МІНІСТЕРСТОВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
Факультет кібербезпеки, комп‘ютерної та програмної інженерії
Кафедра комп’ютерних інформаційних технологій

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
_____ А.С. Савченко
«____» _____ 2021 р

ДИПЛОМНИЙ ПРОЕКТ

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СПУПЕНЯ «БАКАЛАВР»

Тема: «Програмне забезпечення фірми “Софтлюкс”»

Виконавець: студент УС-411 Охремчук Олександр Олександрович
(студент, група, прізвище, ім'я, по батькові)

Керівник: к. т. н., доцент Райчев Ігор Едуардович
(науковій ступень, вчене звання, прізвище, ім'я, по батькові)

Нормоконтролер: ст. викл. Шевченко О.П. _____
(П.І.Б.) (підпис)

КИЇВ 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп‘ютерної та програмної інженерії

Кафедра комп’ютерних інформаційних технологій

Освітній ступінь: Бакалавр

Галузь знань, спеціальність, спеціалізація: 12 “Інформаційні технології”, 122 “Комп’ютерні науки”, “Інформаційні управляючі системи та технології”

ЗАТВЕРДЖУЮ

Завідувач кафедри

А.С. Савченко

“_____”_____ 2021 р.

ЗАВДАННЯ

на виконання дипломного проекту студента

Охремчук Олександр Олександрович

(прізвище, ім’я, по батькові)

1. Тема проекту: «Програмне забезпечення фірми “Софтлюкс”» затверджена наказом ректора № 636/ст. від 22.04.2021р.
2. Термін виконання роботи: з 11.05.2021 по 14.06.2021р.
3. Вихідні дані до роботи: розробка програми для автоматизації валютних операцій фірми “Софтлюкс”.
4. Зміст пояснювальної записки (перелік питань, що підлягають розробці): аналіз існуючих програм для пунктів обміну валют, огляд використання найбільш вживаних методів розробки програмного забезпечення.
5. Перелік обов’язкового графічного матеріалу: Взаємодія груп процесів в рамках фази або проекту. Зразок декомпозиції робіт, організований через фазу проекту. Зразок декомпозиції робіт, організований через основні поставки проекту. Розклад проекту – графічні приклади.

КАЛЕНДАРНИЙ ПЛАН

	Етапи виконання дипломної роботи	Термін виконання етапів	Примітка
1	Аналіз літератури та джерел за темою дипломного проекту.	11.05.2021р. – 12.05.2021р.	
2	Розробка та затвердження плану дипломного проекту.	13.05.2021р.	
3	Проведення консультації з науковим керівником щодо створення першого розділу.	14.05.2021р.	
4	Зібрання необхідного теоретичного матеріалу.	15.05.2021р. – 18.05.2021р.	
5	Формулювання задачі та вибір технологій для реалізації.	19.05.2021р. – 22.05.2021р.	
7	Проведення аналізу існуючого програмного забезпечення для пунктів обміну валют.	23.06.2021р. – 27.05.2021р.	
8	Створення програмного забезпечення для валютно-обмінних операцій.	28.05.2021р. – 04.06.2021р.	
9	Висновки та оформлення пояснювальної записки дипломного проекту.	05.06.2021р. – 08.06.2021р.	
10	Підписання необхідних документів у встановленому порядку.	09.06.2021р. – 10.06.2021р.	
11	Підготовка до захисту та попередній захист дипломного проекту на випусковій кафедрі дипломного проекту	11.06.2021р. – 12.06.2021р.	

Студент **(*Охремчук О.О.*)**

Керівник дипломної роботи **(*Райчев І.Е.)***

РЕФЕРАТ

Пояснювальна записка до дипломної роботи на тему: «Програмне забезпечення фірми “Софтлюкс”» містить: 91 сторінку, 27 рисунків, 1 додаток, 9 літературних джерел.

Об'єкт дослідження: Програмне забезпечення для фірми, яка займається обміном валют.

Предмет дослідження: Особливості розробки ПЗ для валютно-обмінних пунктів.

Мета роботи: Створення програмного забезпечення для автоматизації валютних операцій.

Методи дослідження: Аналіз технологій створення програмного забезпечення.

Отримані результати: Розроблено програмне забезпечення для фінансових компаній, які займаються обміном валют. Установлено, що розроблений програмний продукт оптимізує роботу підприємства.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, БАЗА ДАНИХ, ПРОГРАМНИЙ КОД, C++, QT, QT CREATOR, МЕТОД, АЛГОРИТМ, РОЗРОБКА.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ 7
ВСТУП 8
РОЗДІЛ 1 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ 9
1.1. Програмне забезпечення 9
1.2. Елементи програмного забезпечення 11
1.3. Класифікація програмного забезпечення 11
1.3.1. Системне ПЗ 12
1.3.2. Прикладне ПЗ 14
1.3.3. Інструментальне ПЗ 16
1.4. Середовище розробки 16
1.5. Етапи розробки програмного забезпечення 17
1.5.1. Постановка завдання 18
1.5.2. Розробка моделі і вибір методу вирішення 19
1.5.3. Розробка алгоритму розв’язання задачі 19
1.5.4. Кодування алгоритму 20
1.5.5. Тестування програми 20
1.5.6. Створення документації та супровід ПЗ 21
Висновок до розділу 1 21
РОЗДІЛ 2 ФОРМУЛЮВАННЯ ЗАДАЧІ 23
2.1. Передумови для створення задачі 23
2.2. Опис предметної області 24
2.3. Постановка задачі 26
2.4. Аналіз інструментальних засобів програмування C++/QT 27
2.5. Аналіз бази даних SQLite 28
2.6. Особливості роботи з PPO 30
Висновок до розділу 2 32
РОЗДІЛ 3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ОБМІННО-ВАЛЮНИХ ОПЕРАЦІЙ 33
3.1. Проектування схеми бази даних 33
3.2. Опис вхідної інформації 37

3.3. Структура програми 42
3.4. Тестування 45
3.5. Опис тестування 48
Висновок до розділу 3 52
ВИСНОВКИ 53
СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ 54
ДОДАТКИ 55

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

СУБД	–	система управління базами даних
SQL	–	мова структурованих запитів
ОС	–	операційна система
БД	–	база даних
ПЗ	–	програмне забезпечення

ВСТУП

У наш час практично у будь-якій людській сфері діяльності існує автоматизована система. Останніми роками бурхливого технологічного розвитку зазнала фінансова система нашої країна. Все більше фінансових компаній опираються на нові технології. Комп'ютери, інформаційні мережі, бази даних, все це створило умови для розробки нового програмного забезпечення, яке значно полегшує життя людини. Одним з головних завдань більшості організацій є швидка та безперебійна обробка потоків інформації. Ряд фінансових компаній вкладають кошти в програмне забезпечення, комп'ютерне обладнання та створення бази для переходу до нових обчислювальних платформ з метою зменшення вартості роботи та її прискорення. У сучасних умовах фінансові компанії працюють з метою отримання максимальних доходів при мінімально затрачених ресурсах.

Найбільш ефективним способом збільшення ефективності обмінних пунктів є їх автоматизація. Однак комп'ютеризація, в свою чергу, вимагає додаткових витрат, на які далеко не кожна компанія готова піти. Для дрібних фінансових компаній покупка дорогих модулів автоматизованих систем часто не є вигідною, адже сучасна техніка для дорогих проектів повинна бути досить дорогою та потужною. Також, з певних причин, не завжди відома тривалість існування обмінного пункту, тому виникає необхідність для пошуку дешевших альтернатив, так як будь-яка компанія прагне зменшити свої витрати до мінімуму.
Саме це і визначає практичну значимість та актуальність обраної теми.

Метою системи, що буде описана, є збільшення швидкості обробки інформації та зменшення часу обслуговування клієнта. Це дозволить збільшити продуктивність оператора валютно-обмінних операцій, за допомогою полегшення його праці. Також розроблювана система дозволить вести реєстр валютно-обмінних операцій більш ефективно.

РОЗДІЛ 1
ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

1.1. Програмне забезпечення

Комп'ютер - це пристрій, який здатний здійснювати надскладні розрахунки, але на відміну від людини, він не здатний мислити самостійно. Для того, щоб комп’ютер міг опрацьовувати інформацію – отримувати її, передавати, зберігати, обробляти – його потрібно запрограмувати на виконання всіх цих дій. Запрограмувати – означає надати комп’ютеру певну інструкцію, в якій зазначено, що і як треба робити. Така інструкція повинна складатися із суворої послідовності команд, зрозумілих комп’ютеру. Вона повинна повідомляти йому яким способом потрібно обробити дані, для отримання очікуваного результату.
Дана інструкція називається програмою.

Перші програми були розроблені на машинній мові, яка представляла собою послідовність нулів і одиниць. Далеко не кожна людина могла займатись розробкою такої програми, тільки професійні програмісти.

У 60-і роки вперше почалася розробка високорівневих мов програмування, які дозволили значно полегшити роботу, і створення програм стало доступно для більш широкого кругу користувачів.

Програма представляє собою послідовність команд, яку виконує комп’ютер з деякими вхідними даними, тобто інформацією. Кожна програма зберігається у постійній та зовнішній пам’яті комп’ютера. Для того, щоб комп’ютер міг обробити певну інформацію, відповідна програма повинна бути поміщена в його оперативну пам’ять.

Персональний комп’ютер - це універсальний пристрій для обробки інформації. Більшість простих приладів, як телевізор чи магнітофон, вміють здійснювати тільки заздалегідь закладені функції. Персональний комп’ютер, в свою чергу, може виконувати будь-які дії по обробці інформації. Для цього достатньо скласти точну послідовність дій на зрозумілій йому мові програмування. Сам комп’ютер не володіє знаннями в жодній області, в якій застосовується. Усі ці знання містять виконувані комп’ютером програми.

Комп’ютер можна легко перетворити на робоче місце бухгалтера чи конструктора, дизайнера або ученого, письменника або агронома, просто надавши йому необхідне програмне забезпечення. Більше того, завдяки тенденції зниження вартості комп’ютерної техніки та зростанні її продуктивності, комп’ютери стали предметом домашнього використання, як, наприклад холодильник або телевізор, які розширюють сферу застосування ПК. Відповідно, з’являється потреба у створенні ще більш різноманітного програмного забезпечення, для вирішення завдань у нових областях застосування. Завдяки безперервному підвищені потужності комп’ютерів, а також стрімкому розвитку засобів зв’язку, розробники отримали можливість для реалізації ще більшої кількості запитів кінцевого споживача.

Програма – це впорядкований набір інструкцій, виражених у формі придатній для зчитування комп’ютером. Для початку необхідно зрозуміти, в чому полягає необхідність програми, і що вона повинна виконувати. Після цього, людина пише потрібну програму на спеціальній мові програмування. Даний процес написання програмного забезпечення називається програмуванням.

Усі знання комп’ютера зосереджені у виконуваних на ньому програмах. Тому часто вживаний вираз «комп'ютер зробив», означає що була виконана відповідна, встановлена на ПК, програма, яка дозволила виконати необхідні дії. Всі програми, які може виконувати персональний комп’ютер називаються програмним забезпеченням.

1.2. Елементи програмного забезпечення

Всього, комп'ютерна система являє собою дві основних складових – програмне та апаратне забезпечення. Програмне забезпечення представляє собою, встановлені на комп'ютері, програми. Апаратне забезпечення – вузли та обладнання, розташовані в системному блоці або підключені ззовні.

Частина, яка дозволяє користувачу взаємодіяти з комп'ютерною системою називається інтерфейсом.

Загалом розрізняють три види інтерфейсу:

- апаратний інтерфейс – взаємодія між різними вузлами
- програмний інтерфейс - взаємодія між програмами
- апаратно-програмний інтерфейс - взаємодія між апаратурою і програмами

Говорячи про персональний комп'ютер, можна виділити і третього учасника роботи – користувача, тобто людину, яка користується ПК. В свою чергу, користувач теж взаємодіє як з апаратним, так і з програмним забезпеченням. Спосіб взаємодії користувача та програми називається інтерфейсом користувача.

1.3. Класифікація програмного забезпечення

Програмне забезпечення може бути умовно поділене на три категорії:

- системне ПЗ – програми загального користування, призначені для виконання різних допоміжних функцій.
- прикладне ПЗ – програми, які слугують для виконання необхідних робіт на персональному комп'ютері. Наприклад, створення картинок чи малюнків, редагування текстових документів, обробка інформації і т.д.
- інструментальне ПЗ – програмне забезпечення, яке дозволяє створювати інші програми для комп'ютера на різних мовах програмування.

Область діяльності з проектування та розробки ПЗ також безпосередньо відноситься до програмного забезпечення.

Рис. 1.1. Класифікація програмного забезпечення

1.3.1. Системне ПЗ

Системне ПЗ забезпечує взаємодію між людиною, пристроями та всіма програмами комп'ютера, і тому являється необхідною приналежністю комп'ютера. Системна середа та правила роботи з нею визначаються комплексом програм. Як правило, взаємодіяти з досконалим системним програмним забезпеченням досить комфортно.

