НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ
УКРАЇНИ
Факультет інформаційних технологій

ПОГОДЖЕНО **Декан факультету**	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ** **Завідувач кафедри комп'ютерних наук**
_____Ігор БОЛБОТ (підпис)	**_____Белла ГОЛУБ** (підпис)
«___» _____2026 р.	«___» _____2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Інформаційна система обліку купівлі-продажу криптовалюти»
Спеціальність «122 Комп'ютерні науки»
Освітньо-професійна програма «Комп'ютерні науки»

Гарант освітньої програми д.е.н., професор	_____ (підпис)	**Роман РУДЕНСЬКИЙ**
Керівник бакалаврської кваліфікаційної роботи (науковий ступінь та вчене звання)	_____ (підпис)	**Віктор ПАНКРАТЬЄВ**
Виконав	_____ (підпис)	**Вадим ЛИТВИН**

Київ-2026

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БІОРЕСУРСІВ І
ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____Комп'ютерних наук_____
(назва кафедри)

___к.т.н.,доцент_____. Голуб Б.Л.
(науковий ступінь, вчене звання) (підпис) (ПІБ)
"_06" _січня___ 2026 р.

З А В Д А Н Н Я
на виконання бакалаврської кваліфікаційної роботи студенту
Литвину Вадиму Сергійовичу

Спеціальність 122 – «Комп'ютерні науки»
ОП «Комп'ютерні науки»

1. Тема бакалаврської кваліфікаційної роботи «Інформаційна система обліку купівлі-продажу криптовалюти» затверджена наказом ректора НУБіП України від 06.01.2026 №46 «С»
2. Термін подання завершеної роботи на кафедру_____2026.05.22_____
(рік, місяць, число)

3. Вихідні дані до роботи: механізм функціонування інформаційної системи купівлі-продажу криптовалюти, включаючи процеси реєстрації користувачів, KYC-верифікації, моніторингу курсів криптовалют, створення та обробки заявок на купівлю і продаж, зберігання даних користувачів, заявок і криптогаманця, а також формування аналітичної інформації у мобільному застосунку.

4. Перелік питань, що розглядаються:
- Аналіз процесів взаємодії між користувачами та системою купівлі-продажу криптовалюти.
- Проєктування і розробка інформаційного забезпечення системи купівлі-продажу криптовалюти.
- Проєктування і розробка програмного забезпечення мобільного застосунку Litvin Balance.
- Впровадження та експлуатація мобільного застосунку для купівлі-продажу криптовалюти.

Дата видачі завдання "06"січня 2026 р. (дата наказу)

Керівник бакалаврської кваліфікаційної роботи _____ / Панкратьєв В.О. /
(підпис) (прізвище та ініціали)

Завдання прийняв до виконання:_____ / Литвин В.С. /

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

Скорочення	Розшифрування
API	Application Programming Interface – програмний інтерфейс застосунку
APK	Android Package Kit – інсталяційний пакет Android-застосунку
DBMS	Database Management System – система управління базою даних
HTTP	HyperText Transfer Protocol
JSON	JavaScript Object Notation
KYC	Know Your Customer – ідентифікація та верифікація користувача
P2P	Peer-to-Peer – прямий формат взаємодії між користувачами
SDK	Software Development Kit – набір інструментів розробника
SQLite	Вбудована реляційна система управління базою даних
UI	User Interface – інтерфейс користувача
URL	Uniform Resource Locator – адреса ресурсу

ВСТУП

Бакалаврська кваліфікаційна робота зосереджена на моніторингу курсів криптовалют, аналізі динаміки криптовалют та управлінні замовленнями для користувача, який потребує розміщення замовлення на цифровий актив. Розробляється інформаційна система, орієнтована на освіту, яка не проводить реальних розрахунків обміну, але імітує поведінку користувача системи та адміністратора в мобільному додатку, що відтворює логіку сучасного сервісу криптовалют.

Система LITVIN BALANCE надасть нам набір загальних умов роботи з реєстрацією, онлайн активною сесією, даними KYC, поточними курсами криптовалют, управлінням замовленнями на купівлю та продаж, перевіркою вартості, існуючими P2P пропозиціями та функціональністю створення замовлень без токенів.

Система повинна відображати всі замовлення, відсортовані не за автором, а за типом монети та типом транзакції, статусом та індикатором активності, підтверджувати та відхиляти замовлення, перевіряти ідентичність користувача KYC та підтримувати чат для користувачів під час підтримки чату. Це роль, яку підтримує адміністратор, який повинен забезпечити точність даних, перевірити замовлення перед публікацією, а також спілкуватися з користувачами системи.

У предметній області найважливішою частиною є процес отримання інформації про ринок. Щоб створити та перевірити заявки, система повинна відповідати поточним курсам криптовалют і відстежувати не лише назву монети, але й її поточну ціну та інші ознаки, які користувачі враховують для прийняття розумних рішень про покупку. Доступ заявок до функцій системи є ключовою областю предметної області. Передбачається, що якщо індивідуальний користувач не має X або будь-якої перевіреної KYC, він/вона не може створювати сценарії використання для купівлі або продажу будь-якої криптовалюти. Такий підхід також знижує ймовірність ненадійних або неперевірених записів і підвищує загальну надійність системи. Після створення

заявки її потрібно зберегти в інформаційній базі даних з відповідним звітом про статус, тоді інші користувачі можуть переглядати цю заявку та виконувати її в межах логіки системи, наданої адміністратором. Можливі й інші функціональності.

Щоб розробити програмну систему, спочатку потрібно визначити функціональні вимоги, описати потоки даних ключових компонентів системи продукту та їх змістові модулі, побудувати інформаційну модель на основі тем, розробити IT-інфраструктуру, обґрунтувати вибрані технології, розробити програмне забезпечення для Android-додатків і створити простий та зручний інтерфейс у вигляді навігаційного або інтерактивного інтерфейсу. Аналітичні процеси, пов'язані з ціноутворенням криптовалют та управлінням замовленнями користувачів, також включатимуть складні процеси, що стосуються замовлень користувачів, API транзакцій, підтримки тощо.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Постановка завдання

Ця робота присвячена способам, якими ми можемо стежити за коливаннями курсів криптовалют, аналізувати їх та з'ясувати, як співпрацювати, щоб управлінню ордерами на купівлю або продаж цифрових валют проходило гладко. У цій роботі ми створили просту систему, яка показує, як можна навчатися та спостерігати за процесами, але варто зауважити, що це не справжня біржа і розрахунки тут ведуться не в реальних грошах. Ця система не є звичайною криптобіржею, але вона дозволяє користувачеві поступово ознайомлюватись з усіма діями, які виконує трейдер, і все це можна знайти в простій мобільній програмі. Цей підхід дозволяє вам зрозуміти, як функціонують сучасні сервіси у світі криптовалюти. Важливо спростити окремі елементи програми, визначити правила їх використання та правила взаємодії людей із системою.

Система LITVIN BALANCE повинна дозволяти користувачеві виконувати безліч простих завдань. Ви маєте можливість додати нового користувача до системи або активувати вже існуючого. Користувач може долучитися до паралельної сесії та отримати або надати дані для перевірки KYC. Користувач може дізнатись про курс криптовалюти на сьогодні, оформити замовлення на купівлю чи продаж, а також перевірити свої минулі P2P-пропозиції. Ви можете виконати будь-яку дію з ордером, який вже відкритий, без необхідності мати ще один запис у системі. Свій локальний баланс токенів можна тримати у безпеці. Якщо вам знадобиться якась допомога, ви завжди можете звернутися до служби підтримки для отримання додаткової інформації. Основні функціональні вимоги до системи LITVIN BALANCE наведено в табл. 1.1.

Таблиця 1.1

Основні функціональні вимоги до системи LITVIN BALANCE

Група вимог	**Зміст вимоги**
Робота з обліковим записом	Система повинна підтримувати реєстрацію, авторизацію та збереження поточної сесії користувача.
KYC-верифікація	Перед створенням заявок користувач має подати дані на перевірку; без підтвердженого KYC створення заявок забороняється.
Робота з курсами	Користувач повинен мати можливість переглядати перелік криптовалют, їхню поточну ціну та відсоткову зміну курсу.
Заявки	Система повинна дозволяти створювати, переглядати, фільтрувати, підтверджувати та виконувати заявки на купівлю або продаж.
Аналітика	Система повинна формувати стовпчикову, кругову та лінійну діаграми на основі накопичених даних.
Підтримка	Необхідно реалізувати обмін повідомленнями між користувачем і адміністратором у форматі вбудованого чату.

Система також просить користувача звертати увагу на дані. Ви можете спостерігати за числами, простими схемами та графіками. Вони покажуть, як обробляються ордери і як функціонує ця система. Це дозволить вам швидко зрозуміти основні аспекти руху криптовалюти в системі. В системі, адміністратор має свою унікальну роль. Система надає адміністратору доступ до всіх ордерів, незалежно від того, хто їх створив. Можна вибрати криптовалюту, тип операції, статус або активний запит. Адміністратор має можливість підтверджувати або відхиляти ордери, здійснювати перевірку особистостей (KYC) та взаємодіяти через чат. Адміністратор грає ключову роль у системі. Він пильнує, щоб усім було комфортно і все проходило як треба. Він ретельно перевіряє всі звіти перед їх появою в системі. Він часто спілкується з людьми і

завжди готовий прийти на допомогу, якщо виникає така потреба. Адміністратор забезпечує порядок і надає допомогу в системі, підтримуючи зв'язок з користувачами, що робить використання зручним і безпечним. Цей розподіл ролей дозволяє не лише продемонструвати дії особи, але й висвітлити роботу певної частини системи для менеджерів. Це має велике значення, адже ми можемо зрозуміти, як працює вся ця система і що в ній відбувається. Коли існує такий розподіл, легше зрозуміти, як усе функціонує як всередині, так і зовні. Взаємодію користувача та адміністратора в системі відображено на діаграмі прецедентів (рис 1.1).

Рис 1.1 Діаграма прецедентів системи LITVIN BALANCE

Велика частина цієї сфери пов'язана з аналізом ринкових даних. Система повинна працювати з актуальними курсами криптовалют, які слугують основою для створення та аналізу ордерів. Користувач має право бачити назву монети, її ціну та зміни протягом певного часу, а також інші важливі параметри, щоб розібратися в ринкових тенденціях. Отже, з урахуванням наявності інформації, купувати чи продавати капітал стає значно легше. Теми також охоплюють управління ордерами та користувачами, а ще — оновлення і показ ринкової інформації, яка напряму впливає на активність учасників системи.

У цій темі доступ до операційної системи грає велику роль. У таких програмах передбачається, що користувач, який не має підтвердженого KYC, не може створювати замовлення на купівлю чи продаж криптовалюти самостійно. Це знижує шанси на шахрайство або несанкціоновані транзакції, а також зміцнює загальну довіру користувачів до системи. Після того, як замовлення буде створено, адміністратор має зберегти його в базі даних, надати йому статус, що відображає його стан, щоб інші користувачі могли його бачити, або ж здійснити дії відповідно до правил системи.

В цьому списку завдань важливо зрозуміти, які функції потрібні для програмного продукту, описати, як дані переміщуються між основними модулями, створити модель предметної області, а також визначити, які технології використовувати для бази даних і реалізувати цю систему як Android-додаток. Ця робота охоплює багато різних процесів, які стосуються слідкування за ринком криптовалют, створення замовлень користувачів на криптовалюту, перевірки інформації про їх виконання, а також ведення облікових записів. До того ж, існує мобільна система підтримки для актуальних даних.

1.2 Огляд інформаційних джерел та існуючих рішень

Під час аналізу предметної області ми розглянули різні категорії рішень, які вже присутні на ринку і частково або повністю виконують функції, пов’язані з роботою з криптовалютами. Перша категорія включає великі платформи для

криптовалют, такі як Binance, Bybit і OKX, які пропонують не лише біржову торгівлю, а й P2P-сервіси, аналітичні інструменти, управління гаманцями, перевірку користувачів і потужну серверну інфраструктуру. До другої групи можна зарахувати різні сервіси оголошень або маркетплейси, де люди можуть самостійно публікувати свої пропозиції, переглядати вже існуючі записи та безпосередньо спілкуватись одне з одним без складних біржових процесів.

Цей аналіз першої категорії потенційних рішень показав, що основні усталені біржі криптовалют мають численні переваги, зокрема: актуальну ринкову вартість, механізми купівлі та продажу реальних активів, надання передових торгових інструментів, аналітичні діаграми та різноманітні способи взаємодії користувачів. Однак усі вищезазначені платформи розроблені для повноцінної комерційної діяльності та нелегко адаптуються до проєкту освітньої програми. Вони містять серверний компонент, централізоване зберігання надзвичайно великих обсягів даних, методи захисту фінансових транзакцій, багаторівневі системи автентифікації для кожного користувача та регулярне оновлення ринкових цін. Усі ці компоненти виходять за рамки проєкту бакалаврського рівня.

Маркетплейси та рекламні сервіси мають досить просту модель взаємодії. Таким чином, цей стиль сервісу має сенс для розробки навчальної системи, оскільки можна створювати, переглядати та виконувати заявки без необхідності розробляти розгалужену інфраструктуру обміну. Однак, як правило, такі сервіси не надають інтегрованих аналітичних інструментів, процесів перевірки KYC, функціональності криптовалюти, локального гаманця та адміністративного контролю; отже, вони не можуть повністю задовольнити вимоги системи управління цифровими активами, яка повинна імітувати як обмін заявками, так і ширший спектр процесів використання цифрових активів.

Виходячи з аналізу доступних варіантів, великі біржі вже можуть відповідати рівням, необхідним для забезпечення масштабованості, надійності та фактичних транзакцій, але вони мають надмірний функціонал та складну архітектуру, які не є необхідними для дипломного проєкту. Навпаки, сервіси

оголошень набагато ближчі до моделі освітнього додатку, але вони не містять необхідних елементів для функціонування як повноцінної інформаційної системи в криптовалютній індустрії. Порівняння LITVIN BALANCE з існуючими рішеннями наведено в таблиці 1.2. З цих причин доцільно розробити мобільний додаток, який інтегрує функціональність P2P, локальну базу даних, аналітичні можливості, верифікацію користувачів та адміністративний контроль у спільний набір функцій. Це дозволить створити автономну систему, що поєднує спільні функціональні характеристики кількох типів рішень і водночас залишається реалістичною для реалізації в межах бакалаврської роботи.

Таблиця 1.2

Порівняння LITVIN BALANCE з існуючими рішеннями

Система	**Переваги**	**Обмеження щодо використання в дипломному проєкті**
Binance P2P	Реальні офери, масштабована інфраструктура, великий набір торгових інструментів	Складна серверна архітектура та надмірна функціональність для локального навчального проєкту
Bybit P2P	Зручна робота з оферами та фільтрацією	Орієнтація на готову біржову екосистему, відсутність доступу до внутрішньої реалізації
LocalBitcoins / P2P-сервіси	Проста логіка взаємодії між користувачам	Обмежена аналітика та відсутність власного мобільного навчального середовища
Маркетплеї си оголошень	Зрозуміла модель заявок та оголошень	Відсутність KYC, курсів, криптогаманця та спеціалізованої логіки цифрових активів
LITVIN BALANCE	Єдиний мобільний застосунок з курсами, KYC, заявками, аналітикою та підтримкою	Навчально-прикладний характер, локальне зберігання даних, відсутність реального біржового ядра

Це дослідження включало огляд багатьох різних технічних джерел, що стосуються як самої предметної області, так і практичного застосування програмного продукту. Зокрема, я проаналізував офіційну документацію

розробників Android, яка дозволила мені підтвердити, що програмний продукт був розроблений на платформі Android з використанням мови програмування Kotlin та використанням розмітки XML для користувацького інтерфейсу. Я також розглянув матеріали, що стосуються використання баз даних SQLite у мобільних додатках, оскільки локальна база даних SQLite є критичним компонентом програми та забезпечує місце для зберігання облікових записів, програм, статусу KYC, гаманців та повідомлень служби підтримки. Крім того, я переглянув документацію Binance API щодо поточної ціни криптовалюти, щоб можна було включити показники реального ринку в програму.

Після аналізу цих джерел ми змогли прийняти правильне рішення щодо технологічного стеку, який буде використано під час побудови системи. Завдяки сучасному синтаксису Kotlin, зручності для розробників Android та гарній підтримці через середовище розробки Android Studio, це буде наша основна мова реалізації програмного забезпечення. Ми використовуватимемо XML для побудови різних компонентів інтерфейсу програми (тобто вкладок, форм, списків або екранів). Ми обрали SQLite як нашу вбудовану реляційну базу даних, оскільки вона не потребує окремого серверного середовища та підходить для локальної освітньої системи. Використовуючи протокол REST для підключення до зовнішнього сервісу курсів, ми можемо працювати як автономно, так і мати доступ до актуальних даних з ринку криптовалют. Зрештою, вивчення предметної області та технічні довідки формують основу для розробки архітектурних рішень для системи LITVIN BALANCE та її проєктування.

1.3 Моделювання предметної області

Для формування опису предметної області було створено моделі, які показують, як взаємодіють одна з одною сутності в системі, а також дані, що передаються між ними, внутрішня логіка, що використовується для виконання операцій, та бізнес-процеси, що виконуються через програму, пов'язані одна з одною. Моделювання предметної області є важливим кроком, оскільки воно

дозволяє отримати детальне розуміння компонентів системи та того, як вони пов'язані один з одним, а також як виконуються окремі операції. Це також дає можливість визначити, які сутності є частиною системи, який тип даних буде зберігатися в системі, які типи функцій потрібно включити до програми та як відбувається взаємодія користувача/адміністратора з локальною базою даних та зовнішніми джерелами даних.

Користувачі, заявки, курси криптовалют, записи криптовалютних гаманців, статуси KYC та повідомлення служби підтримки є основним об'єктом даних у цій предметній області. Вся обробка інформації та функції заявок будуть обертатися навколо цих об'єктів. Користувач вважається основним об'єктом у системі, оскільки він створює заявки, перевіряє курси криптовалют, надсилає дані верифікації, використовує свій криптовалютний гаманець та взаємодіє з адміністратором через службу підтримки.

Заявка слугує основним елементом бізнес-логіки, вказуючи, як користувачі бажають купувати або продавати певні цифрові активи. Ринкові курси криптовалют використовуються для представлення інформації про стан ринку та допомоги у прийнятті рішень щодо того, коли створювати або виконувати пропозицію.

Інформація про обліковий запис користувача зберігається окремо від інформації про обліковий запис адміністратора через канали торгових записів, які можна використовувати для надсилання повідомлень підтримки. Користувачі також можуть переглядати свої існуючі цифрові активи в локальній системі через записи своїх криптогаманців. Користувачі можуть перевірити свій статус KYC, а потім отримати додаткові дозволи для створення торгових записів. Користувачі підключаються до системи за допомогою інтерфейсу Android; це основний спосіб доступу до функцій програми. Увійшовши в систему, користувачі можуть отримувати поточні ціни на монети, надсилати дані для перевірки; створювати пропозицію; або виконувати раніше створену пропозицію від іншого користувача. Таким чином, ця програма пропонує два сценарії:

створення нової пропозиції та вибір уже існуючої пропозиції в локальному середовищі P2P.

Перед публікацією програми необхідно виконати перевірку статусу KYC. Якщо під час розміщення програми вона не отримує дійсної перевірки статусу KYC, її неможливо зберегти; тому ви побачите повідомлення про помилку, яке вказує на необхідність виконання перевірки статусу KYC. Після того, як заявку буде перевірено та прийнято як дійсну, вона буде збережена в базі даних, а потім надіслана на повний процес введення даних у модулі адміністрування для подальшої обробки та отримання остаточного рішення про схвалення/відхилення від адміністратора/адміністративного персоналу; ця методологія забезпечує чітке визначення всіх етапів життєвого циклу заявки та пропонує стратегію забезпечення адміністраторського нагляду за життєвим циклом заявки.

Додатковим гравцем у сфері є адміністратор, роль якого включає нагляд, підтвердження та управління заявками. Його завдання включають перегляд усіх створених заявок, обробку запитів KYC, зміну статусу заявок, керування фільтрацією записів та ведення чату підтримки. Таким чином, адміністратор також відіграє певну роль у забезпеченні надійності та доступності інформації для системи, діючи як незалежний інтерфейс до системи, надаючи користувачам унікальні додаткові функції з технологічної точки зору. Загальний результат полягає в тому, що адміністратор надасть усі засоби та механізми, необхідні для перевірки заявок, перевірки інформації користувачів та збору інформації про обслуговування клієнтів для подальшого покращення загальної структури моделі, тим самим відображаючи реальний сценарій того, як працює інформаційна система, повніше, ніж якби не було адміністратора.

Інформаційні потоки в цій системі поділяються на дві категорії. Локальні інформаційні потоки визначають операції з базою даних SQLite, які містять дані про користувачів, програми, повідомлення підтримки, стан гаманця, результати перевірки та аналітичні показники. Ці локальні інформаційні потоки ілюструють

основну внутрішню логіку системи, оскільки більшість дій користувачів та адміністраторів відбуваються через локальні інформаційні потоки. Застосовуючи цей підхід, ви чітко окреслюєте межі системи та визначаєте потенційні точки інтеграції із зовнішніми сервісами, не перевантажуючи архітектуру серверними компонентами.

Створюючи модель предметної області, ви також розробите бачення основних бізнес-процесів системи. Деякі з них: процес реєстрації та авторизації, процес подання запиту KYC, процес створення заявки, процес перевірки адміністратора, процес виконання пропозиції, процес взаємодії з криптогаманцем та процес надсилання повідомлення (службі підтримки). Усі ці бізнес-процеси мають різні процеси для кожного процесу, а також різні набори учасників та різні набори вхідних даних та дій, тому вони разом утворюватимуть одну комплексну модель системи LITVIN BALANCE, яку потім можна буде розвивати далі за допомогою діаграм випадків, діаграм ER, діаграм пакетів або блок-схем окремих алгоритмів.

На рис. 1.2 подано контекстну діаграму системи LITVIN BALANCE, яка відображає взаємодію системи з основними зовнішніми сутностями. Із діаграми видно, що користувач і адміністратор взаємодіють безпосередньо із застосунком, а отримання ринкових даних здійснюється через Binance API. Така схема дозволяє визначити межі системи, основних учасників та канали обміну інформацією.

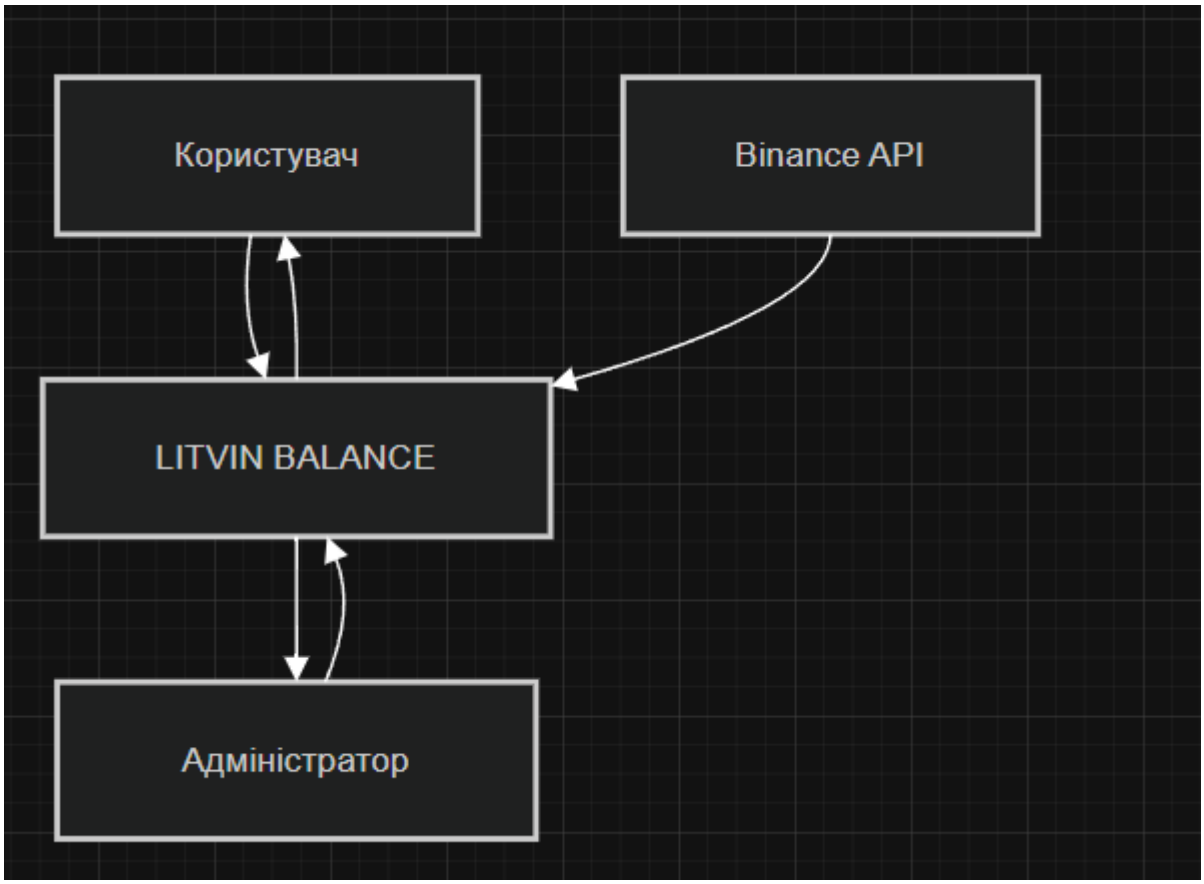


Рис 1.2 Контекстна діаграма системи LITVIN BALANCE

На рис. 1.3 наведено діаграму основного бізнес-процесу створення та виконання заявки. Вона демонструє, що перед створенням заявки обов’язково перевіряється KYC-статус користувача. Якщо верифікацію не підтверджено, система формує повідомлення про необхідність її проходження. Якщо KYC підтверджено, користувач заповнює форму заявки, після чого вона зберігається в базі даних і передається на перевірку адміністратору. Залежно від результату перевірки заявка або публікується як офер, або повертається користувачу з поясненням причини відхилення. У разі публікації офер надалі може бути виконаний іншим користувачем.

