. – Рукопис.

Магістерська робота на здобуття освітнього ступеня магістр за спеціальністю 121 – Інженерія програмного забезпечення. – Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра програмної інженерії, група СПм-62 // м.Тернопіль, 2019 // С. , рис. – , табл. – , додат. – , бібліогр. – .

Метою дипломної роботи є розробка продукту який дозволить вдосконалити через автоматизацію роботу суб’єкта пов’язану з актуальною на сьогоднішній день діяльністю, в якій об’єктом може бути не лише послуга, а й достовірна інформація чи будь-які дані, що можуть представляти комерційний інтерес.

Суть дипломної роботи полягає у створенні програмного продукту, який володіє достатнім рівнем гнучкості, об’єктивності та надійності. Обрана концепція взаємодії дає змогу розподілити навантаження між учасниками процесу обміну інформацією (даними).

Практичне застосування – розроблено надійний програмний продукт, що дозволить підвищити ефективність роботи та який володіє зручним інтерфейсом. Враховано необхідність забезпечення надійності з’єднання та безперебійності програми, а також передбачено ряд запитів до неї.

Технічні вимоги – технології Java, Android.

Ключові слова: ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ, ПРОГРАМНИЙ ПРОДУКТ, ЕФЕКТИВНІСТЬ, ОБ’ЄКТИВНІСТЬ, НАДІЙНІСТЬ. МОВА ПРОГРАМУВАННЯ, БАЗА ДАНИХ, ТЕХНІЧНІ ВИМОГИ, ОБЛАСТЬ ЗАСТОСУВАННЯ.

ABSTRACT

Melnyk I.I. Developing of client-server application for Android operating system with using Java technologies. – Manuscript.

The master degree thesis for the qualification level of magistr in the specialty 121 — Software Engineering. – Ternopil Ivan Pul'ui National Technical University, Faculty of Computer Information Systems and Software Engineering, Software Engineering Department, group SPm-62 // Ternopil, 2019 //

Pages. – , pictures. – , tables. – , supp. – , bibl.ref. –

To improve by automation the work of a subject related to the current activity is the aim of the diploma thesis. The object may be not only the service but also reliable information or any data that may be of commercial interest.

Creating a software product that has sufficient flexibility, objectivity and reliability is the essence of the diploma thesis. The most convenient concept of interaction was chosen. This makes it possible to distribute workloads between participants in the information (data) exchange process, as well as program code on the vendor and customer side.

Practical application - Developed reliable software product that will increase the efficiency and has a user-friendly interface. Because the development system will use a database for its proper functioning, the need for reliable connection and continuity of the program is taken into account, and also requests are foreseeable for it.

Technical requirements – Java and Android technologies.

Keywords: INFORMATION TECHNOLOGY, SOFTWARE, EFFICIENCY, OBJECTIVITY, RELIABILITY. PROGRAMMING LANGUAGE, DATABASE, SPECIFICATIONS, AREA OF APPLICATION.

ВСТУП

Сьогодні більшість населення Землі навіть не замислюється над тим, що можна прожити хоча б частину (якщо не все) свого життя без використання сучасних інформаційних технологій. Для прикладу, користувачі, в своїй більшості, навіть не уявляють свого щоденного існування без доступу до всесвітньої мережі і здебільшого не цікавляться принципами її роботи, які засоби та сучасні інструменти при цьому використовуються. Їх основною метою та бажанням є відносно легке отримання коректної до запиту, актуальної та найбільш повної інформації вчасно, безперебійно, в будь-який момент часу та якісно.

Як показує практика, в сенсі взаємодії, одним з найефективніших виявився підхід «клієнт-сервер» – концепція під якою розуміють дві сторони: з одного боку, клієнт – замовлення (для прикладу якоїсь послуги, інформації, тощо) та сервер з іншого боку – в якості постачальника замовленого. Клієнт та сервер – це окремі програми, наприклад, типовим клієнтом може бути браузер. У якості сервера можна навести такі приклади: всі HTTP-сервери (в Apache); MySQL-сервер; локальний веб-сервер AMPPS або готова збірка Denwer. Деякі з наведених – це ціла сукупність серверів.

Також варто зауважити, що в основі взаємодії по типу «клієнт-сервер» лежить принцип того, що роботу завжди розпочинає лише клієнт, а серверна частина лише відгукується на його запит. Крім того сервер інформує про те чи можливо надати послугу клієнтові і якщо так, то на яких умовах. Зазвичай як клієнтське так і серверне програмне забезпечення може встановлюватись як на різних машинах, так і працювати на одному комп'ютері.

Оскільки ефективність такого підходу є беззаперечною, прийняте рішення за допомогою підходу «клієнт-сервер»,

вдосконалити через автоматизацію роботу суб'єкта пов'язану з актуальною на сьогоднішній день діяльністю, в якій об'єктом може бути не лише послуга, а й достовірна інформація чи будь-які дані, що можуть представляти комерційний інтерес. В результаті виконання огляду та порівняльного аналізу подібних систем встановлено, що вони, у своїй сукупності, не володіють достатнім рівнем гнучкості, об'єктивності та надійності.

Дана концепція взаємодії була обрана як найзручніша в першу чергу із-за того що вона дає можливість розподілити навантаження між учасниками процесу обміну інформацією (даними), а також для того, щоб розділити програмний код на стороні постачальника і на стороні замовника. Крім цього, важливим є те, що до одного сервера може звертатися одночасно кілька «клієнтів», адже учасників процесу може бути, і в більшості є, кілька. Проте варто зауважити, що кількість клієнтів, які можуть одночасно напряму взаємодіяти з сервером залежить від його потужності, а також від того, що хоче конкретно отримати клієнт від сервера. В основі мережевого протоколу лежить принцип взаємодії з врахуванням особливостей притаманних специфіці галузі призначення програмного продукту.

Об'єктом дослідження є актуальна на сьогоднішній день діяльність, в якій до уваги може прийматись не лише послуга, а й достовірна інформація чи будь-які дані, що можуть представляти комерційний інтерес.

Оскільки специфіка галузі передбачає роботу з додатками, які працюють на пристроях Android (смартфони, планшети, вбудовані системи, тощо), для реалізації поставленого завдання було обрано мову програмування Java, оскільки вона є однією з офіційних мов програмування під Android, конкуруючи лише з Kotlin.

Враховуючи те, що обраний напрямок дуже різносторонній, а об'єми роботи обмежені положенням про дипломне проєктування та

методичними вказівками, прийняте рішення сформулювати завдання дослідження, яке полягатиме у аналізі предметної області, розробці та проєктуванні системи, моделюванні архітектури та конструюванні системи.

В результаті отримано надійний програмний продукт, що дозволить забезпечити продуктивну роботу та який володіє зручним інтерфейсом. Оскільки розроблювальна система використовуватиме, для свого повноцінного функціонування базу даних, то враховано необхідність забезпечення надійності з'єднання та безперебійності програми, а також передбачено ряд запитів до неї.

Наукова новизна одержаних результатів полягає в отриманні удосконаленої клієнт-серверної програмної системи для організації автоматизації діяльності, гнучкість якої буде більша, ніж в аналогів.

Отримані результати наукового дослідження можуть бути застосовані в сфері мобільної комерційної діяльності, а також використовуватись для подальших наукових досліджень та розробок з впровадженням в галузях чи напрямках, де така концепція взаємодії матиме сенс.

Оформлення магістерської роботи виконується відповідно до діючих стандартів: ДСТУ 2391-94. «Система технологічної документації. Терміни та визначення»; ДСТУ 3008-95. «Документація. Звіти у сфері науки і техніки. Структура і правила оформлення», а також ЕСКД та іншим чинним стандартам.

Роботу апробовано в рамках VII науково-технічнічної конференції «Інформаційні моделі, системи та технології», м. Тернопіль, 2019 р. – Тернопіль: ТНТУ ім. І. Пулюя (м. Тернопіль, 11-12 грудня 2019 року), 2019.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ, ОГЛЯД ТА ПОРІВНЯЛНИЙ АНАЛІЗ СИСТЕМ ІСНУЮЧИХ В ГАЛУЗІ

1.1 Введення

Ефективна організація системи зв’язків, як свідчить світовий та вітчизняний досвід, неможлива без участі посередника, який забезпечує необхідну якість роботи, зокрема оптимізує показники інтенсивності та протікання процесу, вкладання необхідних для його проходження затрат (матеріальних та не матеріальних), тощо. Певні складності викликає й той факт, що специфіка діяльності такого типу передбачає постійні зміни через допрацювання та вдосконалення. Аналіз сучасної ситуації в галузі показує, що в нових умовах активно створюються нові форми діяльності в рамках напрямку, які потрібно адаптувати до вимог ринку, зокрема як до запитів постачальників послуг, так і до задоволення потреб користувачів. Таким чином відповідні структури налагоджують ефективну роботу з користувачами через, зокрема, передачу, зберігання та обмін інформацією (даними), розширюють функціонал послуг. Під впливом жорсткої щоденної конкуренції та необхідності постійного розвитку, для того, щоб залишатися в тренді на ринку з достатньо важкими вимогами та умовами для виживання, необхідно постійно аналізувати потреби, цікавитись перспективами, розробляти та створювати нові ефективні логістичні схеми та розширювати можливості та збільшувати потенціал. А оскільки за останні роки економіка України демонструє хоч не суттєві, проте стабільні показники зростання [1,2], власне у цьому напрямку перспективною буде розробка, з використанням сучасних інформаційних технологій, впровадження якої дасть можливість вдосконалити через автоматизацію роботу суб’єкта пов’язану з актуальною на сьогоднішній день діяльністю, в

якій об’єктом може бути не лише послуга, а й достовірна інформація чи будь-які дані, що можуть представляти комерційний інтерес. В результаті виконання огляду та порівняльного аналізу подібних систем встановлено, що вони, у своїй сукупності, не володіють достатнім рівнем гнучкості, об’єктивності та надійності. Проте при автоматизації діяльності такого типу можна вирішити основні проблеми інформаційної, товарної і фінансової логістики [2-4]. Також така оптимазація за допомогою програмних засобів відіграє свою важливу роль в розвитку даної сфери з врахуванням специфіки діяльності.

Останні кілька років на вітчизняному ринку спостерігаються тенденції, що кардинально змінюють колишні схеми комерційної діяльності. Сьогодні можна впевнено говорити про закінчення ери безпосереднього технічного продажу (виробник — продавець) без застосування інформаційних технологій та відомостей про подальший шлях товару на всіх етапах починаючи від моменту замовлення на виробництві і завершуючи повторним замовленням найбільш затребуваних позицій. Йде мова про усучаснювання підходу, що передбачає перехід до забезпечення контролю (передача, зберігання та отримання інформації – своєрідний підвид логістики) за всіма ланками його переміщення, в процесі руху який представляє комерційний інтерес.

Дослідження даних статистичного дослідження та тенденцій розвитку зокрема комерційної діяльності в Україні на ринках товарів та послуг свідчить, що за останні декілька років кількість підприємств (організацій) малого та середнього бізнесу має тенденцію до зростання, хоча конкурентоспроможність стає більш жорсткою. На це впливає ціла низка факторів, основними з них є ефективність, якість та стабільність, яких сьогодні не можливо досягти без застосування сучасних технологій. Також слід врахувати, що така тенденція

пояснюється також і тим фактором, що забезпечити якісний обіг даних, більш зручно і ефективніше можна саме в підприємствах таких форм.

Комерційна діяльність у сучасних умовах привертає до себе дедалі більшу увагу науковців (Альбеков А. У., Согомонян С. А. , Осіпова Л. В., Сіняєва І. М. , Панкратов Ф.Г., Серьогіна Т.К., Райзберг Б. А., Тарасюк Г. М., Луіс В. Штерн, інші). Проте, в більшості випадків, аналіз діяльності на основі маркетингу проводиться виходячи з позиції виробника, при формуванні ним каналів розподілу та створення вертикальних маркетингових систем. Достатньо ґрунтовно організація товарообігу розкрита в працях з Блонська В.І., Феофанова К.О., Ващекін Н.П., Лупей Н.А., Копійка В.В., Шинкаренко Т.І., Корчак А.С., Салацький О.А., Мазаракі А.А., Лігоненко Л.О., Ушакова Н.М. та ін. Разом з тим, низка аспектів, що стосуються організації та управління мобільною системою торгівлі залишилися поза увагою дослідників [6-10].

Підготовку замовлення покупець може здійснити самостійно або за допомогою торгового представника фірми-постачальника. На торгового представника покладена велика відповідальність з продажу товарів та наданню послуг. Вони забезпечують якісне обслуговування покупців, оскільки добре ознайомленні з номенклатурою та специфікою товарів і послуг своєї фірми. На сьогоднішній день торгові представники більшості оптових компаній підготовленні професійно. Деякі компанії розробляють та впроваджують власні навчальні програми з підвищення кваліфікації працівників. Проте у більшості випадків персонал дистриб'юторів підготовлює представник від виробника. Підготовка здійснюється на основі тренінг-семінарів та має свої переваги, оскільки основна увага зосереджується на способах продажу, просуванні конкретного товару або торговельної марки та особливостях збору замовлення.

Виконання замовлень товароодержувачів передбачає створення на складах фірми-оптовика великої кількості товарних запасів, необхідних для безперебійного задоволення потреб покупців. Залежно від характеру замовлень покупців, на складах оптових компаній більш дрібніші партії товарів, отримані від деяких постачальників, можуть об'єднуватися у великі партії або товари, отримані великими партіями, продаються покупцям у невеликих кількостях. Виконання замовлення здійснюється на основі товарного чеку, який, у свою чергу, формується згідно заявки, яка заповнюється на підприємстві роздрібної торгівлі. Дана заявка через електронну систему мобільної торгівлі попадає в центральний офіс, де працівник відділу збуту вирішує проводити її чи ні. Заборону на виконання замовлення можуть застосувати залежно від стану дебіторської заборгованості, у випадку не оплати більш ніж половини кількості попередніх замовлень або прострочення термінів розрахунку на один — два тижні. При позитивному розгляді заявки формується товарний чек у трьох примірниках, два з яких потрапляють в експедицію, а один на склад, по якому буде формуватися замовлення. В експедиції один примірник залишається у покупця, а інший із підписом про отримання товарів, повертається на підприємство. Дана система формування замовлення передбачає можливість самоконтролю, оскільки дані про замовлення, відвантаження товарів зі складу та отримання продукції покупцем не мають відрізнятися один від одного. Крім того, у більшості оптових підприємств регіону існує контрольно-ревізійна служба, яка робить звірку відвантажених товарів по накладних та перевіряє залишки на складах. Дана перевірка проводиться, як правило, один раз на тиждень [11].

1.2 Розгляд та пошук рішень для автоматизації процесу

На сьогоднішній день, дистриб'ютори при зборі інформації, в більшості випадків, можуть застосовувати принципи (інструменти) системи мобільності. Вона організована і функціонує на базі програмного забезпечення в якому враховано специфіку подібної діяльності та яке призначене для забезпечення можливості формування запиту на інформацію. Така система, може адаптуватись під використання в різних галузях, проте для прикладу та деталізації наведено опис варіанту її функціоналу який переважно використовується в комерційній діяльності. Систему такого типу застосовують з метою автоматизації діяльності комерційних представників з використанням кишенькових комп'ютерів. За допомогою даної системи можна:

- скоротити час на формування замовлення та його подальшу передачу в систему обліку;
- виключити необхідність повторного вводу інформації на наступних етапах;
- ефективно розподіляти план роботи торгового представника;
- підвищити кількість обслуговування торговельних точок;
- підвищити достовірність інформації, що обробляється;
- оперативно отримувати інформацію, щодо умов здійснення торговельної діяльності та їх зміни;
- збільшити об'єм продажу компанії;
- підвищити ефективність роботи торгових представників;
- готувати основні комплекти документації [12].

При організації мобільної комерційної діяльності, торговий представник може здійснювати продаж товару за допомогою електронного пристрою (смартфон, планшет, вбудовані системи,

тощо), маючи при цьому всю інформацію по складським запасам. При організації попереднього збору замовлення можливі декілька варіантів використання устаткування та відповідного програмного забезпечення: накопичувальний спосіб та на основі передачі інформації через модель клієнт-сервер [2].

Суть накопичувального способу використання полягає в тому, що даний варіант використання вимагає завантаження та періодичного оновлення на терміналі збору даних чи на кишеньковому комп'ютері довідника товарів та клієнтів із центрального офісу. Для обміну даними терміналу з обліковою системою, необхідне підключення комунікаційної підставки до персонального комп'ютера, через який буде здійснюватися передача даних. При продажу товарної продукції покупцю за допомогою мобільного принтера роздруковується пакет відповідних документів. Сформований на терміналі збору даних, продаж зберігається для подальшого завантаження в центральну систему обліку. Список усіх операцій залишається в пам'яті пристрою. Також, при необхідності, є можливість відкоригувати інформацію, накопичену протягом дня. На основі використання клієнт-серверної моделі, з'являється можливість управління даними з формування замовлень клієнтів з мобільного пристрою в центральний офіс, а також зникає необхідність встановлення комп'ютера з обліковою системою у віддалених від офісу магазинах. При продажу продукції покупцю за допомогою мобільного принтера роздруковується пакет відповідних документів, а дані про продаж, використовуючи з'єднання, передаються в центральну систему обліку [2].

Використання мобільності на основі моделі клієнт-сервер дозволяє значно оптимізувати процес виконання замовлень товароодержувачів. Торговий представник може оперативно передавати інформацію в центральний офіс, де без його участі

сформується заявка, яка в свою чергу передається на склад для формування замовлення. Таким чином економиться значна частина часу і торгового представника і працівників фірми загалом. До того ж, слід зауважити, що відбувається загальне підвищення ефективності виконання замовлення товароодержувача.

Серед усіх дистриб'юторів, які використовують систему мобільної торгівлі, більшість впроваджує оптовий продаж продовольчих товарів за передпродажем. Саме цей спосіб, на думку керівників оптових підприємств, є більш ефективним на відміну від венселінгу, хоча вимагає більше затрат з розрахунку на один маршрут. Очевидно, що застосування передпродажу вимагає збільшення додаткових витрат, пов'язаних з купівлею або компенсацією додаткового автомобіля для торгового представника, а також введення в штатний розпис посади експедитора, який буде займатися безпосередньою доставкою товарів покупцю. Отже, ефективність даного підходу до організації постачання та збору замовлення має залежати від непропорційного зростання середньостатистичних обсягів замовлень та додаткового прибутку.

Водночас, не слід відкидати і покращення якісних показників, які характеризують процес доставки товарів та збору замовлення. Серед яких можна виділити і ефективний мерчандайзинг, підвищення відсотку повернення дебіторської заборгованості, підвищення рівня довіри до торгового представника, зменшення помилок при роботі з інформаційними потоками та ін. Не слід залишати поза увагою дослідження впливу факторів сезонності, особливостей асортименту, віддаленості замовника від постачальника та ін. Окремої уваги заслуговує дослідження проблеми підвищення продуктивності праці торгових представників, економії робочого часу, економії витратних матеріалів внаслідок впровадження компонентів системи мобільної

торгівлі. Все це може стати підставою для подальших досліджень даного напрямку.

Не останнє місце, на мою думку, у пошуку способів більш повного задоволення потреб покупців відіграватиме розвиток та впровадження в діяльність оптових підприємств інформаційних мобільних технологій та адаптація до сучасних вітчизняних умов, використання досвіду іноземних компаній.

1.3 Огляд прототипів

1.3.1 Система від SoftServe [13]

Система SalesWorks – дворівнева система (дистриб'ютор – торговий агент). Рішення для автоматизації всіх основних бізнес-процесів окремих дистриб'юторських компаній найрізноманітніших рівнів, що займаються прямими продажами товарів повсякденного вжитку (FMCG).

Складається із двох основних компонентів:

Мобільні модулі – розміщені на базі планшетів чи КПК (комунікаторів), підтримують автоматизацію роботи торгового агента в режимі преселінгу, венселінгу й мерчандайзингу.

Модуль дистриб'ютора – накопичує інформацію, отриману від виїзних співробітників підприємства, автоматизує всі процеси управління, аналізу, планування й контролю роботи торгового відділу й просування товару компанії на ринку.

Функціонал преселінгу підтримує весь набір інструментів для реалізації продукції за класичною схемою попереднього продажу, згідно з якою доставка товару виконується не раніше ніж на наступний день після формування замовлення торговими агентами.

1.3.2 Рішення від ТОВ «Скайрівер» [14]

Компанія ТОВ «Скайрівер» впроваджує рішення в сфері GPS контролю робочого персоналу для пристроїв на базі ОС Android. Програмне рішення «Skyriver MT» - покликане за допомогою мобільних пристроїв, вирішити питання автоматизації сфер діяльності, пов'язаних з торгівлею.

Компанія пропонує комплексне рішення, яке складається з:

– Додаток Skyriver MT на мобільний пристрій на базі ОС Android.
– Модуль для 1С 8.2 "Управління торгівлею".

Доступ до багатофункціонального web – інтерфейсу Skyriver GPS, для здійснення GPS моніторингу співробітників, планування роботи, а також отримання різних звітів (добових, групових, фото-звітів та інших).

1.3.3 Продукт 1С [15]

"1С: Мобільна торгівля" – продукт, що використовується для автоматизації мобільної торгівлі за допомогою мобільних пристроїв (смартфонів, планшетних комп'ютерів) під управлінням операційних систем IOS і Android. Продукт дозволяє автоматизувати різні види діяльності торгових представників оптових і дистриб'юторських компаній, що використовують облікові системи на платформі "1С: Підприємство 8", такі як "1С :. Управління торгівлею ред 11", "1С: Управління підприємством ERP 2.0".