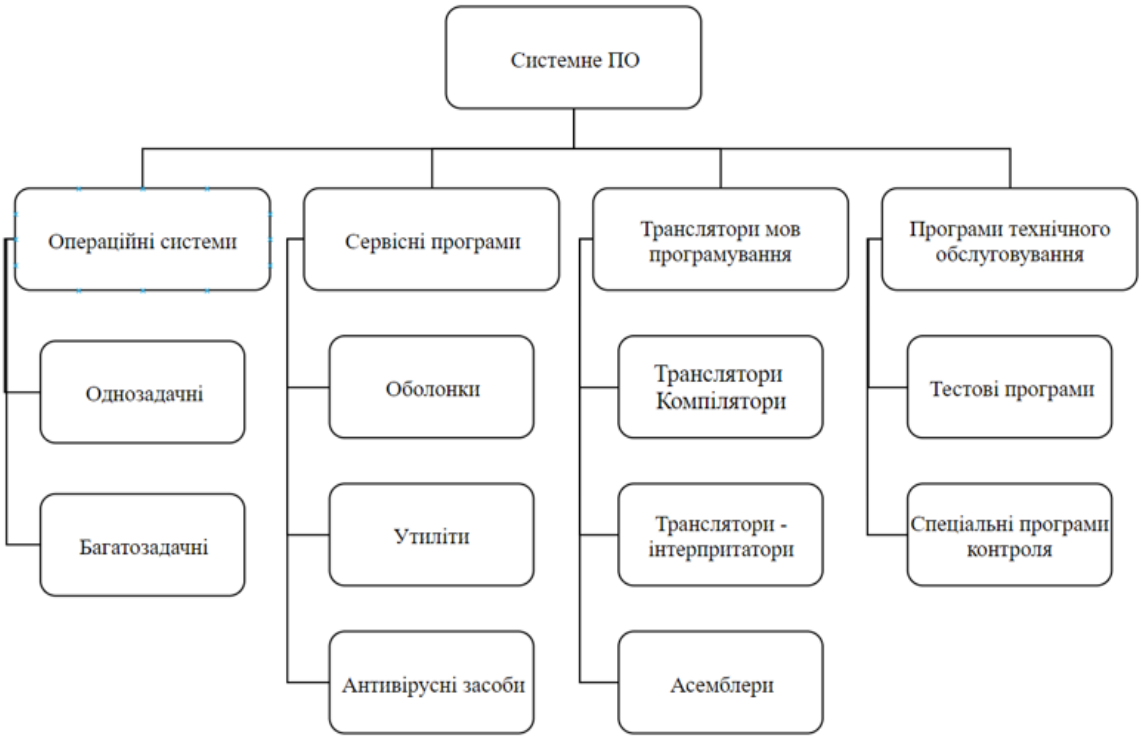

Рис.1.2. Класифікація системного програмного забезпечення

Операційна система (ОС) являється основною та найважливішою системою ПЗ та, як правило, зберігається на жорсткому диску. Коли комп'ютер включається, основна частина ОС записується у внутрішню пам'ять, в якій знаходить протягом усього сеансу роботи ПК. При відсутності в комп'ютері операційної системи, взаємодія з будь-яким програмним забезпеченням, окрім ПЗ для діагностики, є неможливою.

Завдяки ОС забезпечується:

- Взаємодія людини з комп'ютером;
- Виконання прикладних програм;
- Управління ресурсами ПК, наприклад процесором, пам'яттю, та всіма зовнішніми пристроями.

Відсоткове співвідношення кількості користувачів найпопулярніших операційних систем у 2020 році:

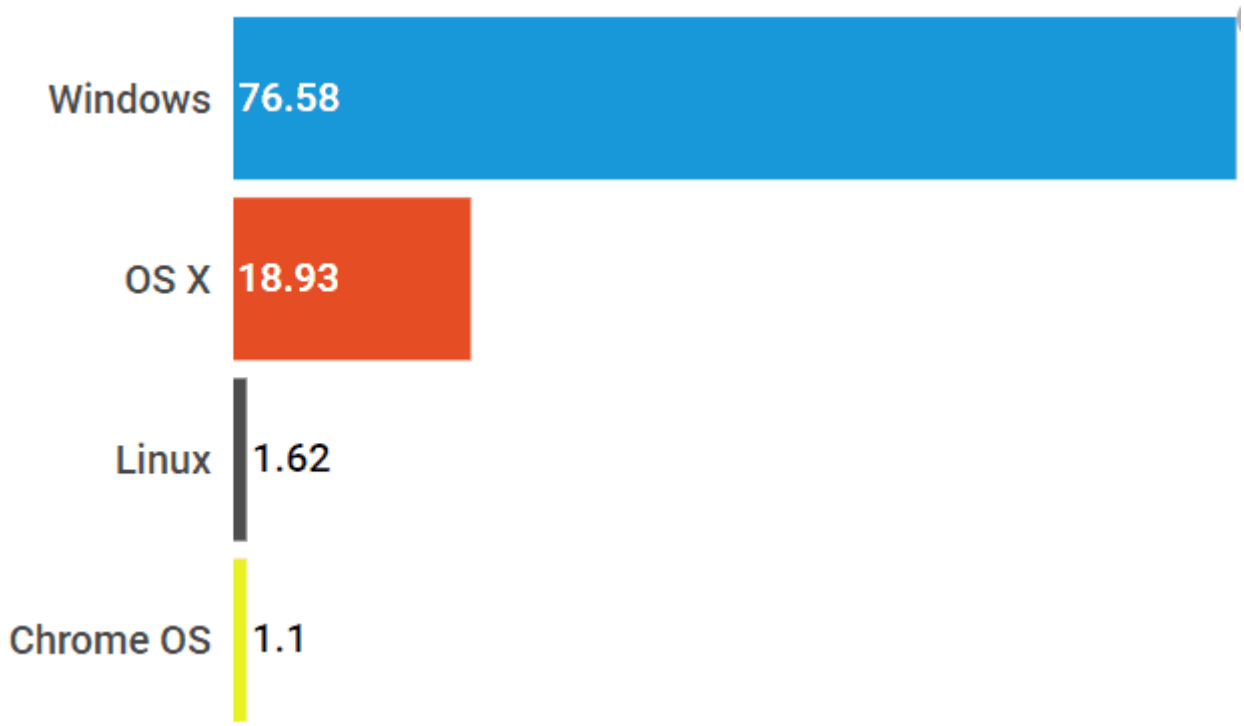

Рис.1.3. Найпопулярніші операційні системи

Системне програмне забезпечення також включає в себе різні комплекси програм, призначених для виконання особливих функцій.

Також до системних програм можна віднести утиліти (програми системного призначення).

1.3.2. Прикладне ПЗ

Прикладне програмне забезпечення визначає прикладну серед комп'ютера і правила роботи в ній. Прикладне середовище, як правило, є нескладним у використанні та досить корисним. Прикладні програми можуть бути встановлені тільки на комп'ютер із операційною системою.

В середовищі операційної системи Windows існують, так звані, додатки. Додатком або пакетом прикладних програм (ППП) називається комплекс прикладних програм.

Наступні групи програмного забезпечення користуються найбільшою популярністю:

- Бази даних (БД) – існують для управління та організації даних;
- Електронні таблиці – відповідають за обчислення та аналіз інформації, в табличній формі;
- Текстові процесори – існують для створення текстових документів;
- Ігри;
- Графічні пакети;
- Навчальні програми, словники, електронні підручники, системи проектування;
- Інтегровані пакети – існують для представлення інформації у вигляді малюнків або графіків;
- Комунікаційні програми – слугують для обміну інформацією між ПК.

Рис. 1.4. Класифікація прикладного програмного забезпечення

1.3.3. Інструментальне ПЗ

Інструментальне програмне забезпечення – це система для автоматизованої розробки нових програм на певній мові програмування. Як правило, для написання програми на обраній мові програмування достатньо мати ряд компонентів:

- Текстовий редактор - для написання вихідного коду програми.
- Компілятор або інтерпретатор - перекладає, зрозумілу людині, мову програмування в бінарний код.
- Збирач (Редактор зв'язків) – формує працездатний додаток, тобто виконаний код, який собою представляє закінчену програму, яка може бути запущена з будь-якого ПК.

1.4. Середовище розробки

Сьогодні розробка програмного забезпечення є дійсно величезною індустрією. Для спрощення процесу розробки програмних продуктів існують середовища розробки. Інтегровані середовища існують для всіх найпопулярніших мов програмування, та є дуже популярними, завдяки своїй багатофункціональності. Вони допомагають виключити з роботи розробника рутинну технічну роботу. По суті, термін середовище розробки означає, що у вас під рукою буде все необхідне для розробки додатків та програм.

Допоміжні засоби, які спрощують роботу:

- Контекстно-залежні підказки та допомога;
- Забарвлення синтаксису відповідно до особливостей мови;
- Виділення структури вихідного коду.

Завдяки всім вище перерахованим засобам процес роботи з мовою є значно ефективнішим.

До найбільш відомих середовищ розробки можна віднести:

- PhpStorm – підтримувані мови програмування: PHP, JavaScript;

- CLion – підтримувані мови програмування: C++, C, Objective C, Kotlin, Python, Swift, Fortran, JavaScript;
- Microsoft Visual Studio - підтримувані мови програмування: Ajax, ASP.NET, JavaScript, Visual Basic, Visual C#, Visual C++, Visual F#
- PyCharm - підтримувані мови програмування: Python, Jython, Cython, IronPython, PyPy, AngularJS, Coffee Script;
- Intellij IDEA - підтримувані мови програмування: Python, Jython, Cython, IronPython, PyPy, AngularJS, Coffee Script;
- Eclipse - підтримувані мови програмування: C, C++, Java, Perl, PHP, Python, Ruby.

1.5. Етапи розробки програмного забезпечення

Процес розробки програмного забезпечення умовно можна розбити на наступні етапи:

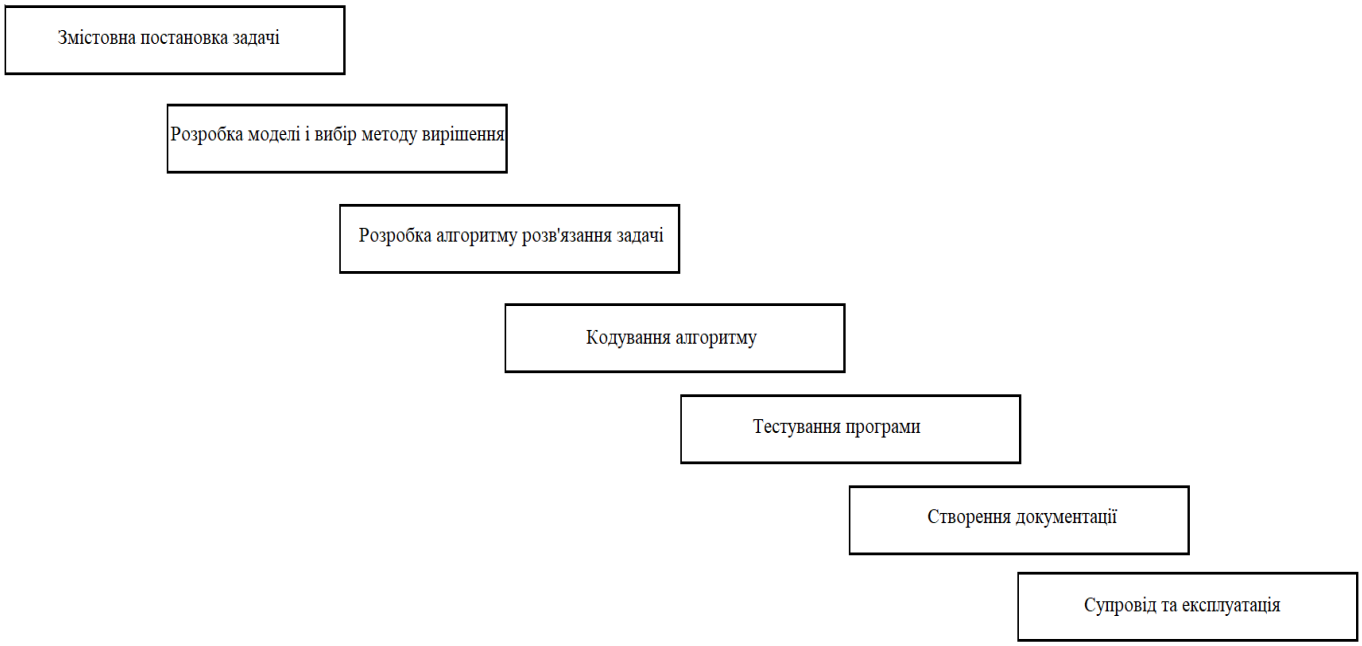

Рис. 1.5. Етапи розробки ПЗ

1.5.1. Постановка завдання

Постановка завдання – це точне формулювання задачі з описом вхідної та вихідної інформації.

У результаті виконання цього етапу повинні бути чітко визначені наступні деталі:

- Назва завдання - коротке визначення розв'язуваної задачі, назва програми;
- Опис – докладне викладення постановки задачі, мета і призначення завдання;
- Управління режимами роботи програмного забезпечення – формулювання основних вимог до способу взаємодії користувача з програмою;
- Вхідні дані – опис даних, які будуть надходити програмі для подальшої обробки. Вказуються тип даних, межі в яких вони можуть змінюватися та джерело даних, тобто пристрій з якого дані будуть надходити програмі;
- Вихідні дані. Опис результату роботи програми, який буде надісланий користувачеві. Вказується, в якому вигляді дані повинні бути передані користувачеві – графічному, текстовому, числовому. Також необхідно вказати точність вихідної інформації та пристрій відображення даних;
- Помилки – перелік можливих помилок при роботі користувача з програмою, тобто помилки при введені даних та ін.
- Приклад роботи програми – наведення декількох прикладів роботи програмного забезпечення, враховуючи, як позитивні так і негативні випадки.

1.5.2. Розробка моделі і вибір методу вирішення

Спочатку, на даному етапі складається математична або логічна модель досліджуваного явища реального світу. Після цього, на основі, складеної раніше, математичної моделі, потрібно визначити метод її вирішення.

Для програм з обчислювальним характером, як правило, наводиться перелік всіх використовуваних формул з докладними коментарями.