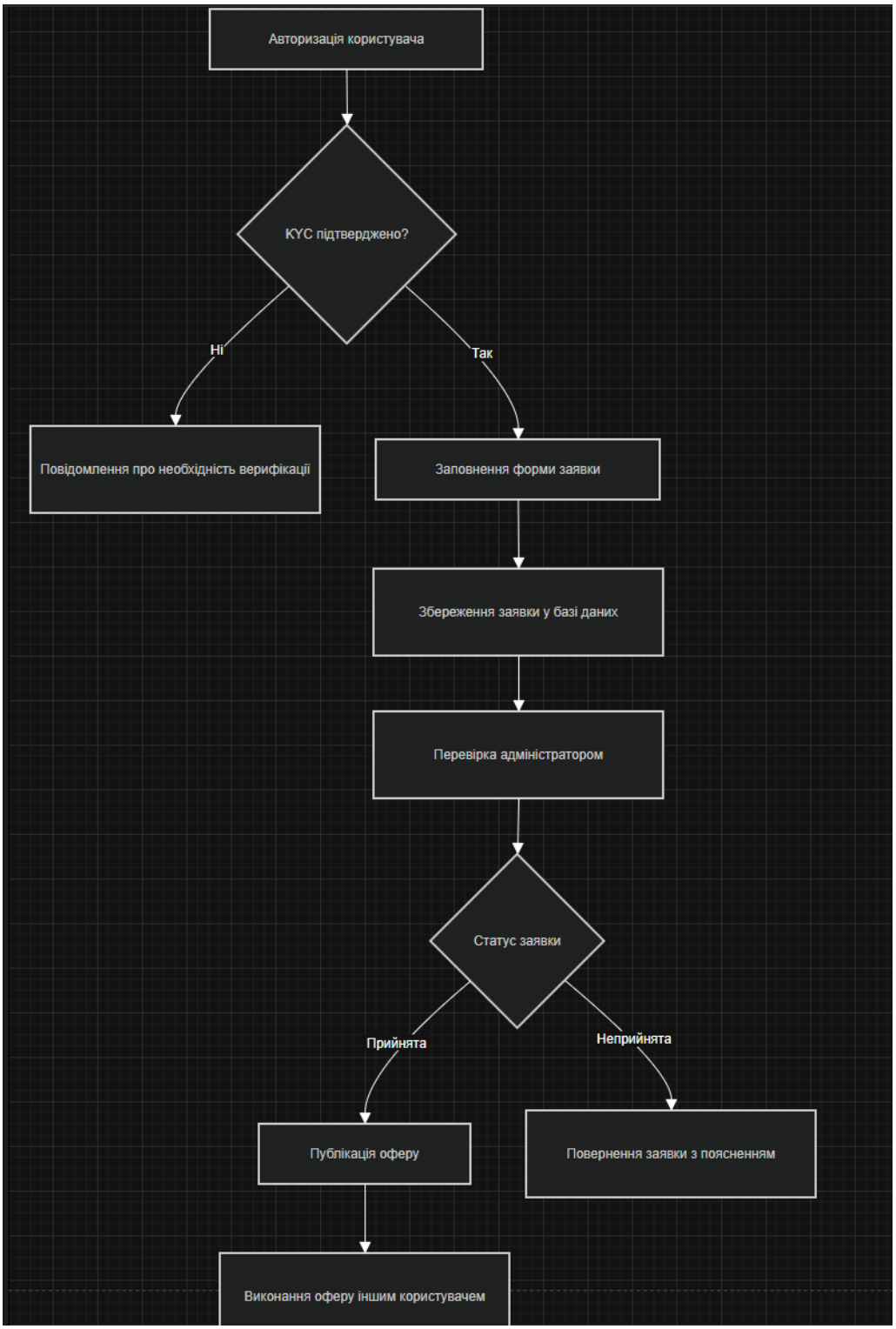


Рис 1.3 Діаграма основного бізнес-процесу створення та виконання заявки

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Логічна модель даних

Інформаційна підтримка системи LITVIN BALANCE спирається на локальну реляційну базу даних SQLite як основний засіб зберігання всієї основної інформації, яку мобільний додаток використовуватиме під час реєстрації користувачів, створення заявок, управління локальними криптогаманцями, обробки статусу KYC, надання аналітичних даних та дій, пов'язаних з підтримкою користувачів. Логічна модель даних представляє основні сутності в предметній області, їх атрибути та зв'язки між собою в базі даних. Це проміжний етап між аналізом предметної області та реалізацією фізичної таблиці в базі даних; вона надає засоби для визначення того, які об'єкти потрібно включити до інформаційної бази, а також як вони взаємодіють один з одним.

Користувач є найважливішим компонентом логічної моделі; отже, всі сутності в логічній моделі пов'язані з користувачем як центральним елементом. Користувач має ідентичність у застосунку, роль, статус перевірки KYC, а також є автором заявок та власником криптогаманця й учасником листування зі службою підтримки. Це створює логічну структуру, оскільки користувачі ініціюють усі основні дії в застосунку, тобто авторизацію, подання заявки на перевірку, перевірку ціни, продаж та купівлю пропозицій, виконання запитів, що очікують обробки, та спілкування з адміністратором. Згідно з цим міркуванням, усі інші сутності в логічній моделі прямо чи опосередковано пов'язані з користувачем.

Найпоширенішою сутністю в системі є ордер, який виражає бажання користувача повідомити про намір купити або продати певну криптовалюту. Ордер, з точки зору логічної моделі, містить тип операції, назву ордера, ім'я трейдера, вибрану країну (монету), суму, яку надає(ють) користувач(і), очікувану суму, яку користувач(і) отримають, грошовий еквівалент ордера,

статус ордера та індикатор активності. Ці характеРистики формують повне визначення для створення торгового запису в системі та підтримують цикл транзакції ордера від створення до перевірки адміністратором, а потім, нарешті, виконання або відхилення.

Ще однією групою сутностей, пов'язаною з інформацією, є таблиця обмінного курсу криптовалюти. Вона містить символ монети, ціну, відсоткову зміну та час оновлення. Хоча ця сутність містить дані із зовнішніх джерел, вона служить частиною логічної моделі, оскільки використовується для представлення ринкових умов, виконання аналітики (статистики) та надання додаткових розрахунків у формах купівлі/продажу. Ці дані дозволяють системі показувати користувачам їхні пропозиції та дають користувачам можливість порівнювати свої пропозиції з поточними ринковими курсами; таким чином, додаток може надавати своїм користувачам кращу інформацію.

Ще однією логічною сутністю моделі є криптогаманець. На відміну від складного об'єкта в структурі бази даних, криптогаманець існує як комбінація записів для типу монет та їх відповідної кількості, що зберігаються кожним користувачем. Логічна модель використовує цей метод комбінації, оскільки він пропонує простий спосіб забезпечення реляційного представлення. Цей метод дозволить будь-якому користувачеві мати кілька записів у сутності гаманця, що представляють баланс користувача для BTC, ETH, SOL або USDT. Це спростить оновлення балансу після успішного виконання транзакції, а також допоможе відобразити активи користувача у його відповідному профілі.

Логічна модель включає повідомлення підтримки в окремому блоці, окремому від решти. Ідентифікатори користувача та відправника, автор (ім'я, логін), автор (роль), звернення (текст повідомлення), дата створення та статус прочитання для всіх повідомлень зберігаються в цій структурі. Завдяки такій організації цих 5 атрибутів можна створити функцію чату між користувачами та адміністраторами, використовуючи єдину локальну базу даних. З логічної точки зору, сутність, яка зберігає повідомлення підтримки, забезпечує зв'язок між користувачем та адміністратором разом з відповідною історією. Наявність поля

«роль» також дуже корисна, оскільки дозволяє нам знати, з якої «сторони» виникло спілкування, та правильно відображати його в інтерфейсі чату.

Статус KYC – це ще одна важлива допоміжна сутність. У цій моделі цей компонент розташований як атрибут користувача. Зберігання атрибутів KYC на рівні користувача зменшує складність структури даних, дозволяючи застосунку швидко визначати правильність використання, тобто чи може людина створювати власні заявки. Однак фактична перевірка та зміна умов статусу KYC все ще є критичним компонентом логічного рівня, оскільки система не змогла б забезпечити доступ до торгових операцій без цього механізму контролю. Тому, хоча ця інформація технічно не знаходиться в незалежній таблиці, KYC є важливим логічним елементом даних у моделі домену.

Загалом, логічна модель даних для LITVIN BALANCE охоплює основні дії, що відбуваються в застосунку, такі як взаємодія з користувачами, створення заявок, зберігання валют (криптовалют), керування обліковими записами користувачів та надання підтримки. Основні сутності логічної моделі даних наведено в таблиці 2.1, а взаємозв’язки між ними відображено на рис. 2.1. Ця логічна модель даних служить освітнім інструментом для навчальних цілей, а також проста в реалізації в SQLite. Крім того, ця логічна модель даних забезпечує основу для створення фізичної бази даних та реалізації програмної логіки для мобільних застосунків.

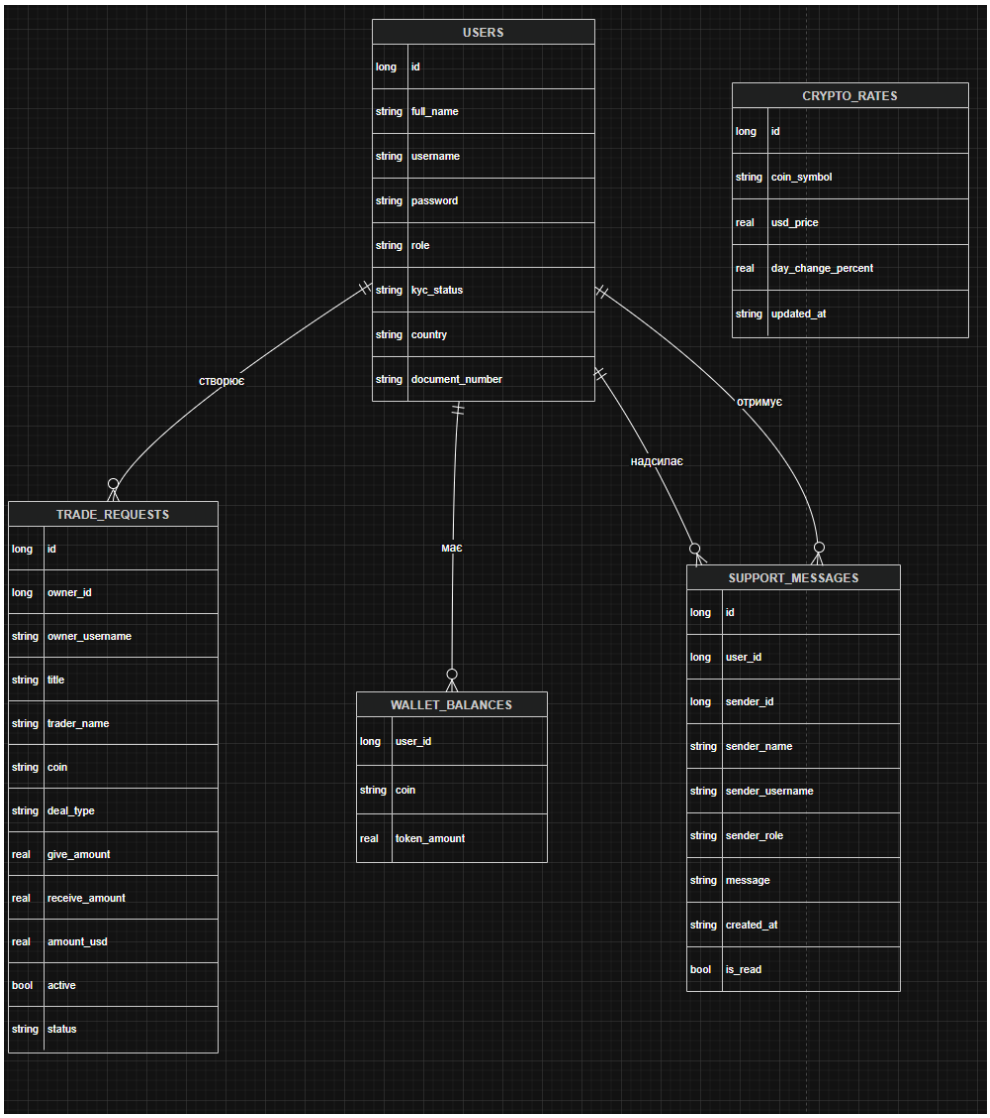


Рис 2.1 ER-діаграма інформаційної бази LITVIN BALANCE

Таблиця 2.1

Основні сутності логічної моделі даних

Сутність	**Ключове поле**	**Призначення**
USERS	id	Зберігання облікових записів, ролей і KYC-статусів користувачів.
TRADE_REQUESTS	id	Зберігання заявок на купівлю та продаж криптовалют.
CRYPTO_RATES	id	Локальне кешування актуальних курсів криптовалют.
WALLET_BALANCES	user_id + coin	Ведення локального балансу токенів по кожному користувачу.
SUPPORT_MESSAGES	id	Зберігання повідомлень між користувачем і адміністратором.

2.2 Вибір системи управління інформаційною базою

Рішення використовувати SQLite як сховище даних для цього рішення було прийнято, головним чином, на основі мобільної архітектури застосунку, яка розроблена для забезпечення локального виконання без встановлення окремого серверного середовища. LITVIN BALANCE, розроблений як освітній та практичний застосунок для Android, робить найбільший акцент на роботі мобільного клієнтського застосунку; на реалізації його функціональних модулів; на наявності зручного інтерфейсу; та побудові логічного потоку взаємодії між кінцевими користувачами (тобто користувачами застосунку) з заявкаи; на верифікації; на електронних гаманцях; та на підтримці. Отже, використання локальної бази даних на відміну від використання бази даних на віддаленому сервері є найбільш доцільним для цієї реалізації, оскільки воно забезпечить усі необхідні вимоги до зберігання даних без залежності від додаткової серверної інфраструктури.

Завдяки вбудованому SQLite у вбудований стек операційної системи Android, відпадає необхідність налаштування окремого сервера, що робить його гарним варіантом для використання в освітньому середовищі завдяки дуже малому розміру та використанню реляційної моделі даних. Таким чином, коли розробники використовують SQLite, їм не потрібно встановлювати додаткове стороннє програмне забезпечення для підтримки зовнішньої бази даних, а також налаштовувати мережевий зв'язок між програмою та її сервером бази даних; натомість SQLite може використовуватися розробниками як як база даних, так і для розробки, тестування та демонстрації програмних продуктів, що робить його надзвичайно простим рішенням для розробки бакалаврських проектів. На додаток до цієї функції, SQLite також може зберігати структуровану інформацію всередині таблиць, що відповідає логічній структурі бази даних програмного забезпечення LITVIN BALANCE.

SQLite – це підходяща база даних для системи, виходячи з її потреб для функціональної роботи. Локальні дані повинні містити облікові записи

користувачів, статус перевірки KYC, запити на купівлю/продаж, записи гаманців, кешовані курси криптовалют та повідомлення підтримки. Кожен із цих типів даних має реляційну структуру та містить не так багато даних порівняно з іншими базами даних; тому його можна легко інтегрувати у вбудовану мобільну СУБД. Саме цей тип системи дозволить цьому проекту належним чином функціонувати як освітній проект завдяки швидкості, продуктивності та простоті використання/обслуговування самої бази даних.

Побудова серверного компонента серверної системи вимагає налаштування мережевого з'єднання між мобільним додатком та сервером, розробки інтерфейсу прикладного програмування для доступу до даних та реалізації необхідних функцій безпеки, автентифікації та адміністрування, що призведе до значного збільшення загальної складності архітектури проекту та змістить основний фокус проекту з мобільного додатку та його продуктивності на забезпечення повнофункціонального клієнт-серверного додатку. Порівняння можливих СУБД для реалізації інформаційної бази наведено в таблиці 2.2.

Таблиця 2.2

Порівняння можливих СУБД для реалізації інформаційної бази

СУБД	**Переваги**	**Недоліки**	**Висновок**
SQLite	Вбудованість у Android, відсутність сервера, простота розгортання	Обмежена масштабованість для багатокористувацьких розподілених систем	Обрано для проєкту
MySQL	Поширеність, серверна модель, підтримка великої кількості клієнтів	Потрібне окреме серверне середовище	Доцільна для серверної версії
PostgreSQL	Розширені можливості, висока надійність, підтримка складних типів даних	Надлишкова для локального мобільного застосунку	Резервний варіант для подальшого розвитку

Локальна СУБД краще відповідатиме вимогам бакалаврської дисертації, оскільки вона дозволяє зосередитися на логіці програми, моделях даних та

інтерфейсі користувача. Це також відповідає освітній меті цього проекту, оскільки демонструє здатність створювати реляційну структуру даних, створювати зв'язок між програмними модулями та зберігати інформацію без зайвої складної архітектури. Крім того, використовуючи SQLite для зв'язку з вашою програмою, ви можете продемонструвати готовий продукт, не турбуючись про необхідність кількох установок для розгортання, оскільки ваша програма працюватиме незалежно на одному пристрої.

SQLite – це технічно обґрунтований та дуже підходящий вибір бази даних для мобільного додатку LITVIN BALANCE, оскільки він простий у впровадженні, працює з реляційною моделлю бази даних та дозволяє зберігати всі необхідні дані локально. Використання цього варіанту створює міцну основу для розробки програмного застосунку, а дипломний проект не обтяжений зайвими серверними компонентами.

2.3 Створення інформаційної бази

Клас TradeRequestDatabase – це реалізація для фізичного представлення інформаційної бази даних. Він створюватиме, ініціалізуватиме та працюватиме з базою даних, створеною та розташованою в локальній програмі. Він створює таблиці users, trade_requests, crypto_rates, wallet_balances, support_messages, а також віртуальну таблицю trade_requests_search для швидкого пошуку програм. Кожна з вищезазначених таблиць має певний набір полів, що відповідають логічній моделі даних, створеній раніше. Таким чином, фізичне представлення бази даних відповідатиме логічному представленню для відповідної предметної області та надаватиме всю необхідну інформацію для ефективної роботи програми.

Таблиця «Користувачі» зберігає інформацію про обліковий запис, включаючи логін, пароль, роль користувача, повне ім'я, країну, номер документа та статус KYC. Інші таблиці покладаються на цю таблицю для своїх зв'язків, оскільки вони містять посилання на програми, записи криптогаманців та

повідомлення служби підтримки. Наявність полів ролі та статусу перевірки дозволяє системі оцінювати, до яких функцій має доступ кожен користувач (наприклад, створення програм або виконання адміністративних функцій). Таким чином, ця таблиця містить ідентифікаційну інформацію про кожного користувача, а також є частиною загальної логіки безпеки системи та визначає доступ окремих користувачів до різних частин системи.

Замовлення в таблиці trade_requests надають функції, пов'язані з торговими замовленнями та пов'язаними з ними характеристиками статусу. Кожне замовлення розрізняється власником за допомогою ідентифікатора та імені користувача, які представляють власника замовлення (або угоди). У цій таблиці наведено такі поля даних: owner_id, owner_username, title, trader_name, coin, deal_type, give_amount, receive_amount, amount_usd, active, status. Схема даних у цій таблиці дозволить вказати як необхідні основні атрибути замовлення, так і інформацію про атрибути, необхідну для відображення, фільтрації, сортування та подальшої обробки кожного замовлення. Наприклад, поле deal_type вказуватиме, чи призначене угода або замовлення для купівлі чи продажу криптовалюти, give_amount та receive_amount вказуватимуть фактичні параметри транзакції, а status вказуватиме, чи запис очікує на розгляд, прийнятий чи відхилений. Кожну пропозицію можна окремо ідентифікувати як активну або закриту за допомогою атрибута active.

Таблиця crypto_rates використовується для оновлення поточних значень криптовалюти, що надходять із зовнішнього API Binance. Символ криптовалюти, значення, відсоткова зміна та час оновлення зберігаються тут і дозволяють програмі відображати найактуальніші ринкові дані, а також проводити аналітику та інші допоміжні типи обчислень під час створення ордера на основі цих даних. Крім того, оскільки ці дані надходять із зовнішнього джерела, рекомендується кешувати їх локально в базі даних, щоб їх можна було швидше отримати для відображення в інтерфейсі, а також щоб ви могли продовжувати використовувати інформацію навіть після перезапуску програми.

Таблиця wallet_balances зберігає локальні дані або кількість токенів, пов'язаних з кожною монетою. Логічна організація бази даних дозволяє одному користувачеві мати скільки завгодно записів, принаймні один запис для кожного з цифрових активів, якими він володіє. Наприклад, користувач може мати окремий запис для BTC, ETH, SOL та USDT. Використовуючи цей метод, значно простіше коригувати баланс цифрового активу після транзакції та зручніше відображати загальний стан криптогаманця в профілі користувача. Фактично, ця таблиця служить локальною моделлю для обліку активів у мобільному додатку.

Для спрощення пошукових запитів було створено повнотекстовий індекс пошуку, реалізований у вигляді віртуальної таблиці під назвою trade_requests_search. Його метою є підвищення швидкості пошуку записів на основі будь-якого з наступних полів запису: назва запиту; ім'я трейдера; назва монети; тип транзакції; ім'я користувача для входу; та/або статус. Це особливо корисно для адміністратора, який має керувати великою кількістю записів і потребує швидкого пошуку певного запиту на основі ключового слова або атрибута. Створення такого індексу в рамках фізичної моделі бази даних значно підвищує зручність використання фільтрів та операцій пошуку.

Крім того, база даних містить таблицю support_messages, яка дозволяє зберігати попередні повідомлення, надіслані між користувачем та адміністратором. Структура фіксує ідентифікатор користувача та ідентифікатор відправника для кожного повідомлення разом з ім'ям та логіном відправника повідомлення. Вона також зберігає роль відправника, розмову (текстове звернення), дату/час створення та чи було повідомлення прочитано. Це дозволяє створити функцію чату підтримки в додатку, щоб користувач міг надсилати повідомлення адміністратору, не виходячи з програми. Для адміністраторів повний огляд усіх поточних відкритих діалогів (повідомлень), а також перемикання між розмовами та перегляд нових повідомлень допоможе забезпечити користувачам (адміністраторам) краще обслуговування клієнтів завдяки розробці цієї програми.

База даних TradeRequestDatabase не лише зберігає дані безпосередньо, але й включає інші типи операцій, необхідні для роботи процесу запиту на торгівлю. Інші типи операцій: оновлення статусу програми, зміна статусу KYC користувача, виконання транзакції, зміна запису гаманця, отримання аналітичної інформації та обробка повідомлень служби підтримки. Таким чином, цей модуль слугуватиме як «фізичним» сховищем запитів на торгівлю, так і центральним механізмом для роботи з даними всередині програми.

В результаті, дані, що зберігаються в класі TradeRequestDatabase, були фізично реалізовані для повного відображення логічної моделі даних, що використовується в базі даних SQLite. Поєднання створених таблиць, а також створених допоміжних механізмів дозволяє нам підтримувати всі основні функції системи, включаючи управління користувачами, управління додатками, управління курсами криптовалют, управління криптогаманцями, підтримку та аналітичну роботу. Ця фізична модель даних є основою безперебійної роботи мобільного додатку LITVIN BALANCE.

Склад фізичної моделі інформаційної бази, основні таблиці та їх призначення наведено в таблиці 2.3, а загальну схему таблиць бази даних, побудовану в середовищі ERwin Data Modeler, подано на рис. 2.2.

Таблиця 2.3

Фізичні таблиці інформаційної бази та їх призначення

Таблиця	**Основні поля**	**Призначення**
users	id, username, password, role, kyc_status	Облікові записи користувачів та адміністраторів.
trade_requests	id, owner_id, coin, deal_type, give_amount, receive_amount, status	Збереження оферів на купівлю та продаж.
crypto_rates	id, coin_symbol, usd_price, day_change_percent, updated_at	Кешування курсів, отриманих із Binance API.
wallet_balances	user_id, coin, token_amount	Облік токенів у локальному гаманці користувача.
support_messages	id, user_id, sender_id, message, created_at, is_read	Чат підтримки між користувачами та адміністратором.
trade_requests_search	request_id, title, trader_name, coin, deal_type, owner_username, status	Швидкий пошук оферів за ключовими параметрами.

