Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Сергієнка Микити Ігоровича*
(ПІБ)

академічної групи *121-18-2*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

освітньої програми *Інженерія програмного забезпечення*
(назва освітньої програми)

на тему: *Створення фронтент частини професійного UI інтерфейсу криптовалютного додатку SPA на мові Javascript з використання фреймворку Nextjs*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*проф. Бердник М.Г*			
розділів:				
спеціальний	*проф. Бердник М.Г*			
економічний	*доц.Касьяненко Л.В.*			
Рецензент	*доц.Шедловський І.А.*			
Нормоконтролер	*доц. Гуліна І.Г*			

Дніпро
2022

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2022 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *121-18-2* *Сергієнко М.І.*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Створення фронтент частини професійного UI криптовалютного додатку SPA на мові Javascript з використання фреймворку Nextjs*

затверджена наказом ректора НТУ «ДП» від *18 травня 2022р,* №268-с

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2022 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2022 р.***

Завдання видав _____ *проф. Бердник М.Г*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Сергієнко М.І.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2022 р.*

Термін подання кваліфікаційної роботи до ЕК: *11.06.2022 р.*

РЕФЕРАТ

Пояснювальна записка: 73 с., 30 рис., 3 дод., 25 джерел.

Об'єкт розробки: веб-додаток з надання купівлі та продажу криптовалюти.

Мета кваліфікаційної роботи: створення веб-додатку з надання послуг купівлі та продажу криптовалюти через зручний інтерфейс.

У вступі розглядається стан проблеми на сьогодняшній день, встановлюється мета кваліфікаційної роботи, актуальність та галузь її застосування, уточнюються цілі завдання.

У першому розділі досліджується предметна галузь, визначається актуальність завдання та описується призначення розробки, розроблюється постановка самого завдання.

У другому розділі виконується проектування і розробка самого веб-додатку, описується алгоритм і структура функціонування, обирається платформа для розробки і описується робота веб-додатка.

В економічному розділі описується трудомісткість самого розробленого програмного продукту, виконується підрахунок вартості роботи по створенню застосунку та проводиться розрахунок часу на його створення.

Практичне значення полягає у створенні веб-додатку та у покращенні практичних знань та отриманні навичок за час створення цієї роботи.

Актуальність програмного продукту визначається великою швидкістю розвитку сфери продаж та надання послуг, а також зробити ці процеси більш зручними та доступними для користувача.

ABSTRACT

Explanatory note: 73 p., 30 fig., 3 add., 25 sources.

Object of development: web application for the purchase and sale of cryptocurrency.

The purpose of the qualification work: to create a web application for the provision of services for the purchase and sale of cryptocurrency through a convenient interface.

The first section explores the subject area, determines the relevance of the task and describes the purpose of development, develops the formulation of the task itself.

The first section explores the subject area, determines the relevance of the task and describes the purpose of the development, develops the formulation of the task itself.

The second section designs and develops the web application itself, describes the algorithm and structure of functioning, selects a platform for development and describes the work of the web application

The economic section describes the complexity of the most developed software product, calculates the cost of work on creating an application and calculates the time for its creation.

Practical importance lies in creating a web application and improving practical knowledge and gaining skills during the creation of this work.

The relevance of the software product is determined by the high speed of development of the sales and service provision sector, as well as to make these processes more convenient and accessible to the user.

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

SEO – пошукова оптимізація сайту;

JSON – javascript object notation, текстовий формат обміну данних;

CSS – cascading style sheets;

ПЗ – програмне забезпечення;

IDE – integrated development environment, інтегроване середовище розробки;

SPA – single page application, односторінковий додаток;

HTML – hypertext markup language, мова розмітки гіпертексту.

ВСТУП

Тема даної кваліфікаційної роботи є створення веб-додатку для надання послуг купівлі та продажу криптовалюти, а також зробити інтерфейс програми зручний для користувача та зрозумілий для відстеження своїх фінансових коштів.

Мета даної кваліфікаційної роботи є вивчення предметної галузі розробки веб-додатків, покращення та набуття практичних навичок, робота з фреймворком Nextjs та взаємодія клієнтською частиною та BACKEND.

Наш світ не стоїть на місці, кожен день людство розвивається в різних напрямках чи то політ у космос чи комп'ютерні технології та фінансові системи теж не стоять на місці. Одним з найважливішим ресурсом нашого часу є кошти. За кошти ми купуємо продовольчі продукти, нерухомість або більш значущі речі, але все це ми робимо через третю особу, яка є будь-яким банком, неможливо здійснити онлайн покупку без участі банку на сьогоднішній день, але як говорилося раніше, що фінансові можливості не стоять на місці, і ми на сьогоднішній день маємо таку можливість.

Криптовалюта є сьогодні єдиним способом здійснювати покупки без участі транзакцій банківської системи, оплата проводиться безпосередньо між покупцем і продавцем через криптографічний гаманець. Вся історія транзакцій зберігається в так званих блокчейнах, яка оновлюється завдяки взаємодії з криптовалютою. Також криптовалюта використовує шифрування своїх даних, за допомогою чого вона стає практично не вразливою для зловмисників.