Для необчислювальних програм, наводиться деяких план дій, тобто словесний опис логічної моделі.

1.5.3. Розробка алгоритму розв'язання задачі

Загальна структура програмного забезпечення формується на даному етапі. Алгоритмом називається система чітко сформульованих правил, які визначають процес перетворення допустимих вхідних даних в бажаний результат за кінцеве число кроків.

Алгоритм може бути описаний різними способами. Найпопулярніші способи опису алгоритму:

- Словесний запис;
- Блок-схема;
- Псевдокод;
- Структурограмма.

Псевдокодом називають неформальний запис алгоритму, який нехтує синтаксисом мови програмування, для простішого та більш зрозумілого вигляду алгоритму. Псевдокод є дуже ефективний для розробки логіки програми. Коли логіка вас влаштує, ви можете звернути особливу увагу на деталі перетворення псевдокоду в справжню мову програмування.

Завдяки псевдокоду, розробник може сконцентруватись на написання безпосередньо логіки програми, не вдаючись при цьому в деталі синтаксису мови

програмування. Адже, для написання якісної та правильної програми необхідно спочатку створити відповідний алгоритм

1.5.4. Кодування алгоритму

Кодування алгоритму іншими словами можна назвати написанням програмного забезпечення. Даний етап полягає у перенесенні, написаного раніше алгоритму, в програму на конкретній мові програмування. Спочатку, необхідно написати файли з вихідним текстом програми, які потім потрібно скомпілювати.

Компіляція – це переклад вихідного тексту в машинний код, який пізніше може бути виконаний. Цей процес здійснюється спеціальним компілятором, для вибраної мови програмування. У разі успішної компіляції утворюється виконуваний файл програми (розширення для різних мов може відрізнятись). Якщо ж під час цього процесу були виявлені певні помилки, програміст повинен переписати вихідний код програми, та повторити процес компіляції поки він не завершиться успішно.

1.5.5. Тестування програми

Розрізняються два основних видів тестування:

- Автоматизовне;
- Мануальне (ручне тестування).

Ціллю автоматизованого тестування є написання тестів, які можуть бути запущені після додавання нового функціоналу. Таким чином легко виявити, чи призвели останні зміни вихідного коду до неочікуваної поведінки програми.

Ручне тестування полягає у перевірці правильності роботи функціоналу шляхом використання програми, як звичайний користувач. Вихідні дані для тестування, як правило, підбираються такі, для яких результат заздалегідь відомий.

1.5.6. Створення документації та супровід ПЗ

Документація за своїм призначенням може бути розбита на декілька наступних груп:

- Опис застосування;
- Керівництво користувача;
- Керівництво програміста

Опис застосування – це загальна характеристика програмного продукту та сфер його застосування. Керівництво користувача – це детальний опис можливостей програми та опис технології роботи з програмою для кінцевого користувача. Керівництво програміста, в свою чергу, призначене для розробників ПЗ та фахівців, які будуть його супроводжувати.

Після здачі програмного забезпечення в експлуатацію, наступає етап супроводу програмного забезпечення. Під час експлуатації може виникнути необхідність додавання нових функцій або усунення неполадок, виявлених кінцевим користувачем. Даний тип робіт називається супроводом ПЗ.

Висновок до розділу 1

На підставі розглянутого теоретичного матеріалу, можна зробити висновок, що програмне забезпечення має задовольняти наступні вимоги:

- відкритість;
- масштабованість;
- незалежність від платформи.

Тому для задоволення вищеперерахованим вимогами було обрано використовувати крос-платформне середовище розробки, яке надає можливості портування програми з мінімальними змінами вихідного коду.

Також, на основі класифікації ПЗ, було встановлено, що сервіс валютно-касових операцій, належить до прикладних програм спеціального призначення.

Отже, особливості розробки у даній області повинні бути досліджені для подальшої роботи над сервісом.

РОЗДІЛ 2
ФОРМУЛЮВАННЯ ЗАДАЧІ

2.1. Передумови для створення задачі

Інформаційне забезпечення, яке полягає у зборі та обробці інформації, необхідної для прийняття управлінських рішень, стало важливою областю у сучасних умовах. Для передачі інформації про діяльність та стан підприємства на вищий рівень управління та взаємний обмін інформацією між певними розділами організації використовуються сучасна електронно-обчислювальна техніка та інші технічні засоби зв'язку.

У даний час, роль автоматизації документообігу помітно зросла. Витрати робочого часу для обробки операції, повинні бути зменшені до мінімуму, для отримання найбільш ефективного результату діяльності підприємства. Також це допомагає збільшити кількість оброблюваних даних, при мінімально затрачених ресурсах, що дає підприємству прямий економічний ефект.

Саму тому, в даний час існує велика потреба підприємств у програмному забезпеченні, яке допомагає погоджувати та підтримувати роботу різних ланок компаній, зокрема фінансової та управлінської. Таким самим чином забезпечується інформація про способи оптимального використання наявного у компанії програмного забезпечення.

На даний момент, У ТОВ «Софтлюкс» не високий рівень автоматизації праці. Для роботи, касири операційних кас використовують одночасно декілька різних програм.

На початку робочого дня, таблиця в Microsoft Excel (надалі Excel) з наказом встановлення курсів повинна бути сформована. Окремо потрібно ввести операцію в реєстр розрахункової операції (РРО) та в ще одну таблицю Excel, яка повинна містити операції обміну за день, на основі сформованих квитанцій. В кінці зміни, з документу Excel формуються реєстри, довідки та звіт НБУ.

Існує досить велика ймовірність допущення помилок, адже всі проміжні розрахунки проводяться касирами вручну. В результаті, існує багато механічної роботи, яка може призвести до помилок, які тяжко відслідкувати. Всі дані, необхідні для звітів також вводяться вручну.

Очевидно, що автоматизація документів зберігання даних та розрахунків дозволить значно скоротити час та необхідні ресурси, адже автоматизований підрахунок буде повністю незалежний від людини, що дасть гарантію правильності кінцевих результатів. Усі необхідні звіти, відомості та квитанції будуть формуватися автоматично.

2.2. Опис предметної області

У наш час існує ряд країн, більшість з яких мають власну валюту. З розвитком торгових відносин між різними країнами, зростають випадки купівлі/продажу різних товарів та послуг між ними. Таким чином, виникає необхідність конвертації валюти, адже у багатьох країн вона відрізняється.

Для торгівлі між країнами виникає потреба конвертувати валюту. Така ж потреба існує в роботі туристичних агентств, так як при подорожах до інших країн людина повинна обміняти валюту на місцеву, аби мати змогу оплачувати різні товари та послуги.

Можливість обміну валют різних країн надають фінансові компанії. З розвитком комп’ютерних технологій, з’являються також електронні валюти. Відповідно, з’являється потреба їх конвертації. Для цього існують віртуальні пункти обміну валют, які дозволяють здійснювати транзакції коштів та переглядати курси валют.

Предметною областю «Обмінного пункту валют» являється діяльність роботи обмінного пункту, в першу чергу, призначена для автоматизації роботи касирів. Обмін здійснюється як продажу, так і покупки грошових одиниць у валюті країни. Всі дані автоматично заносяться в таблицю. Існує можливість перегляду всіх здійснених операцій, за певний термін.

На даний час, існує декілька програмних продуктів для здійснення валютно-обмінних операцій. Серед них, можна виділити наступне ПЗ:

- Система «OBVAL» від ООО «Компьютерінтерсервіс»;
- Пункт обміну валют «АРС-ТАН» від «AS-Сервіс».

Кожна з вище перерахованих програм може вирішувати наступні задачі:

- Продаж валюти;
- Купівля валюти;
- Автоматизоване отримання курсів від НБУ та відправка файлів статичної звітності;
- Сторнування;
- Інкасація;
- Відповідність реєстрів та звітів до вимог НБУ;
- Інтеграція системи з РРО (фіскальним принтером);
- Можливість зміни курсу валют протягом дня.

При всій зручності даного програмного забезпечення, основним недоліком є його висока вартість – 11000 грн за програмне забезпечення (разова оплата), 3000 грн за кожен пункт обміну (разова оплата), 3000 грн додатково за кожен місяць використання програмного продукту. Клієнт ТОВ «Софтлюкс» має 10 пунктів обміну, тому ця ціна є досить великою – приблизно 46000 грн за перший місяць. Це послугувало причиною для створення власного програмного забезпечення для обліку валютно-касових операцій.

2.3. Постановка задачі

Розроблювана система призначена для обліку купівлі та продажу валюти в операційних касах клієнту ТОВ «Софтлюкс».

ТОВ «Софтлюкс» є продуктовою компанією, яка створює програмне забезпечення на замовлення. Клієнт, в даному випадку, фінансова компанія, яка займається наданням послуг з обміну валют та має 10 пунктів обміну у місті Києві. Обладнання кожної каси включає в себе касовий апарат, комп'ютер та принтер. Перед початком робочого дня касир формує та роздруковує наказ на встановлення курсів, після чого відкриває зміну. Обмін супроводжується фіскальними чеками, які друкуються з допомогою РРО. Кожна операція записується в базу даних. При закінченні зміни виконується наступний алгоритм дій:

- З бази даних формується звіт до НБУ;
- Друкується Z-звіт;
- Формуються та роздруковуються реєстр купівлі, продажу та звітна довідка про обороти за день.

Необхідно розробити програмний продукт «Автоматизація обліку операцій пункту обміну валют». Наступні функцій будуть присутні в даному програмному засобі:

- Купівля/Продаж іноземної валюти;
- Відображення залишків гривні та інших валют на касі;
- Відображення поточного курсу валют;
- Можливість встановлення курсу;
- Можливість друку необхідних документів;
- Можливість формування звітів з БД;
- Робота з РРО.

2.4. Аналіз інструментальних засобів програмування C++/QT

Для створення даного програмного забезпечення була вибрана кросплатформна бібліотека розробки графічного інтерфейсу на C++ в поєднанні з середовищем розробки QT Creator. QT надає розробнику не тільки певний каркас структури додатків, а й обширний набір зручних бібліотек. Для запобігання важко відловлюваних помилок, таких як виток пам'яті або незвільнених дескрипторів ресурсних об'єктів, необхідно дотримуватись основних принципів і правил «гарного» стилю програмування на C++ та QT. Потрібно зазначити, що перераховані вище помилки є досить частими у програмах, написаних на C++ без бібліотеки QT.

Основною перевагою бібліотеки QT є стрункий та логічний набір класів, завдяки яким розробник може досягти досить високого рівню абстракції. Це допомагає значно зменшити кількість коду, в порівнянні, наприклад, з бібліотекою MFC. Сам же програмний код виглядає досить структуровано та зрозуміло, що сприяє його легкому читанню, при подальшій розробці. Такий код легко розширяти та підтримувати.

Крім того, завдяки кросплатформленості QT, додаток згодом може бути оптимізований для різних операційних систем, що робить програмний продукт досить гнучким. На відміну від аналогів, наприклад MFC, де потрібно досить багато нелегкої роботи для імпортування ПЗ на іншу операційну систему, QT дозволяє імпортувати програму для різних ОС шляхом простої перекомпіляції вихідного коду.

Також використання бібліотеки QT гарантує кращу якість вихідного коду, та кращу його сумісність зі стандартами C++. Цей аспект є дуже важливим для коду, який буде довгий час розвиватись та підтримуватись.

2.5. Аналіз бази даних SQLite

База даних представляє собою сховище розроблене спеціально для різних типів даних. За моделлю представлення виділяють наступні види баз даних:

- Ієрархічна;
- Об’єктно-орієнтована;
- Об’єктно-реляційна;
- Реляційна;
- Мережева;
- Функціональна.

Для управління базами даних різних форм і розмірів використовуються спеціальні додатки – системи управління базами даних (СУБД). Дані системи призначені для забезпечення реляційної моделі роботи з даними.

Для зберігання даних та роботи з ними СУБД має певну структуру у вигляді таблиці, в якій кожен стовпець може містити дані різного типу. Кожен запис таблиці складається з атрибутів, виражених у вигляді стовпців, та має унікальний ідентифікатор. Ідентифікатор називається ключем, та зберігається разом з іншими даними в таблиці. Як описано в реляційній моделі, усі ці дані взаємопов1язані між собою.

Кожне поле запису таблиці повинне мати заздалегідь вказаний тип даних, наприклад цілочисельне значення, значення з плаваючою точкою, рядок і т.д. У різних СУБД типи даних можуть незначним чином відрізнятися.

Найпопулярнішими системами управління базами даних є:

- PostgreSQL;
- SQLite;
- MySQL.

Для створення програмного забезпечення валютно-обмінних операцій було обрано базу даних SQLite. Перевагою цією бази даних є простота та зручність у використанні. В порівнянні з мережевими СУБД, ця система надає достатньо широкий набір інструментів для роботи з нею. Завдяки технологіям

обслуговуючих бібліотек та зберіганню даних безпосередньо у файлах, замість портів і гнізд в мережевих СУБД, SQLite є дуже швидкою та потужною базою даних.

Основні переваги SQLite:

- Стандарти – попри відносну примітивність даної БД, вона використовує мову SQL, що означає підтримку основних її особливостей;
- Файлова структура – база даних представляє собою один файл, може бути досить легко перенесена на інші машини;
- Масштабування – завдяки файлової структури бази та відмінності при розробці та тестуванні, SQLite надає все необхідне для зручного масштабування.

Основні недоліки SQLite:

- Відсутня можливість збільшення продуктивності – дана база даних спроектована для максимально зручного використання, а тому є недуже продуктивною.
- Відсутня система користувачів – більшість популярних СУБД надають можливість управління правами доступу різних користувачів. Так як SQLite розрахована на невеликі додатки, то дана функція на жаль відсутня.