1.3.4 Програмний продукт «Агент Плюс» [16]

Самостійний програмний продукт з власною базою даних. За допомогою модулів обміну даними "Агент Плюс: Мобільна торгівля" інтегрується з системами 1С.

У "Агент Плюс: Мобільна торгівля" реалізований двосторонній обмін даними за допомогою 3G, Wi-Fi, GPRS з обліковою системою, встановленою в офісі. Передача даних здійснюється за допомогою служби обміну даними "Агент Плюс: Служба обміну даними" (СОД). Функціональні можливості даного продукту включають в себе різні версії Програмний продукт "Агент Плюс: Мобільна торгівля" випускається в двох версіях: "Базова" і "Проф".

"Агент Плюс: Мобільна торгівля" працює на мобільних пристроях (смартфони, планшетні комп'ютери) під управлінням операційної системи Android починаючи з версії 4.0 до 5.1.

1.3.5 Програмний комплекс для мобільної торгівлі від SoftIndusty [17]

Це програмний комплекс для мобільної торгівлі – розроблений компанією SoftIndusty для автоматизації дистрибуції.

Можливості веб-модуля Гермес:

- Створювати шаблони заявок;
- Задавати маршрути для торгових представників;
- Відстежувати їх за допомогою GPS-координат.

Відстежувати:

- Дебіторські заборгованості клієнтів;
- Фінансові історії клієнтів;
- Стану заявок.
- Відвідування роздрібних торгових точок;
- Створення і виконання планів для торгових представників.

Можливості мобільного агента:

- Працювати з заявками;
- Швидко набирати заявку від роздрібної точки з урахуванням наявності товару на складі і його актуальною ціни;
- Переглядати історію заявок клієнта;
- Використовувати шаблони заявок для клієнтів, створені супервайзером;
- Обмінюватися повідомленнями з іншими користувачами;
- Демонструвати клієнту фото товарів із каталогу на екрані мобільного пристрою;
- Переглядати заборгованості по кожному клієнту;
- Отримувати від супервайзера маршрути, завдання і плани без відвідування офісу.

В результаті виконання огляду та порівняльного аналізу систем оптової торгівлі встановлено, що вони у своїй сукупності не володіють достатнім рівнем гнучкості. Усунення даної проблеми – сформувало основу концепції дипломної роботи [18].

2 РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ

2.1 Аналіз вимог до програмної системи

2.1.1 Аналіз предметної області

Поняття предметної області є специфічним та визначається у всіх предметних областях у відповідності до їх особливостей. Цим поняттям характеризується частина чи виокремлений напрямок діяльності в рамках певного контексту (ним може визначатись область дослідження чи та, що є об’єктом виокремленої діяльності).

У попередньому розділі було виявлено проблему недостатньої гнучкості мобільних систем аналогів.

Розроблювальна клієнт-серверна програмна система для організації автоматизації діяльності повинна бути розроблена з урахуванням недоліків її аналогів, а також бути достатньо гнучкою в інтеграції з іншими системами.

2.1.2 Постановка задачі

Виходячи з аналізу предметної області можна визначити основні задачі, які повинна виконувати програма [19].

Для початку програмний продукт повинен володіти зручним інтерфейсом, який дозволить забезпечити продуктивну роботу. Оскільки розроблювальна система використовуватиме для свого повноцінного функціонування базу даних, то необхідно потрібно забезпечити надійність з’єднання та безперебійність програми. Потрібно передбачити ряд запитів до неї. Також програма повинна запобігати випадковим спотворенням інформації.

Проаналізувавши предметну область перед розробкою програми постають такі задачі:

- Можливість оформлювати, редагувати та надсилати замовлення;
- Можливість перегляду інформації про торгову точку ;
- Можливість обміну даними з сервером;
- Можливість використання фільтрів при перегляді інформації;
- Можливість перегляд продукції а також залишків продукції;
- Можливість оновлення версії клієнта з сервера;
- Можливість відстежування місцезнаходження торгового агента;
- Можливість додавання коментарів до замовлення.

Також необхідно реалізувати систему авторизації, для забезпечення санкціонованого доступу на сервер, фільтрації перегляду даних і внесення коректної інформації.

2.1.3 Пошук актантів та варіантів використання

Оскільки ефективність такого підходу є беззаперечною, прийняте рішення використати підхід «клієнт-сервер» [20]. В основі взаємодії по типу «клієнт-сервер» знаходиться принцип того, що її завжди розпочинає лише клієнт, а серверна частина лише відгукує[ься на його запит. Крім того сервер інформує про те чи може він надати послугу клієнтові і якщо може, то на яких умовах. Зазвичай як клієнтське так і серверне програмне забезпечення може встановлюватись як на різних машинах, так і працювати на одному комп'ютері. Розроблювана програмна система поділена на дві частини: серверна частина, та клієнтська.

Серверна програмна частина реалізовуватиме частину варіантів використання клієнтської частини, а саме тому актантів для неї

виділяти не варто. Клієнтською частиною користуватиметься торговий представник. Він буде виконувати свою роботу по збору даних для формування замовлення.

Клієнтська програмна частина системи матиме одного актора – Користувач. Актор реалізовуватиме всі можливі варіанти використання, використовуючи увесь доступний функціонал програми. Користувач матиме можливість створення замовлення, корегування замовлення, перегляду звітностей, перегляду довідників, списків продукції та інше. Також матиме можливість змінювати деякі налаштування програми в залежності від потреб.

Отже, актор користувач реалізовуватиме такі варіанти використання:

- Створення нового замовлення;
- Редагування сформованого замовлення;
- Видалення замовлення;
- Додавання коментарів до замовлення;
- Зміна цін при формуванні замовлення;
- Перегляд інформації;
- Перегляд інформації про замовлення;
- Перегляд інформації про торгову точку
- Дебіторська заборгованість;
- Перегляд продукції та її залишків;
- Обмін даними;
- Отримання нових даних;
- Оновлення існуючих даних;
- Оновлення версії програми;
- Надсилання замовлення;
- Надсилання GPS даних;
- Зміна налаштувань програми.

2.1.4 Опис ключових варіантів використання

При першому запуску програми її потрібно буде налаштувати, зокрема заповнити дані для авторизації на сервері, а також ввести саму адресу сервера з яким відбуватиметься синхронізація. Саме так практично виглядає реалізація варіанту використання «Зміна налаштувань програми». Також від дозволяє змінити інші додаткові налаштування безпеки, наприклад задати пароль адміністратора для подальших змін налаштувань, або змінити налаштування системи відслідковування переміщення працівника.

Варіант використання «Отримання нових даних» дозволить користувачу системи виконати синхронізацію даних та отримати актуальні дані, що потрібні для роботи – тобто довідники: груп продукції, продукції, назв прасів, цін продукції, клієнтів, груп клієнтів, типів документу, а також довідника фірм.

Варіант використання «Оновлення існуючих даних» буде використовуватись в тому випадку, якщо довідники вже були завантажені, а користувач ініціював оновлення даних заново. У цьому разі потрібні дані може бути змінено, залишено без змін або деактивовано деякі довідники.

Варіант використання «Створення нового замовлення» дозволить користувачу знаходячись у конкретній торговій точці оформити замовлення. Для оформлення замовлення користувач використовуватиме наявні у мобільному пристрої дані, які були отримані пристроєм при отриманні нових даних або оновленні існуючих. Користувач обирає із заздалегідь сформованих опцій, саме ті, що потрібні в даний момент, наприклад: конкретну торгову точку, до якої вже «прив’язаний» конкретний прайс лист з цінами на продукти.

Варіант використання «Редагування замовлення» дозволить користувачу внести корективи в раніше створене, але, що важливо, ще не надіслане замовлення. Користувач системи може змінити деякі параметри такі як: тип створюваного документу; додати чи видалити потрібні товари. При потребі можна встановити персональну для клієнта ціну на будь-який товар. Саме так реалізується варіант використання «Зміна цін при формуванні замовлення».

Варіант використання «Видалення замовлення» дозволяє видалити сформоване, але ще не відправлене замовлення у разі помилкового створення або відсутності потреби у надсиланні такого замовлення.

Варіант використання «Надсилання замовлення» потрібен користувачу системи щоб власне надіслати оформлене у торговій точці замовлення на сервер. Надіслане замовлення не можна змінити або видалити.

При потребі вказати додаткові відомості до замовлення користувачу системи стане у нагоді варіант використання «Додавання коментарів до замовлення».

Варіант використання «Надсилання GPS даних» включений у ВВ «Надсилання замовлення» і передбачає відправлення раніше зібраних даних про місцезнаходження користувача разом із замовленням.

Варіанти використання актора користувач зображені на рисунку 2.1.

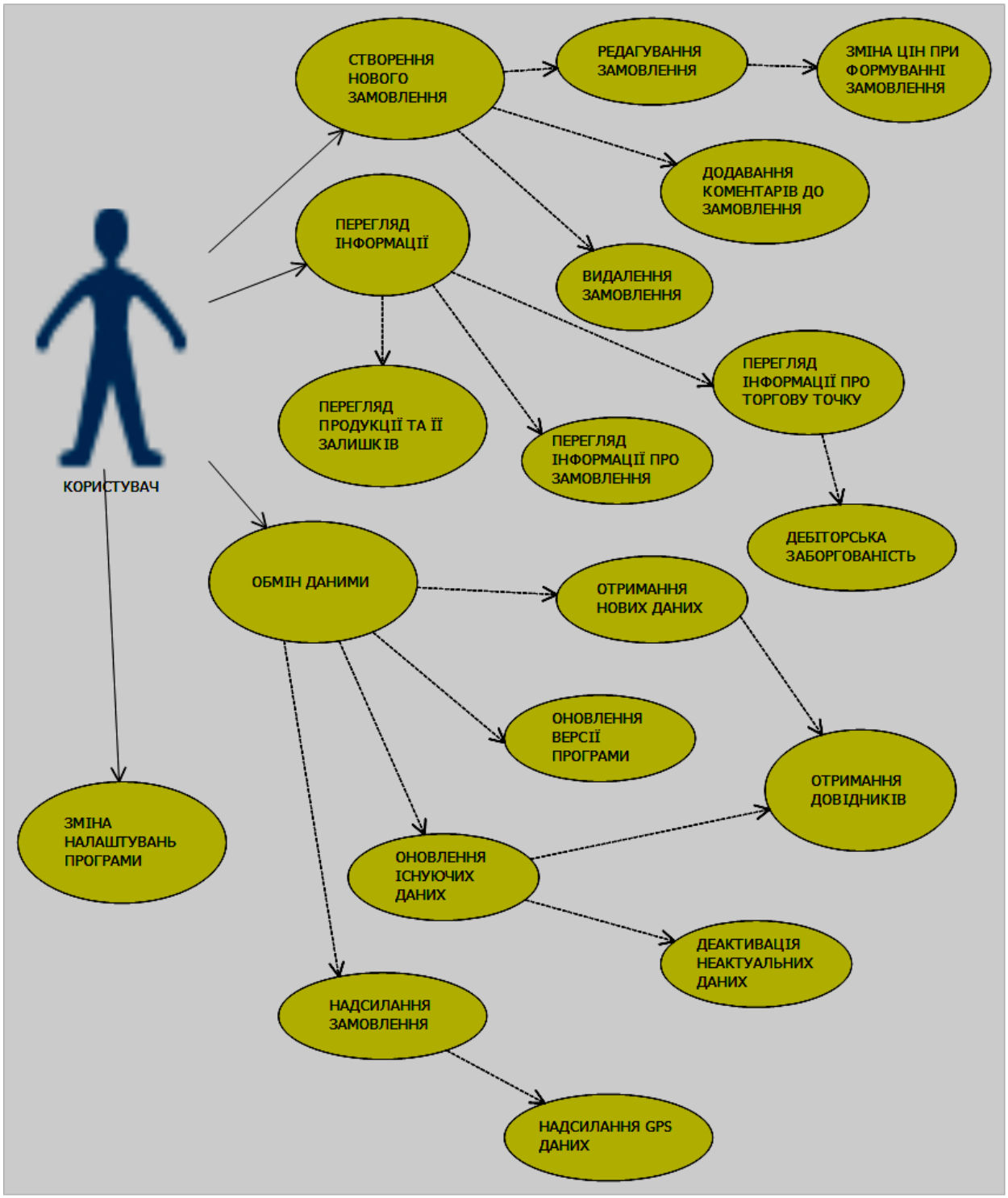

Рисунок 2.1 – Варіанти використання

Варіант використання «Надсилання GPS даних» включений у ВВ «Надсилання замовлення» і передбачає відправлення раніше

зібраних даних про місцезнаходження користувача разом із замовленням.

Варіант використання «Оновлення версії програми» реалізовує можливість отримання нової версії клієнта з серверу при оновленні даних через ВВ «Отримання нових даних» та «Оновлення існуючих даних». Нова версія буде завантажена на мобільний пристрій, далі користувачу буде запропоновано перевстановити програму.

При створенні, редагуванні чи перегляді замовлення варіант використання «Перегляд інформації» дозволить користувачу системи дізнатись дізнаватись адресу торгової точки, контактні дані, та інше. Також за допомогою ВВ «Перегляд продукції та її залишків» можна буде дізнатись яка продукція потрібна даній торговій точці, а також в чи присутні товари на складі та в якій кількості. Користувачу будуть потрібні дані про дебіторську заборгованість кожної торгової точки, для цього знадобиться ВВ «Дебіторська заборгованість».

Варіант використання «Отримання довідників» є розширенням варіантів «Оновлення існуючих даних» та «Отримання нових даних». При наявності довідників, в одному з випадків вони будуть завантаженні, в іншому – оновлені. Якщо оновлення буде містити нові дані котрі не матимуть відношення до вже наявних, то наявні дані будуть деактивовані за допомогою варіанту використання «Деактивація неактуальних даних».

2.2 Проєктування програмної системи

Проєктування – при розробці програмної системи, це один із найважливіших етапів життєвого циклу, не менш важливий та який слідує одразу після інженерії вимог. Основним завданням цього етапу є трансформація побажань замовників системи, які подано як моделі вимог, у проєктні рішення, які дадуть можливість реалізувати згадані

побажання у формах обраної системи програмування. Отже, під час проєктування трансформація простору вимог у простір проєктних рішень виконується. При цьому можна виділити процеси, які можна вважати відносно самостійними один від одного і виконувати як послідовно, так і паралельно, окремими командами виконавців [21]. Це такі процеси:

- концептуальне проєктування полягає в уточненні розуміння й узгодження деталей вимог;
- архітектурне проєктування полягає у визначенні головних структурних особливостей системи, яку будують;
- технічне проєктування полягає у відображенні вимог середовища функціонування і розроблення системи та у визначенні всіх конструкцій як композицій компонент;
- детальне проєктування полягає у визначенні подробиць функціонування та зв’язків для всіх компонент системи.

В основі проєктування будь-якого продукту лежить парадигма подолання складності загального завдання шляхом декомпозиції цільового продукту на окремі його складові або компоненти. Це твердження діє і для програмних систем як продуктів програмної інженерії. Для сучасного стану розвитку програмної інженерії домінуючою є об’єктно-орієнтована парадигма, за якою будь-яка система розглядається як сукупність взаємодіючих об’єктів.

2.2.1 Виявлення основних сутностей

База даних, як правило, містить багато різних типів сутностей. Кожен тип сутності має свій унікальний набір атрибутів, а кожна сутність – має свої власні значення для кожного атрибуту. Ці властивості визначають атрибути сутності, – це властивості, якими буде володіти об’єкт, що розглядається.

Також база даних володіє певним набором відношень. Відношення – це логічно побудований зв’язок між сутностями у базі даних. Відношення задаються за допомогою поєднання ключів.

Ключем або ключовим полем називається унікальне значення, яке дозволяє тим або іншим способом ідентифікувати суть або частку суті предметної області, тобто ключ – це значення деякого атрибуту або атрибутів в кортежі відношення, який представляє екземпляр суті в реляційній моделі даних. Розглядають два типи ключів: первинний ключ і зовнішній ключ. Первинним ключем – називається мінімальна множина атрибутів, що є підмножиною заголовка даного відношення, складене значення яких унікально визначає кортеж відношення. Як правило, термін первинний ключ позначає поле або групу полів таблиці бази даних, значення якого (або комбінація значень яких) використовується як унікальний ідентифікатор запису цієї таблиці. Зовнішнім ключем – називається поле таблиці, призначене для зберігання значення первинного ключа іншої таблиці з метою організації зв'язку між цими таблицями [22].

Також таблиці повинні бути певним чином організовані, що забезпечує оптимальне використання інформації, зокрема усунення повторень. Для цього використовують нормальну форму – властивість відношення в реляційній моделі даних, що характеризує його з погляду надмірності, яка потенційно може привести до логічно помилкових результатів вибірки або зміни даних. Нормальна форма визначається як сукупність вимог, яким повинно задовольняти відношення [23].

В даному випадку база даних, що знаходитиметься на мобільному пристрої міститиме таблиці продукції, цін продукції, груп продукції, клієнтів груп клієнтів та інші таблиці довідників потрібних для роботи. Також будуть ще дві таблиці в котрих записуватиметься інформація про замовлення.

Отже, під час дослідження предметної області можна виявити наступні сутності:

– Замовлення;
– Деталі замовлення.

Між цими встановлено відношення один до багатьох. Наприклад в кожному замовленні є певна кількість товарів. ER - діаграма сутностей представлена на рисунку 2.2.

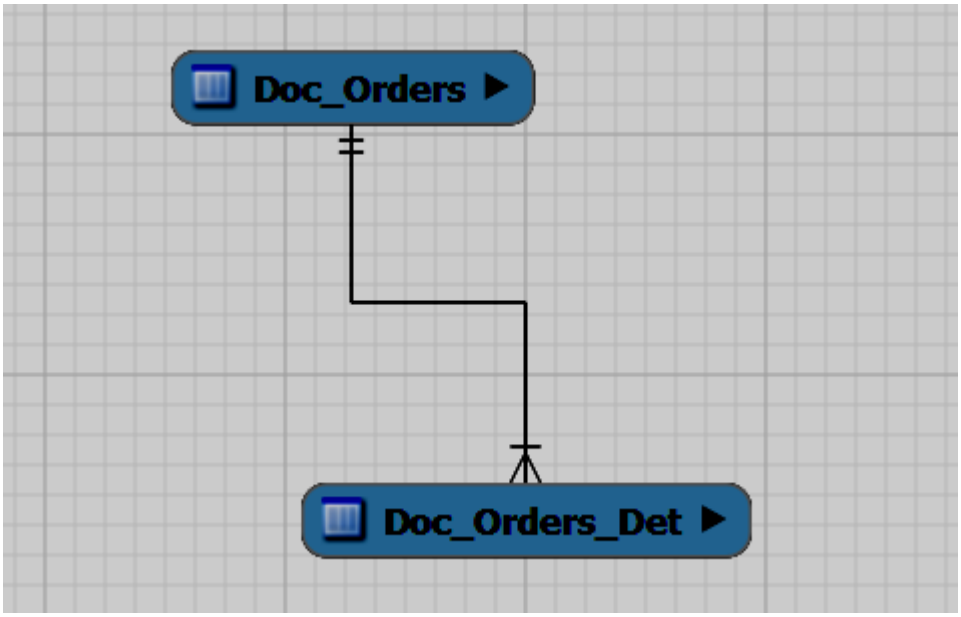

Рисунок 2.2 – ER діаграма сутностей

Нижче наведені сутності та їх атрибути:

– Замовлення – код замовлення, код клієнта, код прас листа, код пристрою, код фірми дата, час, знижка, кількість, сума, коментар, тип документу, сума з податками, сума без податків;
– Деталі замовлення – код замовлення, код продукту, кількість, знижка, ціна, ціна з податками, ціна без податків.

2.2.2 Побудова UML-діаграми класів

UML (англ. Unified Modeling Language) — уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, яка називається UML-моделлю. UML був створений для визначення, візуалізації, проєктування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація [24].

На рисунку 2.3 зображено UML-діаграму пакетів клієнтської частини програми.