Криптовалюта сьогодні є однією з безпечних валют у всьому світі, вона кожен день розвивається та дає нові можливості для переводу коштів та багато інших операцій. Таким чином ця тема є дуже важлива на сьогоднішній день тому, що криптовалюта - це сучасна валюта.

Актуальність веб-додатку з надання послуг купівлі та продажу криптовалюти через зручний інтерфейс визначається великою швидкістю розвитку сфери продаж та надання послуг, а також зробити ці процеси більш зручними та доступними для користувача.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Загальні відомості з предметної галузі

Що таке криптовалюта? Криптовалюта – це цифрова платіжна система у якій не беруть участь банки. Це система з рівноправними учасниками, яка допомагає здійснювати транзакції самостійно без банківської системи – банк не бере участь у транзакціях коштів. Криптовалютні платежі існує тільки в цифровому світі інтернету, а також існує база даних транзакцій усіх користувачів. Транзакції не працюють з фізичними грошима, тобто немає входження і вихід фізичних коштів. При переказі грошей у криптовалюті, транзакції записуються до реєстру. Криптовалюта зберігається у спеціальному цифровому гаманці.

Термін криптовалюта з'явився завдяки своїм особливостям, що для перевірки транзакцій використовується шифрування для передачі засобів та зберігання, а також занесення транзакцій до загального доступного реєстру. Шифрування забезпечує безпеку та надійність коштів.

Перша криптовалюта з'явилася в 2009 році і стала найпопулярнішою на сьогоднішній день Bitcoin. Криптовалюта стала одним із найпопулярніших для заробітку в інтернеті завдяки трейдингу через те, що криптовалюта є не стабільною, її ціна може варіюватися від 0.01 цента до 60 000 тисяч доларів. На даний момент існує дуже багато різних криптовалют, так звані монети: Bitcoin, Ethereum, Cortex, Ethereum Classic, Bitcoin Gold, BitcoinZ та багато інших.

Як використовується криптовалюта?

Криптовалюта починається свій шлях з публічного реєстру – блокчейна, де зберігаються всі записи скоєних транзакцій, що оновлюється користувачами які використовують криптовалюту. Криптовалюта створюється внаслідок так званої дії «Майнінг». Це процес, коли великі

комп'ютерні потужності застосовуються на вирішення складних алгоритмічних завдань після чого вони отримують винагороду монеток. Також криптовалютні монети можна купувати на біржах та зберігати їх у криптографічних гаманцях.

Криптовалюта – немає фізичної форми чи явної, це цифровий ключ, який дає можливість передавати дані з одного боку в іншій та не використовує банківську систему. Перша криптовалюта з'явилася в 2009 році криптовалюта зараз перебуває на стадії активного розвитку. У деяких країнах вже визнали можливість проводити реальні фінансові операції за допомогою криптовалюти.

Що можна придбати за криптовалюту?

Криптовалюта була спочатку вигадана для щоденних транзакцій, даючи можливість купувати все, що захочеться від покупки продуктів до покупки нерухомості. Незважаючи на те, що криптовалюта зараз дуже активно розвивається, дуже великі грошові згоди проходять рідко через те, що наш світ поки що не має повного розуміння, як це влаштовано, але на даний момент можна спокійно робити звичайні покупки в інтернеті.

1.2. Призначення розробки та галузь застосування

Створення веб-додатку для купівлі та продажу нової криптовалюти, яка називається SWH. Мета створити веб-додаток, в якому користувач зможе фінансувати свої гроші в нову криптовалюту та проводити грошові транзакції, а також відстежувати свій заробіток та витрати. Додаток в якому користувач зможе створювати та відкривати свої депозити та ордери, а також взаємодії з фінансовими біржами через зручний UI – інтерфейс. У додатку можна буде купити USDT через спеціальний інтерфейс, після чого користувач зможе придбати криптовалюту SWH, також можливість придбати Bitcoin за закриття своїх ордерів або депозитів через деякий час. На сайті буде доступно три валюти, з якими можна буде взаємодіяти для збільшення свого доху. Додаток

має бути дуже зручний та зрозумілий інтерфейс для користувача, а також можливість керувати своїми особистими даними та фінансами.

1.3. Підстави для розробки

Відповідно до освітньої програми, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу.

Тема роботи узгоджується з керівником кваліфікаційної роботи, випускаючою кафедрою, та затверджується наказом ректора.

Отже, підставами для розробки (виконання кваліфікаційної роботи) є:

- освітня програма 121 Інженерія програмного забезпечення;
- навчальний план та графік навчального процесу;
- наказ ректора Національного технічного університету «Дніпровська політехніка» № 268-с від 18.05.2022 р;
- завдання на кваліфікаційну роботу на тему “Створення фронтент частини професійного UI інтерфейсу криптовалютного додатку SPA на мові Javascript з використання фреймворку Nextjs”.