Основні типи даних SQLite:

- INTEGER – цілочисельне значення зі знаком, яке займає до 8 байт;
- TEXT – текстовий рядок, підтримує кодування UTF-16BE, UTF-16LE, UTF-8;
- REAL – число з плаваючою точкою, яке займає 8 байт;
- BLOB – масив двійкових даних, використовується для зберігання даних великого об’єму (відео, аудіо, зображення).

Коли доречно використовувати SQLite:

- При прямому доступі до диска – якщо існує необхідність звертатися до диска на пряму, SQLite надає зручний, простий у використанні функціонал;

- Для вбудованих додатків – коли пріоритетним є можливість легкого перенесення програмного забезпечення, а масштабованість відходить на другий план;
- Тестування – SQLite дозволяє не більше одного процесу в проміжок часу. Це дозволяє значно збільшити продуктивність, так як додаткові процеси при тестуванні функціоналу дуже уповільнюють додаток.

Коли не доцільно використовувати SQLite:

- Запис великих обсягів даних – через обмеження в кількості процесів в проміжок часу, продуктивність самого додатку може уповільнюватись;
- Багатокористувацька програма – при необхідності забезпечення доступу відразу декільком користувачам, з різними правами доступу, SQLite є не найкращим вибором, адже в даній СУБД відсутня система користувачів.

2.6. Особливості роботи з РРО

Клієнт «Софтлюкс» використовує «КСТВ-В1» - єдиний в Україні спеціалізований електронно-контрольно-касовий реєстратор операцій купівлі-продажу іноземної валюти. Даний реєстратор має високу продуктивність, швидкість друку до 250 мм/с та досить високотехнологічну конструкцію.

«КСТВ-В1» дозволяє проводити наступні операції:

- Купівля/продаж валюти;
- Конвертація;
- Зворотній обмін валюти;

Для зв’язку з комп’ютером використовуються наступні інтерфейси:

- TCP/IP;
- USB;
- RS232.

Рис. 2.1. Реєстратор розрахункових операцій «КСТВ-В1»

Даний РРО обслуговується ЗАТ «Резонанс». Обслуговування включає в себе надання сервера OLE-automation, який дає можливість програмно взаємодіяти з реєстратором.

Для коректної роботи з сервером потрібно дотримуватись певного порядку:

- Крок 1 – встановлення з’єднання з ЕККА, з використанням функції INIT об’єкта;
- Крок 2 – виконання необхідних операцій з ЕККА. При цьому потрібно викликати певні функції об’єкта, які відповідають операції.
- Крок 3 - закриття з’єднання, використовуючи функцію DONE.

Усі операції з іноземною валютою, вимагають вказання коду відповідного до класифікатора валют Національного банку України. Фінансові функції сервера виконуються із вказанням сум в гривнях.

Висновок до розділу 2

У даному розділі було розглянуто приклади готового програмного забезпечення. На основі наведених даних, можна зробити висновок, що програмний продукт, який задовольняє мінімальним вимогам, та коштує дешевше ніж інші аналоги, має місце на ринку попиту. Вартість програмного забезпечення буде зведено до мінімуму, шляхом конкретній направленості на обмін валют.

Було прийнято рішення, використовувати інструментальний засіб розробки QT CREATOR, оскільки він надає можливість портувати програмне забезпечення на інші платформи, що зробить його кросплатформленим.

В якості СУБД, було обрано використовувати SQLite, так як ця база є зручною у використанні та містить весь необхідний функціонал.

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ОБМІННО-ВАЛЮНИХ ОПЕРАЦІЙ

3.1. Проектування схеми бази даних

Для проектування схеми бази даних, необхідно розробити деяку логічну модель для фізичної реалізації, яка міститься на рис. 3.1.

Розроблення логічної моделі включає в себе ряд завдань, таких як:

- Вибір моделі даних;
- Перевірка логічної моделі даних, на можливість виконання усіх транзакцій;
- Нормалізація таблиць;
- Визначення вимог цілісності даних;
- Визначення вимог документування даних;
- Створення фінального варіанта логічної моделі даних.

З предметної області програмного забезпечення валютно-обмінних операцій, можна виділити наступні сутності:

- Курс;
- Зміна;
- Операція;
- Користувач;
- Каса;
- Валюта;

Усі вище перераховані сутності пізніше будуть представляти собою набір відносин між собою.

Наступний кроком є ідентифікація зв’язків між об’єктами. Зв’язком називається деяка асоціація між двома таблицями в базі даних, яка дозволяє по одному відношенню, знаходити інші, пов’язані між собою. Загалом виділяють наступні типи зв’язків:

- n-n (багато до багатьох);
- 1-1 (один до одного);
- 1-n (один до багатьох);
- n-1 (багато до одного).

Розглянемо коротко кожен із зв’язків. Зв’язок типу n-n означає, що кожен екземпляр першої сутності може бути пов’язаний з багатьма екземплярами другої сутності, та відповідно кожен екземпляр другої сутності може бути пов’язаний з багатьма екземплярами першої. Цей тип зв’язку є допустимим на ранніх етапах моделювання схеми бази даних, та пізніше повинен бути мінімалізований.

Тип зв’язку 1-1 означає, що один екземпляр першої сутності пов’язаний з одним екземпляром другої сутності, та навпаки.

Зв’язок типу 1-n означає, що один екземпляр першої сутності пов’язаний з багатьма екземплярами другої сутності. Цей тип зв’язку є досить часто використовуваним. Об’єкт з боку «один» називають батьківським, а об’єкт з боку «багато» називають дочірнім.

Тип зв’язку n-1 означає, що кілька екземплярів першої сутності, пов’язана з одним екземпляром другою сутності.

Також кожен зв’язок має модальність. Загалом виділяють дві модальності зв’язку:

- «Повинен»;
- «Може».

Модальність «повинен» означає, що об’єкт однієї відносини зобов’язаний бути асоційований, як мінімум з одним об’єктом зазначеного типу.

Модальність «може» є менш суворою та означає, що об’єкт однієї відносини може бути асоційований до одного або декількох об’єктів, вказаного типу, але цей зв’язок не є обов’язковим.

Наступним кроком при проектуванні бази даних є визначення атрибутів для кожної сутності.

Сутність «Користувач» буде складатись з наступних атрибутів:

- Ключ користувача (унікальний ідентифікатор);
- ПІБ;
- Логін;
- Пароль.

Сутність «Зміна» буде мати наступні атрибути:

- Ключ зміни (унікальний ідентифікатор);
- Номер зміни;
- Ключ користувача;
- Дата відкриття;
- Дата закриття.

Сутність «Курс» буде мати наступні атрибути:

- Ключ курсу (унікальний ідентифікатор);
- Продаж;
- Покупка;
- Дата;
- Ключ валюти.

Сутність «Валюта» буде складатись з наступних атрибутів:

- Ключ валюти (унікальний ідентифікатор);
- Назва валюти.

Сутність «Операція» буде мати наступний набір атрибутів:

- Ключ операції (унікальний ідентифікатор);
- Тип операції;
- Кількість валюти;
- Дата;

- Ключ курсу;
- Ключ зміни;

Сутність «Каса» буде складатись з наступних атрибутів:

- Ключ каси (унікальний ідентифікатор);
- Початок дня;
- Залишок;
- Підкріплено;
- Інкасовано;
- Ключ валюти;
- Ключ зміни.

Логічна модель для реалізації бази даних була розроблена на даному етапі. Були розглянуті типи та модальності зв'язків, за допомогою яких по одному відношенню можуть бути отримані інші, пов'язані з ним, відношення. Також було виділено основні кроки при проектуванні логічної моделі бази даних.

На рисунку нижче (рис. 3.1) можна побачити графічне зображення всіх сутностей та відношень між ними.

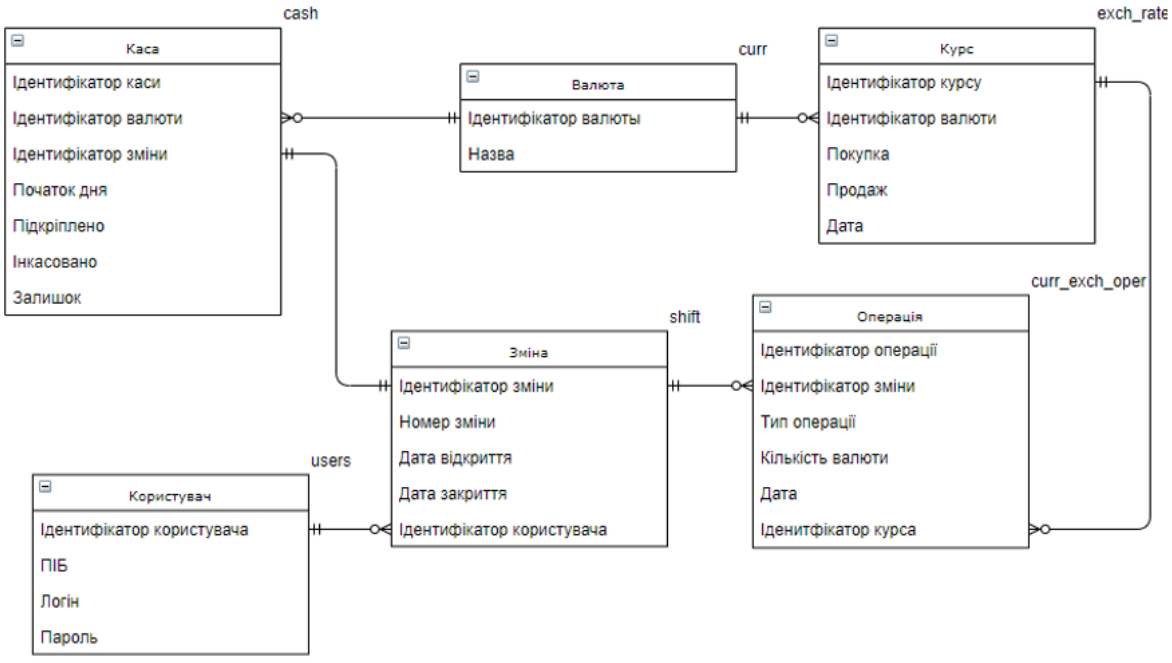

Рис. 3.1 – Діаграм зв'язків між сутностями

3.2. Опис вхідної інформації

В попередньому розділі було виділено 6 таблиць, які будуть зберігати всю інформацію, що буде вводитись користувачем або розрахована в процесі роботи програмного забезпечення. Для чого слугує, та чи інша таблиця можна судити по її назві.

На рисунку нижче наведено (рис. 3.2) наведено список основних таблиць, їх найменувань в базі даних та призначення кожної.

Найменування таблиці	Найменування	Призначення таблиці
cash	Каса	Облік грошей на зміні
users	Користувачі	Довідник користувачів
curr	Валюта	Довідник валюти
exch_rate	Курс обміну	Довідник встановлених курсів
shift	Зміни	Довідник закритих та відкритих змін
curr_exch_oper	Операції	Облік купленої та проданої валюти

Рис.3.2. Список основних таблиць

Розглянемо коротко кожну з таблиць. «Cash» містить інформацію про залишок валюти в касі.

Найменування поля	Опис поля	Тип	Дод. умови
id	Код валюти	INTEGER	Не пусте Первинний ключ Автолічильник Унікальне
id_shift	Код зміни	INTEGER	Не пусте
id_curr	Код валюти	INTEGER	Не пусте
start_day	Залишок на початок дня	REAL	
cash_in	Підкріплення	REAL	
cash_out	Інкасація	REAL	
end_day	Залишок на кінець дня/залишок	REAL	

Рис.3.3. Структура таблиці «Cash»

Таблиця «Users» містить інформацію про користувачів, включаючи їх права, ролі та дату останньою сесії.

Найменування поля	Опис поля	Тип	Дод. умови
id	Код користувача	INTEGER	Не пусте Первинний ключ Автолічильник Унікальне
login	Логін користувача	TEXT	Не пусте Унікальне
name	ПІБ користувача	TEXT	Не пусте
password	Пароль	TEXT	

Рис.3.4. Структура таблиці «Users»

В таблиці «Curr» міститься інформація про валюту та її код.

Найменування поля	**Опис поля**	**Тип**	**Дод умови**
id	Код валюти	INTEGER	Не пусте Первинний ключ Автолічильник Унікальне
name	Назва валюти	TEXT	Не пусте

Рис.3.5. Структура таблиці «Curr»

Таблиця «Exch_rate» містить інформацію про дату встановлення та встановлений курс на валюту.

Найменування поля	**Опис поля**	**Тип**	**Дод. умови**
id	Код встановленого курсу	INTEGER	Не пусте Первинний ключ Автолічильник Унікальне
id_curr	Код валюти	INTEGER	Не пусте
buy	Курс покупки	REAL	Не пусте
sell	Курс продажу	REAL	Не пусте
date	Дата встановлення	INTEGER	Не пусте

Рис.3.6. Структура таблиці «Exch_rate»

Таблиця «Curr_exch_oper» містить інформацію про валютні операції.

Найменування поля	Опис поля	Тип	Дод. умови
id	Код валюти	INTEGER	Не пусте Первинний ключ Автолічильник Унікальне
id_shift	Код зміни	INTEGER	Не пусте
type_operation	Тип операції	TEXT	Не пусте
date	Дата	INTEGER	Не пусте
curr_amou	Кількість валюти	REAL	Не пусте
id_rate	Код курсу	INTEGER	Не пусте

Рис.3.7. Структура таблиці «Curr_exch_oper»

В таблиці «Shift» міститься інформація про зміни, їх номери, дати відкриття та закриття.