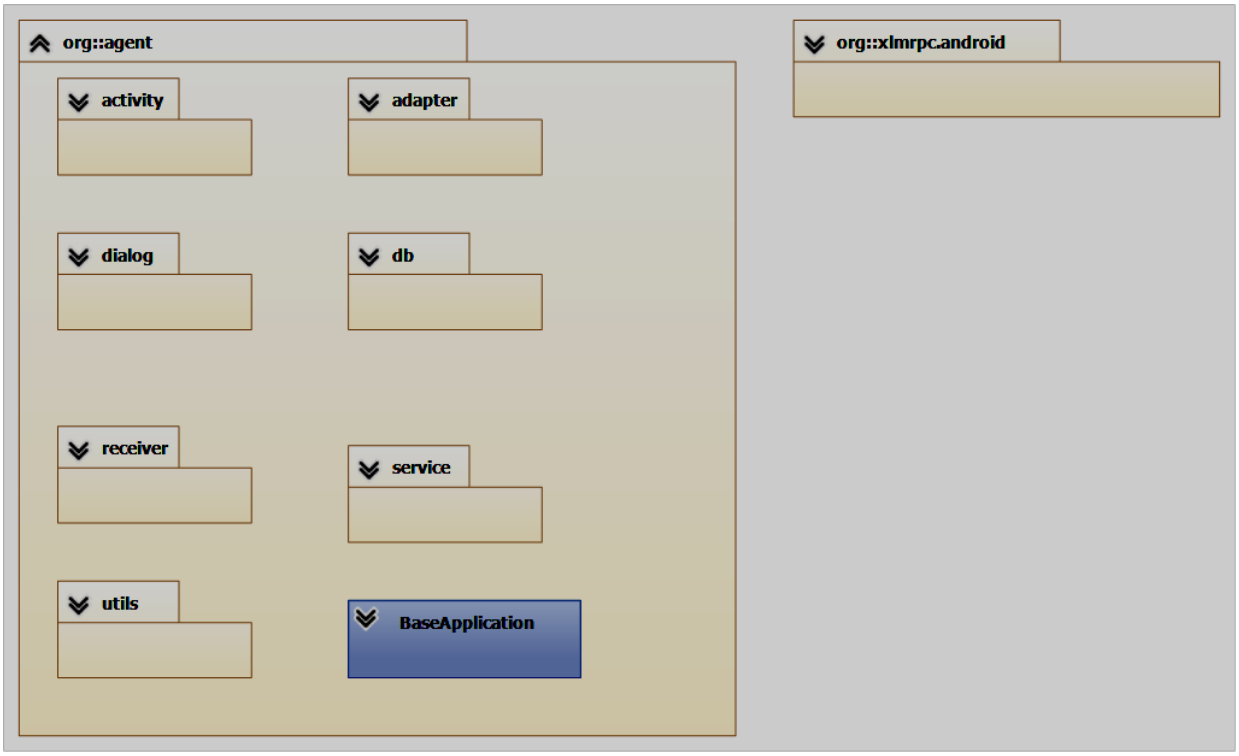

Рисунок 2.3 – UML-діаграма пакетів клієнтської частини програми

Всі усі дані програми містяться в пакеті org. Пакет org містить в собі пакети, agent та xmlrpc.android. Котрі у свою чергу містять у інші пакети, для agent: activity, adapter, dialog, db, receiver, service, utils; xmlrpc.android не містить пакетів, лише класи. Дана структура створена для того щоб чітко виділити та узагальнити, частини функціоналу відносно класів.

На рисунку 2.4 – представлено детальну UML-діаграму пакетів та класів клієнтської частини. Кожен пакет об’єднує в собі подібні за побудовою і функціями класи.

Рисунок 2.4 – детальна UML-діаграма пакетів та класів клієнтської частини програми

Класи пакету activity – це класи активності, діяльності (дії). Усі ці класи наслідують клас Activity – один із головних класів. В пакеті

adapter зібрані класи, що допомагають класам пакету activity працювати зі списками (ListActivity). Пакет utils містить класи для реалізації допоміжних функцій. Пакет db – класи для роботи з базою даних, саме об’єкти цих класів використовують усі інші класи для виконання операцій по роботі з базою даних. Пакет service – містить класи, що потрібні для роботи сервісу відслідковування місцезнаходження. Пакет receiver – містить клас для повернення даних роботи сервісу відслідковування місцезнаходження. Пакет dialog – описує роботу з деякими діалоговими вікнами зокрема пегляду інформації про торгову точку, а також для виведення назв документів. xlmrpc.android – пакет, що містить класи для реалізації роботи виклику віддалених процедур.

На рисунку 2.5 зображено основну UML-діаграму класів клієнтської частини, спеціальними позначеннями позначені відношення між ними.

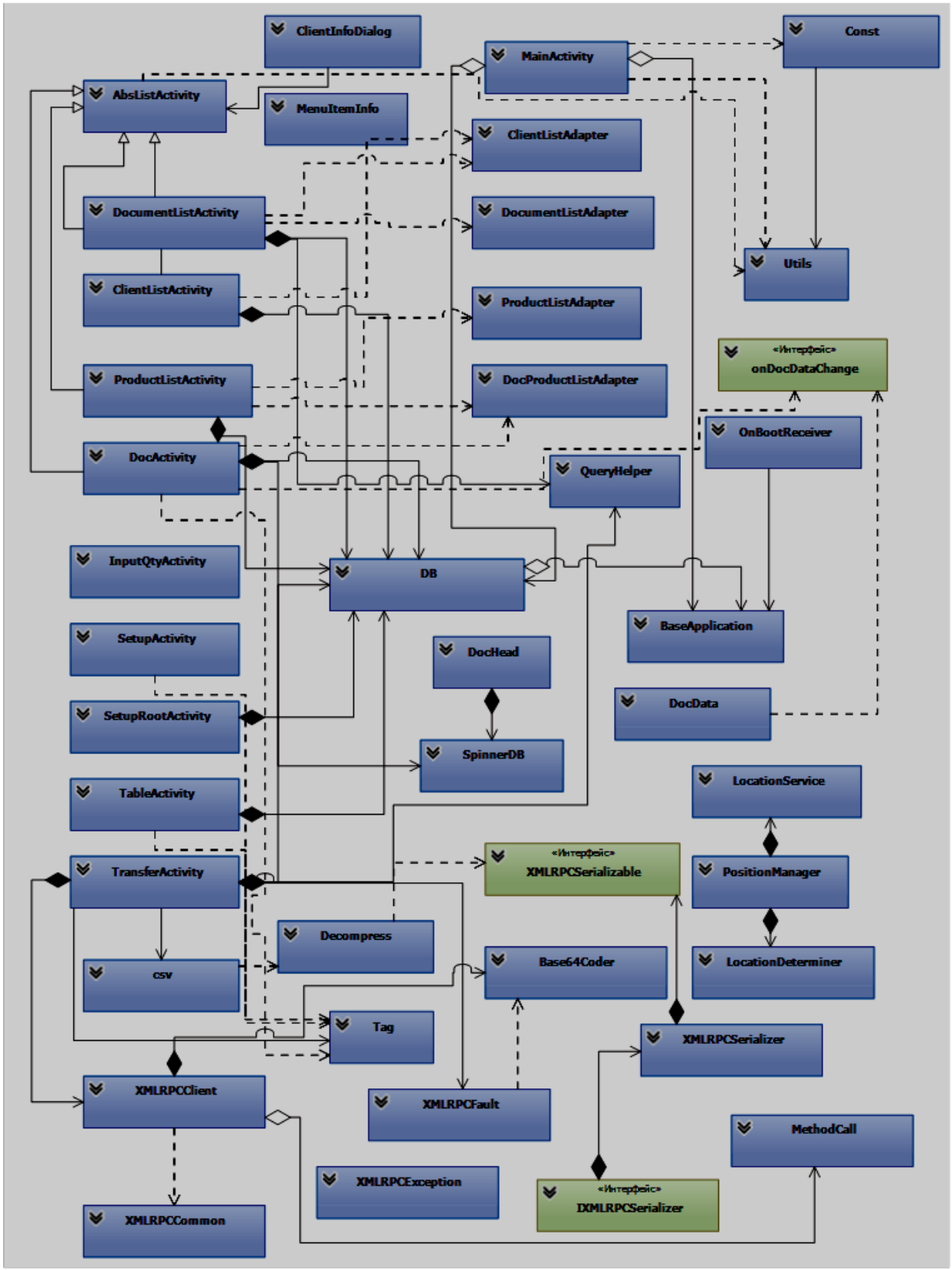

Рисунок 2.5 – основна UML-діаграма класів

UML-діаграма класів серверної частини зображена на рисунку 2.6. Agent є файлом запуску сервера, запускає сервер, згідно налаштувань cfg, також містить метод для запуску класу daemon. Класи використовують дані класу cfg, а також для логування дій існує клас log. Proxy_cvs клас, який обробляє звернення від клієнтської частини, забезпечує обробку, запис, та архівацію даних.

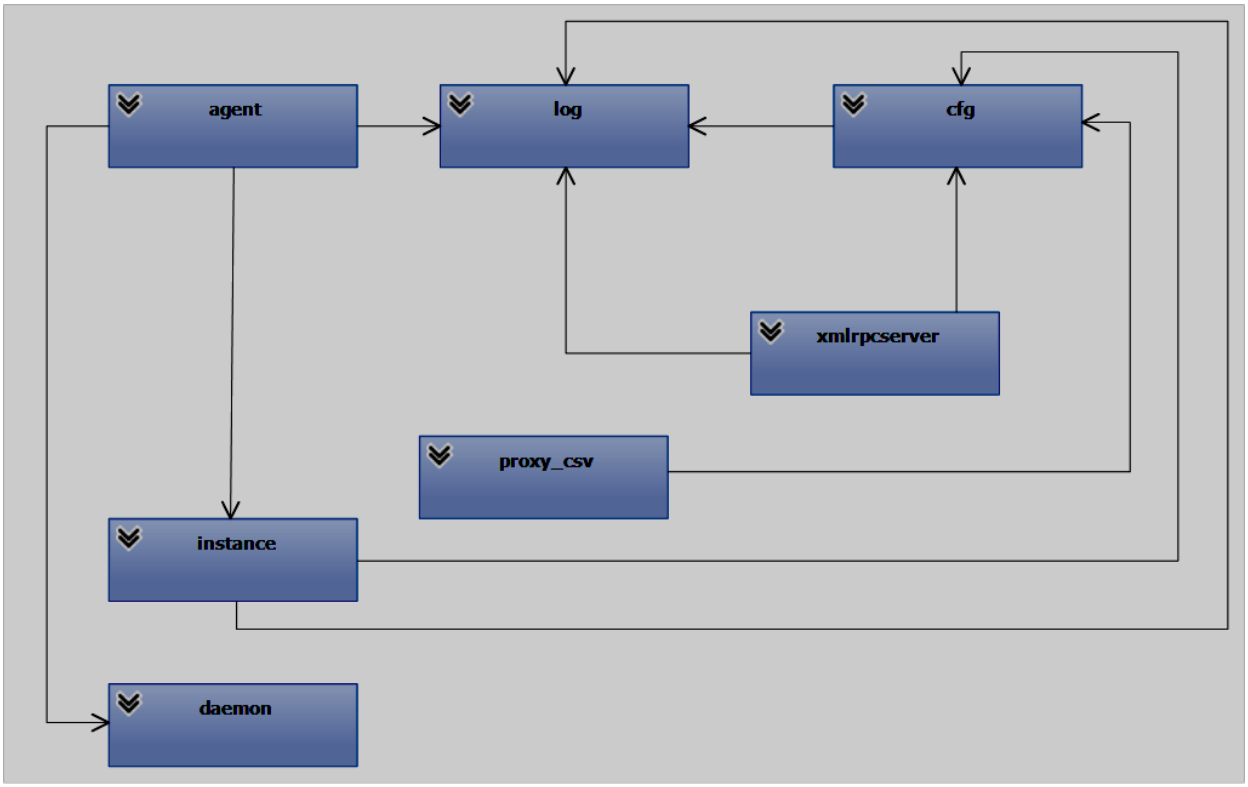

Рисунок 2.6 – основна UML-діаграма класів серверної частини

2.2.3 Моделювання архітектури

2.2.3.1 Клієнтська частина

Клас MainActivity є точкою ініціалізації програми, відповідає за головне меню, а також містить функцію перевірки стану служби GPS в налаштуваннях Android-пристрою. При спеціальних налаштування

проґрами, її робота без апаратно включеного модуля GPS буде неможлива.

Основні атрибути:

- public static int versionCode;
- private BroadcastReceiver mBatInfoReceiver
- private boolean mAgentID;
- private boolean mCustomTitleFlag;

Основні методи:

- protected void onActivityResult(int requestCode, int resultCode, Intent data) – перевіряє файл налаштування проґрами;
- private void customTitleBar(String left, String right) відповідає за футер головного меню, отримує дані про заряд батареї;
- private void checkGPS() – метод, що перевіряє чи увімкнений модуль GPS.

Клас AbsListActivity є абстрактним, містить лише оголошені методи, котрі перевизначені в класах, які реалізовують роботу зі списками.

Основні атрибути:

- protected SQLiteDatabase mDataBase;
- protected CursorAdapter mAdapter;
- protected QueryHelper mQueryHelper.

Основні методи:

- protected List<MenuItemInfo> createContextMenus(int position, long id) створення контекстного меню;
- protected String getContextMenuTitle(int position, long id) – перегляд контекстного меню.

Клас ClientListActivity відповідає за відображення і взаємодію зі списком.

Основні атрибути:

- private static final int MENU_GROUP;

– private static final int MENU_SEARCH;
– private static final int MENU_NEW_DOC;
– private static final int MENU_LIST_DOC;
– private static final int MENU_INFO.

Основні методи:

– public boolean onCreateOptionsMenu(Menu menu) – реалізовує додаткове меню та роботу фільтрів;
– public boolean onOptionsItemSelected(MenuItem item) – основне меню;
– private void NewDoc(long id, int position) метод запускає створення нового докуметну;

Класи ClientListActivity, DocumentListActivity, ProductListActivity, DocActivity реалізовані подібним чином вищеописаному класу, оскільки наслідувані від абстрактного класу AbsListActivity

Клас InputQtyActivity реалізовує можливість введення редагування кількісних характеристик для зміни, або ручного введення.

Основні атрибути:

– private Button mBtQtyPoint;
– private EditText mEdQty;
– private Vibrator mVibrator;
– private TextView mTvMessage, mTvQtyDescr, mTvInputDescr;
– private Bundle mParams;
– private StringBuffer mQtyBuffer;

Основні методи:

– private void Buf2Screen() – метод виведення буферу на екран;
– private float getQty() – метод отримання введених раніше даних;
– private void setQty(float val) – метод зміни значення кількості.

Клас SetupActivity описує та частково реалізує меню налаштувань проґрами, а також відповідає за роботу доступу в наступне меню налаштувань, якщо доступ в наступне меню здійснюється з використання пароля.

Основні методи:

- startSetupRoot()- створює вікно поверх основної активності для вводу пароля;
- public static boolean getVibration(Context context) – забезпечує налаштування молживості увімкнення\вимкення вібрації при вводі даних з віртуальної клавіатури раніше описаного класу InputQtyActivity.

Клас SetupRootActivity реалізовує для користувача можливість зміни налаштувань проґрами, успадковує абстрактний клас PreferenceActivity стандартної бібліотеки, реалізовує оголошені методи

Основні методи:

- public static String getMobileServer()- отримання даних, зокрема адресу сервера з яким буде проводитись обмін;
- public static String getApkFileName() – перевіряє чи є на сервері нова версія проґрами;
- public static String getGUID() - повертає GUID – унікальний код мобільного пристрою.

А також методи для налаштування відслідковування місцезнаходження:

- public static boolean getNoStartGPS();
- public static boolean getIsGPSLocation();
- public static int getGPSchedule() ;
- public static int getGPSAccuracy();
- public static int getGPSTimeout().

Клас TransferActivity є одним із класів що відповідають за обмін даними, який проводиться між серверною та клієнтською частинами.

Основні атрибути:

- private static final int SERVER_MOBILE;
- private static final int SERVER_WIFI;
- private static final int SERVER_TEST;
- private static final String temp_dir.

Основні методи:

- protected Void doInBackground(String... params) – перевизначаємо бібліотечний метод стандартного класу асинхронних завдань. Метод реалізовує функціонал отримання довідників;
- private void getReferenceFromFile(SQLiteDatabase db, String table_name,String file_name, String Message)- метод реалізовує функціонал наповнення бази даних з файлів довідників отриманих раніше;
- private boolean getZipArchive(byte[] data) – призначений для отримання розпакування і обробки тестових даних наповнення програми. Також виконує розпакування архіву з даними, що передається з серверу.

Клас DB працює з базою даних.

Основні атрибути:

- public static final String TABLE_CLIENT_GROUP;
- public static final String TABLE_CLIENT;
- public static final String TABLE_PRODUCT_GROUP;
- public static final String TABLE_PRODUCT;
- public static final String TABLE_PRICELISTS;
- public static final String TABLE_PRICES;
- public static final String TABLE_FIRMS;
- public static final String TABLE_TYPEDOC.

Основні методи:

- public void onCreate(SQLiteDatabase db) – перевизначаємо метод створення бази даних, створює базу даних і таблиці в ній(якщо база даних не існує).
- public void onUpgrade(SQLiteDatabase db, int old_version, int new_version) – перевизначаємо метод. Метод описує що потрібно змінити коли змінилась версія бази даних. Необхідність зміни версії бази даних може виникнути на подальших стадіях розробки, наприклад при додаванні нових, таблиць чи полів структуру бази даних, функція onUpgrade «знає» як потрібно модифікувати попередню версію бази даних при оновленні клієнтської серверної частини.

Клас QueryHelper – описує роботу в базі даних, та містить методи для створення курсорів баз даних, реалізовує вибірку з фільтрами

Основні атрибути:

- private String table_name;
- private String order_by;
- private String[] fileds;
- private String[] fields_on_table.

Основні методи :

- public Cursor CreateDocProductsCursor(SQLiteDatabase db);
- public Cursor CreateDocListCursor(SQLiteDatabase db);
- public Cursor CreateDocProductsCursor(SQLiteDatabase db);
- private String getTables().

Дані методи є частиною реалізації роботи з базою даних у вищеописаних Activity, таких як: ClientListActivity, DocumentListActivity, ProductListActivity, DocActivity та інших.

Клас Const містить константи для роботи з документами і не містить методів.

Основні атрибути даного класу є строковими з public static final String:

- EXTRA_PRODUCT_ID,EXTRA_ PRODUCT_NAME;
- EXTRA_CLIENT_ID, EXTRA_CLIENT_NAME;
- EXTRA_DOC_PRESVE, EXTRA_DOC_TYPE;
- EXTRA_DOC_ID, EXTRA_DOC_DESC.

Клас csv відповідає за створення, читання і редагування файлів csv, а також містить класи для обробки виключень(Exception), котрі передбачають обробку різноманітних виключень.

Основні методи:

- public Writer(File file), public Writer value(String value), public Writer newLine(),public Writer comment(String comment) – методи використовуються для запису інформації;
- public List<String> readLine() – метод для читання строк з csv файлу.

Клас public class XMLRPCClient extends XMLRPCCommon

Основні атрибути:

- private HttpClient client;
- private HttpPost postMethod;
- private HttpParams httpParams.

Основні методи:

- public XMLRPCClient(URI uri, String username, String password, String GUID)- конструктор класу, передає дані з заповенних в налаштуваннях полів логін та пароль обміну даних, також додає до них унікальний ідентифікатор пристрою;
- public Object callEx(String method, Object[] params) – формує post запит, відправляє його, опрацьовує виключення.

2.2.3.2 Серверна частина

Клас agent є також основним файлом запуску сервера. Основні методи:

- start() – запускає сервер, потрбні компоненти, використовує атрибути класу cfg;
- run(self) –метод запуску демона(процесу) з класу daemon;
- SetDocs(AgentID, DocsData) – функція створення та аріхування файлів на сервері з даних отриманих від клієнта.

Клас cfg є класом, що містить містить атрибути налаштувань серверу.

Основні атрибути:

- SERVER_ADRES;
- SERVER_PORT;
- SERVER_RPC_PATH;
- SERVER_USER;
- SERVER_PASSWD.

Клас daemon – батьківський клас процесу(демона) сервера, саме цей клас наслідується підкласом з класу agent.

Основні методи:

- daemonize(self) – метод запуску форків;
- start(self) – запуску процесу;
- stop(self) – зупинка процесу.
- delpid(self) – очищення файлу процесів(pidfile).

Клас instance обробляє звернення від клієнтської частини, забезпечує отримання клієнтом потрібних файлів

Основні методи:

- GetZipData(self, DeviceGUID) – в залежності від id користувача віддає потрібний архів з довідниками;

– GetUpdateApp(self, version) – проводить перевірку версій клієнтської частини наявної на сервері та на клієнті ;

Клас xmlrpcserver

Основні методи:

parse_request(self) – розкодує запит на авторизацію який закодований в base64 та надісланий клієнтом;

__authenticate(self) – приймає та обробляє запит авторизації;

Клас proxy_cvs обробляє звернення від клієнтської частини, забезпечує обробку, запис, та архівацію даних.

Основні атрибути:

– CSV_CLIENT = 'Ref_Clients.csv';
– CSV_CLIENT_GR = 'Ref_Clients_Groups.csv';
– CSV_PRODUCT_GR = 'Ref_Products_Groups.csv';

Основні методи:

– SetLocation(AgentID, location) – збереження даних місцезнаходження;
– SetDocs(AgentID, DocsData) – функція створення та архівування файлів на сервері з даних отриманих від клієнта.

2.3 Конструювання програмної системи

2.3.1 Основні принципи

Суспільство щохвилини розвивається та прогресує. Не викликає сумніву й той факт, що кожний наступний день вже ніколи не буде таким як попередній. Те що було новаторством ще буквально вчора, сьогодні вже є відверто вчорашнім днем. Ми ніколи не замислюємось, а, можливо й варта, що ще якихось 50 років тому навіть не в кожній оселі був телевізор, а новини, в більшості, населення дізнавалось із радіопередавачів, які намагались не вимикати. Сьогодні ж, щоб бути

успішним, варта не просто замислитись, але й натхненно працювати в напрямку самовдосконалення, самоосвіти та технічного розвитку. Вже очевидним є й те, що без постійної генерації та реалізації нових задумів та ідей з використанням сучасних, зокрема інформаційних, технологій не можливо бути в тренді у якості успішного, ефективного та прогресивного члена «елітного клубу» сучасного світу.