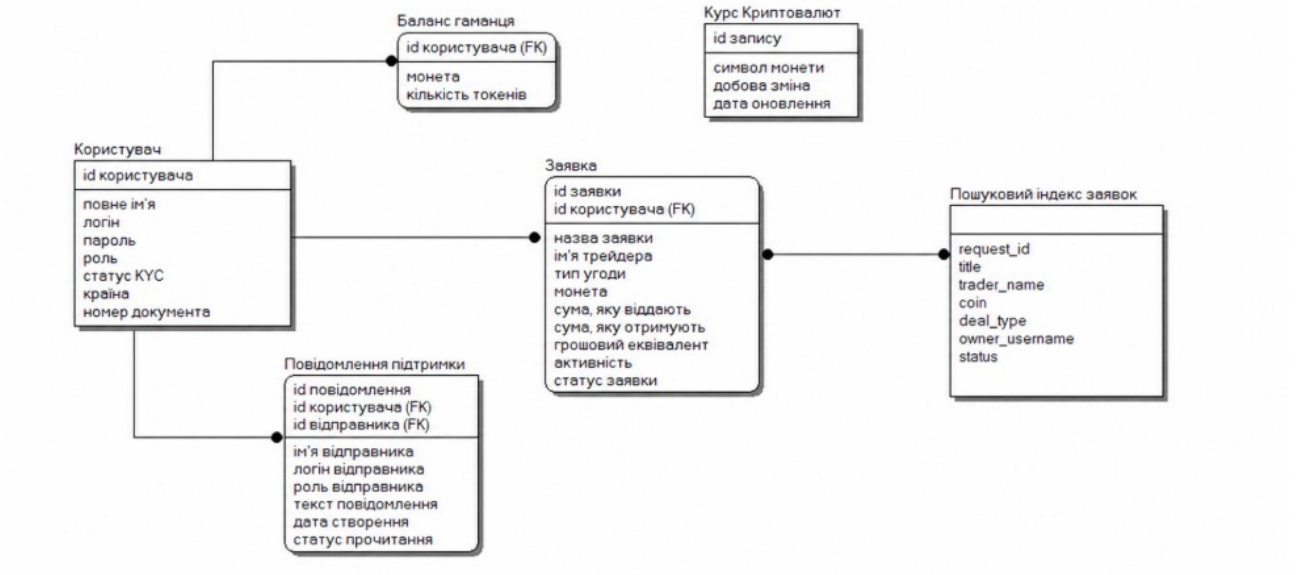


Рис 2.2 – Загальна схема таблиць інформаційної бази LITVIN BALANCE, побудована в ERwin Data Modeler

3 ПРИКЛАДНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Організаційна структура програмного забезпечення

Архітектура LITVIN BALANCE дотримується модульного принципу, де вся функціональність системи розділена на багато логічних блоків, причому кожен блок відповідає за частину функціонування програми. Такий тип структури ідеально підходить для мобільного програмного продукту, оскільки він забезпечує чіткий розподіл обов'язків між різними компонентами системи, допомагає зробити код легшим для читання та спрощує загальне обслуговування, тестування та майбутнє розширення коду. Модульна організація також полегшує внесення майбутніх змін, оскільки будь-які покращення, внесені до одного функціонального блоку, матимуть дуже незначний вплив на інші області програми. Загальну структуру взаємодії модулів застосунку подано на рис. 3.1, а основні програмні модулі та їх призначення наведено в табл. 3.1.

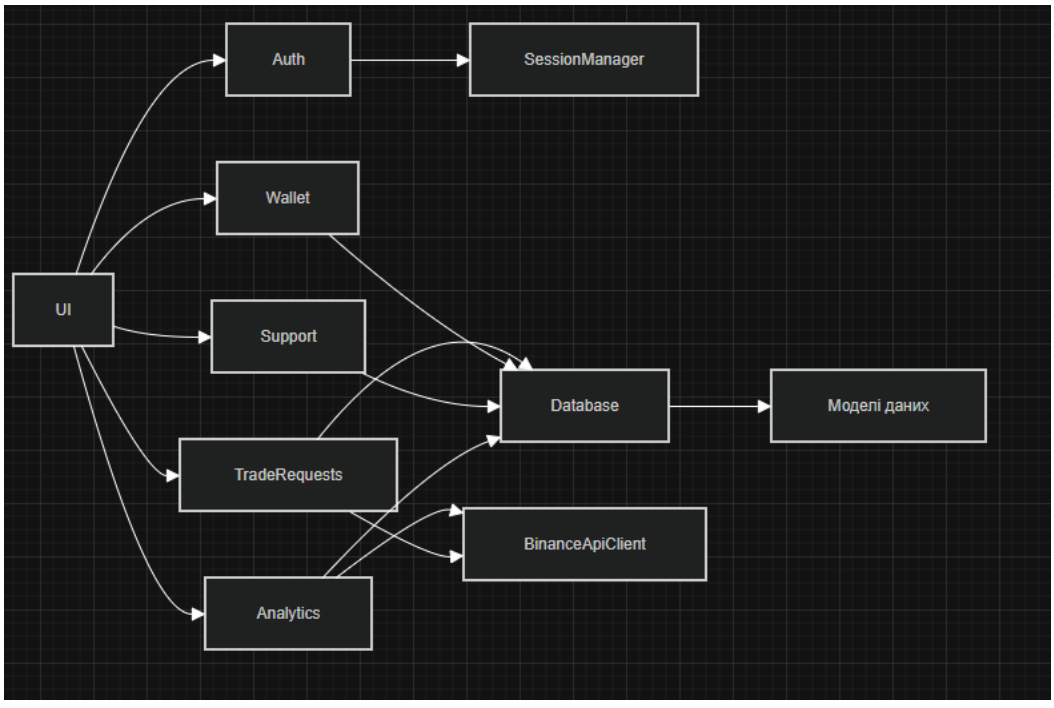


Рис 3.1 Діаграма пакетів застосунку LITVIN BALANCE

Таблиця 3.1

Основні програмні модулі та їх призначення

Модуль / файл	**Призначення**
AuthActivity.kt	Реалізація входу та реєстрації користувача.
MainActivity.kt	Головний екран, вкладки, форми заявок, аналітика, KYC, підтримка та профіль.
TradeRequestDatabase.kt	Створення й супровід локальної бази даних SQLite.
BinanceApiClient.kt	Отримання курсів криптовалют через Binance API.
SessionManager.kt	Збереження даних про поточну сесію користувача.
TradeRequestAdapter.kt / KycRequestAdapter.kt / WalletBalanceAdapter.kt	Адаптери для відображення списків і карток.
BarChartView.kt / PieChartView.kt / LineChartView.kt	Компоненти побудови аналітичних графіків.

Пакет інтерфейсу користувача (UI) є ключовим компонентом архітектури; він містить усі відображення вкладок, форм, карток програм, графіків, таблиць та інші візуальні компоненти. Пакет інтерфейсу користувача – це те, як користувачі взаємодіють із системою. Тому цей пакет є основною точкою входу для користувачів, щоб отримати доступ до функціональності програми. Цей пакет містить логіку для відображення екранів, створення списків, перемикання між розділами, введення та отримання даних, а також прив'язки об'єктів інтерфейсу користувача до відповідних дій програми. Таким чином, пакет інтерфейсу користувача інтегрує всі прояви системи в єдине просте у використанні середовище.

Пакет авторизації відповідає за вхід користувача в систему, реєстрацію та керування його поточним сеансом. Цей модуль перевіряє введені облікові дані, створює новий профіль та зберігає інформацію про поточного користувача після успішної авторизації. Таким чином, цей пакет є важливою частиною архітектури, оскільки інакше неможливо реалізувати персоналізовану взаємодію з додатками, перевірку KYC, криптогаманці чи підтримку. Це те, що зрештою пов'язує ваш обліковий запис з вашою подальшою поведінкою під час взаємодії з системою.

Модуль TradeRequests містить програмні компоненти та функції, що сприяють створенню, зміні, затвердженню, відстеженню та виконанню замовлень. Таким чином, TradeRequests ефективно функціонуватиме як одна з основних функціональних областей загальної програми, оскільки вона включає функціональність, яка реалізує модель купівлі/продажу криптовалюти в системі. Модуль TradeRequests виконує всі операції, необхідні для роботи з параметрами замовлень, обробляє зміну статусу замовлень, підтримує фільтрацію та сортування, а також надає сценарій виконання для виконання замовлень, створених іншим користувачем. Значна частина бізнес-логіки системної програми також знаходиться в цьому модулі.

Пакет гаманця дозволяє локально керувати токенами користувачів з вашої програми. Пакет гаманця зможе зберігати та відображати інформацію про цифрові активи, доступні користувачам. Гаманець діє як зв'язок користувача між базою даних та його профілем, де він бачить свій криптогаманець, а також використовуватиметься для певної логіки замовлення, щоб перевірити, чи можна розмістити замовлення. Гаманець функціонуватиме як локальне сховище для активів користувача в мобільному додатку.

За допомогою модуля аналітики користувачі можуть створювати діаграми та використовувати різні форми статистичного представлення для візуальної аналітики. Цей модуль створює візуальну аналітику з ордерів та інших типів даних у системі, включаючи значення криптовалют та інші показники. Таким чином, цей модуль надає можливість створювати записи/розробляти візуальну

аналітику для оцінки всього рівня активності в системі, стаючи чимось більшим, ніж просто інструментом для створення/перегляду записів.

Пакет Database — це окрема функція, що реалізує SQLite. Він дозволяє створювати таблиці, записувати/читати дані, змінювати записи, змінювати статуси та виконувати інші операції в локальному сховищі даних. Цей модуль є об'єднуючою точкою бізнес-логіки та фізичного зберігання даних. Пакет Database формує основу для всіх інших пакетів, які отримують доступ до цих даних, та підтримує їх подальше успішне використання. Відокремлення цього модуля від інших модулів сприяє більш організованій архітектурі та дозволяє легше нарощувати складність.

Інтеграція BinanceApiClient відповідає за вилучення даних про курс із зовнішніх джерел даних. BinanceApiClient дозволяє додатку взаємодіяти не лише з даними, що зберігаються в локальній системі, але й з актуальними ринковими даними, доступними через API біржі Binance. Він працює, викликаючи Binance API та завантажуючи поточну ціну монет з біржі, а потім надсилаючи цю інформацію до інших частин системи, таких як інтерфейс користувача та аналітика. Таким чином, BinanceApiClient є ключовим компонентом, який використовується для інтеграції системи із зовнішніми джерелами інформації та надає можливість розширити функціональність додатку за межі локального середовища.

Окрім основних компонентів проєкту, існує кілька допоміжних компонентів, які забезпечують спільну поведінку інтерфейсу та забезпечують стабільну роботу різних частин програми. Допоміжні компоненти включають адаптери списків, моделі даних та допоміжні класи форматування. Адаптери списків використовуються для відображення програми, її тарифів, записів гаманця, повідомлень підтримки та інших речей у форматі списку або картки. Моделі даних забезпечують організований спосіб представлення сутностей у системі. Допоміжні класи форматування мають єдиний стиль представлення тексту, стану та даних значень. Хоча ці компоненти є вторинними по

відношенню до основних компонентів, вони сприятимуть загальній цілісності архітектури та підтримуватимуть стабільну поведінку програми.

3.2 Вибір інструментарію для створення прикладного програмного забезпечення

Kotlin було обрано для створення застосунку, оскільки він офіційно рекомендований Google як основна мова для повноцінної гнучкої розробки Android, а також добре задокументований Google. Деякі з переваг використання Kotlin полягають у тому, що він має багато потужних бібліотек для створення об'єктів і класів, забезпечує чудову підтримку колекцій, має вбудовану/першокласну підтримку обробки винятків і пропонує функцію null-safe для зменшення помилок, пов'язаних з неправильним керуванням порожніми змінними. Крім того, Kotlin має сучасний/стислий синтаксис, що робить програми набагато легшими для читання, логічними та лаконічними. Це допомагає розробникам мобільних пристроїв дуже швидко створювати модулі своїх заявок і, отже, загалом спрощує підтримку застосунку.

Kotlin – дуже хороший вибір для розробки мобільних додатків, оскільки він добре працює з Android Studio та рештою екосистеми розробки Android. Він має об'єктно-орієнтовану модель програмування, яка дозволяє легко створювати модульні архітектури для мобільних додатків, де окремі класи самостійно обробляють базу даних, інтерфейс користувача, керування сеансами, мережу, аналітичні або допоміжні функції; ця модульність забезпечує загалом чистішу та простішу кодову базу, що робить його логічним вибором під час розробки LITVIN BALANCE, сучасного мобільного програмного забезпечення з використанням Kotlin.

Він складається з XML-розмітки, яка точно визначає структуру екрана в коді, а також те, як оформлювати стандартні елементи Android. Запис із використанням XML дає змогу вказувати інтерфейс користувача окремо, не вбудовуючи його безпосередньо в логіку програми. Це, у свою чергу, допомагає

чіткіше бачити ієрархію екранів/кнопок/текстових полів/списків/вкладок (як у карточках додатків), які пізніше розробник зможе змінювати. Крім того, XML підходить для адаптації інтерфейсів на екранах різних масштабів і дає змогу повторно використовувати стилізовані компоненти в різних розділах застосунку. Це забезпечує істотну перевагу для дипломного проєкту, оскільки робить складання багатошарового мобільного UI простішим.

Вибір основного IDE (інтегрованого середовища розробки) було зроблено на користь Android Studio, яке містить інструменти для налагодження та емуляції, а також інтерфейсний аналізатор і функції збирання APK-пакетів. Офіційне середовище розробки для Android, яке підтримує роботу з Kotlin, XML та інструментами емуляції пристроїв, а також із системою збирання Gradle. Воно надає інтегровані інструменти для запуску, тестування, налагодження та перегляду структури вашого проєкту у деталях — це робить розробку зручнішою для будь-якого мобільного застосунку. Крім того, завдяки тому, що Android Studio працює більш зручно з ресурсами, макетами, залежностями та структурою вихідного коду, воно стало природним вибором для розробки системи LITVIN BALANCE.

Застосунок використовує SQLite для локального зберігання. База даних є частиною стандартних компонентів Android, тож не потрібно розгортати окреме серверне середовище. Це важлива перевага для мобільного навчального застосунку, оскільки дає змогу зберігати дані щодо користувачів, заявці і статусу верифікації, записів криптогаманця, повідомлень підтримки, а також кешованих курсів із криптовалюти без необхідності розширювати серверну інфраструктуру. SQLite має реляційну модель з мінімальними вимогами до ресурсів і достатньою продуктивністю для підтримки сценаріїв використання, визначених у цій роботі. Це дозволяє зосередити розробку лише на логіці того, що саме може робити застосунок, замість того щоб створювати якийсь громіздкий серверний компонент.

Щоб взаємодіяти з мережею та отримувати інформацію про курси криптовалют, використовуються стандартні можливості HTTP та обробки JSON,

які упаковуються в BinanceApiClient. Цей модуль обробляє запити до фактичного зовнішнього сервісу Binance API, витягує з нього актуальну ринкову інформацію та надає такий оброблений потік для споживання іншими частинами застосунку. Застосунок поєднує локальну автономну логіку із зовнішніми ринковими даними, завдяки чому програма сама набагато більше відповідає і вірніше відображає те, як вона існує на практиці (у реальному світі під час роботи з криптовалютами) — усе це завдяки цьому підходу. Функціональність мережі також винесена в окремий компонент у той самий час, дотримуючись принципів модульної архітектури та підтримки зручності супроводу коду.

Треті сторони інструменти візуалізації та форматування (використовуються для створення документів, презентацій і супровідних матеріалів) застосовуються, однак основна частина логіки програмного забезпечення реалізована за допомогою власного коду. Це означає, що вся ключова функціональність застосунку (авторизація, верифікація KYC, робота із заявками, підтримка криптогаманця, побудова аналітики та підтримка) була реалізована без використання готових шаблонних рішень. Цей підхід є важливим для бакалаврської дипломної роботи, оскільки дає змогу підтвердити свою здатність створювати, запускати та взаємодіяти з програмними модулями як частинами єдиної інформаційної системи.

Отже, обраний інструментарій є достатньо обґрунтованим і відповідає характеру поставленої задачі. Поєднання Kotlin, XML, Android Studio, SQLite та інтеграції з Binance API дозволяє реалізувати мобільний застосунок, який поєднує локальну роботу з даними, сучасний інтерфейс користувача та доступ до актуальної ринкової інформації. Узагальнене обґрунтування вибору інструментальних засобів наведено в табл. 3.2. Саме цей технологічний стек став основою для реалізації програмного продукту LITVIN BALANCE.

Таблиця 3.2

Обґрунтування вибору інструментальних засобів

Інструмент	**Причина використання**
Kotlin	Сучасна мова Android-розробки з підтримкою об’єктно-орієнтованого підходу та безпечної роботи з даними.
Android Studio	Офіційне середовище розробки з підтримкою збірки, налагодження та тестування мобільних застосунків.
XML	Гнучке формування структури інтерфейсу та повторне використання макетів.
SQLite	Локальна реляційна база даних, що не потребує окремого серверного середовища.
Binance API	Надання актуальних ринкових курсів криптовалют для інформаційного модуля та аналітики.

3.3 Алгоритмізація та програмування програмних модулів

Ключовими алгоритмами для створення та використання основних функцій програми були перевірка KYC, локальний запис пропозицій, оновлення статусу, виконання транзакцій та відображення даних на різних вкладках головного екрана. Алгоритми є основними визначальними факторами роботи програми та створюють зв'язок між інтерфейсом користувача, локальною базою даних та зовнішнім джерелом інформації про курси криптовалют. Значна частина логіки знаходиться в компонентах MainActivity та TradeRequestDatabase, де екрани взаємодіють один з одним, обробляються дії користувачів, а дані зберігаються в SQLite. Організація логіки таким чином дозволяє об'єднати логіку інтерфейсу користувача в один компонент, а всі взаємодії, пов'язані з даними та послугами, міститися в окремому модулі.

Під час розробки застосунку визначення початкового стану заявки також є частиною алгоритму, який використовується для його побудови. Після створення нової заявки вона не буде схвалена як офіційна пропозиція, доки адміністратор не перевірить її статус. Алгоритм також враховуватиме тип транзакції (купівля чи продаж), пов'язаної з застосунком, і як слід взаємодіяти з записом у майбутньому, враховуючи тип транзакції. Відповідну блок-схему алгоритму створення заявки наведено на рис. 3.2.

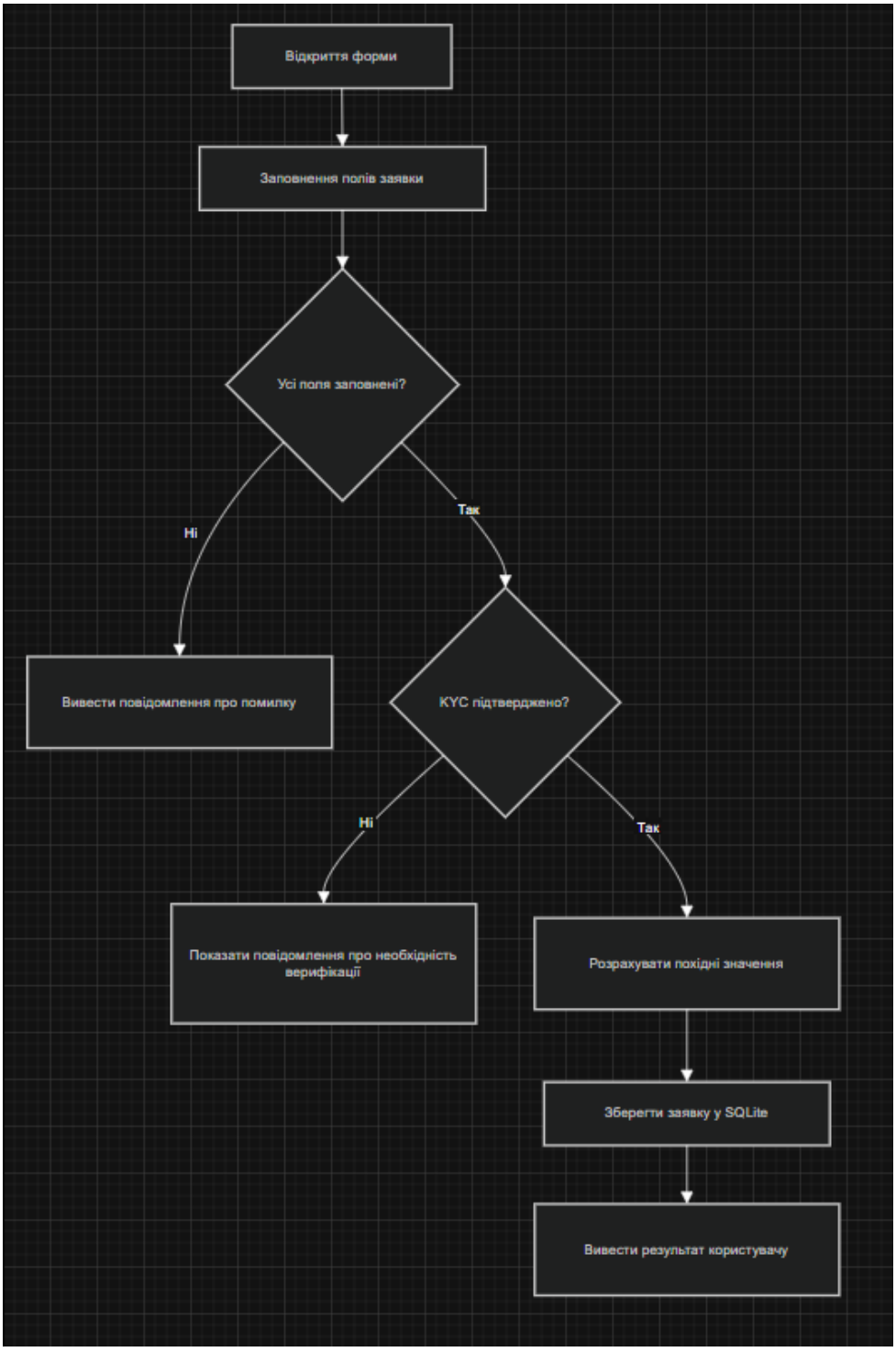


Рис. 3.2 Блок-схема алгоритму створення заявки

Заявки можуть бути схвалені або відхилені на основі різних сценаріїв, як і можливість змінити статус особи на KYC. Це вказує на те, що в системі використовуються два набори правил для обробки адміністративних дій. Адміністратор може переглядати свій набір заявок, фільтрувати їх за монетою, типом транзакції або статусом заявки, а потім приймати рішення щодо окремого запису заявки. Запити KYC працюють однаково. Адміністратор бачить повний список користувачів, вибирає конкретного користувача, а потім схвалює або відхиляє статус KYC цього користувача. Алгоритми як для заявок, так і для запитів KYC є важливими, оскільки вони встановлюють модель, що представляє повноваження контролюючого органу в інформаційній системі.

Особливо важливий алгоритм оновлення аналітичних даних після дій користувача або адміністратора. Зокрема, важлива здатність підтримувати актуальність статистики після створення, підтвердження та виконання заявок. Це означає, що під час заповнення баз даних даними аналітичні екрани відображатимуть точне відображення поточного стану заявки за допомогою статистичних даних. Зміни до вищезазначеного також повинні відображатися в історії заявок, відповідних списках пропозицій, профілі користувача та інших пов'язаних компонентах інтерфейсу користувача.

Модуль підтримки було вдосконалено шляхом створення алгоритму обміну повідомленнями, який має індикатор «прочитання» та лічильник «нових повідомлень». Таким чином, система зберігатиме кожне повідомлення в локальній базі даних, де воно пов’язуватиметься з автором, діалогом, часом створення та статусом «прочитано» чи ні. Коли інша сторона звертається до вікна чату, будь-які непрочитані повідомлення оновлюватимуть свій статус, а також збільшуватимуть лічильник «нових повідомлень». В результаті, модуль підтримки надає засоби для надсилання тексту, а також спрощене представлення способу внутрішнього спілкування користувача та адміністратора один з одним. Алгоритм обробки повідомлень у модулі підтримки подано на рис. 3.3.