Найменування поля	Опис поля	Тип	Дод. умови
id	Код зміни	INTEGER	Не пусте Первинний ключ Автолічильник Унікальне
number	Номер зміни	INTEGER	Не пусте
date_start	Дата відкриття	INTEGER	Не пусте
date_stop	Дата закриття	INTEGER	За умовчанням: NULL
id_user	Код користувача	INTEGER	Не пусте

Рис.3.8. Структура таблиці «Shift»

Таблиця «Cash» містить інформацію про залишок валюти в касі.

Найменування поля	**Опис поля**	**Тип**	**Дод. умови**
id	Код валюти	INTEGER	Не пусте Первинний ключ Автолічильник Унікальне
id_shift	Код зміни	INTEGER	Не пусте
id_curr	Код валюти	INTEGER	Не пусте
start_day	Залишок на початок дня	REAL	
cash_in	Підкріплення	REAL	
cash_out	Інкасація	REAL	
end_day	Залишок на кінець дня/залишок	REAL	

Рис.3.9. Структура таблиці «Cash»

Наступним кроком є створення бази даних, яка буде називатися «obmen». Це буде реалізовано за допомогою команди “CREATE DATABASE obmen;”.

Після цього буде створена фізична модель бази даних, яка є специфічною відповідно до синтаксису створення таблиць системи управління базами даних SQLite. Усі таблиці з атрибутами зображені на рисунку нижче (рис.3.10).

Атрибут	Тип данных	Атрибут	Тип данных
Таблиця “curr”		Таблиця “shift”	
id	int	id	int
name	text	number	int
Таблица “cash"		date_start	int
id	int	date_stop	int
id_shift	int	id_user	int
id_curr	int	Таблиця “users”	
start_day	real	id	int
cash_in	real	login	text
cash_out	real	password	text
end_day	real	name	text
Таблиця “curr_exch_oper”		Таблиця “exch_rate”	
id	int	id	int
id_shift	int	id_curr	int
type_operation	text	buy	real
data	int	sell	real
curr_amou	real	date	text
id_rate	int		

Рис.3.10. Усі таблиці та їх атрибути

3.3. Структура програми

Програмне забезпечення для обміну валютних операцій буде підтримувати наступні функції:

- Встановлення курсів;
- Відображення списку користувачів;
- Підтвердження операції;
- Головне меню;

На рисунку нижче зображений схематичний план головного меню (рис. 3.11).

Рис. 3.11. Інтерфейс головного меню

Наступний рисунок (рис 3.12) зображує деякий схематичний план вводу в програму.

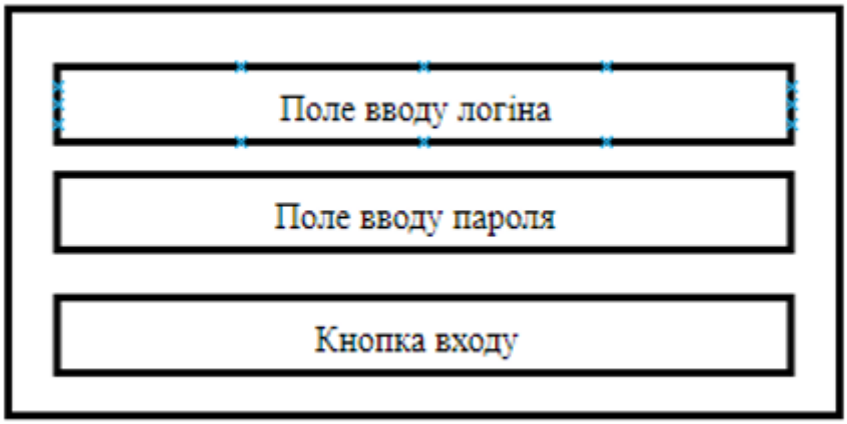

Рис. 3.12. Інтерфейс входу в програму

Рисунок нижче (рис. 3.13) представляє схематичний план проведення операції.

Рис.3.13. Інтерфейс проводження операції

В даному програмному продукті буде використано ряд модулів та класів QT, серед яких:

- QPrinter – пристрій для малювання в принтер;
- QAxContainer – дозволяє працювати з елементами управління ActiveX;
- QWidget – базовий модуль для всіх об’єктів інтерфейсу користувача;
- QMessageBox – надає діалогове вікно для інформування користувача та отримання відповіді;
- QSQLQuery – використовується для управління SQL виразами та їх виконання;
- QSQLDatabase – забезпечує з’єднання з базою даних;
- QDebug – дозволяє розробнику відладжувати програмне забезпечення;
- QDateTime – поєднання класів QDate та QTime. Надає функції для дати і часу.

Нижче перераховані основні класи, які будуть розбивати дане програмне забезпечення на певні підзадачі:

- MainWindow – клас головного меню, необхідного для всіх основних функцій ПЗ;
- DataBase – клас для взаємодії з базою даних. Містить всі необхідні команди для роботи з БД;
- Loggin – клас для автентифікації, який здійснює перевірку логіну та паролю, перевірка наявності відкритої зміни, перевірка стану БД та РРО;
- OleAutomation – клас для взаємодії з РРО, в якому описані всі команди необхідні для роботи з ним;
- BuySellWindow – клас, який представляє форму проведення операції, в якій користувач записує кількість валюти, яку потрібно купити або продати.

Також для графічного інтерфейсу будуть використані наступні компоненти:

- Qlabel – мітка, яка містить текстовий вміст;
- Qbutton – кнопка, яка реагує на подію Click;
- QSaveFileDialog – елемент управління, призначений для збереження документів;
- QTableView – елемент для відображення табличних даних;
- QTextBox – поле, призначене для введення інформації.

3.4. Тестування

Процес тестування іншими словами можна назвати процесом виявлення помилок. Деякі помилки можуть бути легко виявлені під час першого ж запуску програми, адже помітні неозброєним оком та не вимагають використання жодних спеціальних засобів. Наприклад, помилки графічного інтерфейсу досить легко виявити.

Інші помилки можуть вести себе неоднозначно, та не мати конкретних кроків для відтворення. Такі помилки знайти найважче.

В даному програмному забезпеченні тестування здійснюється за допомогою вбудованих засобів інтегрованого середовища QT Creator. Випробовування проводяться з метою розглянути всі можливі варіанти роботи програмного забезпечення та протестувати його у всіх можливих умовах. Загалом виділяють наступні умови роботи ПЗ:

- Нормальні;
- Виняткові;
- Екстримальні;

Результати тестування програмного продукту для проведення валютно-касових операцій описано на рисунку нижче (рис. 3.14)

Назва операції	**Подія**	**Очікуваний результат**	**Фактичний результат**
Захід під користувачем, існуючий у БД	Завантаження головної сторінки	Перехід на головну сторінку	Перехід на головну сторінку
Захід під користувачем, якого немає в БД	Вікно попередження про помилку логіну чи паролю	Попередження про помилку входу	Попередження про помилку входу
Запуск програми без БД в корні програми	Вікно попередження про помилку підключення до БД	Попередження про помилку підключення БД	Попередження про помилку підключення БД
Відкриття зміни	Завантаження залишків	Відображення курсів та залишків	Відображення курсів та залишків
Встановлення курсів	Заповнення потрібних форм та натиск кнопки зберегти	Курси встановлені	Курси встановлені
Додавання нової операції	Введення інформації та натиск кнопки прийняти	Додавання операції, друк квитанції РРО та оновлення залишків	Додавання операції, друк квитанції РРО та оновлення залишків
Закриття зміни	Відображення звітів на друк та друк звітів РРО	Друк звітів	Друк звітів
Закриття програми	Попередження про закриття	Повідомлення про закриття	Повідомлення про закриття
Спроба зайти користувачем, якщо зміна відкрита іншим користувачем	Попередження про відкриту зміну іншим користувачем	Повідомлення про відкриту зміну	Повідомлення про відкриту зміну

Рис. 3.14. Результати тестування

3.5. Опис тестування

Призначення даного програмного продукт полягає у веденні обліку операцій обмінного пункту, здійснення друку звітів та квитанцій.

Головним завданням є спрощення роботи співробітників обмінно-валютного пункту. Дані можуть буи легко структуровані та систематизовані, завдяки наявності спеціальних функцій. Це робить роботу з додатком максимально зручною та ефективною.

Мінімальні системні вимоги для роботи з ПЗ валютно-обмінних операцій:

- ОС Windows 8;
- CPUx86 / x64 Intel або AMD 1GHz;
- Розширення екран 800x600;
- Відеоадаптер з підтримкою кольорової палітри 16 біт або вище;
- 250мб вільного місця на жорсткому диску;

Користувач повинен ввести логін та пароль після запуску додатку (рис. 3.15).

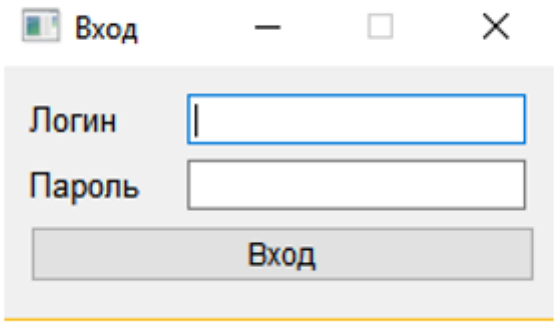

Рис. 3.15. Форма логіну

Після введення логіну та паролю натискаємо кнопку «Вхід», яка запускає процес аутентифікації. При правильно введених даних, відкривається головне вікно. При відсутності відкритих змін, таблиця залишків не активна, тому користувач не може проводити операції.

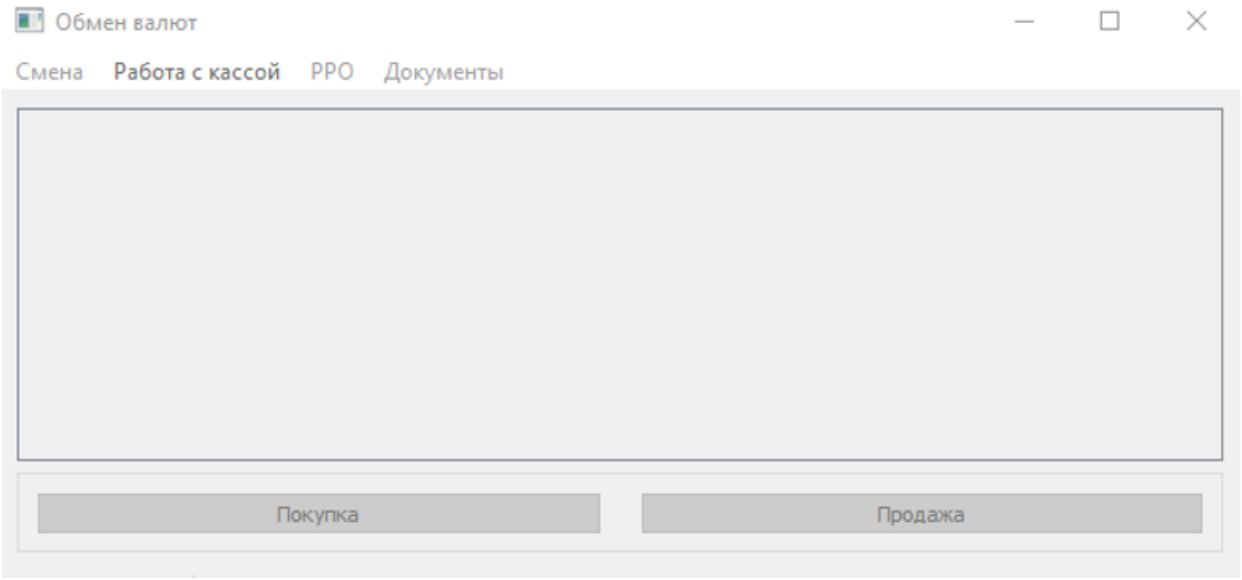

Рис.3.16. Головне вікно закритої зміни

Натискаємо «Зміна» - «Відкрити зміну». Стають активними кнопки «Купити» та «Продати», з’являються назви валют, курси та залишки.

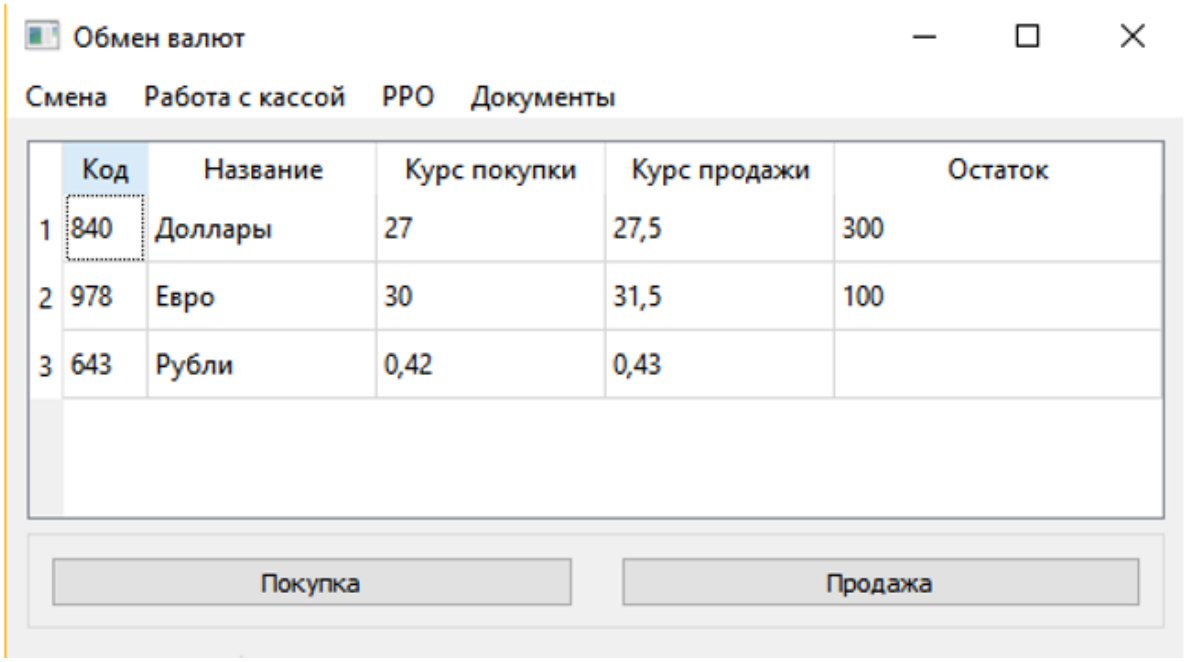

Рис. 3.17. Головне вікно відкритої зміни

Далі натискаємо на валюту, яку хочемо обміняти, та обираємо операцію – купівлю або продаж. З’являється вікно вводу кількості валюти, її кількість та курс.