Життєвий цикл будь якого винаходу розпочинається від зародження ідеї, яка виникає з потреби, і при серйозності намірів, яка є комерційною таємницею, аж до моменту, коли тенденція до розвитку вимагає подальшого вдосконалення з врахуванням стану сучасного етапу розвитку науки і техніки. Однак, на якомусь етапі таємниця перестає бути такою і таким чином людство постійно розвивається і колишні винаходи стають загальним надбанням цивілізації. Саме завдяки цьому, приступаючи до розробки чогось нового, варта проаналізувати стан речей, вивчити аналоги та не витрачати на марно свій час розробляючи вже відомі речі, а займатися, безпосередньо щонайменше, вдосконаленням того, що є вимогою сьогодення.

Загальновідомі поняття описуються за допомогою стандартизованих мов. Наприклад, для опису математичних тотожностей використовується мова математичних символів, для опису електричних кіл використовується мова схем, для опису музичних творів використовується нотний стан.

Подібним чином розвиваються ідеї у конструюванні програмного забезпечення: від винаходу до стандартизованих понять. Для опису стандартизованих понять в галузі конструювання програмного забезпечення використовують мову UML [23].

2.3.2 Вибір мови та середовища розробки

2.3.2.1 Клієнтська частина

Android — операційна система і платформа для мобільних телефонів та планшетних комп'ютерів, створена компанією Google на базі ядра Linux. Підтримується альянсом Open Handset Alliance (OHA). Хоча Android базується на ядрі Linux, він стоїть дещо осторонь Linux-спільноти та Linux-інфраструктури. Базовим елементом цієї операційної системи є реалізація Dalvik віртуальної машини Java, і все програмне забезпечення і застосування спираються на цю реалізацію Java [25].

Оскільки розробка клієнтської частини програмної системи здійснюється для пристроїв на ОС Android. Тому, власне мовою програмування буде Java.

Java – об'єктно-орієнтована мова програмування, випущена компанією Sun Microsystems у 1995 році як основний компонент платформи Java. Зараз мовою займається компанія Oracle, яка придбала Sun Microsystems у 2009 році. Синтаксис мови багато в чому схожий на C та C++. У офіційній реалізації, Java програми компілюються у байткод, який при виконанні інтерпретується віртуальною машиною для конкретної платформи.

Oracle надає компілятор Java та віртуальну машину Java, які задовольняють специфікації Java Community Process, під ліцензією GNU General Public License.

Програми, написані на Java, можна скомпілювати в Dalvik байткод і виконувати на Dalvik virtual machine, яка являє собою розроблену спеціально для використання на мобільних пристроях віртуальну машину, незважаючи на те, що не є стандартною Java Virtual Machine.

Мова значно запозичила синтаксис із C і C++. Зокрема, взято за основу об'єктну модель C++, проте її модифіковано. Усунуто можливість появи деяких конфліктних ситуацій, що могли виникнути через помилки програміста та полегшено сам процес розробки об'єктно-орієнтованих програм. Ряд дій, які в C/C++ повинні здійснювати програмісти, доручено віртуальній машині. Передусім, Java розроблялась як платформо-незалежна мова, тому вона має менше низькорівневих можливостей для роботи з апаратним забезпеченням. За необхідності таких дій java дозволяє викликати підпрограми, написані іншими мовами програмування.

Android Studio — інтегроване середовище розробки (IDE) для платформи Android. Воно було представлене 16 травня 2013 року на конференції Google I/O менеджером по продукції корпорації Google — Еллі Паверс (Ellie Powers). 8 грудня 2014 компанія Google випустила перший стабільний реліз Android Studio 1.0.

Android Studio прийшов на зміну плагіну ADT для платформи Eclipse. Середовище побудоване на базі сирцевих текстів продукту IntelliJ IDEA Community Edition, що розвивається компанією JetBrains. Android Studio розвивається в рамках відкритої моделі розробки та поширюється під ліцензією Apache 2.0.

Бінарні складання підготовлені для Linux (для тестування використаний Ubuntu), Mac OS X і Windows. Середовище надає засоби для розробки застосунків не тільки длясмартфонів і планшетів, але і для носимих пристроїв на базі Android Wear, телевізорів (Android TV), окулярів Google Glass і автомобільних інформаційно-розважальних систем (Android Auto). Для застосунків, спочатку розроблених з використанням Eclipse і ADT Plugin, підготовлений інструмент для автоматичного імпорту існуючого проєкту в Android Studio.

2.3.2.2 Серверна частина

Серверна частина програмної системи буде розроблятись на мові Python [26]. Python — інтерпретована об'єктно-орієнтована мова програмування високого рівня з динамічною семантикою. Розроблена в 1990 році Гвідо ван Россумом. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднання існуючих компонентів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій так і у вихідній формі на всіх основних платформах. В мові програмування Python підтримується декілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна тааспектно-орієнтована.

Серед основних переваг мови можна назвати такі:

– чистий синтаксис (для виділення блоків слід використовувати відступи);
– переносимість програм (що властиве більшості інтерпретованих мов);
– стандартний дистрибутив має велику кількість корисних модулів (включно з модулем для розробки графічного інтерфейсу);
– можливість використання Python в діалоговому режимі (дуже корисне для експериментування та розв'язання простих задач);
– стандартний дистрибутив має просте, але разом із тим досить потужне середовище розробки, яке зветься IDLE і яке написане на мові Python;
– зручний для розв'язання математичних проблем (має засоби роботи з комплексними числами, може оперувати з цілими

числами довільної величини, у діалоговому режимі може використовуватися як потужний калькулятор).

Python має ефективні структури даних високого рівня та простий, але ефективний підхід до об'єктно-орієнтованого програмування. Елегантний синтаксис Python, динамічна обробка типів, а також те, що це інтерпретована мова, роблять її ідеальною для написання скриптів та швидкої розробки прикладних програм у багатьох галузях на більшості платформ.

Інтерпретатор мови Python і багата стандартна бібліотека[en] (як вихідні тексти, так і бінарні дистрибутиви для всіх основних операційних систем) можуть бути отримані з сайту Python www.python.org, і можуть вільно розповсюджуватися. Цей самий сайт має дистрибутиви та посилання на численні модулі, програми, утиліти та додаткову документацію.

Інтерпретатор мови Python може бути розширений функціями та типами даних, розробленими на C чи C++ (або на іншій мові, яку можна викликати із C). Python також зручна як мова розширення для прикладних програм, що потребують подальшого налагодження.

Python підтримує динамічну типізацію, тобто, тип змінної визначається лише під час виконання. З базових типів слід зазначити підтримку цілих чисел довільної довжини і комплексних чисел. Python має багату бібліотеку для роботи з рядками, зокрема, кодованими в юнікоді.

З колекцій Python підтримує кортежі (tuples), списки (масиви), словники (асоціативні масиви) і від версії 2.4, множини.

Система класів підтримує множинне успадкування і метапрограмування. Будь-який тип, включаючи базові, входить до системи класів, й за необхідності можливе успадкування навіть від базових типів.

2.3.3 Вибір системи управління базами даних

Для зберігання даних на клієнті в ОС Android використовується SQLite [27].

Для зберігання даних на сервері файли будуть формуватись у загальному форматі text/csv. Це дозволить легко імпортувати або експортувати дані з існуючих програм для організації торгівлі.

CSV – файловий формат, котрий є відмежовувальним форматом для представлення табличних даних, у якому поля відокремлюються символом коми та переходу на новий рядок. Поля, що містять коми, декілька рядків, або лапки (позначаються подвійними лапками), мають обмежуватися з обох боків лапками.

SQLite – полегшена реляційна система керування базами даних, яка міститься у вигляді бібліотеки, де реалізовано багато зі стандарту SQL-92. Сирцевий код SQLite поширюється як суспільне надбання (англ. public domain), тобто може використовуватися без обмежень та безоплатно з будь-якою метою. Фінансову підтримку розробників SQLite здійснює спеціально створений консорціум, до якого входять такі компанії, як Adobe, Oracle, Mozilla, Bentley і Bloomberg.

Особливістю SQLite є те, що воно не використовує парадигму клієнт-сервер, тобто рушій SQLite не є окремим процесом, з яким взаємодіє застосунок, а надає бібліотеку, з якою програма компілюється і рушій стає складовою частиною програми. Таким чином, як протокол обміну використовуються виклики функцій (API) бібліотеки SQLite. Такий підхід зменшує накладні витрати, час відгуку і спрощує програму. SQLite зберігає всю базу даних (включаючи визначення, таблиці, індекси і дані) в єдиному стандартному файлі на тому комп'ютері, на якому виконується

застосунок. Простота реалізації досягається за рахунок того, що перед початком виконання транзакції весь файл, що зберігає базу даних, блокується; ACID-функції досягаються зокрема за рахунок створення файлу-журналу. Кілька процесів або нитей можуть одночасно без жодних проблем читати дані з однієї бази. Запис в базу можна здійснити тільки в тому випадку, коли жодних інших запитів у цей час не обслуговується; інакше спроба запису закінчується невдачею, і в програму повертається код помилки. Іншим варіантом розвитку подій є автоматичне повторення спроб запису протягом заданого інтервалу часу.

У комплекті постачання йде також функціональна клієнтська частина у вигляді виконуваного файлу sqlite3, за допомогою якого демонструється реалізація функцій основної бібліотеки. Клієнтська частина працює з командного рядка, і дозволяє звертатися до файлу БД на основі типових функцій ОС. Завдяки архітектурі рушія можливо використовувати SQlite як на вбудовуваних (embedded) системах, так і на виділених машинах з гігабайтними масивами даних.

У роботі використано SQLite версії 3.7. Слід також враховувати, що при використанні програми на різних версіях ОС Android версія SQLite буде відрізнятись. Нижче приведено версії БД для деяких версій ОС.

SQLite 3.8.10.2:

– 23-6.0-M (note: M Preview 1 (SDK level 22) used 3.8.10)

SQLite 3.8.6:

– 22-5.1.1-Lollipop

SQLite 3.8.4.3:

– 21-5.0-Lollipop

SQLite 3.7.11:

– 20-4.4W.2-Android Wear
– 19-4.4-KitKat

– 18-4.3-Jelly Bean
– 17-4.2-Jelly Bean
– 16-4.1-Jelly Bean

2.3.4 Побудова схеми реляційної бази даних та опис концептуальної моделі

У підпункті 2.2.1 виявлено наступні сутності системи: замовлення, деталі замовлення.

У таблиці замовлення будуть записуватись такі дані: код замовлення, код клієнта, код прас листа, код пристрою, код фірми дата, час, знижка, кількість, сума, коментар, тип документу, сума з податками, сума без податків. Первинним ключем для таблиці є поле _id. Структуру таблиці зображено на рисунку 2.7.

Table name: Doc_Orders WITHOUT ROWID

	Name	Data type	P	F	U	H	N	C	Default value
1	_id	INTEGER	🔑						NULL
2	presventype	SMALLINT							NULL
3	client_uid	VARCHAR (36)							''
4	docdate	DATE							CURRENT_DATE
5	doctime	TIME							CURRENT_TIME
6	discount	FLOAT							NULL
7	itemcount	FLOAT							NULL
8	mainsumm	CURRENCY							NULL
9	description	VARCHAR (80)							NULL
10	paymentdate	DATE							NULL
11	doctype_id	INTEGER							NULL
12	paytype	INTEGER							NULL
13	docstate	SMALLINT							0
14	sumwithoutvat	CURRENCY							NULL
15	sumwithvat	CURRENCY							NULL
16	regnum	VARCHAR (15)							NULL
17	number	INTEGER							NULL
18	pricelist_uid	VARCHAR (36)							''
19	CREATED_TIME	TIMESTAMP							DATETIME('now', 'localtime')
20	central_uid	VARCHAR (36)							''
21	firm_uid	VARCHAR (36)							''

Рисунок 2.7 – Структура таблиці замовлення

У таблиці деталі замовлення будуть записуватись такі дані – код замовлення, код продукту, кількість, знижка, ціна, ціна з податками, ціна без податків. Первинним ключем для таблиці є поле _id. Зовнішнім ключем є поле order_id. Структуру таблиці зображено на рисунку 2.8.

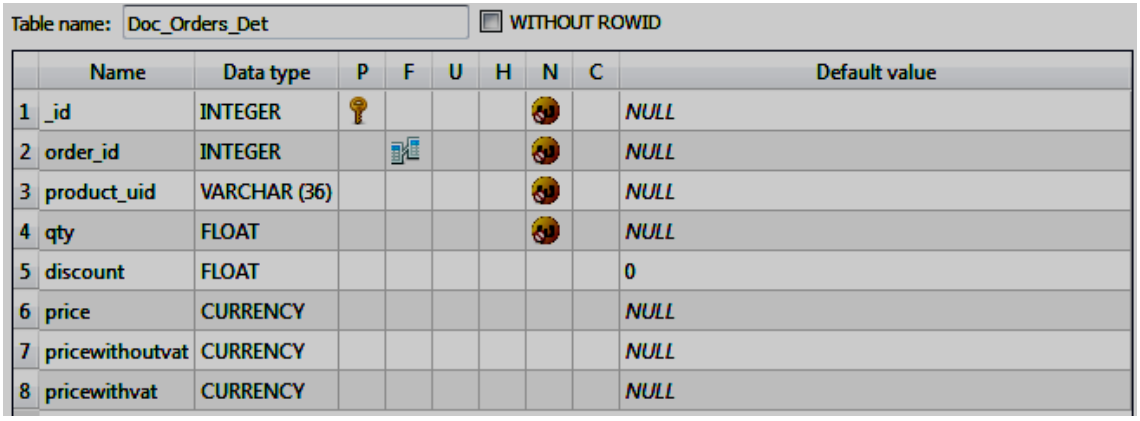
Table name: Doc_Orders_Det WITHOUT ROWID

	Name	Data type	P	F	U	H	N	C	Default value
1	_id	INTEGER							NULL
2	order_id	INTEGER							NULL
3	product_uid	VARCHAR (36)							NULL
4	qty	FLOAT							NULL
5	discount	FLOAT							0
6	price	CURRENCY							NULL
7	pricewithoutvat	CURRENCY							NULL
8	pricewithvat	CURRENCY							NULL

Рисунок 2.8 – Структура деталі замовлення

2.3.5 Реалізація основних класів та методів

Далі буде наведено ряд прикладів програмної реалізації функціоналу клієнтської частини програми. Метод checkGPS() перевіряє чи увімкнено на пристрої GPS. Викликається, якщо в налаштуваннях програми заборонено запуск без увімкненого GPS. Програмна реалізація методу checkGPS() представлена в лістингу 1.1.

Лістинг 1.1 – Перевірка стану GPS модуля.

```
      private void checkGPS() {
   if (!SetupRootActivity.getNoStartGPS(this)) return;
   LocationManager manager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
   if (!
manager.isProviderEnabled(LocationManager.GPS_PROVIDER)) {
      new AlertDialog.Builder(this)
         .setTitle(R.string.setup_nostartwithoutgps)
```

```
          .setMessage(R.string.msg_NoOnGPS)
          .setCancelable(false)
            .setPositiveButton(R.string.lb_on_gps, new
DialogInterface.OnClickListener() {
               public void onClick(DialogInterface dialog,
int which) {
                  startActivityForResult(new
Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS),
                        MSG_SETUP);
               }
            })
            .setNegativeButton(R.string.lb_setup, new
DialogInterface.OnClickListener() {
               public void onClick(DialogInterface dialog,
int which) {
                  startActivityForResult(new
Intent(MainActivity.this, SetupActivity.class),
                        MSG_SETUP);
               }
            })
          .show();
   }
}
```

До відправки замовлення на сервер його варто зберегти в базі даних клієнта аби мати можливість переглядати його та при необхідності редагувати. Програмна реалізація даного функціоналу представлена на лістингу 1.2.

Лістинг 1.2 – метод для збереження документу в базі даних

```
      private void SaveDocument(int doc_state) {
       ProductListAdapter adp = (ProductListAdapter)
mAdapter;
       ContentValues cval = new ContentValues();
       cval.put("number",      mDocId);
       cval.put("client_uid",   mClientUID);
       cval.put("pricelist_uid",mPriceUID);
       cval.put("itemcount",   adp.mDocData.getQtySumm());
       cval.put("mainsumm",    adp.mDocData.getSumm());
       if(doc_state != -1){
           cval.put("docstate",    doc_state);
       }
       mDataBase.beginTransaction();
       mDataBase.update(DB.TABLE_DOCUMENT, cval,
QueryHelper.KEY_ID + "=" + mDocId, null);
       mDataBase.delete(DB.TABLE_DOCUMENT_DET, "order_id =
" + mDocId, null);
       adp.mDocData.Save(mDocId, mDataBase);
```

```
        mDataBase.setTransactionSuccessful();
        mDataBase.endTransaction();
        setResult(RESULT_OK, null);
    finish();
}
```

У користувача повинна бути можливість відредагувати раніше створене замовлення. Функція UpdateDocForm() призначена саме для цього. Програмна реалізація даного функціоналу представлена в лістингу 1.3.

Лістинг 1.3 – Збереження нових даних замовлення після його редагування

```
      private void UpdateDocForm(boolean FirstInit) {
    if(FirstInit){
        if(mState_of_doc == Const.DOCSTATE_SEND){
            EditText formDocDescription =
(EditText)findViewById(R.id.eDocDescription);
            formDocDescription.setClickable(false);
            formDocDescription.setEnabled(false);
            spPriceDoc.setEnabled(false);
            spFirm.setEnabled(false);
            spTypeDoc.setEnabled(false);
        }
    }
    getDocTotal();
    long docLines = getDocLinesCount();

    final TabHost tabHost = (TabHost)
findViewById(android.R.id.tabhost);
    final TabWidget tabWidget = tabHost.getTabWidget();
    final ViewGroup tab = (ViewGroup)
tabWidget.getChildAt(1);
    final TextView tv = (TextView)
tab.findViewById(R.id.tabsText);
tv.setText(getString(R.string.tab_lbl_docprodlist)+((docLin
es>0)?"("+docLines+")":""));
    setTitle(getString(R.string.tab_lbl_orderN)+ mDocID+"
"+ mDocClientName);
    UpdateSaveButton();
    UpdateCheckSend();
}
```

У клієнта буде можливість переглядати усі створені замовлення, для зручності додана можливість відфільтрувати

результати, за день, тиждень, або за увесь час. Програмна реалізація даного функціоналу представлена у вигляді методу onCreateDialog() та наведена в лістингу 1.4.

Лістинг 1.4 – Реалізація роботи фільтрів при перегляді створених замовлень

```
protected Dialog onCreateDialog(int dialogID) {
    super.onCreateDialog(dialogID);
    switch (dialogID) {
        case DLG_APPLY_FILTER:
            final CharSequence[] filter_names = new
CharSequence[]
                    {
getString(R.string.lb_filterall),
getString(R.string.lb_filtertoday),
getString(R.string.lb_filterweek)
                    };
            Dialog dlg = new AlertDialog.Builder(this).
                setTitle(R.string.lb_filter_title).
                setItems(filter_names, new
DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface
dialog, int item) {
                        ApplyFilter(item);
                    }
                }).create();
            return dlg;
        default:
            return null;
    }
}
```

Для створення бази даних при першому запуску клієнтської частини програмної системи буде викликаний спеціальний метод. Клас, що працює з базою даних наслідує SQLiteOpenHelper, отже може перевизначати його методи. В перевизначеному методі onCreate() викликається один із файлів ресурсів програми, а саме файл ресурсу – ddl_of_database. Фізичний файл має назву database.xml і містить набір SQL-запитів для створення бази даних. Реалізація

даного функціоналу представлена в лістингу 1.5. Лістинг SQL-запитів для створеня бази даних наведено в додатку

Лістинг 1.5 – Створення бази даних при першому запуску програми

```
public void onCreate(SQLiteDatabase db) {
   Log.d("DB onCreate", db.toString());
   String[] ddl =
BaseApplication.getInstance().getResources().getStringArray
(R.array.ddl_of_database);
   for (String ddl_txt: ddl) db.execSQL(ddl_txt);
}
```

Для авторизації користувача на сервері клієнт відправлятиме у закодованому вигляді, використовуючи алгоритм base64, повідомлення, яке міститиме логін, пароль, а також унікальний ідентифікатор мобільного пристрою GUID. Програмна реалізація наведена в лістингу 1.6

Лістинг 1.6 – Клієнтська реалізація авторизації на сервері

```
public XMLRPCClient(URI uri, String username, String
password, String GUID) {
       this(uri);
       postMethod.addHeader("Authorization", "Custom " +
Base64Coder.encodeString((username + ":" + password +
":"+GUID)));
       ((DefaultHttpClient)
client).getCredentialsProvider().setCredentials(
       new AuthScope(uri.getHost(),
uri.getPort(),AuthScope.ANY_REALM),
       new UsernamePasswordCredentials(username,
password));
   }
```

Далі буде наведено ряд програмних реалізацій функціоналу серверної частини програмної системи. Метод start( ) виконує запуск сервера, згідно налаштувань cfg. Також в процесі розробки було розроблено клас для логування і виведення на екран деяких операцій. В даному випадку можна буде дізнатись у вікні програми, що сервер

запущений, також буде відображено на якій P адресі та який використовується порт для передачі даних. Реалізацію методу запуску сервера наведено в лістингу 1.7.

Лістинг 1.7 – запуск сервера

```
def start():
    global server
    server = ThreadingXMLRPCServer(addr=(SERVER_ADRES,
SERVER_PORT))
    server.register_introspection_functions()
    server.register_instance(Instance())
    Log('Start server on %s:%s' % (SERVER_ADRES,
SERVER_PORT))
    server.serve_forever()
```

Доступ на сервер, як згадувалось раніше відбуватиметься з використанням комбінації логіна, пароля, а також унікального ідентифікатору мобільного пристрою клієнта. На сервері міститься спеціальний файл для кожного пристрою, саме цей файл вказує серверу, що під'єднаний коректний клієнт, якому потрібно віддати, або від якого потрібно прийняти дані згідно того який id у цього клієнта. Id клієнта вказується в файлі з іменем, яке і є унікальним ідентифікатором пристрою. Програмна реалізація даного функціоналу наведена в лістингу 1.8.