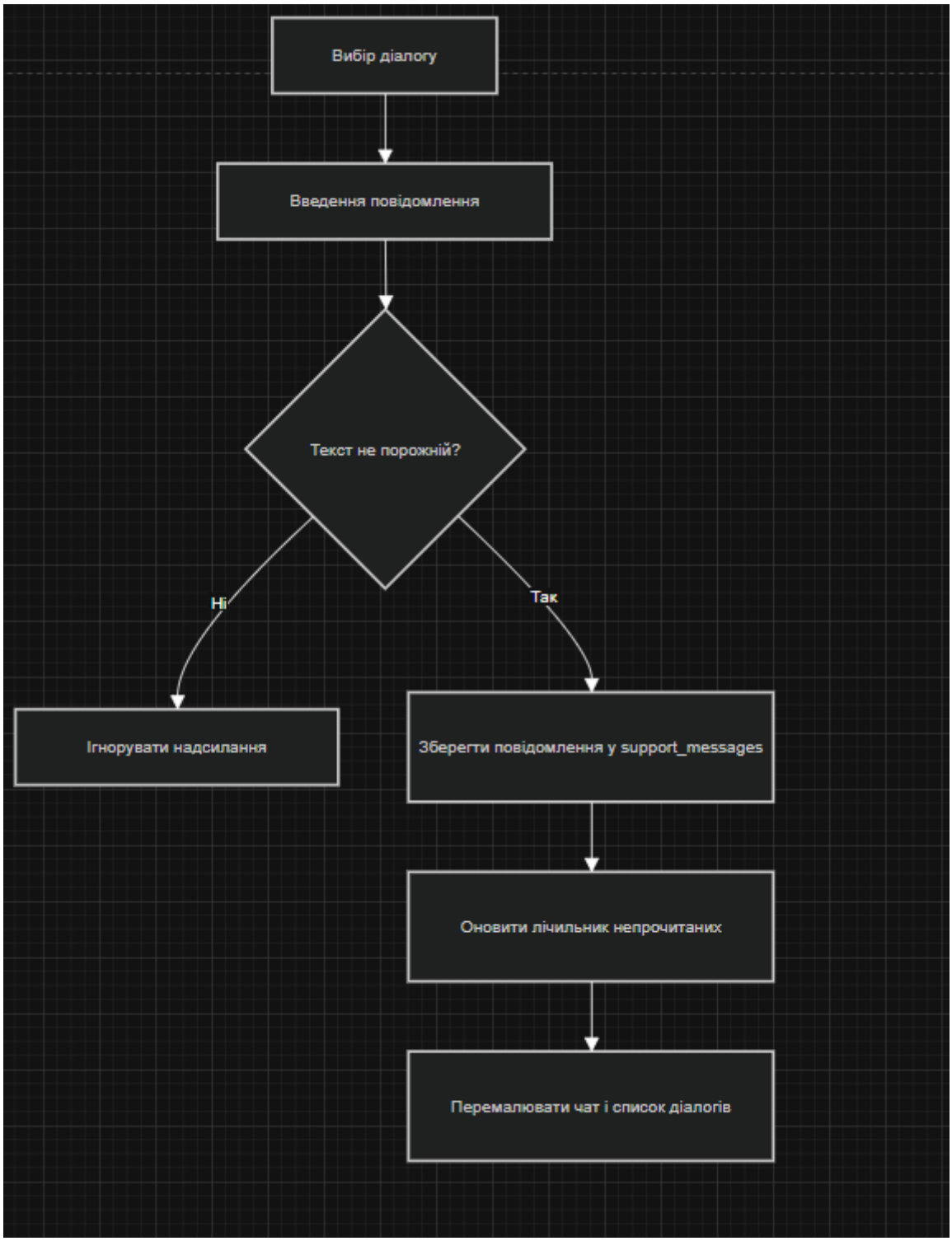


Рис. 3.3 Блок-схема алгоритму обробки повідомлень підтримки

Алгоритмізація основних процесів у LITVIN BALANCE створює основу для координації діяльності всіх функціональних модулів (Авторизація, KYC, Заявки, Пропозиції, Крипто-гаманець, Аналітика, Підтримка). Завдяки вбудовуванню реалізованих алгоритмів у класи MainActivity, TradeRequestDatabase та допоміжні класи, узгоджена внутрішня логіка системи

гарантує належний порядок виконання дій як користувача, так і адміністратора, що призводить до належного відображення результатів через мобільний інтерфейс. Приклади реалізації основних екранів застосунку наведено на рис. 3.4–3.7.

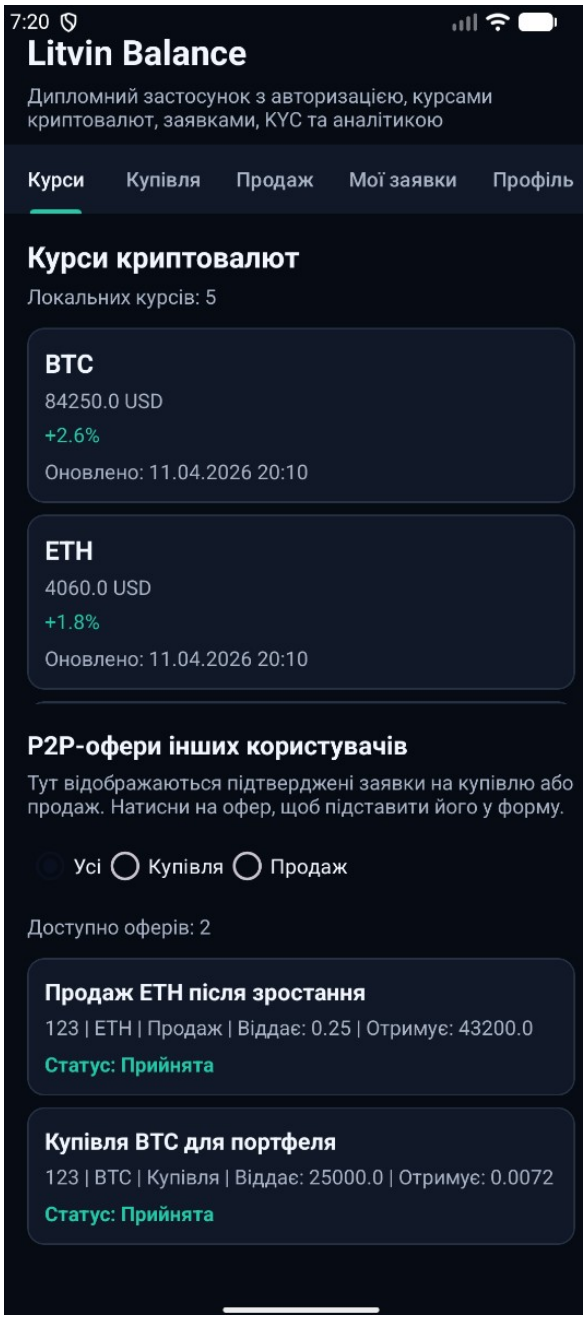


Рис. 3.4 Екран вкладки «Курси» з переліком криптовалют, поточними цінами та відсотковою зміною курсу

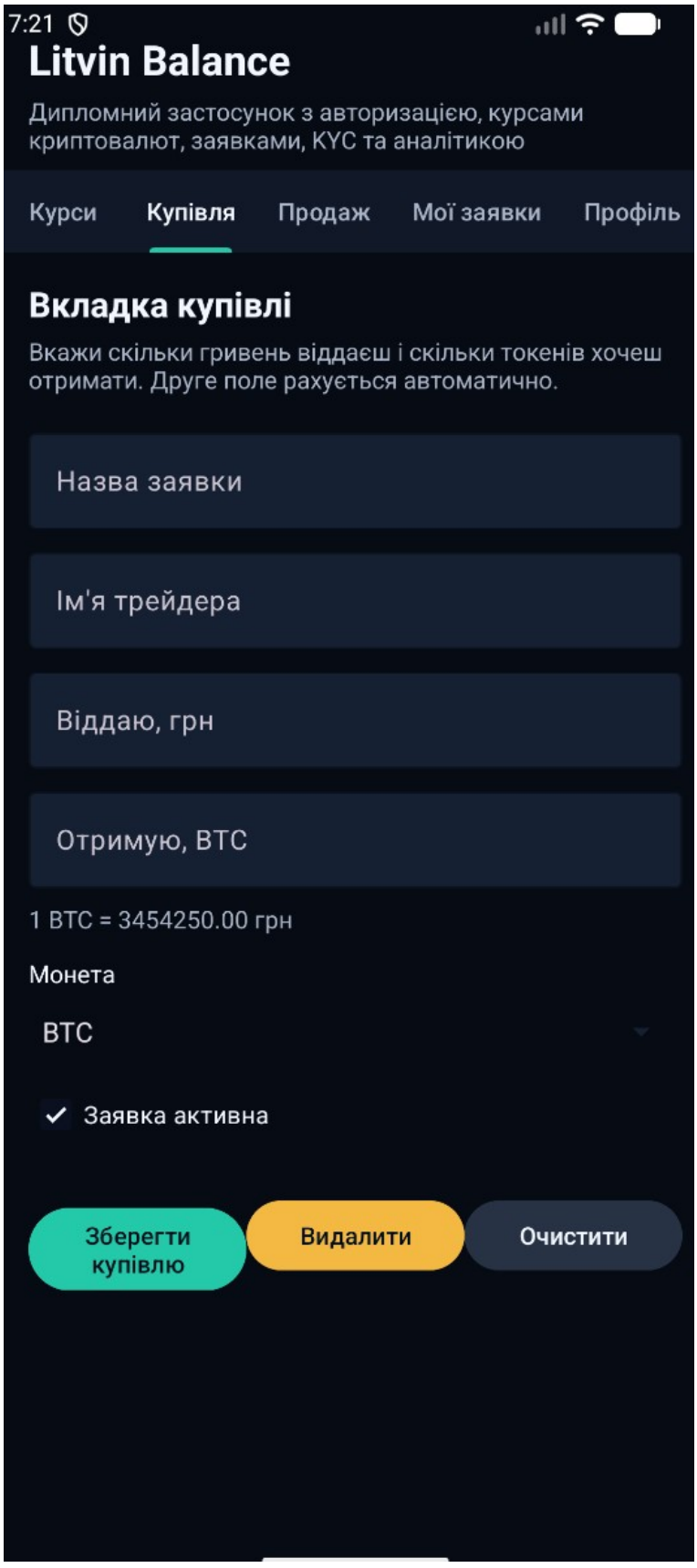


Рис. 3.5 Екран створення заявки на купівлю, де видно поля назви заявки, імені трейдера, суми та вибір монети

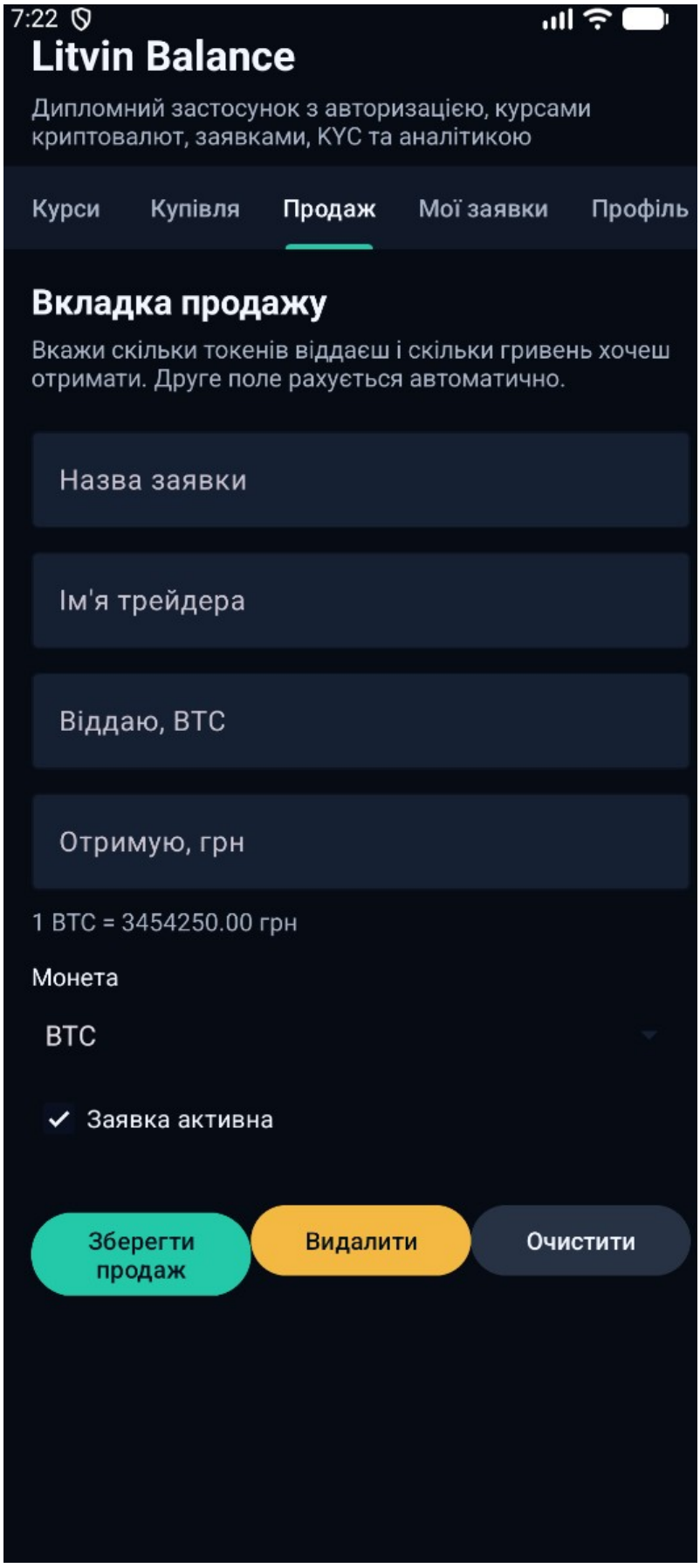


Рис. 3.6 Екран створення заявки на продаж

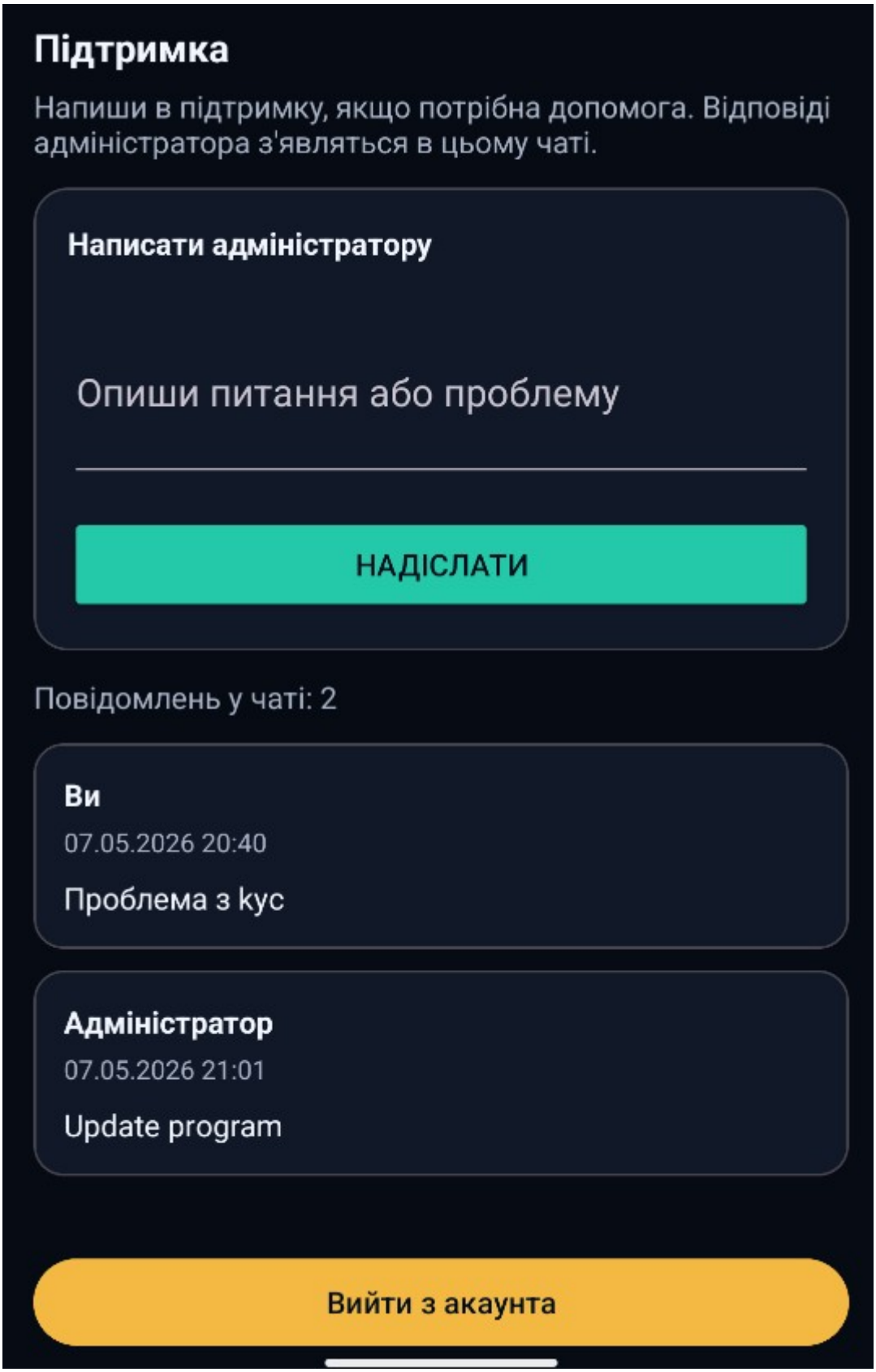


Рис. 3.7 Екран підтримки, у якому для адміністратора відображено список діалогів і чат із користувачем

4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ

4.1 Тестування системи

Фаза тестування розробки – це останній крок, який надає засоби для оцінки того, чи функціональність відповідатиме вимогам. Ви можете оцінити, наскільки добре працює кожен модуль програми, чи дії користувача призведуть до очікуваного результату, і чи система поводитиметься належним чином, виходячи зі сценаріїв, визначених у обсягу робіт. Для цього конкретного проекту функціональне тестування має бути основним завданням, оскільки є чітко визначені ролі, сценарії взаємодії та логічні переходи від стану до стану. Тестування не лише виявляє помилки в програмі, але й підтверджує, що програмний продукт готовий до демонстрації, захищений від несанкціонованого доступу та використовується в освітніх/навчальних цілях.

Спочатку потрібно перевірити основні функції системи: а саме: вхід у систему; реєстрацію нових користувачів; виконання перевірок KYC; створення/модерацію заявок; обробку пропозицій; та як використовувати криптовалютний гаманець на додаток до аналітики користувачів та обслуговування клієнтів. Для кожної з цих функцій слід перевірити як умову успіху, так і деякі поширені умови невдачі. На прикладі входу/автентифікації буде перевірено успішний та невдалий вхід (тобто правильна комбінація пароля/імені користувача проти неправильної комбінації пароля/імені користувача). Для реєстрації слід перевірити, чи система зберегла правильні значення для ролі користувача, імені користувача та інших пов'язаних атрибутів. Що стосується модуля KYC, потрібно перевірити, чи модуль KYC отримав інформацію користувача, надіслану з веб-інтерфейсу, і чи оновив він свій статус для перевірки адміністратором. Приклади тестових сценаріїв для системи LITVIN BALANCE наведено в таблиці 4.1.

Таблиця 4.1

Приклади тестових сценаріїв для системи LITVIN BALANCE

Сценарій тестування	**Очікуваний результат**
Успішна авторизація зареєстрованого користувача	Користувач переходить на головний екран із доступом до вкладок.
Спроба створення заявки без KYC	Система виводить повідомлення про необхідність пройти верифікацію.
Подання KYC-запиту та його підтвердження адміністратором	Статус користувача змінюється на «Підтверджено».
Фільтрація заявок за монетою BTC	На екрані залишаються лише записи для BTC.
Сортування оферів за ціною	Список оферів змінює порядок відповідно до обраного сортування.
Надсилання повідомлення в підтримку	Повідомлення зберігається в чаті та відображається у співрозмовника як нове.

Тут наголошується на випадку, коли користувач створює заявку, не завершивши перевірку KYC. Це критично важливий контроль для застосунку, оскільки він безпосередньо пов'язаний з граничним контролем прав доступу до заявки та з логікою безпеки застосунку. У цьому випадку застосунок повинен правильно заборонити створення заявки в базі даних та повідомити користувача про необхідність завершити перевірку KYC. Приклад такого повідомлення наведено на рис. 4.1. Тестування цієї логіки важливе з технічних причин та для відповідності бізнес-логіці застосунку. Невдача процесу перевірки KYC дозволить користувачеві створити перевірену заявку; це суперечить основним вимогам застосунку.

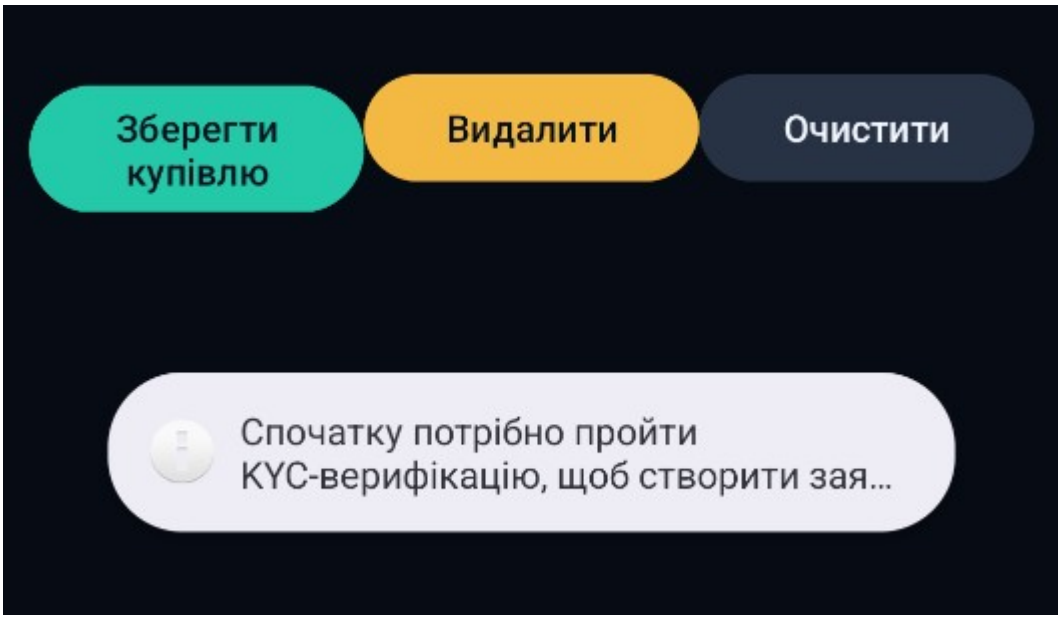


Рис. 4.1 Екран із повідомленням про неможливість створити заявку без підтвердженого KYC.

Також потрібно перевірити функціональність адміністративного модуля, який дозволяє видаляти створені заявки шляхом фільтрації, підтвердження або відхилення заявок, а також змінювати статус KYC користувачів. Після кожної взаємодії адміністратору важливо переконатися, що статус оновлено правильно, і що будь-які внесені зміни будуть негайно відображені в будь-якій іншій області програми, пов’язаній з цією взаємодією. Наприклад, якщо адміністратор підтвердив KYC користувача, то цей користувач тепер матиме доступ до створення заявок у програмі. Якщо адміністратор змінив статус пропозиції, то цей оновлений статус пропозиції має відображатися у відповідних списках заявок, звітах, аналітиці та в історії дій адміністратора.

Потрібно перевірити фільтрацію заявок за монетою, сортування за ціною, функціональність лічильників кількості нових повідомлень та оновлення статусів після відповіді адміністратора на елемент. Хоча ці функції не є фундаментальними при створенні елемента, вони відіграють важливу роль у тому, наскільки легко використовувати систему та наскільки точно вона працює в повсякденних життєвих сценаріях. Наприклад, фільтрація зможе відображати лише ті записи, які відповідають критеріям вибраної монети або статусу,

сортування точно змінюватиме порядок пропозицій, а нові лічильники повідомлень реагуватимуть на створення нових запитів на підтримку. Виконання цих тестів покаже, що програма працює як єдина повноцінна інформаційна система, а не просто як набір окремих функцій.

Тим не менш, існує ще один критичний аспект тестування, який завжди слід враховувати. Знову ж таки, це стосується тестування модуля криптогаманця та виконання пропозиції. Подібно до попереднього тесту, вам потрібно переконатися, що після завершення замовлення локальні токени належним чином скориговані. Вам потрібно переконатися, що жодна частина системи не може створювати негативні або нелогічні ключі чи значення; і, загалом, що тип транзакції був точно врахований. У випадку продажів гаманець повинен перевірити наявність достатньої кількості токенів, і що, якщо після завершення замовлення токенів недостатньо, транзакцію не слід вважати успішно завершеною. Ці типи тестів дозволяють не лише перевірити правильність відображення даних, але й підтвердити належне забезпечення внутрішньої узгодженості даних.

Модуль підтримки перевіряє, чи повідомлення були надіслані, чи була записана історія листування, чи точно відображена роль відправника, а також чи є статус прочитання актуальним. Це критично важливо для повноцінної функціональності чату підтримки. Він не повинен служити просто сховищем статичного тексту. Крім того, модуль аналітики повинен перевіряти, чи оновлюються графіки під час створення, підтвердження або виконання запиту, щоб аналітичні показники відображали поточний стан системи.