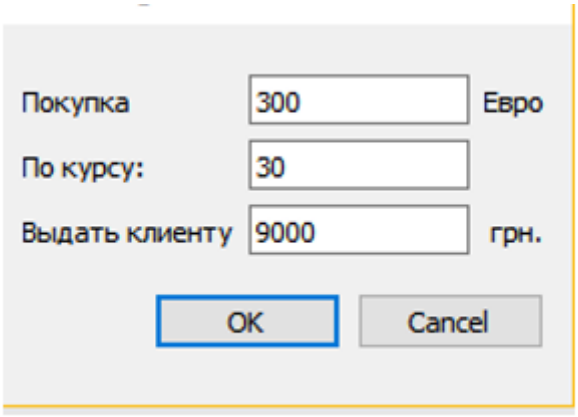

Рис. 3.18. Форма проведення операції обміну

Після введеня потрібної кількості, підтвердження операції, отримуємо РРО чек та потрапляємо до головного меню.

Тепер можна закрити зміну, для цього натискаємо «Зміна» - «Закрити зміну». Після цього «Реєстр покупки», «Реєстр продажу» та «Звітна довідка про касові обороти за день» друкуються на принтері. На РРО друкується Z-звіт.

ТОВ "ПРЕМІУМ ФІНАНС",Операційна каса №1
м. Київ, TEST

Реєстр
купленої іноземної валюти
за 10 січня 2019 року Зміна № 1

N з/п	Час здійснення операції	Назва іноземної валюти (код)	Сума іноземної валюти	Курс	Сума виданих гривень	Номер виданої довідки/квитанції/ платіжного пристрою	Відмітка про сторно
1	09:03:57	978-EUR	300,00	30	9000,00	1	

Всього:
978-EUR 300,00
980-UAH 9000,00

Підпис касира _____ /Кузнецов Максим /
Місце для відбитка штампа*

*Для електронного документа, який має електронний підпис касира, реквізит не проставляється.

Рис. 3.19. Звіт «Реєстр купленої валюти»

ТОВ "ПРЕМІУМ ФІНАНС",Операційна каса №1
м. Київ, TEST

Реєстр
проданої іноземної валюти
за 10 січня 2019 року Зміна № 1

N з/п	Час здійснення операції	Номер виданої квитанції	Сума в гривнях	Назва іноземної валюти (код)	Сума іноземної валюти	Курс	Номер одержаної довідки	Відмітка про сторно

Підпис касира _____ / Кузнецов Максим /

Місце для відбитка штампа*

_____

*Для електронного документа, який має електронний підпис касира, реквізит не проставляється.

Рис. 3.20. Звіт «Реєстр проданої валюти»

ТОВ "ПРЕМІУМ ФІНАНС",Операційна каса №1
м. Київ, TEST

Звітна довідка про касові обороти за день і залишки цінностей
за 11 січня 2019 року Зміна:1

№ з/п	Код валюти	Залишок готівки в касі відокремленого підрозділу, пункті обміну валюти на початок дня	Отримано валюти		Куплено відокремленим підрозділом, пунктом обміну валюти іноземної валюти	Продано відокремленим підрозділом, пунктом обміну валюти іноземної валюти	Передано валюти		Залишок готівки в касі відокремленого підрозділу, пункті обміну валюти на поточний момент часу/кінець робочого дня
			аванс на початок робочого дня	підкріплення протягом робочого дня			на кінець робочого дня	протягом робочого дня	
1	2	3	4	5	6	7	8	9	10
1	643	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
2	840	200,00	0,00	0,00	0,00	0,00	0,00	0,00	200,00
3	978	100,00	0,00	0,00	300,00	0,00	0,00	0,00	400,00
4	980	10000,00	0,00	0,00	0,00	9000,00	0,00	0,00	1000,00

Касир _____ / Кузнецов Максим /
(підпис)

Рис. 3.21. Звіт «Звітна оборотка про касові оборотки за день та залишки цінностей»

Таким чином, розроблений програмний продукт відповідає всім сучасним вимогам до оформлення та є простим і зрозумілим у використанні.

Висновок до розділу 3

Розробка ПЗ валютно-касових операцій була здійснена у даному розділі. Для цього була використана біблотека QT та СУБД SQLite.

Була розроблена база даних «bank», яка налічує 6 таблиць: users, exch_rate, cash_curr, shift, cash, curr_exch_oper, curr.

Наступні компоненти були використані для взаємодії з користувачем:

- Qlabel – мітка, яка містить текстовий вміст;
- Qbutton – кнопка, яка реагує на подію Click;
- QTableView – елемент для відображення табличних даних;
- QTextBox – поле, призначене для введення інформації.

Тестування підтвердило повну працездатність даного програмного забезпечення.

ВИСНОВКИ

Засіб для обліку обмінно-валютних операцій було розроблено у даному дипломному проекті.

Відмінністю від інших аналогів є забезпечення простоти у навігації по списку всіх доступних функцій та швидкий доступ в будь-яку точку прикладання додатку.

Програма надає користувачеві наступні функції:

- Встановлення курсів;
- Взаємодія з РРО;
- Перегляд залишків;
- Система автентифікації;
- Можливість додавання нової операції;
- Роздрукування звітів по історії.

Були оброблені всі варіанти, які може допустити користувач при роботі з додатком – від неправильно автентифікації, до розходження дати РРО з датою комп’ютера.

Таким чином, мета досягнута – завдання виконано в повному обсязі. Програмний продукт підтримує всі необхідні функції для правильного обліку валютних операцій.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Вікіпедія: Автоматизація [Електронний ресурс]: режим доступу: http://uk.wikipedia.org/wiki/ (дата звернення 24.05.2021р).– Назва з екрана.

2. Вікторія: Програмне забезпечення персональних комп'ютерів [Електронний ресурс]: режим доступу: http://www.victoria.lviv.ua/ (дата звернення 21.05.2021р).– Назва з екрана.

4. Эллис М. Справочное руководство по языку C++ с комментариями / Эллис М., Строуструп Б. : пер. с англ. – М. : Мир, 2014.: 445 с.

5. Ирэ Пол Объектно-ориентированное программирование с использованием C++ / Ирэ Пол : пер. с англ. – Киев : НИИПФ ДиаСофт Лтд, 2015.: – 480 с.

6. Шлее М. Qt 4:Профессиональное программирование на C++ / Шлее М.: СПб., .БХВ– Петербург., 2007.: 880 с.

7. Вікіпедія: Програмне забезпечення [Електронний ресурс]: режим доступу: http://uk.wikipedia.org/wiki/ (дата звернення 21.05.2021р).– Назва з екрана.

8. Вікіпедія: Розробка програмного забезпечення [Електронний ресурс]: режим доступу: http://uk.wikipedia.org/wiki/ (дата звернення 24.05.2021р).– Назва з екрана.

9. Вікіпедія: Qt Creator [Електронний ресурс]: режим доступу: http://uk.wikipedia.org/wiki/ (дата звернення 26.05.2021р).– Назва з екрана.