Лістинг 1.8 – Перевірка клієнта, його GUID-ідентифікатора

```
def Guid2Id(GUID):
    try:
        check_guid = PATH_TO_DEVICES + GUID
        #Log(" check_guid=" + agentid, False)
        Agent_ID = 0
        if os.path.isfile(check_guid):
            return int( open(check_guid, "r").read() )
        else:
            Log("GUID file doesn't exist:"+GUID)
            return False
    except:
        Log("Wrong GUID file: "+GUID)
        return None
```

Функція аутентифікації отримає повідомлення, яке надсилав клієнт (лістинг 1.6). Розкодує його, а також виділить такі частини: логін, пароль, GUID. Перші дві з яких звірить з налаштуваннями сервера, а третю перевірить використовуючи функціонал наведений в лістингу 1.8, реалізація перевірки отриманих даних наведена в лістингу 1.9.

Лістинг 1.9 – перевірка отриманих від клієнта даних щодо авторизації

```
def __authenticate(self):
        auth = self.headers.get("Authorization", "")
        if DEBUG_LEVEL > 4:
            Log("Authenticate:\t%s" % auth)
        if not auth:
            return False
        (auth_metod, encode) = auth.split()
        if auth_metod <> 'Custom':
            return False
        (user, passwd, guid) = b64decode(encode).split(':')
        check_guid = self.PATH_TO_DEVICES + guid
        Log(" GUID="+guid, False)
        if user == SERVER_USER and passwd == SERVER_PASSWD
and os.path.exists(check_guid) and
os.path.isfile(check_guid):
            return True
        else:
            return False
```

Для роботи серверу в цілому будуть запущені 2 форки(fork) – fork це системний виклик, який створює новий процес, який є копією батьківського процесу. Це зазвичай системний виклик, який реалізований на рівні ядра. Fork це основний (і історично, єдиний) метод створення процесу в UNIX-подібних операційних системах. Дані 2 форки реалізовані в так званій схемі «double fork magic» [28].

Дана схема описує вирішення трудностей роботи процесів-демонів, в батьківських класах, виконання яких має тривати

безкінечно довго та у класах-потомках виконання яких закінчується власне після виконання потрібних методів. Реалізація створення форків представлена у лістингу 1.10.

Лістинг 1.10 – Реалізація створення форків

```
def daemonize(self):
          try:
               pid = os.fork()
               if pid > 0:
                    sys.exit(0)
          except OSError, e:
               sys.stderr.write("fork #1 failed: %d (%s)\n"
% (e.errno, e.strerror))
               sys.exit(1)
          os.chdir("/")
          os.setsid()
          os.umask(0)
          try:
               pid = os.fork()
               if pid > 0:
                    sys.exit(0)
          except OSError, e:
               sys.stderr.write("fork #2 failed: %d (%s)\n"
% (e.errno, e.strerror))
               sys.exit(1)
          sys.stdout.flush()
          sys.stderr.flush()
          si = file(self.stdin, 'r')’
          so = file(self.stdout, 'a+')
          se = file(self.stderr, 'a+', 0)
          os.dup2(si.fileno(), sys.stdin.fileno())
          os.dup2(so.fileno(), sys.stdout.fileno())
          os.dup2(se.fileno(), sys.stderr.fileno())
          # записываем pidfile
          atexit.register(self.delpid)
          pid = str(os.getpid())
          file(self.pidfile, 'w+').write("%s\n" % pid)
```

2.4 Використання програмної системи

2.4.1 Розгортання програмної системи. Системні вимоги

Розгортання програмного забезпечення – це усі дії, що роблять програмну систему готовою до використання. Даний процес є частинною життєвого циклу програмного забезпечення [29].

Загалом процес розгортання складається з кількох взаємопов'язаних дій із можливими переходами між ними. Ця активність може відбуватися як з боку виробника так і з боку споживача. Оскільки кожна програмна система є унікальною, то усі процеси та процедури під час розгортання важко передбачити. Тому, "розгортання" можна трактувати як загальний процес відповідно до певних вимог та характеристик. Розгортання може здійснюватись програмістом і в процесі розробки програмного забезпечення.

В Україні інколи англомовний термін software deployment перекладають як "впровадження", оскільки згідно з давніми ще радянськими стандартами орієнтованими на планову економіку, впровадження мало бути кінцевим етапом будь-якого виробництва. Що породило деякі неоднозначності у літературі. У вітчизняній літератури ці терміни часто вживаються або як взаємозамінні синоніми, або як окремі назви етапів. Ці два терміни дещо відрізняються за своєю суттю. Впровадження - вужчий термін. Тому часто фірми пов’язані з розробкою та встановленням програмного забезпеченням вказують, що вони здійснюють впровадження і розгортання програмного забезпечення.

Для розгортання системи необхідні такі мінімальні вимоги:

– для розгортання серверної частини підійде комп’ютер, віртуальний сервер, виділений сервер з підключенням до мережі Internet, а також:

1. ОС Microsoft Windows XP та вище або Linux-подібні системи;
2. Інтерпретатор python версії 2.5-2.7;
3. 200 МБ вільного дискового простору;
4. 1024 МБ оперативної пам’яті;
5. Тактова частота процесора 1 ГГц;

– для розгортання клієнтської частини:

1. Мобільний пристрій з ОС Android 4.0 та вище.

3 ОПИС ТА ТЕСТУВАННЯ ПРОҐРАМНОЇ СИСТЕМИ

Одним із найважливіших етапів є тестування програмної системи, оскільки саме тут перевіряється її відповідність до поставлених вимог. Зазвичай розробник може заздалегідь розробити архітектуру проґрами, продумати класи і методи, які реалізуються, але він не зможе наперед передбачити кількість багів – проґрамних дефектів, які можуть виникати в процесі розробки. Об'єктно-орієнтовний підхід до розробки проґрамного забезпечення дозволяє тестувати неповністю готовий продукт. Завдяки цьому можна провести тестування окремих компонентів проґрамної системи.

Тестування – це в основному процес технічного дослідження. Він виконується на вимогу замовників, і призначений для вияву інформації про якість продукту відносно контексту, в якому він має використовуватись. До цього процесу входить виконання проґрами з метою знайдення помилок [30].

3.1 Опис та робота із системою

Тестування клієнта здійснювалось в емуляторах операційної системи Android версій 4.0-5.1.1, а також на мобільному пристрої марки Lenovo, моделі P780 з операційною системою Android 4.4.2.

Тестування сервера здійснювалось на віртуальному сервері з ОС Ubuntu 12.04/LTS та Python 2.7.10, котрий орендований у однієї із хостингових компаній, а також на комп'ютері з ОС Windows 7, інтерпритатором Python 2.7.10 у середовищах Python IDLE та NetBeans

Далі буде наведено ряд графічного матеріалу по роботі з проґрамою.

На рисунку 3.1 зображено вигляд екрану при першому запуску в вертикальній і горизонтальній орієнтації.

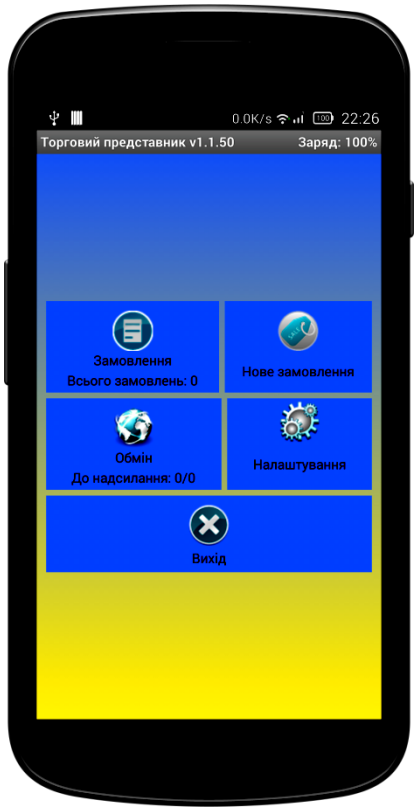

Рисунок 3.1 – Запуск програми

Головний екран є досить інформативним. Він показує інформацію про кількість уже створених замовлень, а також показує кількість документів, що готові до надсилання. В меню «Замовлення» показані замовлення, що тільки створені, створені і помічені «як готові до надсилання», а також, ті замовлення, що уже надіслані. Меню налаштувань є два, одне із них – звичайні налаштування, де можна налаштувати лише вібрацію при заповнені самого замовлення, а також переглянути GUID-ідентифікатор пристрою. Для доступу в меню «Налаштування адміністратора» потрібен пароль. По замовчуванні пароль не вказано. Вигляд меню налаштувань зображено на рисунку 3.2.

Рисунок 3.2 – Вікна налаштувань програми

Далі потрібно перейти в налаштувати адресу для обміну даними. Зробити це можна в пункті «Мобільна мережа». В даному

меню можна також налаштувати «Пароль адміністратора», щоб доступ в меню налаштувань був з використанням паролю. В меню також можна змінити налаштування модуля відслідковування координат, частоту відстежування, а також тайм-аут і точність визначення. Якщо на мобільному пристрої апаратно GPS вимкнено, а у програмі відстежування – увімкнено, то програма збиратиме дані на основі мобільної мережі

В меню «Обмін» можна отримати з серверу необхідні для роботи дані – довідники: клієнтів та груп клієнтів, продукції та груп продукції, цін, та інше.

Рисунок 3.3 – Обмін даними та створення замовлення

Меню «Нове замовлення» при першому запуску зображено на рисунку 3.3. Вибрати клієнта, тип документу чи фірму є наразі неможливим, оскільки клієнт ще було синхронізовано з сервером. Замовлення зберегти також не вдасться. Після оновлення даних замовлення можна коректно створювати(рисунок 3.3).

На вкладці «продукти» потрібно вибрати продукцію та необхідну кількість, та зберегти документ, помітити готовим до надсилання, та надіслати через меню «Обмін». Підбір продуктів зображено на рисунку 3.4.

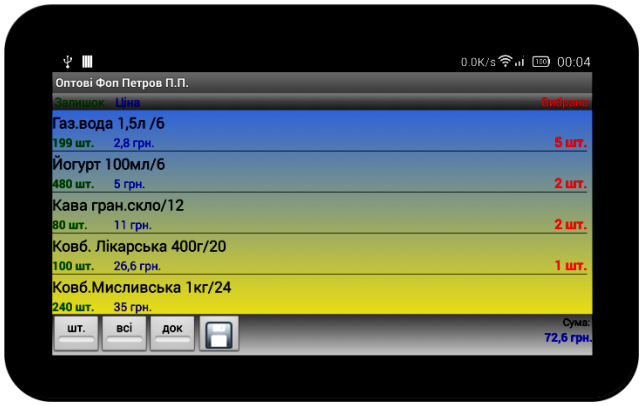

Рисунок 3.4 – Підбір продуктів

Серверна частина програмної системи не містить графічного інтерфейсу. Момент отримання клієнтом довідників фіксує сервер він виводить в лог на екран. В лозі вказується IP-адреса з якої відбувається з'єднання, повідомлення про аутентифікацію, яке закодоване алгоритмом base64, GUID пристрою, ID агента та розмір архіву з даними, що були надіслані на клієнт. Лог роботи серверу зображено на рисунку 3.5.

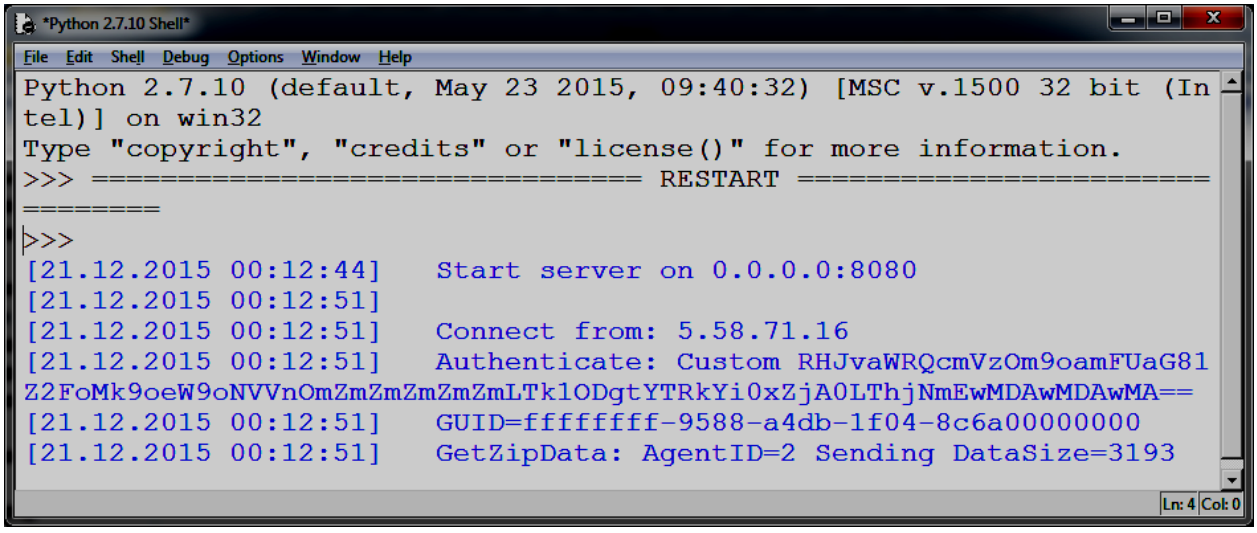

Рисунок 3.5 – Лог роботи серверу: отримання довідників.

При надсиланні даних на сервер, він виведе інформацію про під’єднання, GUID, розпізнавання операцію збереження даних, а також про передачу зібраних координат по місцезнаходження клієнта. Лог роботи сервера представлено на рисунку 3.6

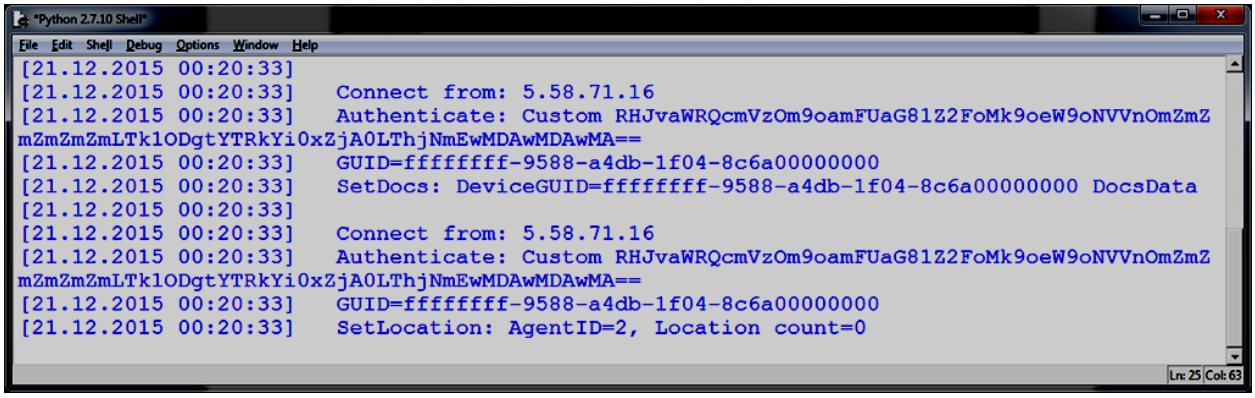

Рисунок 3.6 – Лог роботи серверу: збереження даних на сервері

3.2 Тестування (ручне)

Для виявлення помилок при роботі клієнтської частини програми під час розробки використовувався клас Log та його об’єкти для налагодження програми та тестування виконання деяких операцій. Тестування отримання довідників переставлено на рисунку 3.7

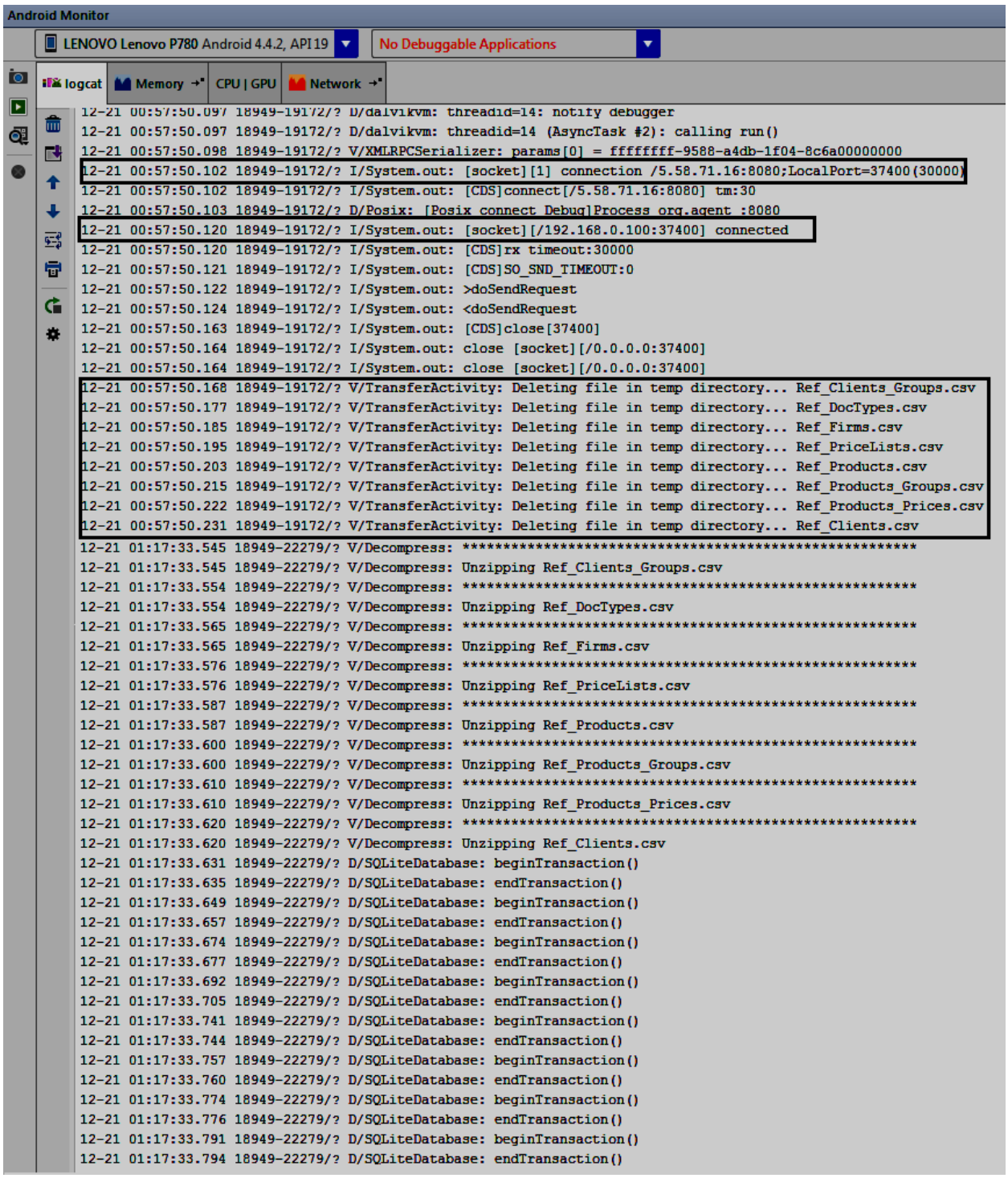

Рисунок 3.7 – Тестування отримання довідників

Спробуємо оновити довідники та переглянути лог виконання програми використовуючи Android Monitor в середовищі Android Studio. Як видно з рисунку 3.7 дані щодо адреси обміну сервером коректно передаються операційній системі. Трохи нижче можна помітити, що клас TransferActivity почав видалення старих

тимчасових файлів, та отримав нові. Після клас Decompress розархівує отримані файли. Після виконання потрібних запитів відбудеться збереження даних в БД мобільного пристрою. Звіривши дані з бази даних, можна сказати, що вони записались коректно.

3.3 Основні результати тестування

Під час тестування не було помічено жодного аварійного закінчення роботи програми. З цього можна зробити висновок, що розроблена програмна система є працює стабільно, а всі можливі виключення обробляються належним чином. Загалом протестовано роботу з програмною системою, були виконані сценарії ручної перевірки подібні до тих, що описані в підрозділі 3.1.

Програма не викликає значних навантажень на операційну систему.

Розроблена клієнт-серверна програмна система для організації автоматизованого передпродажу та є повістю готовим продуктом, здатним виконувати відведені їй функції.

4 ОРГАНІЗАЦІЙНО-ЕКОНОМІЧНА ЧАСТИНА

Метою дипломної роботи є вдосконалення через автоматизацію роботи суб’єкта пов’язану з актуальною на сьогоднішній день діяльністю, в якій об’єктом може бути не лише послуга, а й достовірна інформація чи будь-які дані, що можуть представляти комерційний інтерес, при використанні підходу «клієнт-сервер».

Для здійснення розрахунків використано реальні дані роботи, описано технологічний процес розробки із зазначенням трудомісткості кожної операції, визначено суму матеріальних затрат, та витрат на оплату праці основного і допоміжного персоналу, включно з відрахуванням на соціальні заходи, обчислено додаткові затрати, а також визначено економічну ефективність та термін окупності продукту [31].