Отже, тестування (зокрема функціональне тестування) для проєкту LITVIN BALANCE набуває критичного значення, оскільки тестування дозволяє нам перевірити, чи всі основні модулі функціонують належним чином; виявити помилки в бізнес-логіці; та підтвердити відповідність заданим вимогам. Виконуючи функціональне тестування, ми також можемо гарантувати успішне виконання сценаріїв використання користувачами та адміністраторами, а також

стабільність, логічну узгодженість та готовність до використання в рамках бакалаврського проєкту за програмою.

4.2 Вимоги до апаратного та програмного забезпечення

Для роботи застосунку потрібен мобільний пристрій на базі Android, який використовує однойменний APK-файл. Локальна база даних зберігатиметься на вашому пристрої, що дозволить вам використовувати основні функції офлайн без потреби в додаткових серверах. Щоб отримати доступ до поточних курсів криптовалюти, застосунок повинен бути підключений до Інтернету, а також до зовнішнього API Binance для отримання даних про курси. Схему розгортання системи LITVIN BALANCE наведено на рис. 4.2.

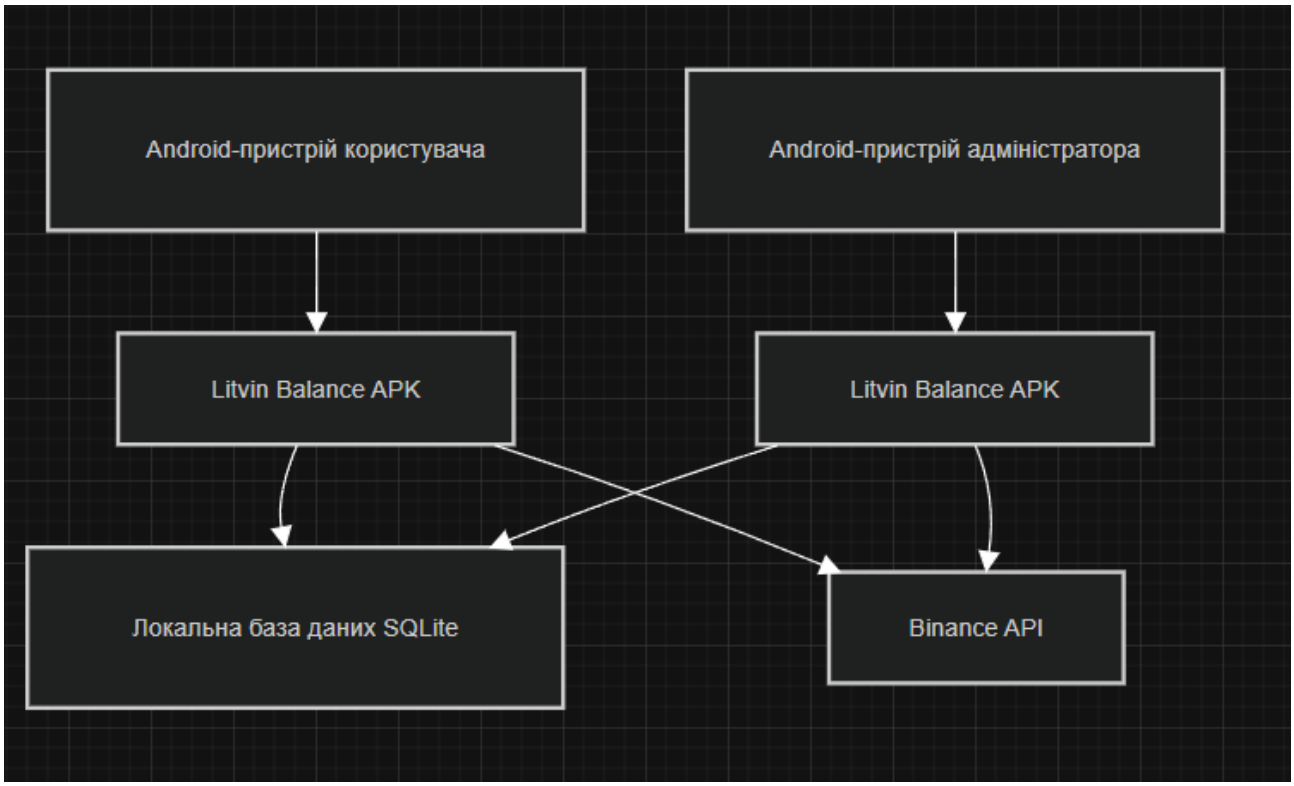


Рис. 4.2 Діаграма розгортання системи LITVIN BALANCE

Для створення, оцінки та компіляції прикладного програмного забезпечення потрібен ПК з Android Studio; JDK; SDK для розробки програмного забезпечення для Android; та емулятор або фізичний пристрій для тестування.

Якщо цю систему розширити до багатокористувацької серверної версії, вона зможе перейти на серверну СУБД та централізовану базу даних. Однак наразі для задоволення ваших потреб достатньо мати локальну архітектуру. Мінімальні вимоги до апаратного та програмного забезпечення наведено в табл. 4.2.

Таблиця 4.2

Мінімальні вимоги до апаратного та програмного забезпечення

Компонент	**Мінімальні вимоги**
Мобільний пристрій	Смартфон або планшет з Android 8.0 або новішою версією ОС.
Пам’ять пристрою	Не менше 2 ГБ оперативної пам’яті та 200 МБ вільного простору для інсталяції й локальної бази даних.
Мережеве з’єднання	Доступ до Інтернету для завантаження курсів криптовалют через Binance API.
Середовище розробки	Windows 10/11 або інша сумісна ОС, Android Studio, JDK, Android SDK.

4.3 Склад інсталяційного пакету

Для запуску системи LITVIN BALANCE на Android-пристроях сформовано інсталяційний пакет у форматі .apk. Цей пакет містить скомпільовані класи програми, ресурси інтерфейсу, конфігураційні файли, маніфест, графічні елементи, макети екранів та інші компоненти, необхідні для коректної роботи застосунку на мобільному пристрої.

Під час підготовки проєкту я вважаю доцільним окремо зберігати вихідний код програми, скомпільований .apk-файл, скріншоти інтерфейсу користувача, ліцензійну угоду, текстові матеріали для сторінки продукту та презентаційні зображення. Такий підхід дає мені змогу швидко продемонструвати готовий програмний продукт, підготувати матеріали до захисту, а також за потреби

продовжити подальшу розробку та супровід системи. Основні складові інсталяційного пакета та супровідних матеріалів наведено в табл. 4.3.

Таблиця 4.3

Основні складові інсталяційного пакету та супровідних матеріалів

Складова	**Призначення**
LitvinBalance.apk	Інсталяція застосунку на Android-пристрій.
AndroidManifest.xml	Опис дозволів, активностей і базових параметрів застосунку.
res/layout	XML-макети екранів і компонентів інтерфейсу.
res/drawable	Графічні ресурси, іконки та допоміжні елементи оформлення.
src/main/java	Вихідний код програмних модулів.
Матеріали для маркету	Промо-банер, скріншоти, EULA та текстові описи продукту.

Ліцензійна угода з кінцевим користувачем (EULA) LITVIN BALANCE

1. Встановлюючи або використовуючи мобільний додаток LITVIN BALANCE, користувач погоджується з умовами цієї ліцензійної угоди.

2. LITVIN BALANCE – це застосунок, який дозволяє перевіряти ціни на криптовалюту, розміщувати ордери на купівлю/продаж, проходити верифікацію KYC, отримувати доступ до свого криптогаманця, аналізувати транзакції та спілкуватися із системними адміністраторами через повідомлення.

3. Користувачеві дозволено використовувати програму на неексклюзивній основі, але це не надає права продавати, декомпілювати, змінювати, розповсюджувати або іншим чином використовувати Програмний код

4. Користувач погоджується використовувати програму лише в законних цілях; крім того, користувач не виконуватиме жодних дій, які можуть спричинити порушення роботи Системи.

5. Дані про курси криптовалют, що відображаються в додатку, отримані із зовнішніх джерел, зокрема через Binance API, та надаються лише в інформаційних цілях. Офіційна біржа криптовалют, фінансова установа чи платіжна служба не включають цю програму.

6. Розробник не несе жодної відповідальності за фінансові рішення користувачів, прийняті на основі інформації, наданої через програму.

7. Через внутрішню логіку системи користувачі повинні пройти верифікацію KYC, перш ніж зможуть створювати заявки на купівлю та продаж криптовалюти.

8. Системний адміністратор може схвалювати або відхиляти заявки, подані користувачами, та виконувати верифікацію KYC для таких заявок.

9. Доступ користувача до певних функцій програми може бути обмежений, якщо він порушує ці правила використання.

10. Розробник має право змінювати або додавати функції до функціональності програми або вносити зміни до цієї угоди.

Приклад промо-банера системи LITVIN BALANCE наведено на рис. 4.3.

Рис. 4.3 Промо-банер системи LITVIN BALANCE

ВИСНОВКИ

Мобільний додаток LITVIN BALANCE для купівлі та продажу криптовалют був розроблений автором бакалаврської роботи. Під час створення цього додатку було проаналізовано предметну область, визначено вимоги, створено логічну модель даних, обрано інструменти та побудовано основні структурні блоки додатку.

Створення програмного продукту на етапі розробки призвело до появи програми, яка пропонує такі функції: реєстрація та автентифікація користувачів; перевірка KYC; моніторинг курсів криптовалют; створення та виконання P2P-заявок; перегляд локального криптогаманця; відображення аналітичних даних; спілкування між Користувачем/Клієнтом та Адміністрацією через обмін повідомленнями. Крім того, були додані адміністративні функції, такі як запити KYC; модерація заявок; служби підтримки.

Загалом, практичне значення отриманих результатів полягає в тому, щоб надати повний освітній та прикладний приклад мобільної інформаційної системи для управління цифровими активами. Запропоноване рішення може слугувати відправною точкою для майбутнього розвитку серверної архітектури, що включає методи оплати в режимі реального часу, розширені можливості аналітики та багатокористувацький доступ через централізовану синхронізацію даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

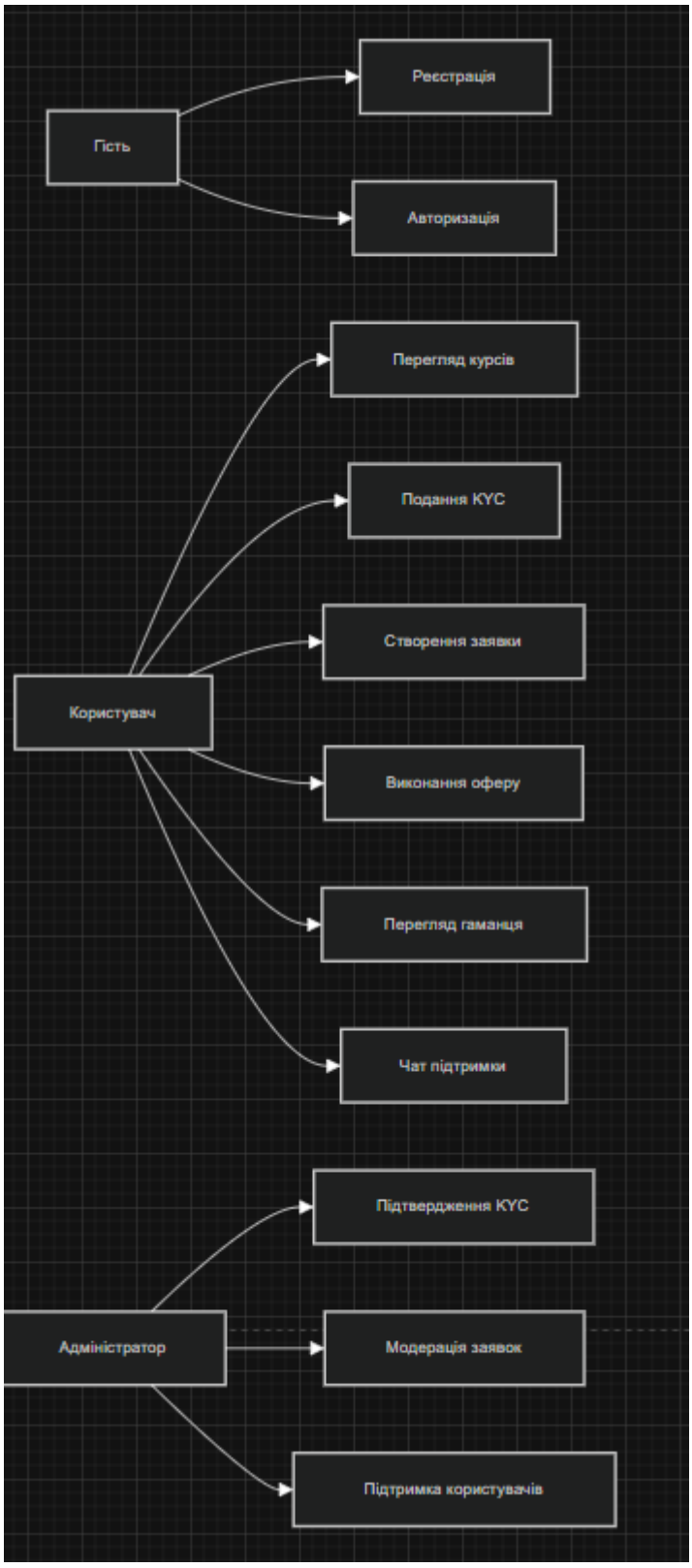
1. Методичні вказівки до розробки бакалаврської кваліфікаційної роботи для студентів спеціальності 122 «Комп'ютерні науки» освітнього ступеня «Бакалавр». Київ : НУБіП України, 2025.
2. Android Developers. Build your first app [Електронний ресурс]. URL: https://developer.android.com/training/basics/firstapp (дата звернення: 13.05.2026).
3. Android Developers. Guide to app architecture [Електронний ресурс]. URL: https://developer.android.com/topic/architecture (дата звернення: 13.05.2026).
4. Android Developers. Activities overview [Електронний ресурс]. URL: https://developer.android.com/guide/components/activities/intro-activities (дат а звернення: 13.05.2026).
5. Android Developers. Create a view-based layout [Електронний ресурс]. URL: https://developer.android.com/develop/ui/views/layout/declaring-layout (дата звернення: 13.05.2026).
6. Material Design 3. Design system for Android [Електронний ресурс]. URL: https://m3.material.io/ (дата звернення: 13.05.2026).
7. Kotlin Documentation. Kotlin language documentation [Електронний ресурс]. URL: https://kotlinlang.org/docs/home.html (дата звернення: 13.05.2026).
8. SQLite Documentation. SQLite documentation [Електронний ресурс]. URL: https://www.sqlite.org/docs.html (дата звернення: 13.05.2026).
9. Binance Developers. Spot API documentation [Електронний ресурс]. URL: https://github.com/binance/binance-spot-api-docs (дата звернення: 13.05.2026).

10. FATF. Updated Guidance for a Risk-Based Approach to Virtual Assets and Virtual Asset Service Providers [Електронний ресурс]. Paris, 2021. URL: https://www.fatf-gafi.org/en/publications/Fatfrecommendations/Guidance-rba-virtual-assets-2021.html (дата звернення: 13.05.2026).
11. Nakamoto S. Bitcoin: A Peer-to-Peer Electronic Cash System [Електронний ресурс]. 2008. URL: https://bitcoin.org/bitcoin.pdf (дата звернення: 13.05.2026).
12. Antonopoulos A. M. Mastering Bitcoin: Programming the Open Blockchain. 2nd ed. Sebastopol : O'Reilly Media, 2017.
13. Pressman R. S., Maxim B. R. Software Engineering: A Practitioner's Approach. 9th ed. New York : McGraw-Hill Education, 2019.
14. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. 3rd ed. Boston : Addison-Wesley, 2004.
15. OWASP Foundation. Mobile Application Security Verification Standard (MASVS) [Електронний ресурс]. URL: https://mas.owasp.org/MASVS/ (дата звернення: 13.05.2026).
16. OWASP Foundation. Mobile Application Security Testing Guide (MSTG) [Електронний ресурс]. URL: https://mas.owasp.org/MSTG/ (дата звернення: 13.05.2026).
17. Fielding R. T. Architectural Styles and the Design of Network-Based Software Architectures [Електронний ресурс] : Doctoral dissertation. University of California, Irvine, 2000. URL: https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm (дата звернення: 13.05.2026).
18. Bray T., ed. The JavaScript Object Notation (JSON) Data Interchange Format [Електронний ресурс]. RFC 8259. 2017. URL: https://www.rfc-editor.org/rfc/rfc8259 (дата звернення: 13.05.2026).

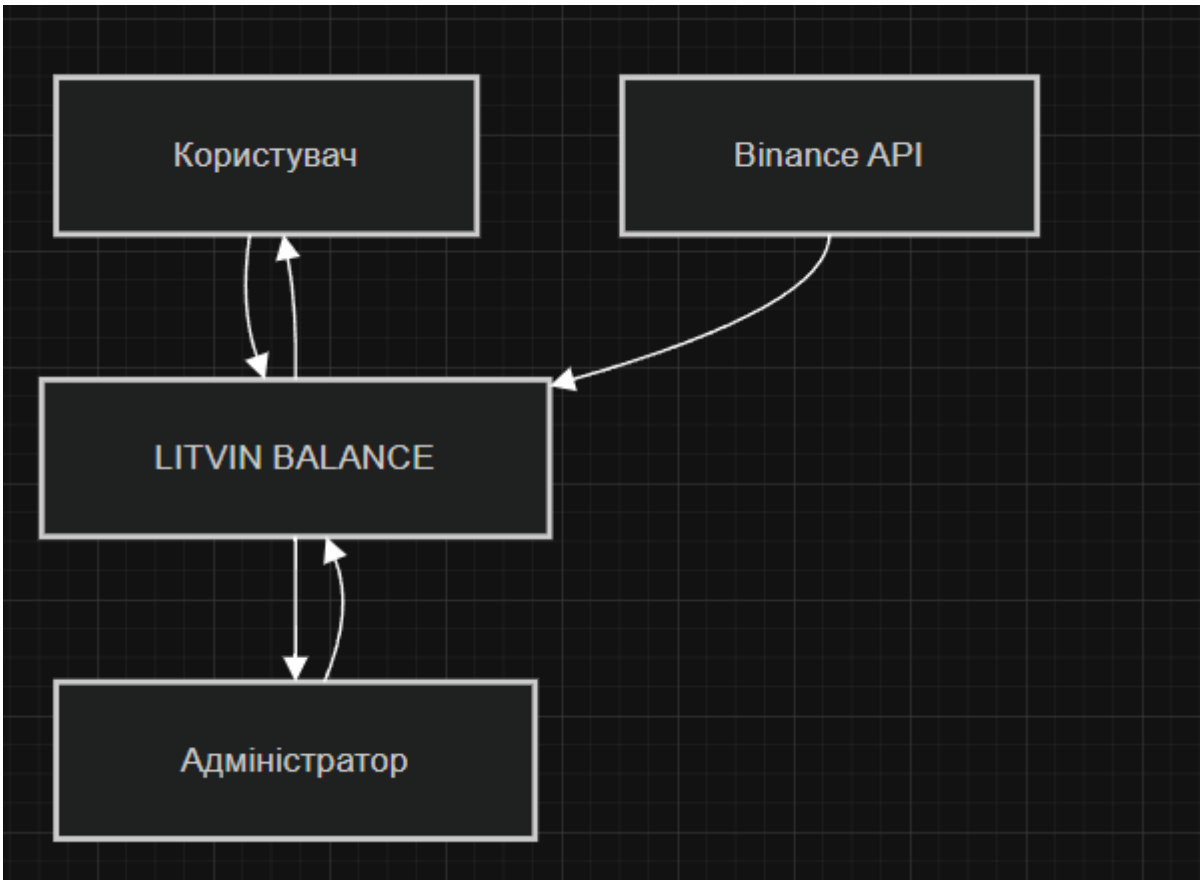
ДОДАТОК А

ДІАГРАМИ ПРОЕКТУ

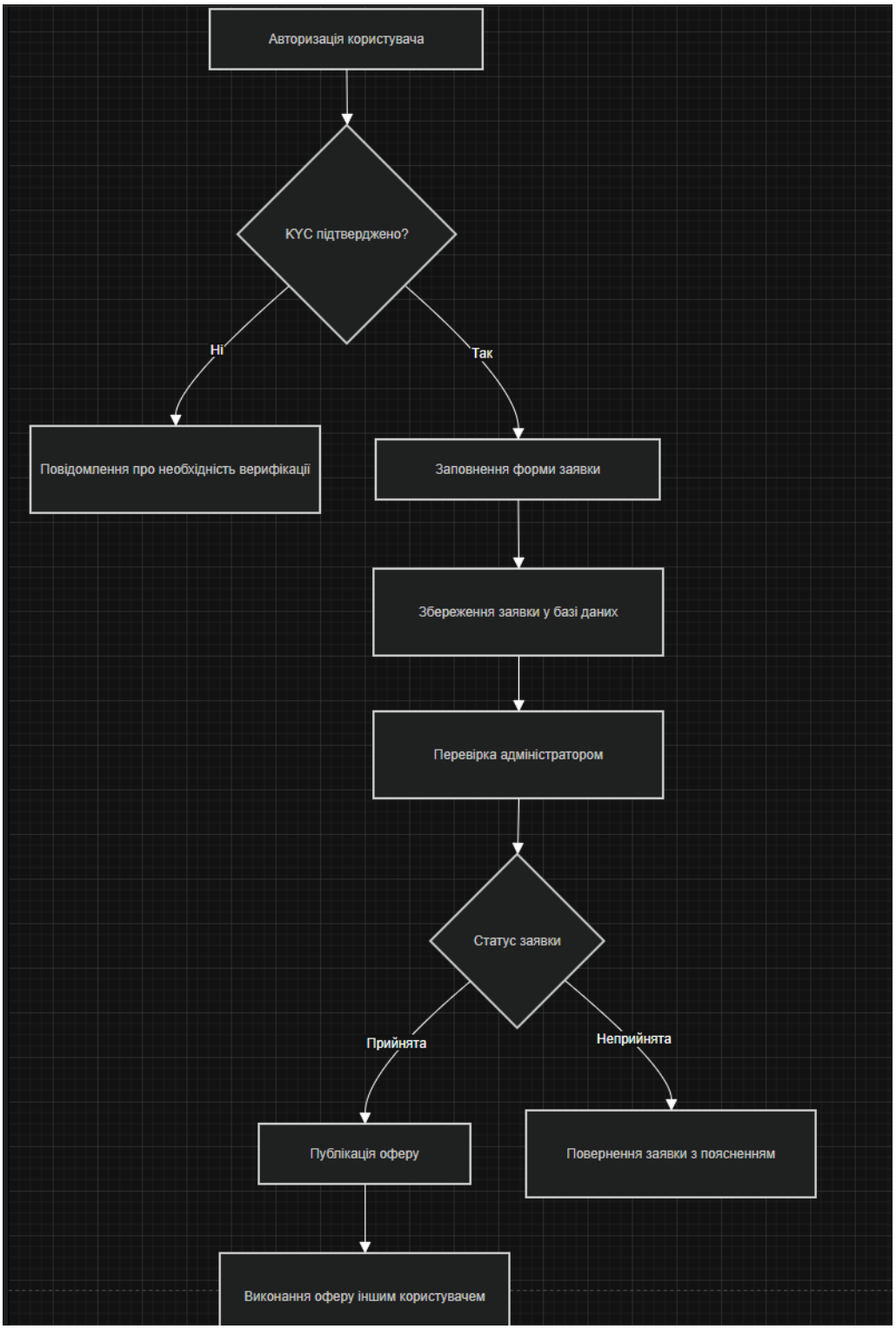
Сторінок – 8



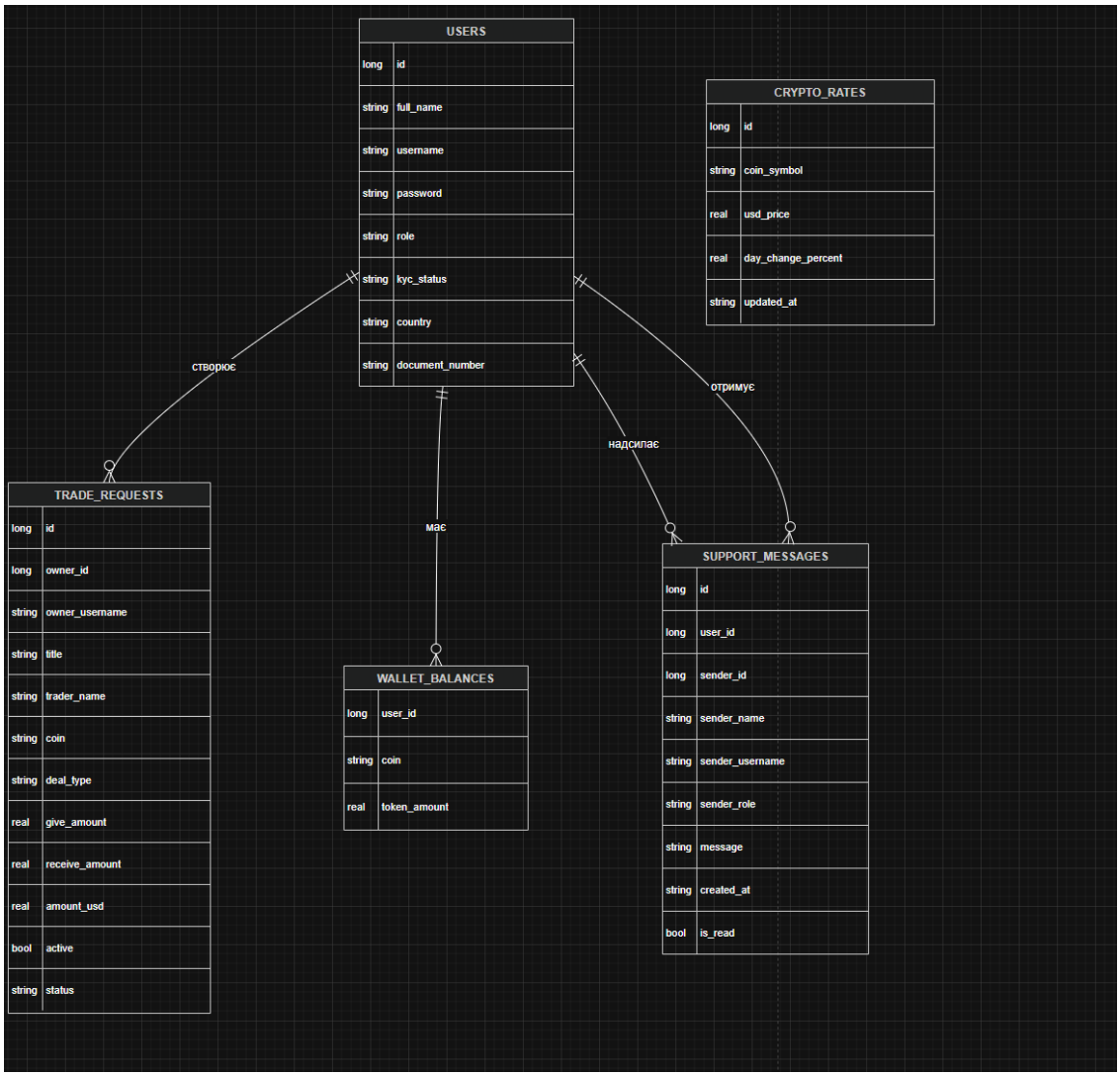
Діаграма прецедентів системи LITVIN BALANCE