ДОДАТКИ

Додаток А

```
#ifndef BUYSELLWINDOW_H
#define BUYSELLWINDOW_H
#include <QDialog>

namespace Ui {
class BuySellWindow;
}

class BuySellWindow : public QDialog
{
   Q_OBJECT

public:
   explicit BuySellWindow(QString typeOper = "", QString val= "", double kurs = 0.,
QString klient= "", QWidget *parent = nullptr);
   ~BuySellWindow();

private:
   Ui::BuySellWindow *ui;
};

#endif // BUYSELLWINDOW_H



#ifndef DATABASE_H
#define DATABASE_H

#include <QtCore/QCoreApplication>
#include <QtNetwork>
#include <QString>
#include <QMessageBox>
#include <QDebug>
```

```
#include <QtSql/QSqlDatabase>
#include <QtSql/QSqlQuery>
#include <QFileInfo>
#include <QTime>

class DataBase
{
public:
    DataBase();
    ~DataBase();
    bool checkInternet();
    bool loggin(QString login, QString password);
    bool openDB();
    QString getUserName();
    QDateTime getStartShgift();
    qint8 getNumberShift();
    bool shiftIsOpen();
    bool shifterVSUser();
    QSqlQuery* updateTable();
    qint8 getShiftID();
    QDateTime getLastExRate();
    bool preBuy(QString kod);
    bool preSell(QString kod);
    QString getBufferS();
    double getBufferD();


private:
    QSqlDatabase mydb;
    QSqlQuery *qry;
    QString name;
    //QString login;//
    qint8 userID;
    QDateTime startShift;
    qint8 numberShift;
```

```
    qint8 shiftID;
    QDateTime lastExRate;
    QString buffString;
    qint8 buffInt;
    double buffDouble;
};

#endif // DATABASE_H

#ifndef LOGGIN_H
#define LOGGIN_H

#include "mainwindow.h"
#include "database.h"
#include <QWidget>
#include <QtWidgets>

namespace Ui {
class Loggin;
}

class Loggin : public QWidget
{
    Q_OBJECT

public:
    explicit Loggin(QWidget *parent = nullptr);
    ~Loggin();

private slots:

    void on_pushButton_loggin_clicked();

    void on_lineEdit_login_returnPressed();
```

```
    void on_lineEdit_password_returnPressed();

private:
    Ui::Loggin *ui;
    DataBase* db;
};

#endif // LOGGIN_H



#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QMessageBox>
#include <QCloseEvent>
#include <database.h>
#include <QLabel>
#include <QStatusBar>
#include <QSqlQueryModel>
#include "oleautomation.h"
#include "buysellwindow.h"

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = nullptr, DataBase *db_l = nullptr);
    ~MainWindow();
```

```
private slots:
    void on_actionX_triggered();

    void on_actionZ_triggered();

    void on_pushButton_buy_clicked();

    void on_tableView_activated(const QModelIndex &index);

private:
    Ui::MainWindow *ui;
    void closeEvent(QCloseEvent*);
    QLabel *userName;
    OleAutomation* ppo;
    DataBase* db;
};

#endif // MAINWINDOW_H

#ifndef OLEAUTOMATION_H
#define OLEAUTOMATION_H

#include <ActiveQt/qaxobject.h>

enum typeOper {buy = 1, returnBuy, sell, returnSell};
enum typeSerOper {avans = 1, podkrep, inkas};

class OleAutomation
{
public:
    OleAutomation(QObject*);
    ~OleAutomation();
    bool initAuto(QString name); //Осуществляет поиск и установку соединения с ЭККА
    bool setExRate(qint8 currency_code, double buyrate, double sellrate, double regrate);
//Устанавливает курс национальной валюты к иностранной
```

```
        bool startConfigureCurrencies(); //Открытие процедуры программирования классификатора иностранных валют
        bool clearAllCurrencies(); //Очистка всех объектов классификатора валюты при программировании в фискальную память
        bool addCurrency(qint8 index , qint8 currency_code , QString literal_currency_code, QString name); //Добавляет валюту в список валют, программируемых в фискальную память командой
        bool commitCurrencies();//Закрытие процедуры программирования классификатора иностранных валют
        bool setInternalTime(qint8 hours , qint8 minutes , qint8 seconds); //Изменяет текущее время ЭККА
        bool zReport(); //Закрывает фискальную смену и печатает Z-отчёт
        bool xReport(); //Печатает X-отчёт
        bool nullCheck(); //Печатает "нулевой" чек
        bool checkCopy();//Печатает копию последнего чека
        bool registerExchenge(typeOper to, double sum1, qint8 currency_code, double sum2); //Регистрация валютообменной операции, печать чека
        bool openServiceOperation(typeSerOper tso); //Открывает служебную операцию аванса/подкрепления/инкассация ценностей
        bool addServiceOperationItem(double sum, qint8 currency_code); //Добавляет позицию в служебную операцию аванса/подкрепления/инкассации ценностей
        bool closeServiceOperation(); //Закрытие служебной операции аванса/подкрепления/инкассации ценностей и печать документа
        void settlementAsync(); //Иниициирует фоновую отправку данных на сервер ДПА
        bool done(); //Закрывает соединение с ЭККА
    private:
        QAxObject *ppo;
    };

    #endif // OLEAUTOMATION_H
```

```
#include "buysellwindow.h"
#include "ui_buysellwindow.h"

BuySellWindow::BuySellWindow(QString typeOper, QString val, double kurs, QString
klient, QWidget *parent) :
    QDialog(parent),
    ui(new Ui::BuySellWindow)
{
    ui->setupUi(this);

    ui->label_5->setText(typeOper);
    ui->label_6->setText(val);
    ui->label->setText("По курсу: ");
    ui->label_2->setText("");
    ui->label_3->setText(klient);
    ui->label_4->setText(" грн.");
    ui->lineEdit->setText(QString::number(kurs));
    ui->lineEdit->setReadOnly(true);
}

BuySellWindow::~BuySellWindow()
{
    delete ui;
}

#include "database.h"

DataBase::DataBase()
{
    mydb = QSqlDatabase::addDatabase("QSQLITE", "mydb");
    mydb.setDatabaseName("D:/DOCUMENTS/Test8/database.db3");

    numberShift = 0;
}
```

```
DataBase::~DataBase()
{}

bool DataBase::checkInternet()
{
    QNetworkAccessManager nam;
    QNetworkRequest req(QUrl("http://www.google.com"));
    QNetworkReply *reply = nam.get(req);
    QEventLoop loop;
    QTimer timeoutTimer;
    QObject:: connect(&timeoutTimer, SIGNAL(timeout()), &loop, SLOT(quit));
    //QObject::connect(reply, SIGNAL(readyRead()), &loop, SLOT(quit()));
    QObject::connect(reply, SIGNAL(finished()), &loop, SLOT(quit()));
    timeoutTimer.setSingleShot(true);
    timeoutTimer.start(5000);
    //if(!reply->isFinished())
    loop.exec();
    if(reply->bytesAvailable())
        return true;
    else
        return false;
}

bool DataBase::loggin(QString login, QString password)
{
    qry = new QSqlQuery(mydb);
    if(qry->exec("SELECT name, id FROM users WHERE login = '" + login + "' AND
password = '" + password + "';")) {
        if(qry->next()) {
            name = qry->value(0).toString();
            userID = qry->value(1).toInt();
            //DataBase::login = qry->value(1).toString();//
            //QSqlDatabase::removeDatabase("mydb");
            delete qry;
            return true;
```

```
            } else {
                qDebug() << "Invalid ligin and/or password";
                delete qry;
                return false;
            }
        }
        qDebug() << "Invalid ligin and/or password";
        delete qry;
        return false;
    }

    bool DataBase::openDB()
    {
        QSqlDatabase::database("mydb");
        if(!mydb.open())
        {
            qDebug() << "Error connecting to the database";
            return false;
        }
        qry = new QSqlQuery(mydb);
        if(qry->exec("SELECT date_start, number, id FROM shift WHERE date_stop IS
NULL;")) {
            if(qry->next()) {
                //startShift = qry->value(0).toDateTime();
                startShift.setTime_t(qry->value(0).toUInt());
                numberShift = qry->value(1).toInt();
                shiftID = qry->value(2).toInt();
                qry->clear();
            }
        }
        delete qry;
        return true;
    }

    QString DataBase::getUserName()
```

```
{
    return name;
}

QDateTime DataBase::getStartShgift()
{
    return startShift;
}

qint8 DataBase::getNumberShift()
{
    return numberShift;
}

bool DataBase::shiftIsOpen()
{
    if(numberShift)
        return true;
    return false;
}

bool DataBase::shifterVSUser()
{
    openDB();
    qry = new QSqlQuery(mydb);
    qint8 shifterID = 0;
    if(qry->exec("SELECT id_user FROM shift WHERE date_start = '" +
QString::number(startShift.toTime_t()) + "';")) {
        if(qry->next()) {
            shifterID = qry->value(0).toInt();
            qDebug() << shifterID;
            qry->clear();
        }
    }
    delete qry;
```

```
    qDebug() << shifterID << "   " << userID;
    if(shifterID == userID)
        return true;
    return false;
}

QSqlQuery* DataBase::updateTable()
{
    openDB();
    QSqlQuery* qryTable = new QSqlQuery(mydb);

    qryTable->prepare("SELECT cash.id_curr, curr.name, exch_rate.buy, exch_rate.sell,
cash.end_day "
                      "FROM cash "
                      "LEFT JOIN curr ON curr.id = cash.id_curr "
                      "LEFT JOIN exch_rate ON exch_rate.id_curr = curr.id "
                      "WHERE cash.id_shift = '" + QString::number(shiftID) + "' AND
exch_rate.date = '" + QString::number(getLastExRate().toTime_t()) + "';");
    qryTable->exec();

    qDebug() << qryTable;
    return qryTable;
}

qint8 DataBase::getShiftID()
{
    return shiftID;
}

QDateTime DataBase::getLastExRate()
{
    QDateTime qdt;
    openDB();
    qry = new QSqlQuery(mydb);
    if(qry->exec("SELECT MAX(date) FROM exch_rate;")) {
```

```
        if(qry->next()) {
            qdt.setTime_t(qry->value(0).toUInt());
            qDebug() << qdt;
            return qdt;
        }
        return qdt.currentDateTime();
    }
    return qdt.currentDateTime();
}

bool DataBase::preBuy(QString kod)
{
    openDB();
    qry = new QSqlQuery(mydb);
    if(qry->exec("SELECT curr.name, exch_rate.buy FROM curr "
                 "LEFT JOIN exch_rate ON exch_rate.id_curr = curr.id "
                 "WHERE curr.id = '" + kod + "'"
                 "ORDER BY exch_rate.date DESC LIMIT 1;")) {
        qDebug() << kod;
        if(qry->next()) {
            DataBase::buffString = qry->value(0).toString();
            DataBase::buffDouble = qry->value(1).toDouble();
            qry->clear();
        }
        delete qry;
    }
    delete qry;
}

QString DataBase::getBufferS()
{
    return DataBase::buffString;
}

double DataBase::getBufferD()
```

```
{
    return DataBase::buffDouble;
}

#include "loggin.h"
#include "ui_loggin.h"

Loggin::Loggin(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::Loggin)
{
    ui->setupUi(this);
    db = new DataBase;
}

Loggin::~Loggin()
{
    delete ui;
}

void Loggin::on_pushButton_loggin_clicked()
{
    if(db->checkInternet())
        qDebug() << "Conected to internet";
    else {
        qDebug() << "No internet connection";
        QMessageBox::critical(this, "Ошибка", "Нет подключения к Интернету");
        return;
    }

    QString user_login = ui->lineEdit_login->text();
    QString user_password = ui->lineEdit_password->text();
```

```
    if(!db->openDB()) {
        QMessageBox::critical(this, "Ошибка", "Ошибка подключения к базе данных!\n\nОбратитесь к администратору");
        return;
    }

    if(db->loggin(user_login, user_password)) {
        if(db->shifterVSUser() || !db->shiftIsOpen()) {
            MainWindow *w = new MainWindow(nullptr,db);
            w->show();
            this->close();
        } else {
            QMessageBox::warning(this, "От халепа!", "Уже есть открытая смена под другим пользователем");
                return;
        }
    } else {
        QMessageBox::warning(this, "Авторизация", "Неверный логин и/или пароль");
        return;
    }
}

void Loggin::on_lineEdit_login_returnPressed()
{
    ui->pushButton_loggin->setFocus();
}

void Loggin::on_lineEdit_password_returnPressed()
{
    ui->pushButton_loggin->setFocus();
}

#include "loggin.h"
#include "mainwindow.h"
#include <QApplication>
```

```
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    Loggin login;
    login.show();
    return a.exec();
}

#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent, DataBase*db_l) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    ppo = new OleAutomation(this);
    MainWindow::db = db_l;

    userName = new QLabel(this);
    userName->setText(db->getUserName());
    statusBar()->addWidget(userName);

    if(db->shiftIsOpen()) {
        //подгрузить остатки и открыть кнопки
        ui->openShift->setVisible(false);
        ui->closeShift->setEnabled(true);
        ui->menu_2->setEnabled(true);
        ui->tableView->setEnabled(true);
        ui->pushButton_sell->setEnabled(true);
        ui->pushButton_buy->setEnabled(true);
        QSqlQueryModel * model = new QSqlQueryModel();
        model->setQuery(*(db->updateTable()));
```

```
        ui->tableView->setModel(model);
        model->setHeaderData(0, Qt::Horizontal, QObject::tr("  Код  "));
        ui->tableView->setColumnWidth(0, 15);
        model->setHeaderData(1, Qt::Horizontal, QObject::tr("Название"));
        ui->tableView->setColumnWidth(1, 100);
        model->setHeaderData(2, Qt::Horizontal, QObject::tr("Курс покупки"));
        ui->tableView->setColumnWidth(2, 100);
        model->setHeaderData(3, Qt::Horizontal, QObject::tr("Курс продажи"));
        ui->tableView->setColumnWidth(3, 100);
        model->setHeaderData(4, Qt::Horizontal, QObject::tr("Остаток"));
        ui->tableView->setColumnWidth(4, 100);
        ui->tableView->setSelectionMode(QAbstractItemView::SingleSelection);
        ui->tableView->setSelectionBehavior(QAbstractItemView::SelectRows);
        ui->tableView->horizontalHeader()->setStretchLastSection(true);
        //ui->tableView->resizeColumnsToContents();
        qDebug() << (model->rowCount());
    } else {
        //не открыта смена
        ui->closeShift->setVisible(false);

    }
}

MainWindow::~MainWindow()
{
    delete ui;
    delete db;
    delete ppo;
    delete userName;
}

void MainWindow::MainWindow::closeEvent(QCloseEvent * event)
{
    QMessageBox::StandardButton replay = QMessageBox::question(this, " ", "Вы действи-
тельно хотите выйти?", QMessageBox::Yes | QMessageBox::No);
```

```
    (replay == QMessageBox::Yes) ? (event->accept()) : (event->ignore());
}

void MainWindow::on_actionX_triggered()
{
    if(!ppo->initAuto(db->getUserName()))
        return;
    ppo->xReport();
    ppo->done();
}

void MainWindow::on_actionZ_triggered()
{
    if(ppo->initAuto(db->getUserName()))
        return;
    ppo->zReport();
    ppo->done();
}



void MainWindow::on_pushButton_buy_clicked()
{
    db->preBuy(ui->tableView->selectionModel()-
>selectedRows().value(0).data().toString());//возвращение кода выбраной валюты
    qDebug() << ui->tableView->selectionModel()-
>selectedRows().value(0).data().toString() << " KOD";
    qDebug() << db->getBufferS();
    qDebug() << db->getBufferD();
    BuySellWindow window("Покупка", db->getBufferS(), db->getBufferD(), "Выдать
клиенту");
    window.setModal(true);
    window.exec();
}
```

```
void MainWindow::on_tableView_activated(const QModelIndex &index)
{

}

#include "oleautomation.h"

OleAutomation::OleAutomation(QObject* parent)
{
   ppo = new QAxObject("Resonance.ForeignExchange", parent);
}

OleAutomation::~OleAutomation()
{
   delete ppo;
}

bool OleAutomation::initAuto(QString name)
{
   //Осуществляет поиск и установку соединения с ЭККА
   return ppo->dynamicCall("InitAuto(string, string)", name).toBool();
}

bool OleAutomation::setExRate(qint8 currency_code, double buyrate, double sellrate,
double regrate)
{
   //Устанавливает курс национальной валюты к иностранной
   //Для наказа вызывать несколько раз для каждой валюты
   return ppo->dynamicCall("SetExchangeRate(int,decimal, decimal, decimal)",
currency_code, buyrate, sellrate, regrate).toBool();
}

bool OleAutomation::startConfigureCurrencies()
{
```

```
    //Открытие процедуры программирования классификатора иностранных валют
    return ppo->dynamicCall("StartConfigureCurrencies()").toBool();
}

bool OleAutomation::clearAllCurrencies()
{
    //Очистка всех объектов классификатора валюты при программировании в фискальную память
    return ppo->dynamicCall("ClearAllCurrencies()").toBool();
}

bool OleAutomation::addCurrency(qint8 index, qint8 currency_code, QString literal_currency_code, QString name)
{
    //Добавляет валюту в список валют, программируемых в фискальную память командой
    //Вызывать несколько раз для каждой валюты. После закрыть командой CommitCurrencies()
    return ppo->dynamicCall("AddCurrency(int, int, string, string)", index, currency_code, literal_currency_code, name).toBool();
}

bool OleAutomation::commitCurrencies()
{
    //Закрытие процедуры программирования классификатора иностранных валют
    return ppo->dynamicCall("CommitCurrencies()").toBool();
}

bool OleAutomation::setInternalTime(qint8 hours, qint8 minutes, qint8 seconds)
{
    //Изменяет текущее время ЭККА
    return ppo->dynamicCall("SetInternalTime(int, int, int)", hours, minutes, seconds).toBool();
}
```

```
bool OleAutomation::zReport()
{
   //Закрывает фискальную смену и печатает Z-отчёт
   return ppo->dynamicCall("ZReport()").toBool();
}

bool OleAutomation::xReport()
{
   //Печатает X-отчёт
   return ppo->dynamicCall("XReport()").toBool();
}

bool OleAutomation::nullCheck()
{
   //Печатает "нулевой" чек
   return ppo->dynamicCall("NullCheck()").toBool();
}

bool OleAutomation::checkCopy()
{
   //Печатает копию последнего чека
   return ppo->dynamicCall("CheckCopy()").toBool();
}

bool OleAutomation::registerExchenge(typeOper to, double sum1, qint8 currency_code,
double sum2)
{
   //Регистрация валютообменной операции, печать чека
   return ppo->dynamicCall("RegisterExchange(int, decimal, int, decimal", to, sum1,
currency_code, sum2).toBool();
}

bool OleAutomation::openServiceOperation(typeSerOper tso)
{
   //Открывает служебную операцию аванса/подкрепления/инкассация ценностей
```

```
    return ppo->dynamicCall("OpenServiceOperation(int)", tso).toBool();
}

bool OleAutomation::addServiceOperationItem(double sum, qint8 currency_code)
{
    //Добавляет позицию в служебную операцию аванса/подкрепления/инкассации ценностей
    return ppo->dynamicCall("AddServiceOperationItem(decimal, int)", sum, currency_code).toBool();
}

bool OleAutomation::closeServiceOperation()
{
    //Закрытие служебной операции аванса/подкрепления/инкассации ценностей и печать документа
    return ppo->dynamicCall("CloseServiceOperation()").toBool();
}

void OleAutomation::settlementAsync()
{
    //Иниициирует фоновую отправку данных на сервер ДПА
    ppo->dynamicCall("SettlementAsync");
}

bool OleAutomation::done()
{
    //Закрывает соединение с ЭККА
    return ppo->dynamicCall("Done()").toBool();
}
```