4.1 Планування стадій та етапів проєктування програмного забезпечення

Відповідно до ст. 1 Закону № 3792, комп'ютерна програма – це набір інструкцій у вигляді слів, цифр, кодів, схем, символів чи у будь-якому іншому вигляді, виражених у формі, придатній для зчитування комп'ютером, які приводять його в дію для досягнення певної мети або результату. В такому самому значенні розуміється цей термін і в контексті податкового обліку (пп. 8.2.2 Закону про прибуток). Статтею 8 Закону № 3792 передбачено, що комп'ютерні програми є об'єктом авторського права.

Діяльність з програмування – це процес створення (розробки) програми, який може бути представлений послідовністю таких кроків: формулювання вимог до програми та розробка алгоритму; кодування

(запис алгоритму на вибраній мові програмування); відлагодження; тестування; доопрацювання програми; розробка довідкової системи.

Проєктування ПЗ складається з низки послідовних, цілеспрямованих, взаємозв'язаних та взаємообумовлених етапів, на які можна поділити весь часовий відрізок, що відводиться на розробку проєкту (таблиця 4.1).

Таблиця 4.1 – Етапи розробки проєкту

Найменування		Вид роботи	
стадії	Етапу	шифр	зміст роботи
1 Підготовча стадія	1.1 Вивчення стану предметної області	1.1.1	Дослідження проблеми
		1.1.2	Вибір платформи для реалізації проєкту
		1.1.3	Вивчення та аналіз аналогічних розробок
		1.1.4	Економічне обґрунтування доцільності виконання проєкту
	1.2 Розробка технічного завдання	1.2.1	Складання ТЗ
		1.2.2	Узгодження ТЗ із зацікавленими сторонами
		1.2.3	Складання плану та розрахунок розробки
2 Технічна пропозиція	2.1 Аналіз ТЗ та техніко-економічне обгрунтування проєкту	2.1.1	Доведення техніко-економічного обгрунтування
		2.1.2	Аналіз ТЗ та визначення приоритетних аспектів розробки
		2.1.3	Доведення та уточнення загального обсягу робіт, строків виконання та витрат
3 Теоретична розробка	3.1 Теоретичне вивчення задачі	3.1.1	Дослідження технічних особливостей інформаційної системи
		3.1.2	Визначення переліку технологій, які використовуватимуться при розробці та мови програмування
		3.1.3	Визначення форматів даних та запитів і їх узгодження із розробниками проєкту
		3.1.4	Розробка алгоритмів роботи програми на високому рівні
		3.1.5	Розробка структури систем забезпечення та схеми взаємодії її компонент
4 Практична реалізація	4.1 Реалізація окремих модулів ПЗ	4.1.1	Розробка алгоритмів роботи програми на низькому рівні
		4.1.2	Розробка окремих класів та компоновка їх у модулі
		4.1.3	Розробка графічного користувацького інтерфейсу для системи
	4.2 Відладка ПЗ	4.2.1	Автономна відладка окремих модулів системи
		4.2.2	Комплексна відладка ПЗ системи

Продовження таблиці 4.1

		4.3.1	Розробка структурної схеми субмодуля
	4.3 Розробка субмодуля	4.3.2	Розробка функціональної схеми субмодуля
		4.3.3	Розробка алгоритму функціонування субмодуля
		4.3.4	Обгрунтування вибору елементного базису та розробка схеми
Найменування		Вид роботи	
5 Доробка всього комплексу	5.1 Тестування всієї системи	5.1.1	Тестування системи у реальних умовах
		5.1.2	Доробка систем із урахуванням результатів тестування
	5.2 Підготовка документації по системі	5.2.1	Підготовка звіту про розробку системи
		5.2.2	Підготовка технічної документації
		5.2.3	Розробка довідкової системи, допоміжної і супроводжувальної документації, запис CD/DVD диска
6 Заключна стадія	6.1 Ознайомлення зацікавлених осіб з проєктом	6.1.1	Підготовка презентації
		6.1.2	Демонстрація системи

Головна мета планування процесу розробки – це визначення необхідних ресурсів на всіх його етапах.

Традиційний процедурний підхід для розробки ПЗ в основі якого лежать алгоритми, процедури і функції передбачає розробку ПЗ як монолітного композиту, що в подальшому, як правило, вимагає великих витрат на супровід та модернізацію.

Об’єктно-орієнтований підхід, що ґрунтується на основі об’єктів певних класів, що описують певну область, описують певну поведінку (методи) та володіють властивостями (атрибутами), орієнтовані на варіанти використання та покроковий, покомпонентний процес розробки.

Підготовча стадія, технічна пропозиція і заключна стадія при об’єктно-орієнтованому підході залишаються незмінними. Стадії теоретичної розробки, практичної реалізації та доробки всього комплексу програмного забезпечення описані з точки зору специфіки об’єктно-орієнтованого підходу із врахування ЖЦ розробки ПЗ і представлені із уточненням фахівців, залучених у кожний робочий процес, та артефактів (таблиця 4.2).

При створенні ПЗ за об’єктно-орієнтованим підходом необхідно залучити більшу кількість фахівців, більшу увагу приділити покроковій покомпонентній розробці, що, можливо, збільшить робочий час і витрати. Однак, цей підхід значно скорочує витрати на подальший супровід і модернізацію, що приводить до зменшення загальних витрат.

Таблиця 4.2 - Робочі процеси життєвого циклу розробки ПЗ

№	Робочі процеси	Діяльності	Співробітники	Артефакти
1	«Визначення вимог»	Ідентифікація актантів (А) і варіантів використання (ВВ), Модель ВВ Описи (формалізація) ВВ і сценаріїв реалізації, Визначення пріоритетності ВВ Створення прототипів інтерфейсів користувача (ІК)	Системний аналітик Специфікатор ВВ	Модель ВВ Актанти Глосарій ВВ Прототип ІК
2	«Проєктування»	Проєктування архітектури, Визначення вузлів та мережевих конфігурацій Визначення систем та їх інтерфейсів Визначення підсистем та зв’язків між ними Визначення інтерфейсів підсистем Визначення архітектурно значущих класів та класів проєктування, Визначення загальних механізмів проєктування Проєктування ВВ Проєктування класів Визначення вимог до реалізації	Архітектор Інженер з ВВ	Модель про-ектування Модель розміщеня Опис арітектури Проєкти реквізитів ВВ Класи про-ектування
3	«Реалізація»	Модель реалізації, Компоненти, Інтерфейси компонентів, Опис архітектури План збирання системи, Реалізація архітектури, Ітеративна реалізація класів Проєктування з генерацією програмного коду, Збірка системи, Планування білдів Реалізація білдів і системи в цілому	Системний інтегратор Інженер-програміст	Модель реа-лізації Опис архітекту-ри План збірки Компоне-нти Підсист-еми реалізації Інтерфейси

Продовження таблиці 4.2

	«Тестування»	Модель тестування Тестові приклади Процедура тестування Тестові компоненти План і оцінка тестування Розробка тестів Тестування цілісності Тестування системи Оцінка результатів тестування	Інженер з тестування Системний тестер	Модель тестування Тестові приклади План тестування Оцінювання тестів

Терміни розробки залежать від замовника, від наданої необхідної інформації (зразків довідників, документів і т.і.), рівня та порядку оплати та інших умов, від замовлення, і можуть становити від 2 тижнів до 6 місяців і більше (за даними сотфверних компаній).

Вартість складання технічного завдання переважно складає до 10% від планованої вартості розробки і становить від 200 до 5000 гривень (за даними аутсорсингових компаній). Роботу зі складання технічного завдання веде керівник проєкту разом із програмістами та консультуючись із замовником.

Для визначення загальної тривалості проведення робіт доцільно дані витрат часу по окремих операціях техпроцесу звести у таблицю (таблиця 4.3).

Таблиця 4.3 – Тривалість та трудоємність розробки та реалізації програмного проєкту

Найменування роботи	Виконавець, посада, спеціальність	К-сть виконавців	Тривалість виконання роботи, дні
2	3	4	5
Дослідження предметної області, вибір середовища для реалізації проєкту, вивчення та аналіз аналогічних розробок	Керівник проєкту	2	1
	Програміст		

Продавження таблиці 4.3

Економічне обгрунтування доцільності виконання проєкту, складання ТЗ, узгодження ТЗ із зацікавленими сторонами, складання плану та розрахунок розробки	Керівник проєкту	2	1
	Програміст		
	Програміст		
Доведення техніко-економічного обгрунтування, аналіз ТЗ та визначення приоритетних аспектів розробки, доведення та уточнення загального обсягу робіт, строків виконання та витрат	Керівник проєкту	2	1
	Програміст		
Теоретична розробка процедурний підхід	Програміст	1	3
Теоретична розробка об'єктно- орієнтований підхід	Системний аналітик, специфікатор ВВ, архітектор, інженер з ВВ	4	4
Практична реалізація процедурний підхід	Програміст	1	5
Практична реалізація об'єктно-орієнтований підхід	Системний інтегратор, інженер-програміст	3	7
Тестування програмного забезпечення	Інженер з тестування	1	3
Доробка всього комплексу програмного забезпечення	Програміст	1	4
Заключна стадія	Керівник проєкту	2	1
	Програміст		

Як для процедурного, так і для об'єктно-орієнтованого підходу, підготовча і заключна стадії та технічна пропозиція будуть виконуватися одинаково, теоретична розробка і практична реалізація для об'єктно-орієнтованого підходу вимагатимуть залучення більшої кількості ІТ-спеціалістів, що, можливо, збільшить тривалість виконання роботи.

4.2 Розрахунок витрат на реалізацію проєкту та оцінка економічної ефективності

Оцінка вартості комп'ютерних програм базується на витратному підході: використанні первісної вартості об'єктів, виходячи з фактичних витрат на розробку та доведення до комерційного використання з урахуванням амортизації. Оподаткування заробітної плати програмістів підприємства-розробника, яким встановлено певний посадовий оклад, відбувається аналогічно оподаткуванню зарплати працівників інших галузей господарства.

Враховуючи залежність якості кінцевого продукту від кваліфікації програмістів, розробники часто йдуть на додаткові заходи їх стимулювання, які найчастіше втілюються у:

- певній системі преміювання (за виконання графіку розробки чи його дострокове виконання, оптимізацію етапів розробки тощо). Премії оподатковуються аналогічно витратам на оплату праці;

- авторській винагороді за кожну продану одиницю програми (роялті). Сума роялті не обкладається внесками до Пенсійного фонду та фондів соціального страхування, а використання авторських прав оформлюється відповідним договором (ліцензійний авторський договір).

Разом з тим до створення ПЗ можуть бути залучені також позаштатні програмісти як зареєстровані, так і не зареєстровані підприємцями. В обох випадках співпраця з ними здійснюється на підставі цивільно-правового договору, найчастіше – договорами підряду. Щодо оподаткування виплат за договором підряду, то все залежить від того, чи зареєстрований виконавець підприємцем:

- якщо виконавець за договором підряду не зареєстрований фізособою-підприємцем, то виплати за таким договором оподатковуються ПДФО за ставкою 15 %, також нараховуються (у

розмірі 33,2 %) та утримуються (у розмірі 2 %) внески до Пенсійного фонду (п. 1 та п. 4 ст. 4 Закону № 400). Оподаткування провадиться у межах максимальної величини, що дорівнює 15 розмірам прожиткового мінімуму, встановленого для працездатних осіб;

- якщо ж виконавець за договором підряду є фізособою-підприємцем, яка сплачує єдиний податок чи знаходиться на загальній системі оподаткування, виплати, що здійснюються на його користь в рамках договору підряду, не оподатковуватимуться ПДФО.

Вважатимемо, що усі програмісти працюють у штаті підприємства-розробника і їм встановлено певний посадовий оклад. Місячний оклад, денна заробітна плата, трудомісткість і основна заробітна плата кожного учасника техпроцесу представлено у таблиця 4.4.

Таблиця 4.4 – Сума часткових заробітних плат

	Місячний оклад, грн.	Денна зар. плата, грн	Трудомісткість, людино-дні		Основна заробітна плата, грн.	
			Процедурний підхід	Об'єктно-орієнтований підхід	Процедурний підхід	Об'єктно-орієнтований підхід
Керівник проєкту	18200	728	4	4	2912	2912
Інженер-програміст	15200	608	17	19	10336	11552
Інженер-тестувальник	12100	484	3	3	1452	1452
Дизайнер	7500	300	3	3	900	900
Всього			27 днів	29 днів	15600 грн.	16816 грн.

Визначимо витрати на основну заробітну плату. Вони складають суму заробітних плат за кожну із робіт. Витрати на основну зарплату для процедурного та об'єкт-орієнтованого підходів:

Додаткова заробітна плата обчислюється за формулою $ЗП_{дод}=ЗП_{осн}\cdot 0.2$.

Обчислимо витрати на додаткову зарплату для двох підходів:

$ЗП_{осн(п)} = 2912 + 10336 + 1452 + 900 = 15600$ (грн.);

$ЗП_{осн(о)} = 2912 + 11552 + 1452 + 900 = 16816$ (грн.);

Це сума часткових заробітних плат за кожну роботу, і вона приведена в таблиці 4.4.

Додаткова заробітна плата обчислюється за формулою:

$ЗП_{дод} = 0{,}2 \cdot ЗП_{осн}$.

$ЗП_{дод(п)} = 15600 \cdot 0.2 = 3120$ (грн.);

$ЗП_{дод(о)} = 16816 \cdot 0.2 = 3363.2$ (грн.);

Таким чином загальний фонд заробітної плати, що обчислюється за формулою:

$ФЗП = ЗП_{осн}+ЗП_{дод}$.

$ФЗП_{п} = 15600 + 3120 = 18720$ (грн.);

$ФЗП_{о} = 16816 + 3363.2 = 20179.2$ (грн.);

З цієї суми утримуються обов'язкові відрахування на заробітну плату: єдиний соціальний внесок, який складає 3,6% від суми нарахованої заробітної плати та податок на доходи фізичних осіб, який складає 15% від суми нарахованої заробітної плати, зменшеної на суму єдиного внеску на загальнообов'язкове соціальне страхування та податкової соціальної пільги.

Таким чином обов'язкові відрахування на заробітну плату для працівників складають:

Утримання єдиного соціального внеску:

$Відр_{ЄСВ1} = 0.036 \cdot ФЗП = 0.036 \cdot 18720 = 674$ грн.;

$Відр_{ЄСВ2} = 0.036 \cdot ФЗП = 0.036 \cdot 20179.2 = 726.5$ грн.

Податок на доходи фізичних осіб:

$Відр_{ПДФО1} = 0.15 \cdot (ФЗП - Відр_{ЄСВ}) = 0.15 \cdot (18720\text{-}674) = 2706.9$ грн.;

$Відр_{ПДФО2} = 0.15 \cdot (ФЗП - Відр_{ЄСВ}) = 0.15 \cdot (20179.2\text{-}726.5) = 2917.9$ грн.

$Відр_1 = Відр_{ЄСВ1} + Відр_{ПДФО1} = 674 + 2706.9 = 3380.9$ грн.;

$Відр_2 = Відр_{ЄСВ2} + Відр_{ПДФО2} = 726.5 + 2917.9 = 3644.4$ грн.

Законом № 1621 підрозділ 10 розділу XX Перехідних положень Податковий кодекс України від 02 грудня 2010 року № 2755-VI зі змінами та доповненнями (далі – ПКУ) доповнено пунктом 16, яким тимчасово, до 1 січня 2015 року, встановлено військовий збір.

Водночас Законом України від 28 грудня 2014 року № 71-VIII «Про внесення змін до Податкового кодексу України та деяких законодавчих актів України щодо податкової реформи» (далі – Закон № 71), який набрав чинності з 01.01.2015, оподаткування військовим збором подовжено до набрання чинності рішенням Верховної Ради України про завершення реформи Збройних Сил України (пункт 16 підрозділ 10 розділу XX Перехідних положень ПКУ).

Платниками збору в ромірі 1,5% від заробітної плати є особи, визначені пунктом 162.1 статті 162 цього ПКУ (підпункт 1.1 пункт 16 підрозділу 10 ПКУ).

Військовий збір з фізичних осіб:

$ВЗФО = 0{,}015 \cdot ФЗП$

$ВЗФО_1 = 0{,}015 \cdot 18720 = 280{,}8$ грн.

$ВЗФО_2 = 0{,}015 \cdot 20179.2 = 302{,}7$ грн.

Нарахування на фонд оплати праці, які включають відрахування до Пенсійного фонду, фонду з тимчасової втрати працездатності, фонду з безробіття і фонду страхування від нещасних випадків на виробництві; для бюджетної організації тариф на фонд оплати праці встановлено на рівні 36,3%, для підприємців розмір єдиного внеску залежно від класу професійного ризику виробництва становить від 36,76 % до 49,70 %. Зокрема, видання програмного забезпечення – 36,77%).

Нарахування на фонд оплати праці (*ФОП*):

$ФОП_{ЄСВ} = 0.3677 \cdot ФЗП$

$ФОП_{ЄСВ1} = 0.3677 \cdot 18720 = 6883.34$ грн.;

$ФОП_{ЄСВ1} = 0.3677 \cdot 20179.2 = 7419.82$ грн.

Всього витрат:

$В_{ЗП1} = ЗП_1 + ФОП_{ЄСВ1} = 18720 + 6883.34 = 25603.34$ грн.;

$В_{ЗП2} = ЗП_2 + ФОП_{ЄСВ2} = 20179.2 + 7419.82 = 27599.02$ грн.

Матеріальні витрати визначаються як добуток кількості витрачених матеріалів та їх ціни:

$$M_{Bi} = q_i \cdot p_i$$

де q_i – кількість витраченого матеріалу *i*-го виду; p_i – ціна матеріалу *i*-го виду.

Звідси, загальні матеріальні витрати можна визначити:

$$З_{м.в.} = \sum M_{B\,i}$$

Проведені розрахунки занесемо у таблицю 4.5:

Таблиця 4.5 – Зведенні розрахунки матеріальних витрат

№ п/п	Найменування матеріальних ресурсів	Од. виміру	Фактично витрачено матеріалів	Ціна 1-ці, грн.	Загальна сума витрат, грн.
1	SCRUM-дошка	шт	1	500,00	500,00
2	Папір для друку	листів	120	0,25	30,00
3	Чорнила для принтера	шт	1	65,00	65,00
4	Маркери	шт	4	14,00	56,00
Всього					651,00

Згідно постанов Національної комісії, що здійснює державне регулювання у сфері енергетики (згідно Постанов Національної комісії, що здійснює державне регулювання у сфері енергетики (НКРЕ) від 22.01.2001р. №47 зі змінами і доповненнями, від 06.12.2002р. № 1358, від 22.10.2004р. № 1030, від 28.04.2016р. № 755 на підставі Закону України від 03.12.1999р № 1274-XIV “Про внесення змін до Закону України “Податкового кодексу України”, згідно Закону України “Про міський електричний транспорт” від 29 червня 2004 року № 1914 – IY; також затвердженого постановою Кабінету Міністрів України від 01.06.2011р. № 869, Порядку розрахунку роздрібних тарифів на електричну енергію, тарифів на розподіл електричної енергії (передачу електричної енергії місцевими (локальними) електромережами), тарифів на постачання електричної енергії за регульованим тарифом, затвердженого постановою Національної комісії, що здійснює державне регулювання в сфері енергетики та комунальних послуг (НКРЕКП) "Про ринкове формування роздрібних тарифів на електричну енергію, що відпускається для кожного класу споживачів, крім населення, на території України", ця сума становить 289,85 коп/кВт.г.

Витрати на електроенергію одиниці обладнання визначаються за формулою:

$$З_е = W \cdot T \cdot S$$

де W – необхідна потужність, кВт; T – кількість годин роботи обладнання; S – вартість кіловат-години електроенергії.S=2,90грн/кВт.г.

1 $З_{е1} = 1{,}5 \cdot 216 \cdot 2{,}9 = 939{,}6$ грн.;

2 $З_{е2} = 1{,}5 \cdot 232 \cdot 2{,}9 = 1009{,}2$ грн.

Розрахунок суми амортизаційних відрахувань.

Комп'ютери та оргтехніка належать до четвертої групи основних фондів. Для цієї групи річна норма амортизації дорівнює 60 % (квартальна – 15 %).

Амортизація за час (період) використання обладнання (комп'ютера) складає долю затрат, які припадають на дослідницьку роботу (написання програми), і визначається:

$$A = \frac{C_Б \cdot N_A \cdot T_{ФАК}}{T_{ГОД}}$$

де $C_Б$ – балансова вартість обладнання, грн; N_A – норма амортизаційних відрахувань в рік, %; $T_{ГОД}$ – річний робочий фонд часу, год; $T_{ФАК}$ – фактичний час роботи обладнання по написанню програми, год.

Таблиця 4.6 – Перелік обладнання, необхідного для розробки програми

Найменування	Кількість, шт.	Ціна за одиницю продукту, грн.	Сума, грн.
1	2	3	4
Комп'ютер	4	15775	63100
Принтер	1	2000	2000
Apple Developer Program Membership	1	3250	3250
Google Play Developer Membership	1	900	900

Продавження таблиці 4.6

1	2	3	4
Хостинг	1	995	995
Доменне ім'я	1	366	366
Всього			70611

$$A_1 = \frac{70611 \cdot 0{,}6 \cdot 216}{2016} = 4539{,}3\,\text{грн.};\ A_2 = \frac{70611 \cdot 0{,}6 \cdot 232}{2016} = 4875{,}52\,\text{грн.}$$

Накладні витрати пов’язані з обслуговуванням виробництва, утриманням апарату управління підприємства (фірми) та створення необхідних умов праці.