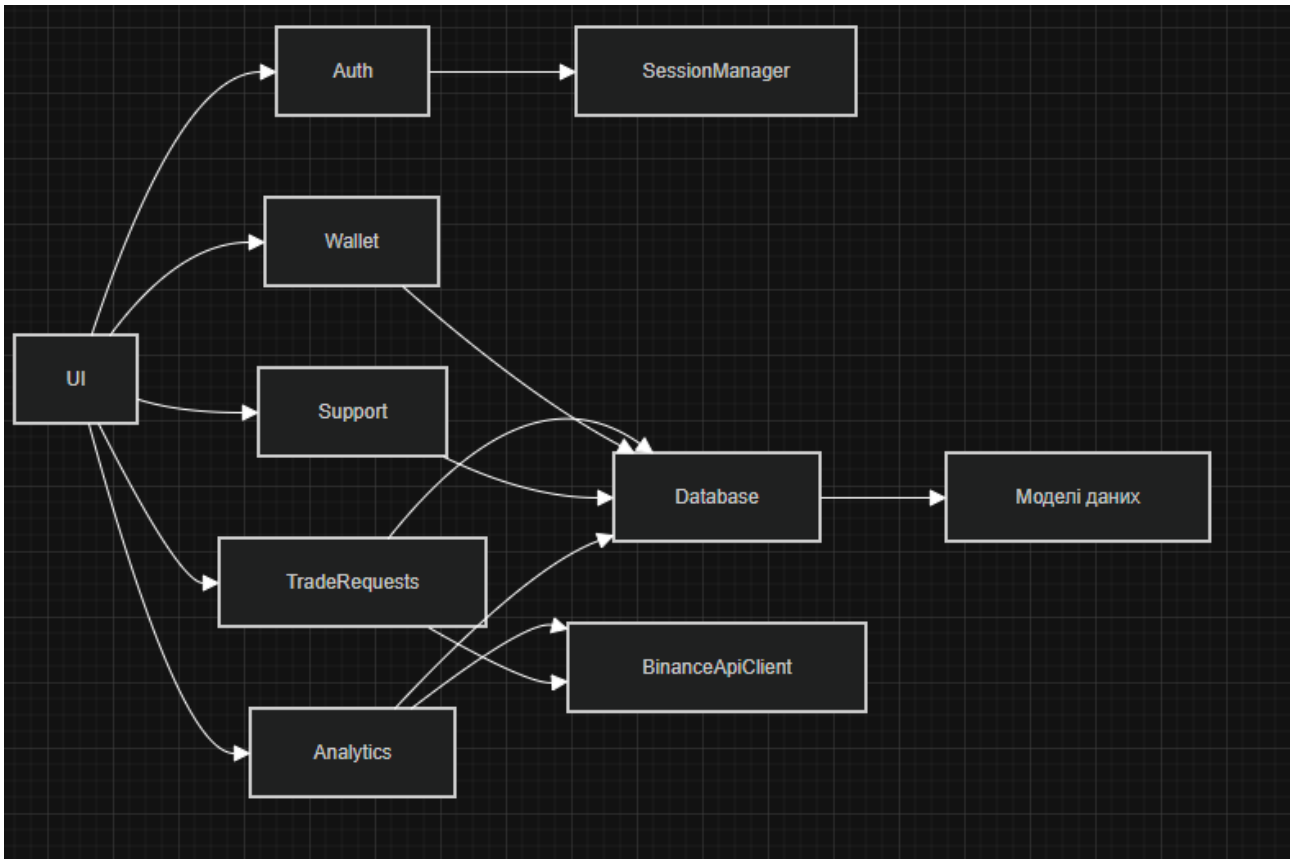
Контекстна діаграма системи LITVIN BALANCE



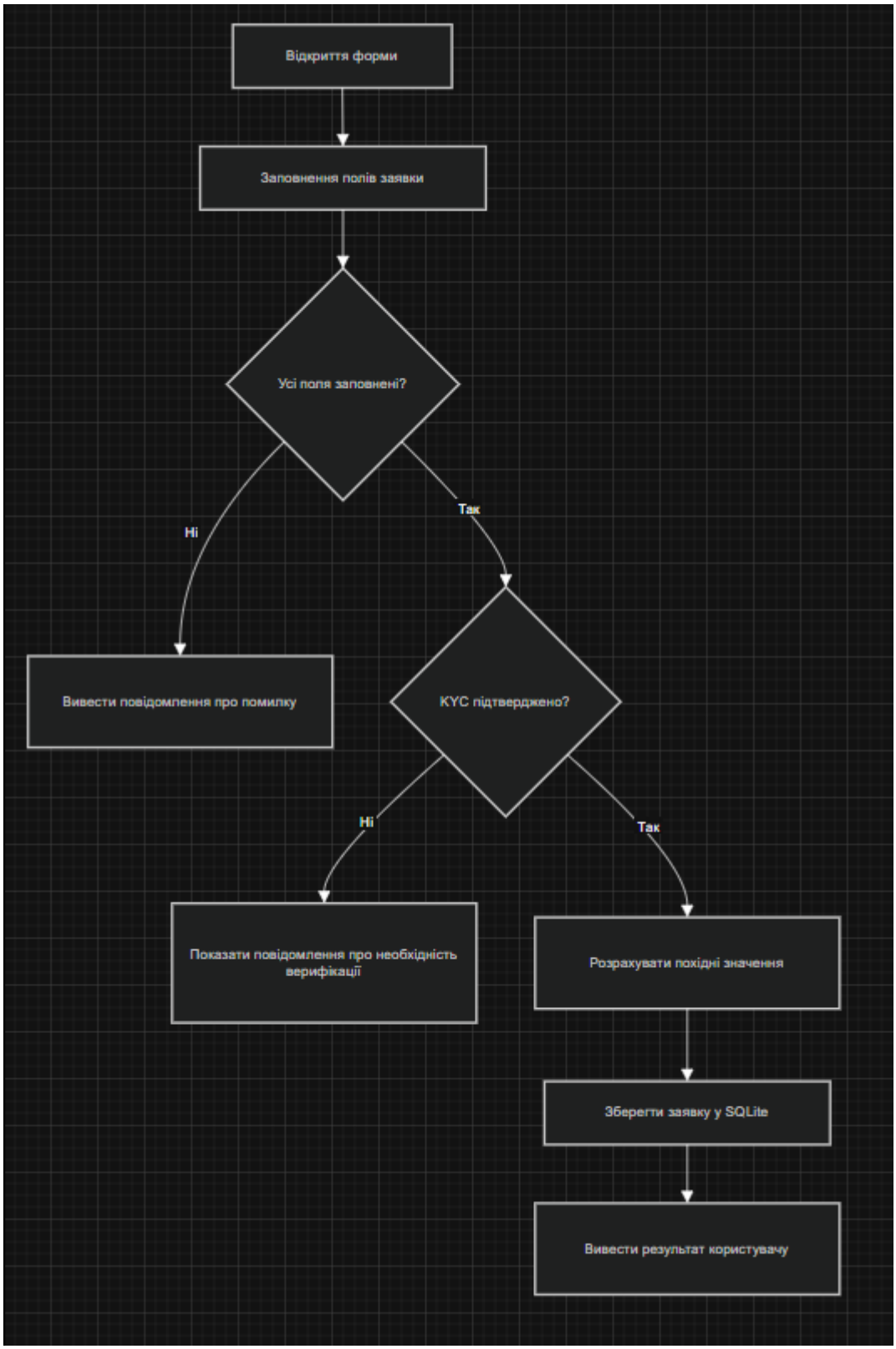
Діаграма основного бізнес-процесу створення та виконання заявки



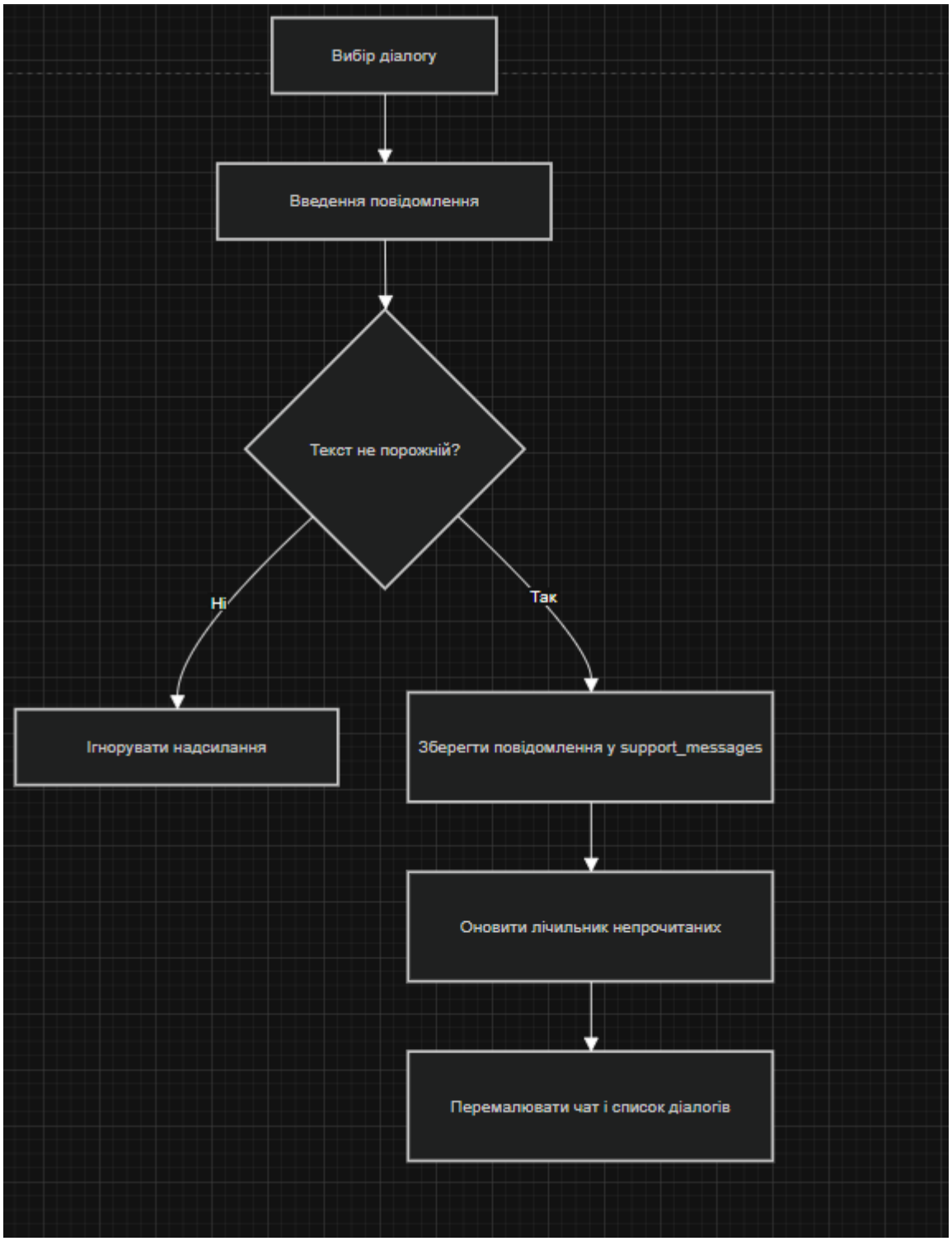
ER-діаграма інформаційної бази LITVIN BALANCE



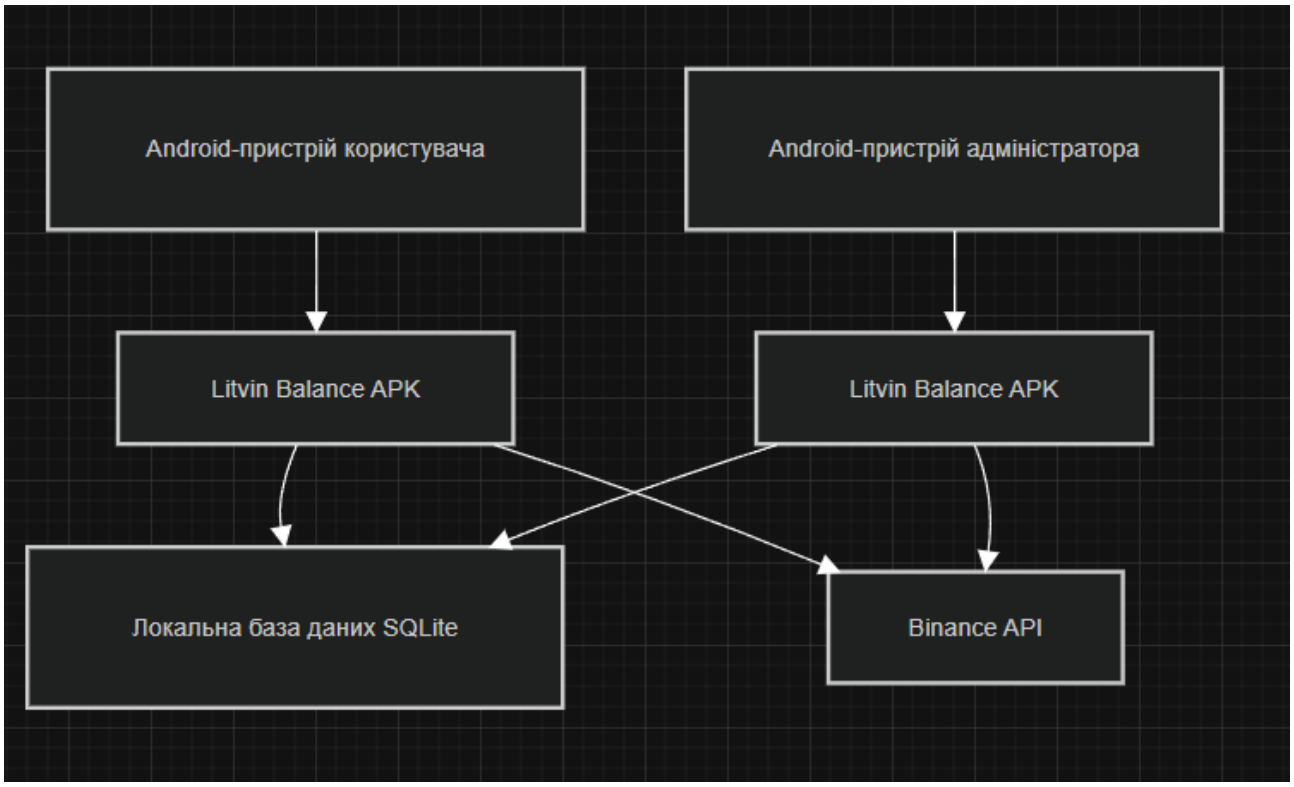
Діаграма пакетів застосунку LITVIN BALANCE