Форми

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<ui version="4.0">
 <class>BuySellWindow</class>
 <widget class="QDialog" name="BuySellWindow">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>400</width>
    <height>300</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>Dialog</string>
  </property>
  <layout class="QVBoxLayout" name="verticalLayout">
   <item>
    <layout class="QGridLayout" name="gridLayout">
     <item row="1" column="1">
      <widget class="QLineEdit" name="lineEdit"/>
     </item>
     <item row="1" column="2">
      <widget class="QLabel" name="label_2">
       <property name="text">
        <string>TextLabel</string>
       </property>
      </widget>
     </item>
     <item row="1" column="0">
      <widget class="QLabel" name="label">
       <property name="text">
        <string>TextLabel</string>
       </property>
      </widget>
     </item>
     <item row="2" column="1">
```

```
 <widget class="QLineEdit" name="lineEdit_2"/>
</item>
<item row="0" column="1">
 <widget class="QLineEdit" name="lineEdit_3"/>
</item>
<item row="2" column="0">
 <widget class="QLabel" name="label_3">
  <property name="text">
   <string>TextLabel</string>
  </property>
 </widget>
</item>
<item row="2" column="2">
 <widget class="QLabel" name="label_4">
  <property name="text">
   <string>TextLabel</string>
  </property>
 </widget>
</item>
<item row="0" column="0">
 <widget class="QLabel" name="label_5">
  <property name="text">
   <string>TextLabel</string>
  </property>
 </widget>
</item>
<item row="0" column="2">
 <widget class="QLabel" name="label_6">
  <property name="text">
   <string>TextLabel</string>
  </property>
 </widget>
</item>
</layout>
</item>
```

```
 <item>
  <widget class="QDialogButtonBox" name="buttonBox">
   <property name="orientation">
    <enum>Qt::Horizontal</enum>
   </property>
   <property name="standardButtons">
    <set>QDialogButtonBox::Cancel|QDialogButtonBox::Ok</set>
   </property>
  </widget>
 </item>
</layout>
</widget>
<resources/>
<connections>
 <connection>
  <sender>buttonBox</sender>
  <signal>accepted()</signal>
  <receiver>BuySellWindow</receiver>
  <slot>accept()</slot>
  <hints>
   <hint type="sourcelabel">
    <x>248</x>
    <y>254</y>
   </hint>
   <hint type="destinationlabel">
    <x>157</x>
    <y>274</y>
   </hint>
  </hints>
 </connection>
 <connection>
  <sender>buttonBox</sender>
  <signal>rejected()</signal>
  <receiver>BuySellWindow</receiver>
  <slot>reject()</slot>
```

```
  <hints>
   <hint type="sourcelabel">
    <x>316</x>
    <y>260</y>
   </hint>
   <hint type="destinationlabel">
    <x>286</x>
    <y>274</y>
   </hint>
  </hints>
 </connection>
</connections>
</ui>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>Loggin</class>
 <widget class="QWidget" name="Loggin">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>220</width>
    <height>100</height>
   </rect>
  </property>
  <property name="sizePolicy">
   <sizepolicy hsizetype="Fixed" vsizetype="Fixed">
    <horstretch>0</horstretch>
    <verstretch>0</verstretch>
   </sizepolicy>
```

```
</property>
<property name="minimumSize">
 <size>
  <width>220</width>
  <height>100</height>
 </size>
</property>
<property name="maximumSize">
 <size>
  <width>220</width>
  <height>100</height>
 </size>
</property>
<property name="cursor">
 <cursorShape>ArrowCursor</cursorShape>
</property>
<property name="contextMenuPolicy">
 <enum>Qt::NoContextMenu</enum>
</property>
<property name="windowTitle">
 <string>Вход</string>
</property>
<layout class="QVBoxLayout" name="verticalLayout">
 <item>
  <layout class="QHBoxLayout" name="horizontalLayout">
   <item>
    <widget class="QLabel" name="label_login">
     <property name="sizePolicy">
      <sizepolicy hsizetype="Fixed" vsizetype="Fixed">
       <horstretch>0</horstretch>
       <verstretch>0</verstretch>
      </sizepolicy>
     </property>
     <property name="minimumSize">
      <size>
```

```
      <width>57</width>
      <height>20</height>
     </size>
    </property>
    <property name="maximumSize">
     <size>
      <width>57</width>
      <height>20</height>
     </size>
    </property>
    <property name="font">
     <font>
      <family>Arial</family>
      <pointsize>10</pointsize>
     </font>
    </property>
    <property name="text">
     <string>Логин</string>
    </property>
   </widget>
  </item>
  <item>
   <widget class="QLineEdit" name="lineEdit_login"/>
  </item>
 </layout>
</item>
<item>
 <layout class="QHBoxLayout" name="horizontalLayout_2">
  <item>
   <widget class="QLabel" name="label_password">
    <property name="sizePolicy">
     <sizepolicy hsizetype="Fixed" vsizetype="Fixed">
      <horstretch>0</horstretch>
      <verstretch>0</verstretch>
     </sizepolicy>
```

```
  </property>
  <property name="minimumSize">
   <size>
    <width>57</width>
    <height>20</height>
   </size>
  </property>
  <property name="maximumSize">
   <size>
    <width>57</width>
    <height>20</height>
   </size>
  </property>
  <property name="font">
   <font>
    <family>Arial</family>
    <pointsize>10</pointsize>
   </font>
  </property>
  <property name="text">
   <string>Пароль</string>
  </property>
 </widget>
</item>
<item>
 <widget class="QLineEdit" name="lineEdit_password">
  <property name="autoFillBackground">
   <bool>false</bool>
  </property>
  <property name="inputMethodHints">
<set>Qt::ImhHiddenText|Qt::ImhNoAutoUppercase|Qt::ImhNoPredictiveText|Qt::ImhSensitiveDat
a</set>
  </property>
  <property name="echoMode">
```

```
    <enum>QLineEdit::Password</enum>
   </property>
  </widget>
 </item>
</layout>
</item>
<item>
 <widget class="QPushButton" name="pushButton_loggin">
  <property name="minimumSize">
   <size>
    <width>202</width>
    <height>23</height>
   </size>
  </property>
  <property name="maximumSize">
   <size>
    <width>202</width>
    <height>23</height>
   </size>
  </property>
  <property name="font">
   <font>
    <family>Arial</family>
    <pointsize>9</pointsize>
   </font>
  </property>
  <property name="focusPolicy">
   <enum>Qt::StrongFocus</enum>
  </property>
  <property name="text">
   <string>Вход</string>
  </property>
  <property name="autoDefault">
   <bool>true</bool>
  </property>
```

```
    <property name="default">
     <bool>false</bool>
    </property>
    <property name="flat">
     <bool>false</bool>
    </property>
   </widget>
  </item>
 </layout>
</widget>
<layoutdefault spacing="6" margin="11"/>
<resources/>
<connections/>
</ui>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>800</width>
    <height>600</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>Обмен валют</string>
  </property>
  <widget class="QWidget" name="centralwidget">
   <layout class="QVBoxLayout" name="verticalLayout">
```

```
<item>
 <widget class="QTableView" name="tableView">
  <property name="enabled">
   <bool>false</bool>
  </property>
  <property name="sizePolicy">
   <sizepolicy hsizetype="Expanding" vsizetype="Preferred">
    <horstretch>0</horstretch>
    <verstretch>100</verstretch>
   </sizepolicy>
  </property>
 </widget>
</item>
<item>
 <widget class="QGroupBox" name="groupBox">
  <property name="sizePolicy">
   <sizepolicy hsizetype="Preferred" vsizetype="Preferred">
    <horstretch>0</horstretch>
    <verstretch>10</verstretch>
   </sizepolicy>
  </property>
  <layout class="QHBoxLayout" name="horizontalLayout">
   <property name="spacing">
    <number>20</number>
   </property>
   <property name="sizeConstraint">
    <enum>QLayout::SetMaximumSize</enum>
   </property>
   <item>
    <widget class="QPushButton" name="pushButton_buy">
     <property name="enabled">
      <bool>false</bool>
     </property>
     <property name="sizePolicy">
      <sizepolicy hsizetype="Preferred" vsizetype="Preferred">
```

```
   <horstretch>3</horstretch>
   <verstretch>15</verstretch>
  </sizepolicy>
 </property>
 <property name="minimumSize">
  <size>
   <width>0</width>
   <height>0</height>
  </size>
 </property>
 <property name="sizeIncrement">
  <size>
   <width>0</width>
   <height>0</height>
  </size>
 </property>
 <property name="baseSize">
  <size>
   <width>0</width>
   <height>0</height>
  </size>
 </property>
 <property name="text">
  <string>Покупка</string>
 </property>
</widget>
</item>
<item>
 <widget class="QPushButton" name="pushButton_sell">
  <property name="enabled">
   <bool>false</bool>
  </property>
  <property name="sizePolicy">
   <sizepolicy hsizetype="Preferred" vsizetype="Preferred">
    <horstretch>3</horstretch>
```

```
        <verstretch>15</verstretch>
       </sizepolicy>
      </property>
      <property name="minimumSize">
       <size>
        <width>0</width>
        <height>0</height>
       </size>
      </property>
      <property name="sizeIncrement">
       <size>
        <width>5</width>
        <height>5</height>
       </size>
      </property>
      <property name="baseSize">
       <size>
        <width>0</width>
        <height>0</height>
       </size>
      </property>
      <property name="text">
       <string>Продажа</string>
      </property>
     </widget>
    </item>
   </layout>
  </widget>
 </item>
</layout>
</widget>
<widget class="QMenuBar" name="menubar">
 <property name="geometry">
  <rect>
   <x>0</x>
```

```
  <y>0</y>
  <width>800</width>
  <height>21</height>
 </rect>
</property>
<widget class="QMenu" name="menu">
 <property name="title">
  <string>PPO</string>
 </property>
 <addaction name="actionX"/>
 <addaction name="actionZ"/>
 <addaction name="separator"/>
 <addaction name="getListCur"/>
 <addaction name="insListCur"/>
 <addaction name="insRate"/>
</widget>
<widget class="QMenu" name="menu_2">
 <property name="enabled">
  <bool>false</bool>
 </property>
 <property name="title">
  <string>Работа с кассой</string>
 </property>
 <addaction name="moneyIn"/>
 <addaction name="moneyOut"/>
 <addaction name="setRate"/>
</widget>
<widget class="QMenu" name="menu_3">
 <property name="title">
  <string>Документы</string>
 </property>
 <widget class="QMenu" name="menu_5">
  <property name="title">
   <string>Реестры</string>
  </property>
```

```
  <addaction name="repBuy"/>
  <addaction name="repSell"/>
 </widget>
 <addaction name="menu_5"/>
 <addaction name="zvt"/>
 <addaction name="nakaz"/>
</widget>
<widget class="QMenu" name="menu_4">
 <property name="title">
  <string>Смена</string>
 </property>
 <addaction name="openShift"/>
 <addaction name="separator"/>
 <addaction name="closeShift"/>
</widget>
<addaction name="menu_4"/>
<addaction name="menu_2"/>
<addaction name="menu"/>
<addaction name="menu_3"/>
</widget>
<widget class="QStatusBar" name="statusbar"/>
<action name="getListCur">
 <property name="text">
  <string>Просмотр список и курсов валют</string>
 </property>
</action>
<action name="insListCur">
 <property name="text">
  <string>Загрузка списка валют</string>
 </property>
</action>
<action name="insRate">
 <property name="text">
  <string>Загрузка курсов валют</string>
 </property>
```

```
</action>
<action name="actionX">
 <property name="text">
  <string>X - отчет</string>
 </property>
</action>
<action name="actionZ">
 <property name="text">
  <string>Z - отчет</string>
 </property>
</action>
<action name="moneyIn">
 <property name="checkable">
  <bool>false</bool>
 </property>
 <property name="enabled">
  <bool>true</bool>
 </property>
 <property name="text">
  <string>Подкрепление</string>
 </property>
</action>
<action name="moneyOut">
 <property name="text">
  <string>Инкасация</string>
 </property>
</action>
<action name="openShift">
 <property name="text">
  <string>Открыть смену</string>
 </property>
</action>
<action name="closeShift">
 <property name="enabled">
  <bool>false</bool>
```

```
  </property>
  <property name="text">
   <string>Закрыть смену</string>
  </property>
 </action>
 <action name="repBuy">
  <property name="text">
   <string>Покупки</string>
  </property>
 </action>
 <action name="repSell">
  <property name="text">
   <string>Продажи</string>
  </property>
 </action>
 <action name="zvt">
  <property name="text">
   <string>Отчетная справка</string>
  </property>
 </action>
 <action name="nakaz">
  <property name="text">
   <string>Наказ</string>
  </property>
 </action>
 <action name="setRate">
  <property name="text">
   <string>Установка курсов</string>
  </property>
 </action>
</widget>
<resources/>
<connections/>
</ui>
```