Залежно від організаційно-правової форми діяльності господарюючого суб’єкта, накладні витрати можуть становити 20–60 % від суми основної та додаткової заробітної плати працівників.

$НВ = 0{,}5 \cdot ЗП_{осн}$.

$НВ_п = 0{,}5 \cdot 15600 = 7800$ грн.;

$НВ_о = 0{,}5 \cdot 16816 = 8408$ грн.

Проведемо розрахунок вартості створюваного програмного продукту. Вартість продукції включає у собі собівартість і планований прибуток.

Повна собівартість програмного продукту дорівнює сумі усіх витрат на його виробництво:

$Св = В_{зп} + З_{мв} + З_е + А + НВ;$

$Св_1 = 25603{,}3 + 651 + 939{,}6 + 4539{,}3 + 7800 = 38882{,}2$ (грн.)

$Св_2 = 27599{,}02 + 651 + 951{,}8 + 4875{,}52 + 8408 = 41891{,}74$ (грн.)

Приймемо прибуток на рівні 27%. Для нових інноваційних продуктів, що користуються високим попитом на ринку.

Отже, вартість розробленого програмного забезпечення:

$В_1 = Св_1 + 0,27 \cdot Св_1 = 38882,24 + 0,27 \cdot 38882,24 = 49380,44$ грн.;

$В_2 = Св_2 + 0,27 \cdot Св_2 = 41891,74 + 0,27 \cdot 41891,74 = 53202,51$ грн.

Економічна ефективність (*Е*) полягає у відношенні результату виробництва до затрачених ресурсів:

$$E = \frac{П}{Св}$$

де *П* – прибуток, *П= В - Св*; *Св* – собівартість.

Якщо ринкова вартість програмного продукту рівна прийнятій, то економічна ефективність визначається встановленим рівнем прибутку.

$E_п = (49380,44 - 38882,24) / 38882,24 = 0,27;$

$E_о = (53202,51 - 41891,74) / 41891,74 = 0,27.$

Поряд із економічною ефективністю розраховують термін окупності капітальних вкладень ($T_{ок}$):

$$T_{ок} = \frac{1}{E}$$

У нашому випадку $T_{ок1} = T_{ок2} = 1/0,27 = 3,7$ років, що є нормальним, оскільки допустимим вважається термін окупності до 5 років.

Загальна вартість пропонованих робіт по розробці програмного продукту становить 18720 грн. для першого варіанту та 20179,2 грн. для другого. Оскільки ефективність для обидвох проєктів відповідно до встановленого рівня прибутку становить 0,27, що є високим

показником, то проводити дані роботи варто і вкладені кошти окупляться за 3,7 року.

4.3 Визначення витрат на супровід і модернізацію проґрамного продукту та уточнений аналіз ефективності вкладених інвестицій

Виходячи із експертних оцінок і складності проґрами, приймемо величину витрат на супровід і модернізацію проґрамного забезпечення, створеного за процедурним методом 55% від початкових витрат, а за об'єктно-орієнтованим – 25%.

Собівартість модернізації:

$СвМ_п = 0,6 \cdot Св_п = 0,6 \cdot 38882,24 = 23329,35$ грн.,

$СвМ_о = 0,2 \cdot Св_о = 0,2 \cdot 41891,1 = 8378,22$ грн.

Для споживача вартість модернізації:

$М_п = 0,6 \cdot В_п = 0,6 \cdot 47719,7 = 29628,27$ грн.,

$М_о = 0,2 \cdot В_о = 0,2 \cdot 51710,7 = 10640,50$ грн.

Таким чином, уже після першої модернізації, загальні витрати на створення і супровід ПЗ для виробника за об'єктно-орієнтованим методом менші, ніж за процедурним, навіть якщо його собівартість є дещо дорожчою.

$ЗВ_{п\ (вир)} = 38882,24 + 23329,35 = 62211,59$ грн.,

$ЗВ_{о\ (вир)} = 41891,74 + 8378,22 = 50269,96$ грн.

Як і для споживача:

$ЗВ_п = 49380,44 + 29628,27 = 79008,71$ грн.,

$ЗВ_о$ = 53202,51 + 10640,50 = 63843,01 грн.

Річна економія витрат за всіма можливими напрямами і додатковими витратами, пов'язаними з супроводом і тільки одноразовою модернізацією (у розрахунку на одиницю продукції) при об’єктно-орієнтованому методі порівняно із процедурним:

$\Delta C_{(вир)} = ЗВ_{1(вир)} - ЗВ_{2(вир)} = 62211,59 - 50269,96 = 11941,63$ грн,

$\Delta C = ЗВ_1 - ЗВ_2 = 79008,27 - 63843,01 = 15165,70$ грн,

Визначення ЧПД відбувається за формулою:

$$ЧПД = \sum_{i=1}^{t} ГП_i \alpha_{TBi} - \sum_{i=1}^{t} ІК_i \alpha_{TBi};$$

де $ГП_i$ – грошовий потік і-го розрахункового року;

$ІК_i$ – сума інвестицій і-го розрахункового року;

α_{TBi} – коефіцієнт дисконтування (коефіцієнт приведення інвестицій і грошового потоку до теперішньої вартості).

Для розрахунку коефіцієнта дисконтування (коефіцієнта приведення) грошових потоків за роками періоду економічного життя інвестицій використовується формула:

$$\alpha = \frac{1}{(1+i)^n};$$

де i – ставка дисконтування або норма дисконту, i = 0,27;

n – час або кількість періодів (років), протягом якого планується отримання доходу.

$$\alpha_0 = 1, \quad \alpha_1 = \frac{1}{(1+0,27)} = 0,79$$

Вважатимемо, що обидва програмних продукта одинаково забезпечують потреби і вимоги споживача, і тому придбання першої чи другої програми одинаково вплинуть на розмір його додаткових доходів на вкладений капітал, Тому приймемо цю величину за

постійну, а порівняння дохідності двох проєктів проведемо тільки за витратами.

$$ЧПД'_1 = ГП + 0{,}78 \cdot ГП - 47719{,}7 - 0{,}78 \cdot 28631{,}8 = 1{,}78ГП - 70052{,}5 \text{ грн.},$$

$$ЧПД'_2 = ГП + 0{,}78 \cdot ГП - 51710{,}7 - 0{,}78 \cdot 10342{,}14 = 1{,}78ГП - 59777{,}57 \text{ грн.}$$

Економія витрат у випадку придбання, супроводу і одноразової модернізації програмного продукту, створеного за об'єктно-орієнтованим підходом, становить 10274,93 грн. Отже, сучасну комп'ютеру програму доцільно виконувати по об'єктно-орієнтованій парадигмі,

Усі резутати розрахунків, приведених у даному розділі, зведені у таблицю 4.7

Таблиця 4.7 – Результати розрахунків

№ п.п.	Назва	Процедурний підхід	Об'єкто-орієнтований підхід
1	2	3	4
1	Зарплата основна, грн	15600	16816
2	Зарплата додаткова, грн	3120	3363.2
3	Фонд заробітньої плати (1+2)	18720	20179.2
4	Обов'язкові відрахування на заробітну плату (4.1+4.2), грн	3480.9	3644.4
4.1	Утримання єдиного соціального внеску (3,6%), грн	674	726.5
4.2	Податок на доходи фізичних осіб (15%), грн	2806.9	2917.9
5	Військовий збір (1.5%), грн	280.8	302.7
6	Відрахування на ФОП (36,77%), грн	6883.34	7419.82
7	Разом на оплату праці (3+6), грн	25603.34	27599.02
8	Матеріальні витрати, грн	651	651
9	Електроенергія, грн	939.6	1009.2

Продовження таблиці 4.7

1	2	3	4
10	Амортизація, грн	4539.3	4875.52
11	Накладні витрати, грн	7800	8408
12	Разом на ін. витрати (8+9+10+11)	13929.9	14943.72
13	Собівартість, грн	38882.24	41891.74
14	Прибуток, грн	10498.20	11310.77
15	Вартість розробленого ПЗ, грн	49380.44	53202.51
16	Економічна ефективність	0.27	0.27
17	Термін окупності, років	3.7	3.7
18	Собівартість модернізації, грн	23329.35	8378.22
19	Супровід і модернізація, грн	29628.27	10640.50

Загальна вартість пропонованих робіт по розробці програмного продукту становить 79008,71 грн. для першого варіанту та 63843,01 грн. для другого. Оскільки ефективність для обидвох проєктів відповідно до встановленого рівня прибутку становить 0,27, що є високим показником, то проводити дані роботи варто і вкладені кошти окупляться за 3,7 року.

При використанні об’єктно-орієнтовного підходу зменшується кількість працівників, які залучаються у проєкт, та зменшуються витрати на реалізацію проєкту, але для підтримки проєкту і його подальшої модернізації все ж в майбутньому потрібні набагато більші витрати. Для функціонального підходу потрібна більша кількість працівників, часу і коштів, але на модернізацію і підтримку в загальному потрібно менше ресурсів, і у висновку програма виконана по функціональній парадигмі стає дешевшою для споживача, і приносить економію ресурсів для розробників.

5 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

5.1 Профілактика нещасних випадків, професійних захворювань і отруєнь на виробництві

Основним завданням профілактики нещасних випадків на виробництві є інформаційна та методична підтримка при розробці роботодавцями заходів, направлених на поліпшення умов праці, зниження рівня травматизму, а також поширення позитивного досвіду щодо створення здорових і безпечних умов праці.

Для створення безпечних та здорових умов праці роботодавець повинен забезпечити функціонування на підприємстві системи управління охороною праці та очолити роботу з управління охороною праці.

Роботодавець повинен забезпечити суворе дотримання працівниками вимог виробничої та трудової дисципліни, за необхідності, використовувати заходи матеріального та адміністративного впливу до порушників трудової та виробничої дисципліни. Також повинен постійно проводити навчання з питань охорони праці, а також різноманітні інструктажі.

Відповідно до пункту 6.6 “Типового положення про порядок проведення навчання і перевірки знань з питань охорони праці”, затвердженого наказом Державного комітету України з нагляду за охороною праці від 26.01.05 №15, позаплановий інструктаж проводиться з працівниками на робочому місці або в кабінеті охорони праці: при введенні в дію нових або переглянутих нормативно – правових актів з охорони праці, а також при внесенні змін та доповнень до них; при порушеннях працівниками вимог

нормативно – правових актів з охорони праці, що призвели до травм, аварій, пожеж тощо.

Також інструктажі можуть проводитися при зміні технологічного процесу, заміні або модернізації устаткування, приладів та інструментів, вихідної сировини, матеріалів, що впливають на стан охорони праці; при перерві в роботі виконавця робіт більш ніж на 30 календарних днів – для робіт з підвищеною небезпекою, а для решти робіт – 60 днів.

Позаплановий інструктаж з учнями, студентами, курсантами, слухачами проводиться під час проведення трудового і професійного навчання при порушеннях ними вимог нормативно – правових актів з охорони праці, що можуть призвести або призвели до травм, аварій, пожеж тощо.

До організаційних причин травматизму, професійних захворювань та отруєнь належать: недостатній нагляд за дотриманням правил техніки безпеки, відсутність необхідної технічної кваліфікації у персоналу, робота без запобіжних засобів, неправильне розташування людей на робочому місці, недостатній інструктаж, погане освітлення, низька або висока температура, слабка вентиляція тощо.

До технічних причин належать: несправність або недосконаліть технологічного обладнання, інструментів, пристроїв і засобів техніки безпеки, незручність або захаращеність робочого місця.

Причиною травми може стати також хворобливий стан працюючого, непідготовленість до даної роботи і неуважне ставлення до неї, втома і стан сп'яніння.

Статистика свідчить про те, що більшість усіх нещасних випадків соціально зумовлені або є наслідком психофізіологічних якостей та особистісних особливостей персоналу, який здійснює

трудову діяльність, а причиною травматизму виступають небезпечні дії працівників. При цьому людський фактор у безпеці праці стає переважно визначальним.

Соціальні та особистісні чинники, що впливають на охорону праці охоплюють широке коло питань, форм і методів роботи. Урахування індивідуальних особистісних відмінностей має велике значення для формування трудових колективів (бригад, змін).

Розслідування та облік нещасних випадків відбувається відповідно до "Положення про порядок розслідування й обліку нещасних випадків на виробництві". Це положення встановлює єдиний порядок розслідування й обліку нещасних випадків на виробництві.

За результатами розслідування складається висновок, що є обов'язковим для роботодавця і може бути оскаржений в органах державної інспекції праці або в суді.

Для профілактики роботодавець повинен проводити технічні, нормативно-методичні, санітарно-гігієнічні, організаційні, соціально-економічні, лікувально-профілактичні заходи.

До технічніх заходів відносять засоби, що забезпечують безпечні і нешкідливі умови праці, та пов'язані з впровадженням нового обладнання, пристроїв і приладів безпеки і безпечною експлуатацією засобів виробництва.

До нормативно-методичних заходів належать: розробка посібників і рекомендацій; розробка нормативно-правової бази з охорони праці на підприємстві; розробка розділів охорони праці в посадових інструкціях; забезпечення необхідною нормативно-правовою документацією.

До організаційні заходів належать: контроль за технічним станом обладнання, інструментів, будівель і споруд; контроль за дотриманням вимог нормативних документів з охорони праці;

нагляд за обладнанням підвищеної небезпеки; організація навчання, перевірка знань з питань охорони праці і інструктажів робітників підприємства; контроль за виконанням технологічного процесу відповідно до вимог охорони праці; організація належних умов до проїздів і проходів відповідно до вимог охорони праці; забезпечення працівників засобами індивідуального та колективного захисту; забезпечення відповідними знаками безпеки, плакатами.

До санітарно-гігієнічних заходів належать: контроль за впливом виробничих факторів на здоров'я працівників; забезпечення санітарно-побутових умов згідно з діючими нормами; атестація робочих місць відповідно до їх нормативним актам з охорони праці; планування заходів щодо поліпшення санітарно-гігієнічних умов праці.

До соціально-економічних заходів належать: надання пільг і компенсацій працівникам, які працюють зі шкідливими і небезпечними умовами праці; соціальне страхування працівників роботодавцем; фінансування заходів з охорони праці; відшкодування роботодавцем працівнику збитків у разі каліцтва.

До лікувально-профілактичних заходів: надання медичної допомоги потерпілим від нещасних випадків на виробництві; контроль за здоров'ям працюючих протягом їхньої трудової діяльності; лікувально-профілактичне харчування працівників, які працюють на роботах зі шкідливими і небезпечними умовами праці; проведення медичних оглядів працівників (попередніх та періодичних); дотримання охорони праці жінок, неповнолітніх та інвалідів; відшкодування потерпілому працівнику витрат на лікування, протезування, придбання транспортних засобів та інші види медичної допомоги.

При аналізі виробничого травматизму й професійних захворювань застосовують статистичний, монографічний та інші методи. Статистичний метод заснований на вивченні причин травматизму по документах, що реєструють уже минулі за визначений період нещасні випадки, професійні отруєння чи захворювання (акти за формою Н-1).

Топографічний метод полягає у вивченні причин нещасних випадків за місцем їх події, а монографічний метод виробничого травматизму містить у собі детальне дослідження всього комплексу умов, завдяки яким стався нещасний випадок.

5.2 Вибір схеми пристрою захисного відключення від ураження електричним струмом для електроустановки напругою до 1000 В, що проєктується

Захисне відключення — швидкодіючий захист, що забезпечує автоматичне відключення електроустановки при виникненні в ній небезпеки ураження людини струмом. Така небезпека може виникнути при замиканні фази на корпус, зниженні опору ізоляції мережі нижче визначеної межі і, нарешті, в разі дотику людини безпосередньо до струмопровідної частини, що перебуває під напругою.

Захисне відключення застосовується в тих випадках, коли інші захисні заходи (заземлення, занулення) є ненадійними, складно здійснюваними (в умовах вічної мерзлоти та ін.), багато коштують або коли до безпеки обслуговування ставляться підвищені вимоги (у шахтах, кар'єрах), а також у пересувних електроустановках. Зона застосування пристроїв захисного відключення практично не обмежена, вони придатні для мереж будь-якої напруги і з будь-яким режимом нейтралі. Проте найбільшого поширення пристрої

захисного відключення набули в мережах до 1000 В (із заземленою й ізольованою нейтраллю). Крім того, захисне відключення є незамінним для ручних електроінструментів.

В усіх цих випадках небезпека ураження зумовлена напругою дотику $U_{\text{дот}}$ або струмом, що проходить крізь людину: $U_{\text{дот}} = I_h R_h$. Основними елементами пристроїв захисного відключення є прилади захисного відключення й автомати. Прилад захисного відключення складається з окремих елементів, що сприймають вхідну величину, реагують на її зміни і при заданому її значенні дають сигнал на відключення вимикача. Цими елементами є: датчик — вхідний пристрій (як правило, реле відповідного типу); підсилювач, що підсилює сигнал датчика; коло контролю; допоміжні елементи (сигнальні лампи і вимірювальні прилади - омметри тощо).

Основні вимоги, які ставляться до пристроїв захисного відключення, такі: висока чутливість; незначний час відключення; селективність дії; здатність здійснювати самоконтроль справності; достатня надійність.

Залежно від прийнятих вхідних (контрольованих) величин пристрої захисного відключення умовно поділяються на типи, які: реагують на потенціал (напругу) корпуса щодо землі; на струм замикання на землю; напругу нульової послідовності; струм нульової послідовності; напругу фази щодо землі; оперативний струм; а також вентильні схеми. Розглянемо деякі з названих типів пристроїв захисного відключення (див. рисунок 5.1).

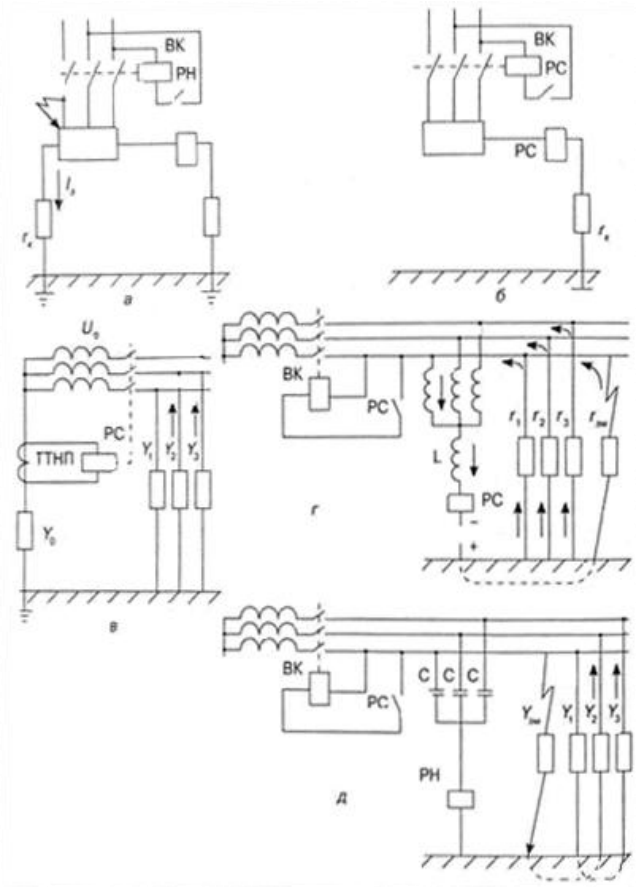

Рисунок 5.1 – Схеми пристроїв захисного відключення

де:

а – реагують тільки на потенціал корпусу;

б – тільки на струм замикання на землю;

в – тільки на напругу нульової послідовності;

г – тільки на струм нульової послідовності;

д – оперативний струм реле витоку.

Пристрої, що реагують на потенціал корпусу. Призначення цих пристроїв захисного відключення - усунення небезпеки ураження людей струмом при виникненні на заземленому чи зануленому корпусі підвищеного потенціалу. Принцип дії: швидке відключення від мережі пошкодженого устаткування, якщо потенціал $\varphi_к$, який виник на його корпусі, виявиться вище потенціалу $\varphi_{к.доп}$, при якому напруга дотику до корпусу має

якнайтриваліше допустиме значення. На рисунку 5.1а показано принципову схему такого пристрою, в якому датчиком служить реле максимальної напруги, включене між корпусом, що захищається, і допоміжним заземлювачем (безпосередньо або через трансформатор напруги). Електроди допоміжного заземлювача мають бути розміщені поза зоною розтікання струмів, які стікають із заземлювача корпусу або заземлювачів нульового провідника мережі. Коли відбувається пробій фази на заземлений чи занулений корпус, спочатку виявляється захисна властивість заземлення (чи занулення), що знижує потенціал корпуса до деякої межі. Якщо $\varphi_{к}$ перевищить $\varphi_{к.доп}$, спрацює пристрій захисного відключення, тобто відбудеться відключення пошкодженої установки від мережі.

Пристрій, що реагує на струм замикання на землю. Призначення - усунення небезпеки ураження струмом людей при дотику до заземленого корпусу в момент замикання на нього фази. Принцип дії: швидке відключення пошкодженого устаткування від мережі в разі, якщо струм, який проходить через провідник, що заземлює корпус цього устаткування, перевищить деяке граничне значення $I_{з.доп}$, при якому напруга дотику має найбільше тривало допустиме значення $U_{дот.доп}$ (див. рисунок 5.1б). Тут датчиком служить струмове реле, що має малий опір і включене безпосередньо у розсічку заземлювального проводу або у вторинну обмотку трансформатора струму, що застосовується при великому струмі замикання на землю. При замиканні фази на корпус струм, що стікає в землю, якщо він перевищує уставку захисту, викликає спрацювання реле, тобто відключення установки від мережі. У схемах, застосовуваних у системах занулення, струмове реле включається в розсічку занулювальних провідників і спрацьовує під

дією однофазового короткого замикання. Такі пристрої відрізняються чіткістю спрацьовування.