Блок-схема алгоритму створення заявки



Блок-схема алгоритму обробки повідомлень підтримки

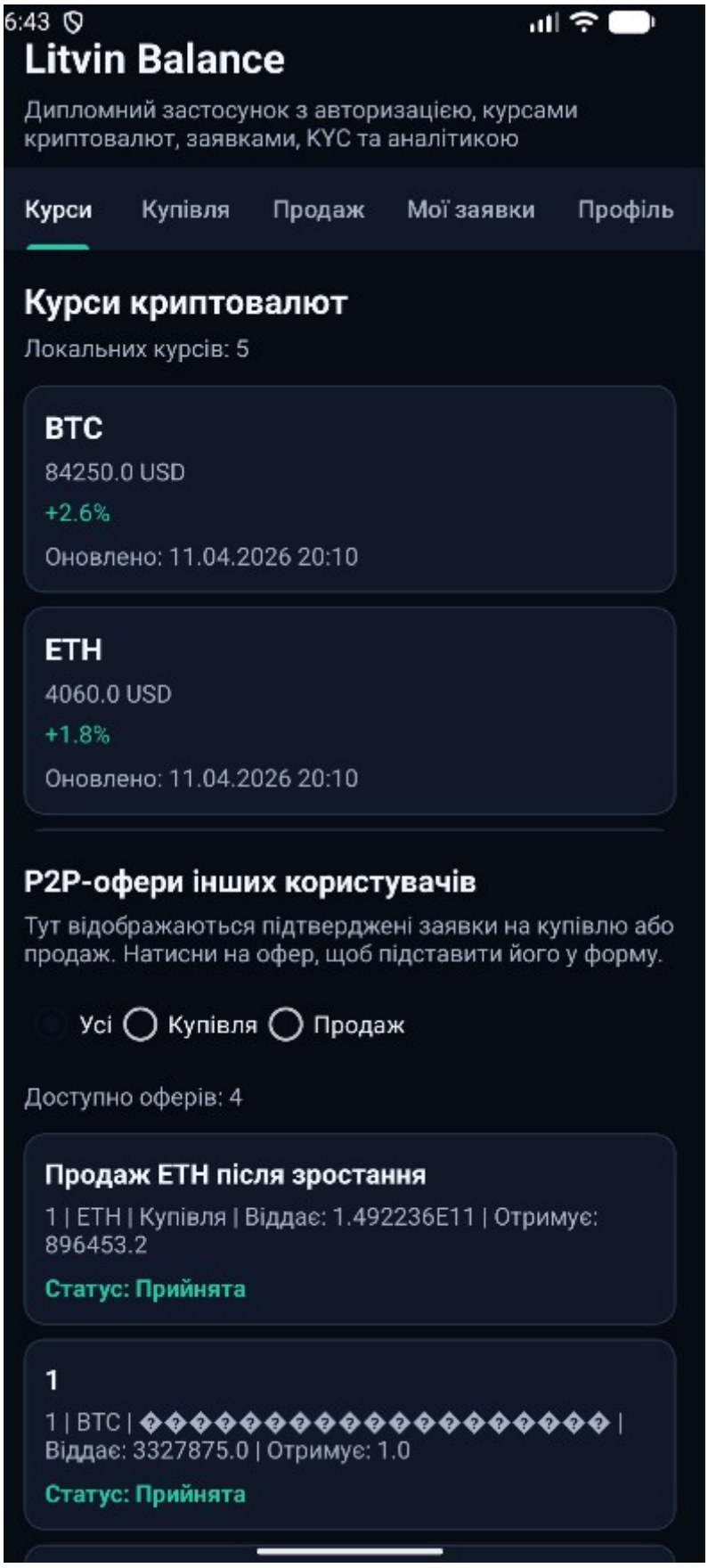


Діаграма розгортання системи LITVIN BALANCE

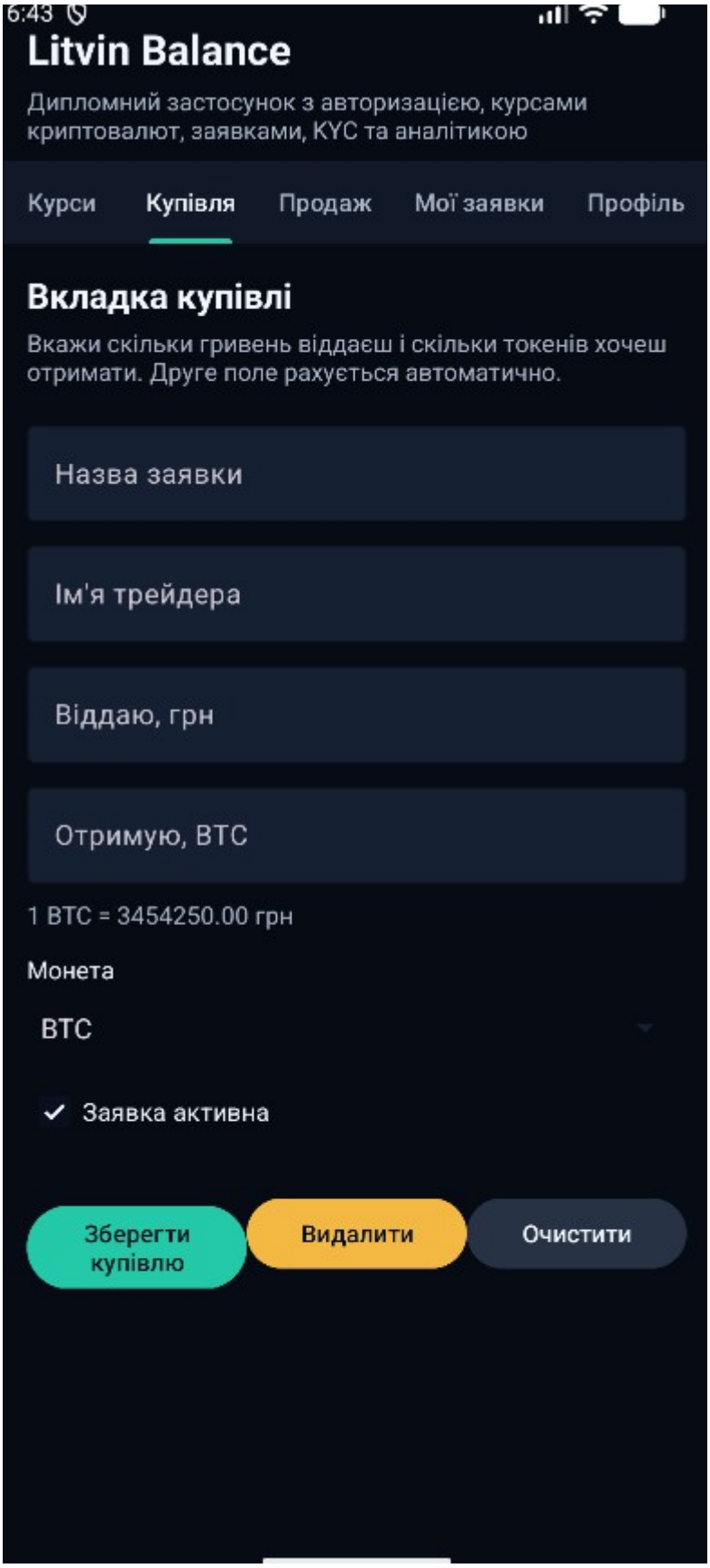
ДОДАТОК Б

СКРІНШОТИ ГОТОВОЇ ПРОГРАМИ

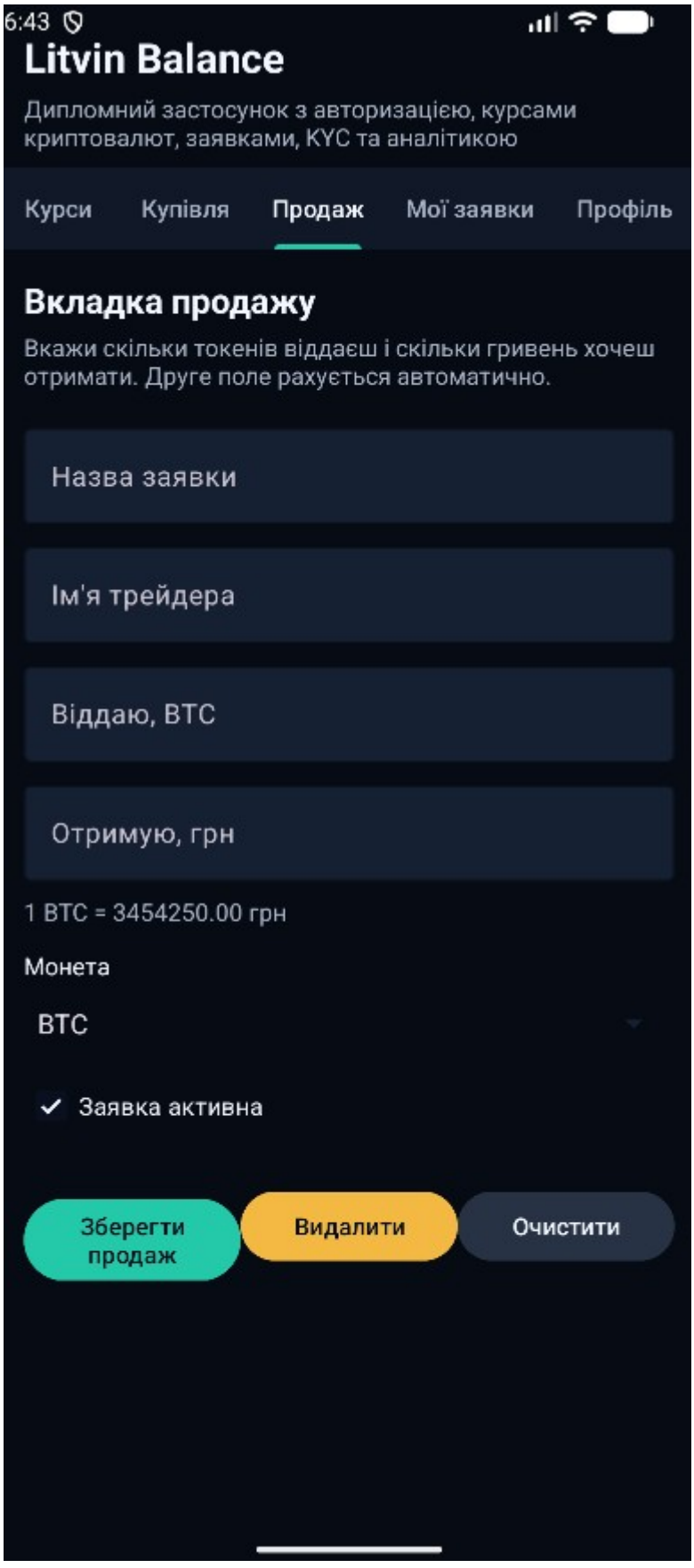
Сторінок – 10



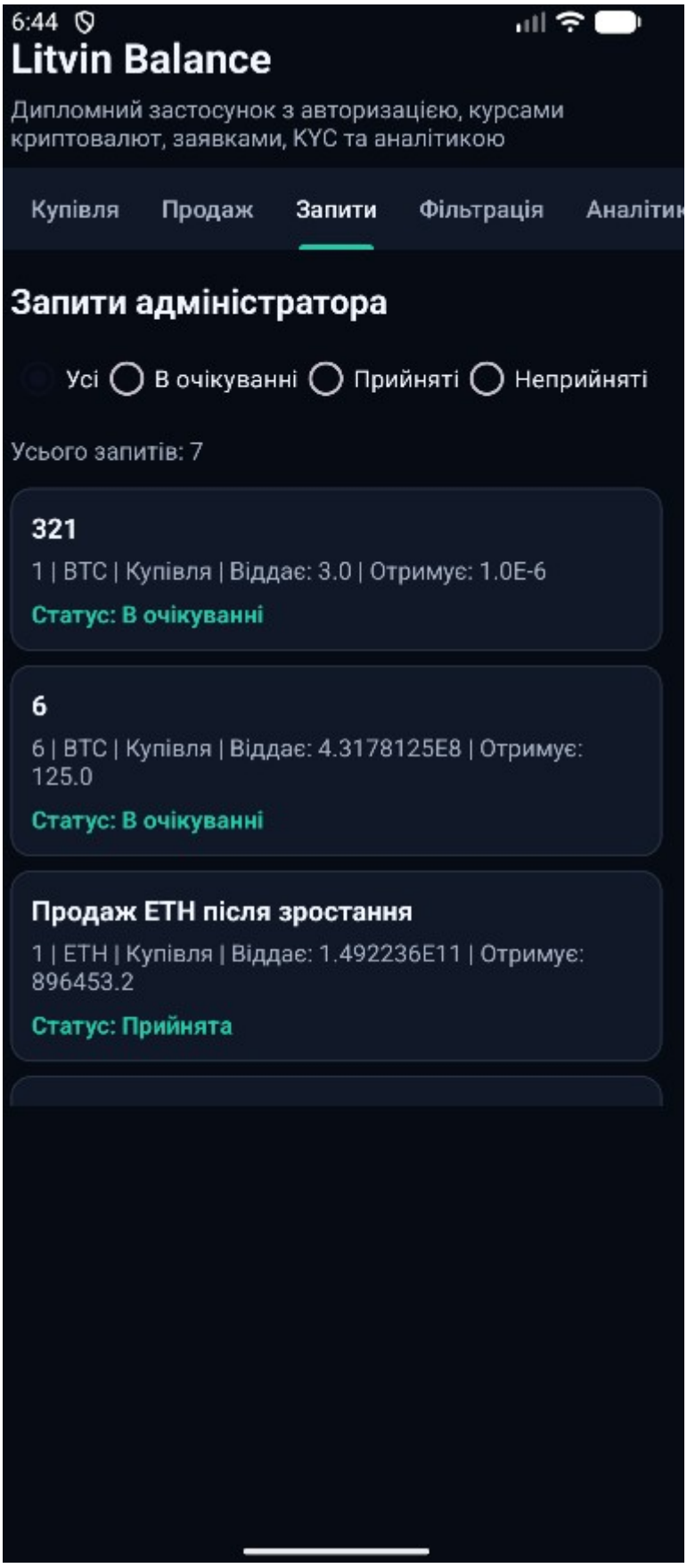
Головний екран вкладки «Курси» з картками BTC та ETH



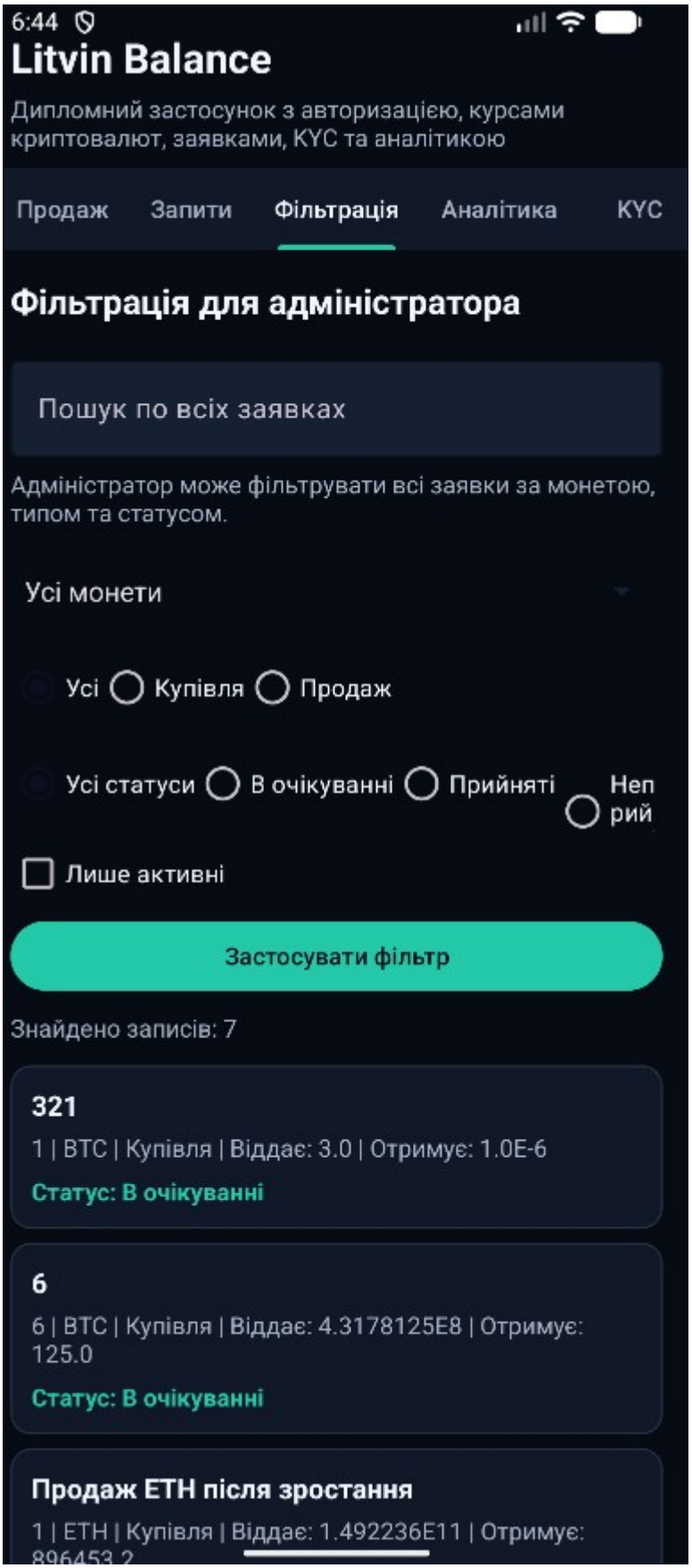
Форма створення заявки на купівлю криптовалюти



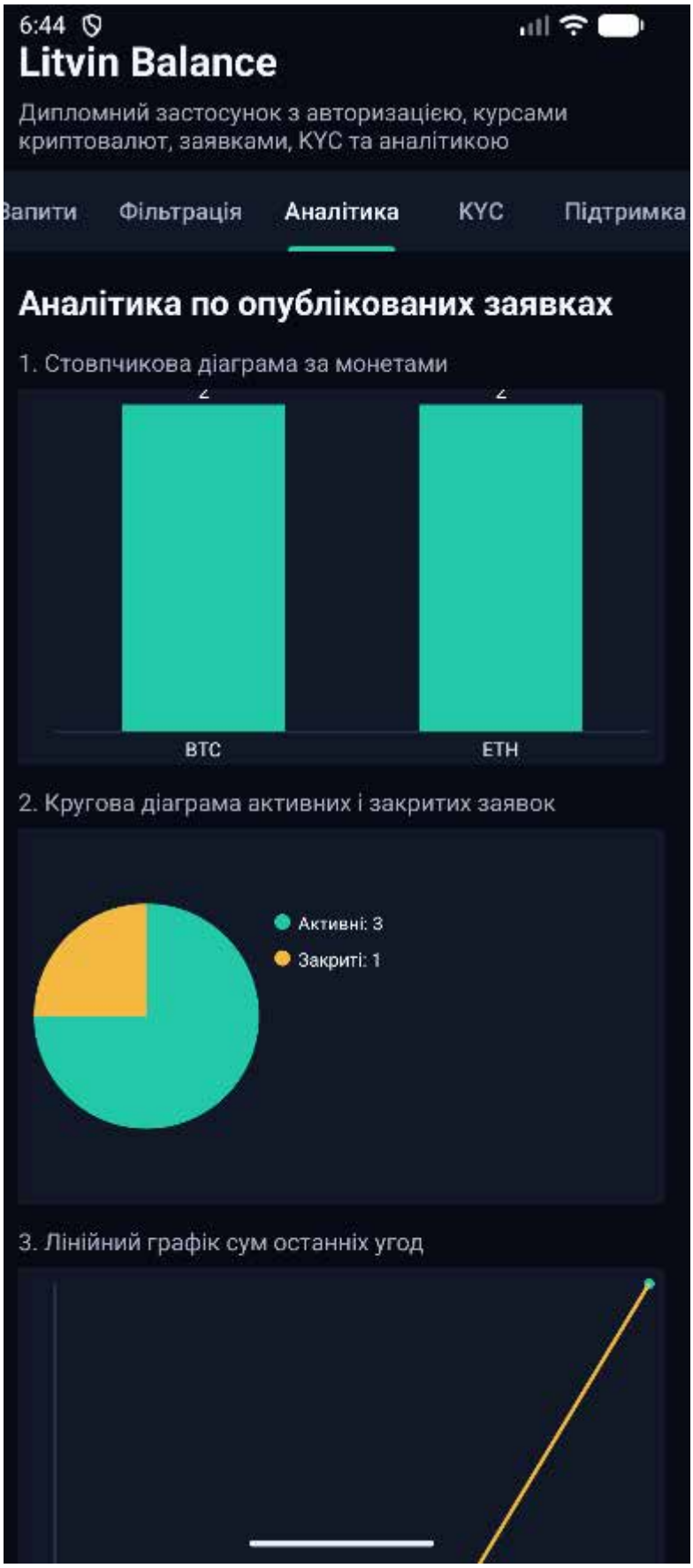
Форма створення заявки на продаж криптовалюти



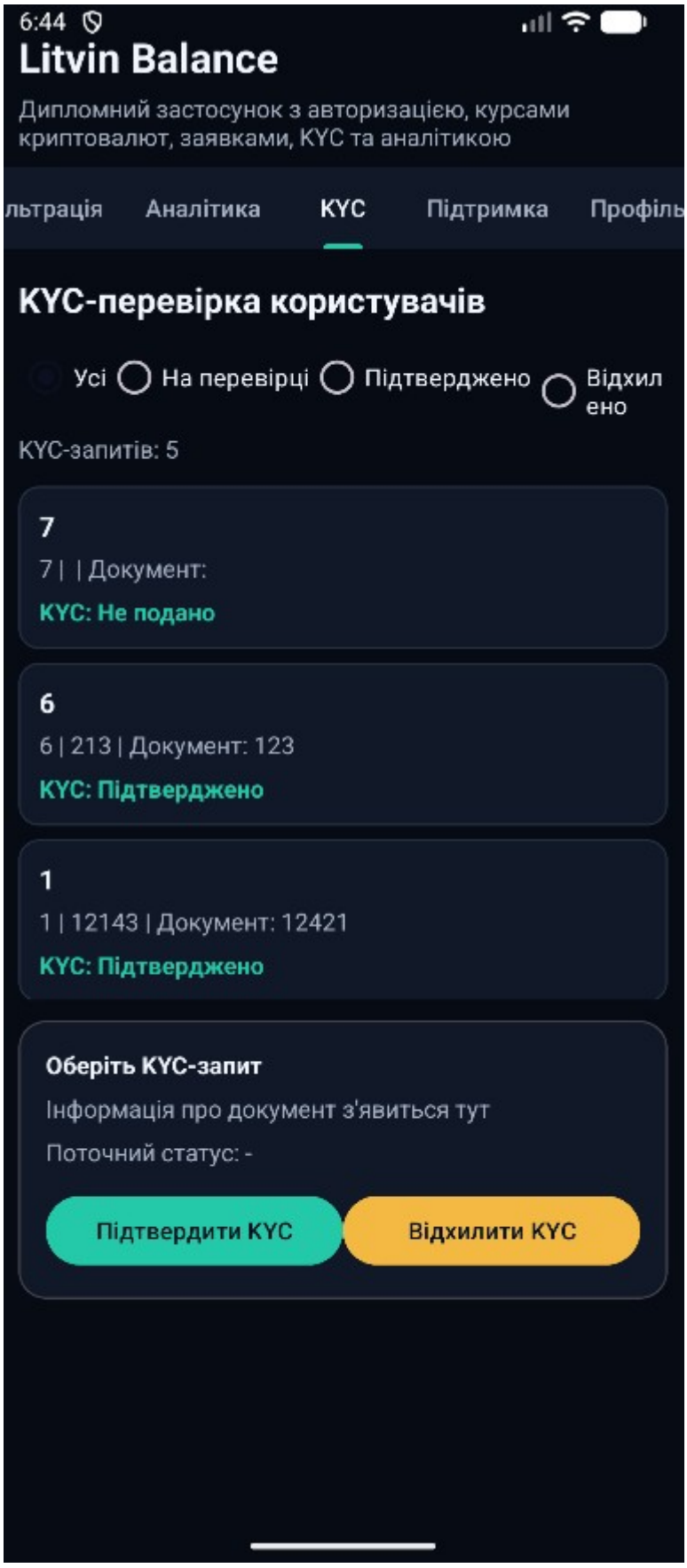
Екран адміністратора з підтвердженими та очікуваними запитами



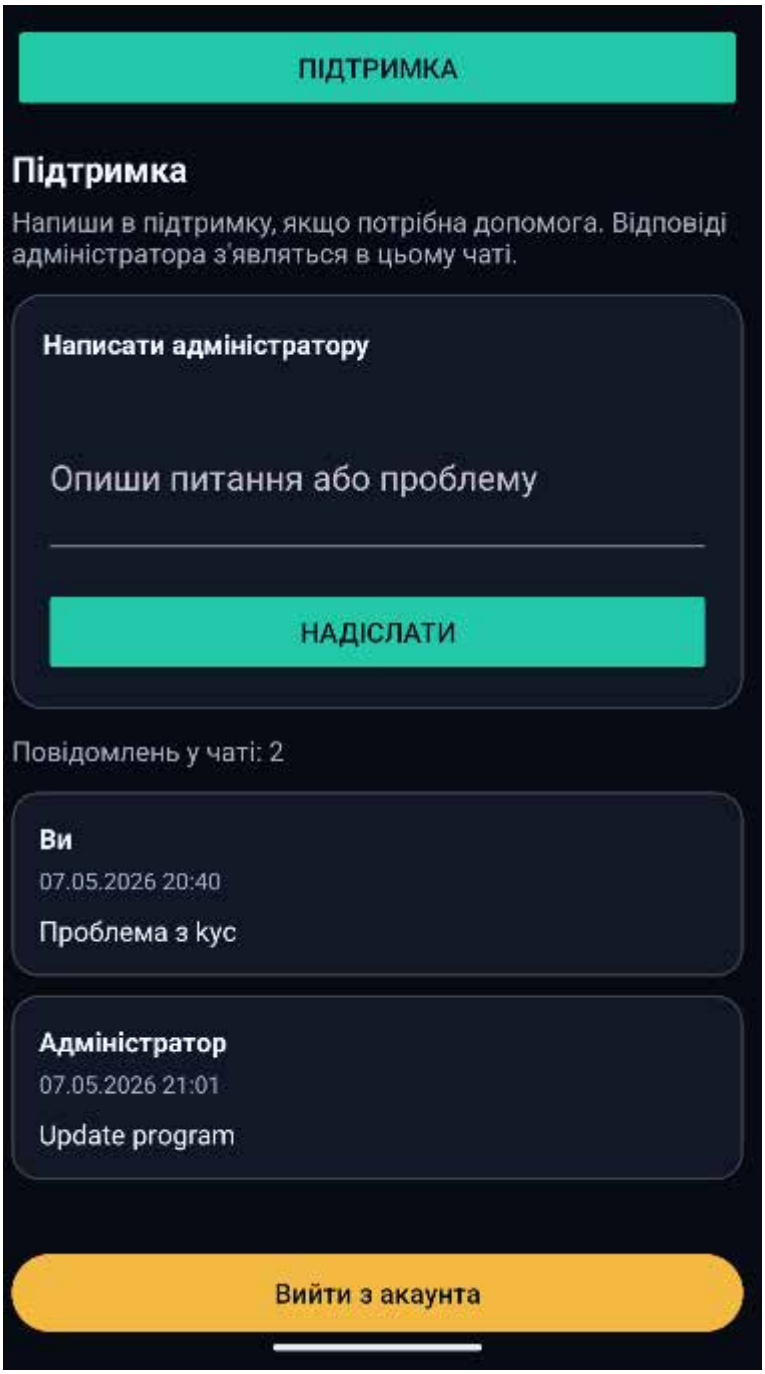
Екран фільтрації заявок за монетою, типом і статусом



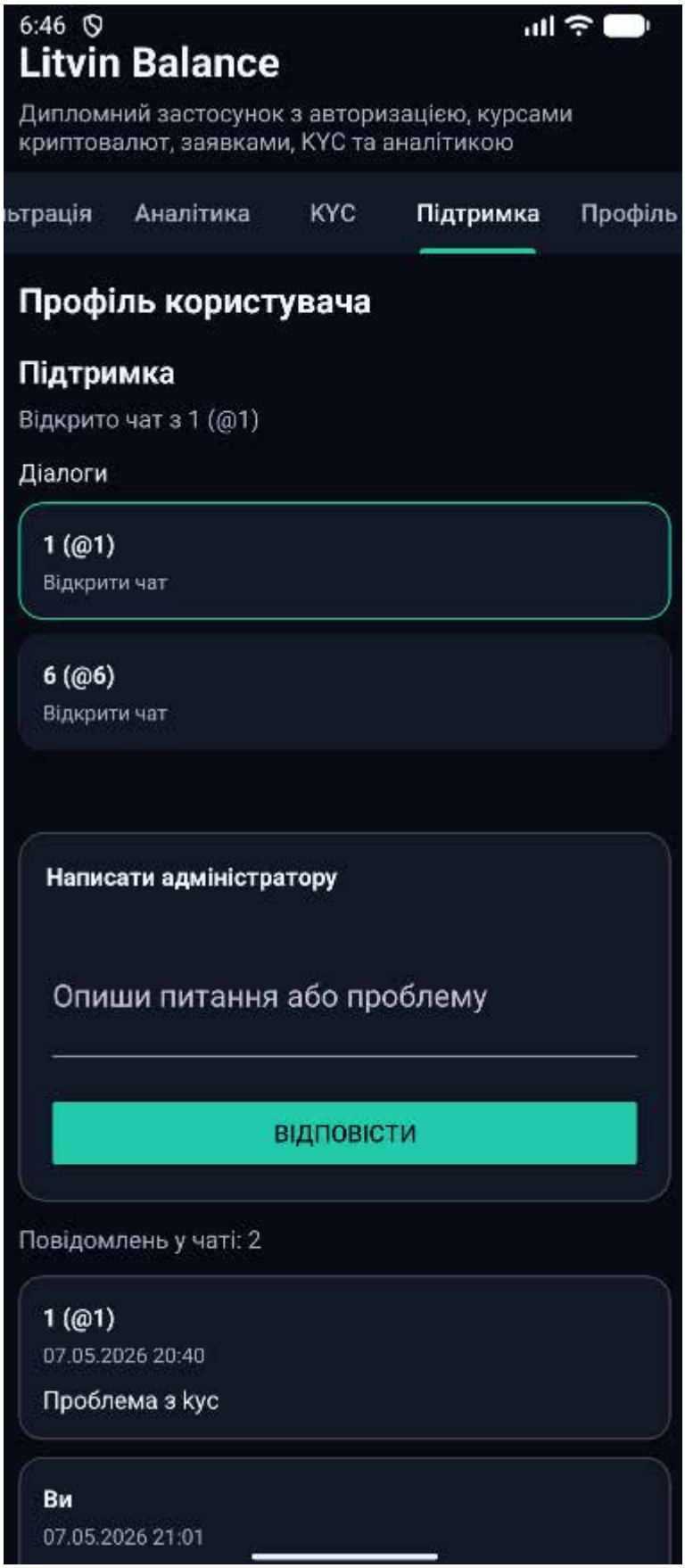
Екран аналітики зі стовпчиковою, круговою та лінійною діаграмами