Пристрої, що реагують на напругу нульової послідовності. Призначення цих пристроїв захисного відключення - усунення ураження струмом, що виникає при глухому замиканні однієї або двох фаз на землю, у тому числі при замиканні фази на заземлений корпус. Принцип дії: швидке відключення мережі від джерела живлення при виникненні напруги нульової послідовності, тобто несиметрії повних провідностей проводів мережі щодо землі вище деякої межі (див. рисунок 5.1в). Тут датчиком служить фільтр напруги нульової послідовності, що складається з трьох конденсаторів, поєднаних у зірку. Реле напруги, включене між нульовою точкою фільтра і землею, спрацьовує, коли напруга нульової послідовності (тобто напруга між нейтральною точкою джерела струму і землею) U_0 досягає значення, при якому напруг на затискачах реле стає рівною чи перевищує напругу його спрацьовування $U_{сл}$. При цьому відбувається відключення мережі від джерела. Зона застосування таких пристроїв захисного відключення - трифазні трипроводові мережі до 1000 В з ізольованою нейтраллю і малою довжиною, яким властиві високий опір ізоляції і невелика ємність щодо землі.

Пристрої, що реагують на струм нульової послідовності. Призначення пристроїв захисного відключення цього типу — забезпечити безпеку людини у разі дотику до заземленого (зануленого) корпусу при замиканні на нього фази або до струмопровідної частини, що перебуває під напругою. Принцип дії: швидке відключення ділянки чи мережі споживача енергії, якщо струм нульової послідовності перевищує деяке значення, при якому напруга дотику до "пробитого" корпусу або струмопровідної частини, що перебуває під напругою, має найбільше тривало

допустиме значення $U_{\text{дот.доп}}$. Тут датчиком може служити фільтр струму нульової послідовності, що є трьома однотипними трансформаторами струму, установлених на всіх фазах мережі. Однойменні затискачі їх вторинних обмоток з'єднані паралельно, і до них підключена обмотка струмового реле. У результаті через реле проходить струм, який дорівнює геометричній сумі вторинних струмів трансформаторів. Цей струм, досягши значення струму спрацювання реле або перевищивши його, викликає відключення ділянки мережі, що захищається, від джерела живлення (див. рисунок 5.1г).

Пристрої, що реагують на оперативний струм (див. рисунок 5.1д). Призначення цього пристрою захисного відключення, що реагує на оперативний струм, - забезпечувати безперервний автоматичний контроль опору ізоляції мережі, а також захист людини, яка торкнулася до струмопровідної частини, від ураження струмом. Отже, цей тип пристрою захисного відключення (реле витоку) може служити самостійним заходом захисту від ураження струмом при дотику до "пробитого" незаземленого і незануленого корпусу або до струмопровідної частини, що перебуває під напругою. Він може також служити додатковим захисним заходом до захисного заземлення (цей принцип використовується в шахтах, кар'єрах тощо). Принцип дії: швидке відключення мережі від джерела струму при зниженні опору ізоляції мережі щодо землі нижче деякого граничного значення, при якому струм (або напруга дотику) через людину, яка торкнулася до струмопровідної частини, досягає найбільшого тривало допустимого значення $U_{\text{дот.доп}}$. Тут датчиком служить реле з малим струмом спрацьовування (кілька міліампер). Застосовуються й інші схеми захисного відключення.

ВИСНОВКИ

Суспільство щохвилини розвивається та прогресує. Не викликає сумніву й той факт, що кожний наступний день вже ніколи не буде таким як попередній. Те що було новаторством ще буквально вчора, сьогодні вже є відверто вчорашнім днем. Ми ніколи не замислюємось, а, можливо й варта, що ще якихось 50 років тому навіть не в кожній оселі був телевізор, а новини, в більшості, населення дізнавалось із радіопередавачів, які намагались не вимикати. Сьогодні ж, щоб бути успішним, варта не просто замислитись, але й натхненно працювати в напрямку самовдосконалення, самоосвіти та технічного розвитку. Вже очевидним є й те, що без постійної генерації та реалізації нових задумів та ідей з використанням сучасних, зокрема інформаційних, технологій не можливо бути в тренді у якості успішного, ефективного та прогресивного члена «елітного клубу» сучасного світу.

Життєвий цикл будь якого винаходу розпочинається від зародження ідеї, яка виникає з потреби, і при серйозності намірів, яка є комерційною таємницею, аж до моменту, коли тенденція до розвитку вимагає подальшого вдосконалення з врахуванням стану сучасного етапу розвитку науки і техніки. Однак, на якомусь етапі таємниця перестає бути такою і таким чином людство постійно розвивається і колишні винаходи стають загальним надбанням цивілізації. Саме завдяки цьому, приступаючи до розробки чогось нового, варта проаналізувати стан речей, вивчити аналоги та не витрачати на марно свій час розробляючи вже відомі речі, а займатися, безпосередньо щонайменше, вдосконаленням того, що є вимогою сьогодення.

Дипломна робота складається з вступу, чотирьох розділів, висновку, списку літератури та додатків.

В результаті виконання огляду та порівняльного аналізу систем оптової торгівлі встановлено, що вони у своїй сукупності не володіють достатнім рівнем гнучкості. Усунення даної проблеми – сформувало основу концепції магістерської роботи.

Розроблено клієнт-серверну програмну систему: проаналізовано вимоги до програмної системи, проаналізовано предметну область виявлено актантів та їхні варіанти використання, спроєктовано та змодельовано архітектуру системи, програмно реалізовано класи та методи, що разом дало змогу розробити клієнт-серверну програмну систему для організації автоматизації мобільної діяльності.

Проведено тестування програмної системи, що дало змогу перевірити на практиці роботу клієнт-серверної програмної системи для організації автоматизованого передпродажу.

Проведено підрахунки собівартості реалізації проєкту та собівартості його модернізації. Порівняно витрати при використанні різних підходів до розробки програмного забезпечення.

Розробку та роботу в цілому виконано з врахуванням норм охорони праці, а також правил безпеки життєдіяльності.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Господарський кодекс України № 436-IV від 16.01.2003 [Електронний ресурс]. – Режим доступу: http://zakon4.rada.gov.ua/laws/show/436-15. – Назва з екрану.

2. Павленко А. Ф., Войчак А. В., Кардаш В. Я., Пилипчук В. П. та ін. Теорія та практика маркетингу в Україні: Монографія. — К.: КНЕУ, 2005. — 584 с.

3. Гурч Л1.М. Логістика: Навч. посіб. для студ. вищ. навч. закл. / Гурч Людмила Миколаївна — К : ДП «Видавничий дім «Персонал», 2008. — С. 365 (560 с.)

4. А.А.Кириллова. Финансовая логістика / А.А.Кириллова // Бухучет в строительных организациях, 2011, № 2- С . 72-76.

5. Концева В. В., Костенко С.С. Фінансові потоки в логістичних системах / В.В. Концева, С.С. Костенко // [Електронний ресурс]. - Режим доступу: www.nbuv.gov.ua/ portalAratural/Vntu/2009_19_l/pdf/85.pdf.

6. Дашков Л. П. Коммерция и технология торговли. / Л. П. Дашков, В. К. Памбухчиянц. – М. : Информ.-внедрен. центр «Маркетинг», 2007. – 448 с.

7. Мазаракі А.А., Лігоненко Л.О., Ушакова Н.М. Економіка торговельного підприємства: підручник / За ред. Н.М. Ушакової. — К.: Хрещатик, 1999. — 800 с.

8. Комерційне підприємництво: сучасний стан, стратегії розвитку [Електронний ресурс]. – Режим доступу : http://www.big-library.com.ua/book/. – Назва з екрану.

9. Лісіца В.В. Спеціалізація роздрібної торговельної мережі: стан та шляхи подальшого розвитку в умовах фор" мування ринкового середовища // Науковий вісник Пол" тавського університету

споживчої кооперації України. Серія: Економічні науки. — 2010. — №1 (5). — С. 83—88.

10. Копійка В.В., Шинкаренко Т.І. Європейський Союз: заснування і етапи становлення. — К., Видавни" чий Дім "Ін Юре", 2001. — 448 с.

11. Балабанов И. Т. Интерактивний бизнес. — СПб.: Питер, 2001. — 128 с.

12. Агент Плюс: Мобільна торгівля [Електронний ресурс] – Режим доступу: URL: http://www.agentplus.com.ua/mobiletrade – Заголовок з екрану.

13. SalesWorks Professional [Електронний ресурс] – Режим доступу: URL: http://www.softservebs.com/uk/products/salesworks/professional/ – Заголовок з екрану.

14. Skyriver MT [Електронний ресурс] – Режим доступу: URL: http://skyfleet.com.ua/kontrol-avtotransporta/mobilnaya-torgovlya/ – Заголовок з екрану.

15. 1С: Мобільна торгівля [Електронний ресурс] – Режим доступу: URL: http://solutions.1c.ru/mobile_trade/features – Заголовок з екрану.

16. Агент Плюс: Мобільна торгівля [Електронний ресурс] – Режим доступу: URL: http://www.agentplus.com.ua/mobiletrade – Заголовок з екрану.

17. Hermes [Електронний ресурс] – Режим доступу: URL: http://hermes.cn.ua – Заголовок з екрану.

18. Методичні вказівки до виконання магістерської роботи освітнього рівня — магістр студентами усіх форм навчання для напряму підготовки 121 – ―Інженерія програмного забезпечення‖ / Укладачі : Петрик М.Р., Михалик Д.М., Кінах Я.І., Гладьо С.В.,

Цуприк Г.Б. – Тернопіль : Вид-во ТНТУ імені Івана Пулюя, 2016 – 26 с.

19. Guckkenheimer S., Peter J. Software Engineering With Microsoft Visual Studio. Team System. – Adison Wesley, 2006. – 273 p.

20. Андон Ф., Лаврищева М. Методы инженерии распределенных компьютерных систем. К., 1997.

21. Р. Круз. Структуры данных и проєктирование программ. М:, БИНОМ. Лаборатория знаний. 2008.

22. Нормалізація баз даних [Електронний ресурс] – Режим доступу: URL: https://uk.wikipedia.org/wiki/Нормалізація_баз_даних/ – Заголовок з екрану.

23. Unified Modeling Language [Електронний ресурс] – Режим доступу:URL: https://uk.wikipedia.org/wiki/Unified_Modeling_Language/ - Заголовок з екрану.

24. Мейер Бертран. Объектно-ориентированное конструирование програмных систем. пер. с анг. – М.: 2005.

25. Java [Електронний ресурс] – Режим доступу : URL: http://uk.wikipedia.org/wiki/Java/ – Заголовок з екрану.

26. Python [Електронний ресурс] – Режим доступу : URL: https://uk.wikipedia.org/wiki/Python – Заголовок з екрану.

27. SQLite [Електронний ресурс] – Режим доступу : URL: https://uk.wikipedia.org/wiki/SQLite – Заголовок з екрану.

28. Double fork magic как заклинание для вызова демонов и борьбы с зомби [Електронний ресурс] – Режим доступу : URL: http://verber.kh.ua/node/834 – Заголовок з екрану.

29. Розгортання ПЗ [Електронний ресурс] – Режим доступу : URL: https://uk.wikipedia.org/wiki/Розгортання_програмного_забезпечення – Заголовок з екрану.

30. Тестування програмного забезпечення [Електронний ресурс] – Режим доступу : URL: http://www.znannya.org/?view=software-testing – Заголовок з екрану.

31. Методичні вказівки для виконання розділу магістерської роботи щодо техніко-економічного обґрунтування вибору проєктного рішення розробки та оцінки якості програмного забезпечення / Упор. Петрик М.Р., Кінах Я.І., Головатий А.І., Рогатинська Л.Р. – Тернопіль: Вид-во ТНТУ ім. І. Пулюя. – 2013. – 34 с.

32. Закон України "Про охорону праці". - С. 11, р. IV.

33. Типове положення про кабінет охорони праці. Затверджено наказом Державного Комітету України з нагляду за охороною праці від 18.07.97. - № 191.

34. Бедрій ЯЛ. Охорона праці. - К.: ЦУЛ, 2002. - С. 29, 30.

35. Гандзюк М.П. Основи охорони праці. - К.: "Каравела", 2003. С. 26.

36. Грибан В.Г. Охорона праці. - К.: "Центр учбової літератури", 2011. - С. 49, 50.

37. Жидецький В.Ц. Основи охорони праці. - Львів: Афіша, 2002. - С. 51,52, 80.

ДОДАТКИ

ДОДАТОК А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ПУЛЮЯ

Кафедра “ Програмної інженерії ”

ЗАТВЕРДЖУЮ
Завідувач кафедри
програмної інженерії
“___“ _____ 2019 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання дипломної роботи
на здобуття освітнього ступеня «магістр»
за спеціальністю 121 – «Інженерія програмного забезпечення» на тему

«РОЗРОБКА КЛІЄНТ-СЕРВЕРНОГО ДОДАТКУ ДЛЯ ОПЕРАЦІЙНОЇ СИСТЕМИ ANDROID З ВИКОРИСТАННЯМ JAVA ТЕХНОЛОГІЙ»

Керівник роботи
к.т.н., доц. каф. ПІ Цуприк Г.Б.
“__” _____ 2019 р.
_____ /підпис/

Виконавець
студент групи СПм-62 Мельник І.І.
“___” _____ 2019 р.
_____ /підпис/

Тернопіль 2019

1 НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

1.1 Розробка клієнт-серверного додатку для операційної системи Android з використанням Java технологій.

1.2 Область застосування – на прикладі комерційної діяльності.

2 ОСНОВА ДЛЯ РОЗРОБКИ

Основою для розробки є завдання на дипломну роботу, попередньо видане та затверджене кафедрою програмної інженерії факультету комп'ютерно-інформаційних систем і програмної інженерії Тернопільського національного технічного університету імені Івана Пулюя.

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою дипломної роботи є вдосконалення через автоматизацію роботу суб'єкта пов'язану з актуальною на сьогоднішній день діяльністю, в якій об'єктом може бути не лише послуга, а й достовірна інформація чи будь-які дані, що можуть представляти комерційний інтерес, при використанні підходу «клієнт-сервер».

4 ДЖЕРЕЛА РОЗРОБКИ

Джерелами даної розробки є технічна документація, існуючі програмні та програмні системи, довідники та довідкові системи.

5 ЗАДАЧІ РОЗРОБКИ

5.1 Аналіз предметної області з метою вибору існуючих або розробки нових (удосконалення існуючих) моделей, що описують предметну область, математична постановка задач та вибір моделей їх розв'язку.

5.2 Вибір та обґрунтування вибору архітектури, вибір програмного середовища та необхідних додаткових бібліотек для реалізації програмної системи, вибір зовнішніх програмних та апаратних засобів для забезпечення роботи програмної системи.

5.3 Аналіз та уточнення вимог технічного завдання з точки зору обраних моделей, методів, алгоритмів та середовища розробки.

5.4 Проєктування, реалізація та тестування окремих компонент програмної системи, розробка принципів взаємозв'язку між ними.

5.5 Тестування програмного комплексу на навантаження.

5.6. Організаційно-економічні розрахунки.

5.7. Аналіз роботи в аспекті дотримання вимог положення з Охорони праці і безпеки в надзвичайних ситуаціях.

6 ВИМОГИ ДО ПРОҐРАМНОЇ СИСТЕМИ

6.1 Функціональні вимоги

6.1.1 Користувач системи повинен мати змогу виконувати наступні функції:

- Можливість оформлювати, редагувати та надсилати запит;
- Можливість перегляду інформації;
- Можливість обміну даними з сервером;
- Можливість використання фільтрів при перегляді інформації;
- Можливість перегляду інформації за запитом;
- Можливість оновлення версії клієнта з сервера;

– Можливість відстежування місцезнаходження;
– Можливість додавання коментарів до замовлення.

6.1.2 Вхідна інформація отримується шляхом:

— Завантаження інформації із сервера;

— Створення її на пристрої/сервері

6.1.3 Вихідна інформація:

— Вихідна інформація повинна подаватись у простому та інтуїтивно зрозумілому для користувача форматі;

— Вихідна інформація виводиться у текстовому та табличному форматах;

6.2 Вимоги до надійності.

6.2.1 Передбачити контроль введеної інформації.

6.2.2 Розробити комплекс заходів контролю коректності дій користувача під час роботи з системою.

6.3 Вимоги до апаратних засобів.

6.3.1 Клієнтська частина системи повинна працювати на пристроях зі встановленою ОС Android 4.0 та вище.

6.3.2 Мінімальні вимоги до робочих станцій: ОС Microsoft Windows XP та вище або Linux-подібні системи, 200 МБ вільного дискового простору, 1024 МБ оперативної пам’яті, тактова частота процесора 1 ГГц, Python версії 2.5-2.7.

6.3.3 Вимоги до програмного забезпечення: операційна система Android версії 4.0 і вище.

6.5 Вимоги експлуатації

6.5.1 Кліматичні вимоги до експлуатації, при яких забезпечується робота програми повинні відповідати кліматичним умовам експлуатації наявних технічних засобів.

6.5.2 Вимоги до кваліфікації та численності персоналу. Мінімальна кількість користувачів, необхідної для роботи програмної системи, може складати одину одиницю.

7 ВИМОГИ ОХОРОНИ ПРАЦІ

В розділі “Охорона праці” дипломної роботи повинен бути даний аналіз умов праці в приміщенні де працює розробник програмного засобу.

8 ПОРЯДОК КОНТРОЛЮ І ПРИЙОМУ

8.1 Представлення дипломної роботи до попереднього захисту

8.2 Представлення дипломної роботи до захисту

УДК 004.41

І.І. Мельник, Г.Б.Цуприк, к. т. н., доцент

(Тернопільський національний технічний університет імені Івана Пулюя)

РОЗРОБКА КЛІЄНТ-СЕРВЕРНОГО ДОДАТКУ ДЛЯ ОПЕРАЦІЙНОЇ СИСТЕМИ ANDROID З ВИКОРИСТАННЯМ JAVA ТЕХНОЛОГІЙ

Сьогодні більшість населення Землі навіть не замислюється над тим, що можна прожити хоча б частину (якщо не все) своє життя без використання сучасних інформаційних технологій. Для прикладу, користувачі, в своїй більшості, навіть не уявляють свого щоденного існування без доступу до всесвітньої мережі і здебільшого не цікавляться принципами її роботи, які засоби та сучасні можливості при цьому використовуються. Їх основною метою та бажанням є відносно легке отримання коректної до запиту, актуальної та найбільш повної інформації вчасно, безперебійно, в будь-який момент часу та якісно.

Як показує практика, в сенсі взаємодії, одним з найефективніших виявився підхід «клієнт-сервер» – концепція під якою розуміють дві сторони. З одного боку, клієнт – замовлення (для прикладу якоїсь послуги, інформації, тощо) та сервер з іншого боку – в якості постачальника замовленого. Клієнт та сервер – це окремі програми, наприклад, типовим клієнтом може бути браузер. У якості сервера можна навести такі приклади: всі HTTP-сервери (в Apache); MySQL-сервер; локальний веб-сервер AMPPS або готова збірка Denwer. Деякі з наведених – це ціла сукупність серверів.

Також варто зауважити, що в основі взаємодії по типу «клієнт-сервер» знаходиться принцип того, що її завжди розпочинає лише клієнт, а серверна частина лише відгукується на його запит. Крім того сервер інформує про те чи може він надати послугу клієнтові і якщо може, то на яких умовах. Зазвичай як клієнтське так і серверне програмне забезпечення може встановлюватись як на різних машинах, так і працювати на одному комп'ютері.

Оскільки ефективність такого підходу є беззаперечною, прийняте рішення за допомогою підходу «клієнт-сервер» вдосконалити через автоматизацію роботу суб’єкта пов’язану з актуальною на сьогоднішній день діяльністю, в якій об’єктом може бути не лише послуга, а й достовірна інформація чи будь-які дані, що можуть представляти комерційний інтерес. В результаті виконання огляду та порівняльного аналізу подібних систем встановлено, що вони, у своїй сукупності, не володіють достатнім рівнем гнучкості, об’єктивності та надійності.

Дана концепція взаємодії була обрана як найзручніша в першу чергу із-за того що вона дає змогу розподілити навантаження між учасниками процесу обміну інформацією (даними), а також для того, щоб розділити програмний код на стороні постачальника і на стороні замовника.

Оскільки специфіка галузі передбачає роботу з додатками, які працюють на пристроях Android (смартфони, планшети, вбудовані системи, тощо), для реалізації поставленого завдання було обрано мову програмування Java, оскільки вона є однією з офіційних мов програмування під Android, конкуруючи лише з Kotlin.

В результаті отримано надійний програмний продукт, що дозволить забезпечити продуктивну роботу та який володіє зручним інтерфейсом. Оскільки розроблювальна система використовуватиме, для свого повноцінного функціонування базу даних, то враховано необхідність забезпечення надійності з’єднання та безперебійності програми, а також передбачено ряд запитів до неї.

ДОДАТОК В – ДИСК З ПРОҐРАМНИМ ЗАБЕЗПЕЧЕННЯМ