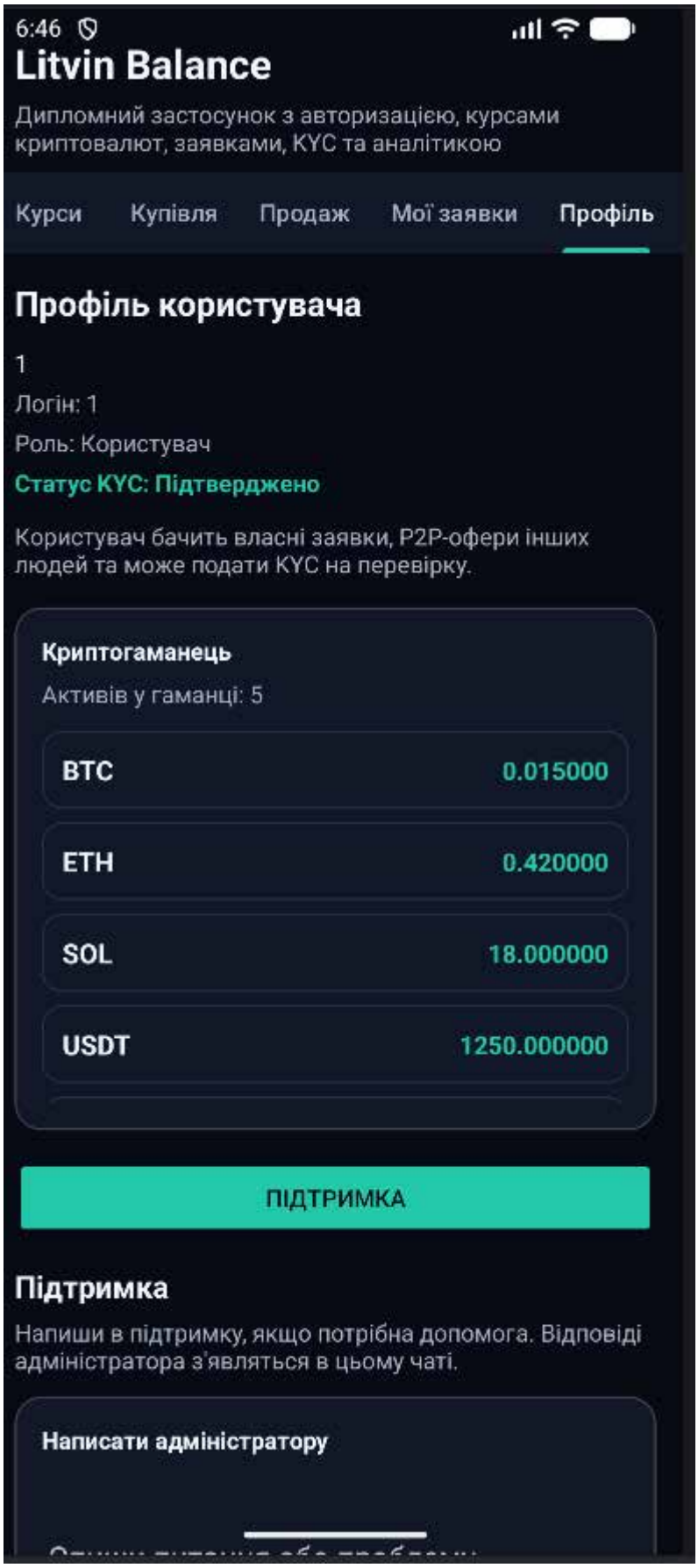
Екран KYC-перевірки користувачів



Екран підтримки у профілі користувача



Екран підтримки для адміністратора з відкритим чатом



Екран профілю з відображенням локального криптогаманця

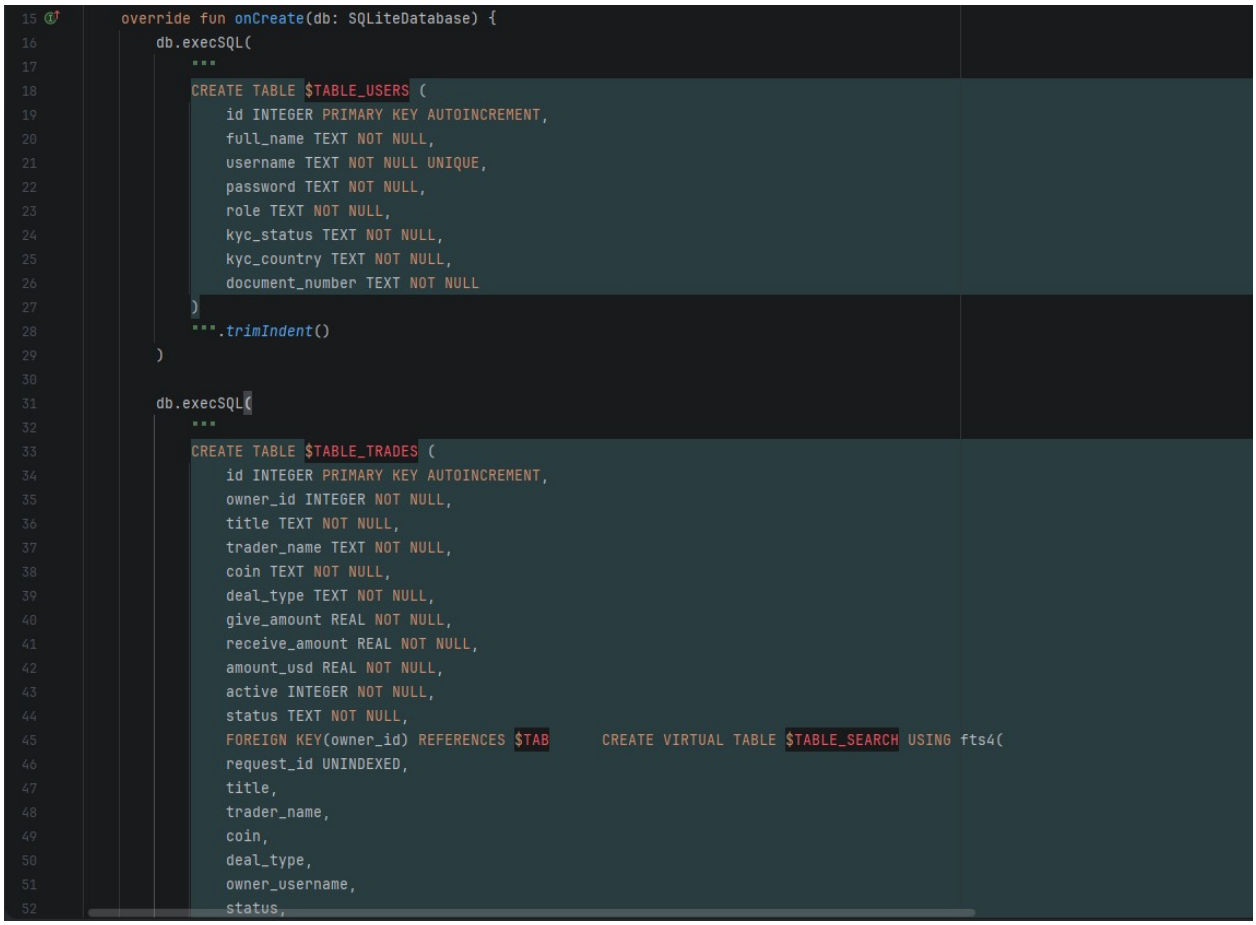
ДОДАТОК В

Лістинги основних програмних модулів

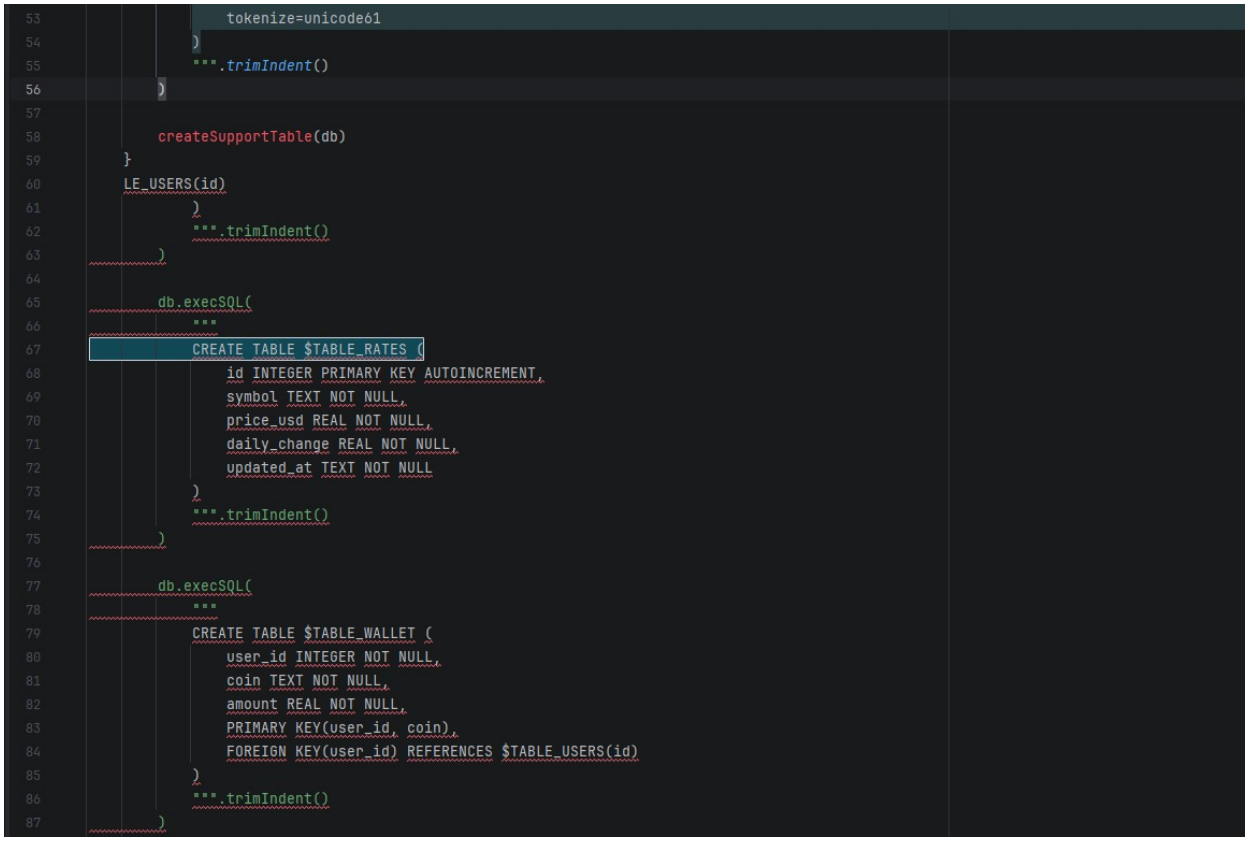
Сторінок – 7

Створення структури локальної бази даних

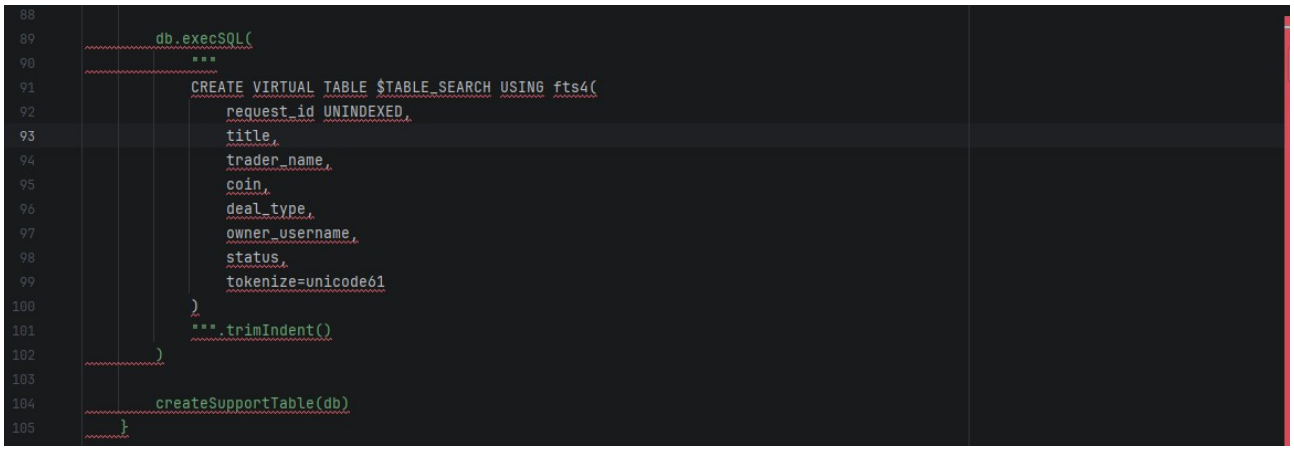
TradeRequestDatabase.kt



```
override fun onCreate(db: SQLiteDatabase) {
    db.execSQL(
        """
        CREATE TABLE $TABLE_USERS (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            full_name TEXT NOT NULL,
            username TEXT NOT NULL UNIQUE,
            password TEXT NOT NULL,
            role TEXT NOT NULL,
            kyc_status TEXT NOT NULL,
            kyc_country TEXT NOT NULL,
            document_number TEXT NOT NULL
        )
        """.trimIndent()
    )

    db.execSQL(
        """
        CREATE TABLE $TABLE_TRADES (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            owner_id INTEGER NOT NULL,
            title TEXT NOT NULL,
            trader_name TEXT NOT NULL,
            coin TEXT NOT NULL,
            deal_type TEXT NOT NULL,
            give_amount REAL NOT NULL,
            receive_amount REAL NOT NULL,
            amount_usd REAL NOT NULL,
            active INTEGER NOT NULL,
            status TEXT NOT NULL,
            FOREIGN KEY(owner_id) REFERENCES $TAB      CREATE VIRTUAL TABLE $TABLE_SEARCH USING fts4(
            request_id UNINDEXED,
            title,
            trader_name,
            coin,
            deal_type,
            owner_username,
            status,
```



```
            tokenize=unicode61
        )
        """.trimIndent()
    )

    createSupportTable(db)
}
LE_USERS(id)
        )
        """.trimIndent()
    )

    db.execSQL(
        """
        CREATE TABLE $TABLE_RATES (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            symbol TEXT NOT NULL,
            price_usd REAL NOT NULL,
            daily_change REAL NOT NULL,
            updated_at TEXT NOT NULL
        )
        """.trimIndent()
    )

    db.execSQL(
        """
        CREATE TABLE $TABLE_WALLET (
            user_id INTEGER NOT NULL,
            coin TEXT NOT NULL,
            amount REAL NOT NULL,
            PRIMARY KEY(user_id, coin),
            FOREIGN KEY(user_id) REFERENCES $TABLE_USERS(id)
        )
        """.trimIndent()
    )
```



```
88
89          db.execSQL(
90              """
91              CREATE VIRTUAL TABLE $TABLE_SEARCH USING fts4(
92                  request_id UNINDEXED,
93                  title,
94                  trader_name,
95                  coin,
96                  deal_type,
97                  owner_username,
98                  status,
99                  tokenize=unicode61
100             )
101             """.trimIndent()
102         )
103
104         createSupportTable(db)
105     }
```

Перевірка KYC та збереження заявки

MainActivity.kt

```
907     private fun saveOrUpdate() {
908         val user = currentUser ?: return
909
910         if (selectedId == null && !database.canCreateTrade(user)) {
912             showMessage("Спочатку потрібно пройти KYC-верифікацію, щоб створити заявку")
913             return
914         }
915
916         val title = binding.titleInput.text.toString().trim()
917         val trader = binding.traderInput.text.toString().trim()
918         val giveText = binding.giveAmountInput.text.toString().trim()
919         val receiveText = binding.receiveAmountInput.text.toString().trim()
920
921         if (title.isEmpty() || trader.isEmpty() || giveText.isEmpty() || receiveText.isEmpty()) {
922             showMessage("Заповни всі поля форми")
923             return
924         }
925
926         val giveAmount = giveText.toDoubleOrNull()
927         val receiveAmount = receiveText.toDoubleOrNull()
928         if (giveAmount == null || receiveAmount == null || giveAmount <= 0 || receiveAmount <= 0) {
929             showMessage("Суми повинні бути більшими за нуль")
930             return
931         }
932
933         val coin = binding.coinSpinner.selectedItem.toString()
934         val uahEquivalent = if (currentTradeMode == "Купівля") giveAmount else receiveAmount
935         val trade = TradeRequest(
936             id = selectedId ?: 0L,
937             ownerId = user.id,
938             ownerUsername = user.username,
939             title = title,
940             traderName = trader,
941             coin = coin,
942             dealType = currentTradeMode,
943             giveAmount = giveAmount,
944             receiveAmount = receiveAmount,
945             amountUsd = uahEquivalent,
946             active = binding.activeCheckBox.isChecked,
947             status = resolveStatusForSave()
948         )
949
950         if (selectedId == null) {
951             database.insertTrade(trade)
952             showMessage(if (isAdmin) "Запис додано" else "Заявку створено і відправлено на підтвердження")
953         } else {
954             database.updateTrade(trade)
955             showMessage(if (isAdmin) "Запис оновлено" else "Заявку оновлено і повернуто в очікування")
956         }
957
958         clearForm()
959         refreshRequestData()
960         loadMarketplaceOffers()
961         if (isAdmin) applyAdminFilters()
962         updateCharts( items = database.getTrades(isAdmin = true, statusFilter = TradeRequestDatabase.STATUS_ACCEPTED))
963     }
```

Подання та підтвердження KYC-запиту

MainActivity.kt

```
643     private fun submitKyc() {
644         val user = currentUser ?: return
645         if (isAdmin) return
646
647         val country = binding.kycCountryInput.text.toString().trim()
648         val documentNumber = binding.kycDocumentInput.text.toString().trim()
649         if (country.isEmpty() || documentNumber.isEmpty()) {
650             showMessage("Заповни країну та номер документа для KYC")
651             return
652         }
653
654         database.submitKyc(user.id, country, documentNumber)
655         refreshCurrentUser()
656         renderProfile()
657         loadKycRequests()
658         showMessage("KYC відправлено на перевірку")
659     }
660
        5 Usages
661     private fun loadKycRequests() {
662         if (!isAdmin) return
663         val status = when (binding.kycStatusRadioGroup.checkedRadioButtonId) {
664             binding.kycPendingRadio.id -> TradeRequestDatabase.KYC_PENDING
665             binding.kycApprovedRadio.id -> TradeRequestDatabase.KYC_APPROVED
666             binding.kycRejectedRadio.id -> TradeRequestDatabase.KYC_REJECTED
667             else -> TradeRequestDatabase.STATUS_ALL
668         }
669         val users = database.getUsersByKycStatus(status).filter { it.role != TradeRequestDatabase.ROLE_ADMIN }
670         kycAdapter.submitList( newItems = users)
671         binding.kycCountText.text = "KYC-запитів: ${users.size}"
672     }
673
        1 Usage
674     private fun selectKycUser(user: AppUser) {
675         selectedKycUserId = user.id
676         binding.selectedKycNameText.text = user.fullName
677         binding.selectedKycInfoText.text = "${user.username} | ${user.kycCountry} | Документ: ${user.documentNumber}"
678         binding.selectedKycStatusText.text = "Поточний статус: ${user.kycStatus}"
679         showMessage("KYC-користувача вибрано для перевірки")
680     }
```

```
681
        2 Usages
682     private fun updateSelectedKycStatus(status: String) {
683         val userId = selectedKycUserId ?: run {
684             showMessage("Спочатку вибери KYC-запит")
685             return
686         }
687         database.updateKycStatus(userId, status)
688         loadKycRequests()
689         binding.selectedKycStatusText.text = "Поточний статус: $status"
690         showMessage(if (status == TradeRequestDatabase.KYC_APPROVED) "KYC підтверджено" else "KYC відхилено")
691     }
```

Отримання курсів криптовалют через Binance API

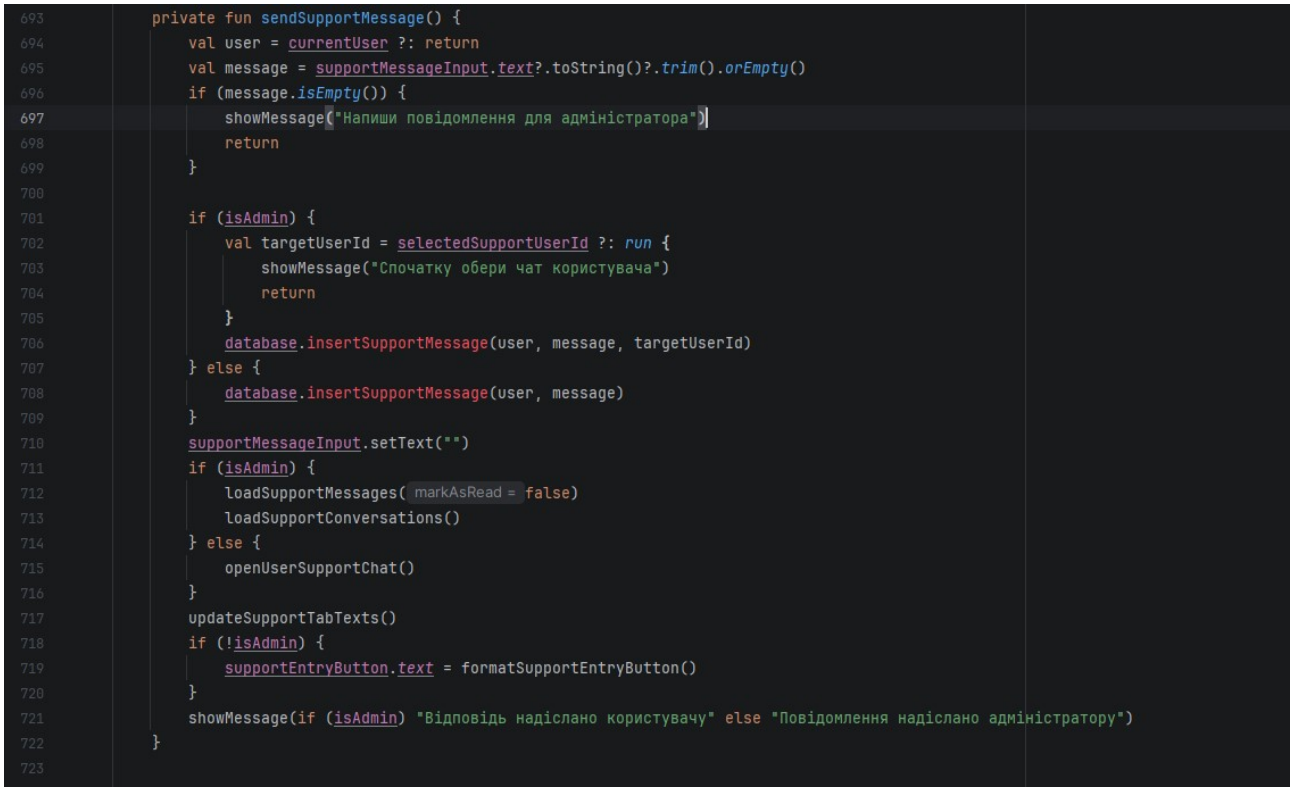
BinanceApiClient.kt

```kotlin
class BinanceApiClient {

    1 Usage
    fun fetchTickerSnapshots(symbols: List<String>): List<BinanceTickerSnapshot> {
        if (symbols.isEmpty()) return emptyList()

        val encodedSymbols = URLEncoder.encode(
            s = JSONArray( copyFrom = symbols).toString(),
            enc = Charsets.UTF_8.name()
        )

        val response = get("https://api.binance.com/api/v3/ticker/24hr?symbols=$encodedSymbols")
        val json = JSONArray(response)

        return buildList {
            for (index in 0 ≤ until < json.length()) {
                val item = json.getJSONObject(index)
                add(
                    BinanceTickerSnapshot(
                        symbol = item.getString( name = "symbol"),
                        lastPrice = item.getString( name = "lastPrice").toDouble(),
                        priceChangePercent = item.getString( name = "priceChangePercent").toDouble(),
                        quoteVolume = item.getString( name = "quoteVolume").toDouble(),
                        closeTime = item.getLong( name = "closeTime")
                    )
                )
            }
        }
    }
```

```kotlin
    1 Usage
    fun fetchKlinePoints(symbol: String, interval: String = "1h", limit: Int = 12): List<ChartPoint> {
        val response = get("https://api.binance.com/api/v3/klines?symbol=$symbol&interval=$interval&limit=$limit")
        val json = JSONArray(response)

        return buildList {
            for (index in 0 ≤ until < json.length()) {
                val item = json.getJSONArray(index)
                val closePrice = item.getString( index = 4).toFloat()
                add(ChartPoint( label = (index + 1).toString(), value = closePrice))
            }
        }
    }
```

Надсилання повідомлення в чат підтримки

MainActivity.kt



```
693     private fun sendSupportMessage() {
694         val user = currentUser ?: return
695         val message = supportMessageInput.text?.toString()?.trim().orEmpty()
696         if (message.isEmpty()) {
697             showMessage("Напиши повідомлення для адміністратора")
698             return
699         }
700
701         if (isAdmin) {
702             val targetUserId = selectedSupportUserId ?: run {
703                 showMessage("Спочатку обери чат користувача")
704                 return
705             }
706             database.insertSupportMessage(user, message, targetUserId)
707         } else {
708             database.insertSupportMessage(user, message)
709         }
710         supportMessageInput.setText("")
711         if (isAdmin) {
712             loadSupportMessages( markAsRead = false)
713             loadSupportConversations()
714         } else {
715             openUserSupportChat()
716         }
717         updateSupportTabTexts()
718         if (!isAdmin) {
719             supportEntryButton.text = formatSupportEntryButton()
720         }
721         showMessage(if (isAdmin) "Відповідь надіслано користувачу" else "Повідомлення надіслано адміністратору")
722     }
723
```

ДОДАТОК Д

Декларація про академічну доброчесність та використання ШІ

Сторінок – 1

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

Декларація про академічну доброчесність та використання ШІ

Я, Литвин Вадим Сергійович , здобувач НУБіП України,
усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота на тему
«Інформаційна система обліку купівлі-продажу криптовалюти» є результатом моєї особистої праці.

2. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.

3. Щодо використання інструментів ШІ зазначаю, що (обрати необхідне):

o [ ] інструменти генеративного ШІ не використовувалися.

o [ ] інструменти ШІ (ChatGPT, DeepL) були використані для:

▪ [ ] перекладу іншомовних джерел;

▪ [ ] стилістичного редагування та перевірки граматики;

▪ [ ] допомоги у написанні/відлагодженні програмного коду;

▪ [ ] інше: _____.

Я розумію, що надання недостовірної інформації може призвести до скасування результатів захисту та відрахування з числа здобувачів НУБіП України.

«___»_____ 2026 р. _____ **Литвин Вадим Сергійович**