1.4. Постановка завдання

Завданням даної роботи є створення клієнтської частини веб-додатку для продажу та купівлі криптовалюти з використанням фреймворку Next.js.

В результаті необхідно спроектувати та розробити клієнтську частину веб-додатку для спрощення транзакцій та виявлення свого доходу.

Продукт має два рівні:

- рівень доступу «Гість»;
- рівень доступу «Клієнт».

Для створення веб-додатку застосовується мова Javascript, так як вона створена для веб-інтерфейсів.

Для створення додатку застосовується фреймворк Nextjs [1] який побудований на Reactjs [2], та має свій власній маршрутизатор, якій має можливість створювати сторінки на сервері або на клієтной частині.

Завдяки Nextjs [1] код писати набагато легше та швидкість роботи сайту збільшується в декілька разів, тому що він будує SPA [3] сторінку, яка використовую лише один HTML [4] документ для входу в веб-додаток, також в Next.js [1] можна використовувати компонентний підхід, що допомагає підтримувати коднабагато легше.

Так як у нашому випадку для передачі даних з серверу клієнту буде використано REST API [5] – підхід до архітектури мережевих протоколів, які надають доступ до інформаційних ресурсів з BACKEND частини.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Кінцевий продукт матиме два рівні доступу до функціоналу:

- рівень доступу «Гість»;
- рівень доступу «Клієнт».

На рівні доступу Гість повинен містити наступні функції:

- реєстрація та логін;
- перегляд головної сторінки.

На рівні доступу Клієнт повинен містити наступні функції:

- перегляд свого балансу;
- створення ордеру;
- історія поповнень та виведення грошей;
- статистика балансу;
- настройка власного профілю;
- можливість зв’язатися с технічною підтримкою;
- отримання повідомлень;
- виведення своїх грошей;

− вихід з додатку.

Веб-додаток повинен мати зручний інтерфейс для зручності клієнтів та гарну можливість отримання інформації о грошевих новинах, історію, повідомлення.

1.5.2. Вимоги до інформаційної безпеки

Основні вимоги до інформаційної безпеки:

− конфіденційність інформації;

− цілісність даних;

− доступність інформації;

− вірогідність даних;

− автентичність даних;

− впровадження політики прав доступу.

Під час роботи додатку є два режими:

− авторизований користувач;

− неавторизований користувач.

Неавторизований користувач має змогу потрапити на головну сторінку та ознайомитись з інформацією сайту та дізнатися, що таке криптовалюта.

Авторизований користувач має змогу поповнити свій баланс та купити нову криптовалюту та створити депозити або орден.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для роботи з веб-додатком використовується браузер. Відповідні характеристики необхідні для роботи більшості браузерів для Windows:

− ЦП: Intel, AMD 4-го покоління або вище;

− відеоадаптер: 3D адаптер nVidia, Intel, AMD/ATI;

− відеопам’ять: 128 МБ;

− накопичувач: 512 Гб;

− оперативна пам’ять: 4 Гб.

У випадку з мобільними платформами необхідні мінімальні версії операційних систем:

− iOS 11.0 і вище;

− Android 8 і вище.

1.5.4. Вимоги до інформаційної та програмної сумісності

Веб-додаток – клієнт-серверний додаток, в якому клієнт взаємодіє з веб-сервером за допомогою браузера. Це дозволяє вирішити багато проблем, так як немає необхідності хвилюватись про окрему платформу, так як браузер має можливість працювати на персональних комп’ютерах, ноутбуках, смартфонах.

Сучасні браузери мають функцію автоматичного оновлення, тому проблеми у користувачів не повинні виникати при роботі в популярних браузерах.

Веб-додаток має підтримуватися такими браузерами:

− Chrome від 87 версії;

− Opera від 74 версії;

− Edge від 88 версії;

− Safari від 13.1 версії;

− IE від 11 версії;

− Android від 90 версії;

− Baidu від 7.12 версії;

− Firefox від 78 версії;

− iOS Safari від 13.4версії;

− Opera mobile від 62 версії;

− Samsung від 13.0 версії.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

В результаті цієї кваліфікаційної роботи повинен бути створений веб-додаток для купівлі-продажу криптовалюти, а також кросбраузерний для основних браузерних систем.

Цільовий застосунок повинен мати таку функціональність:

- можливість реєстрації в системі;
- можливість поповнити свій гаманець, який створюється при реєстрації в системі;
- можливість створення замовлення на покупку;
- можливість створення депозиту;
- можливість відстежувати свої доходи і витрати;
- можливість перегляду поточного балансу;
- можливість звернутися в технічну підтримку;
- отримувати сповіщення, коли відбувається подія;
- перевірити історію поповнень;
- перевірити історію своїх виведення коштів;
- перевірити список закритих замовлень;
- перевірте витрати;
- можливість редагування персональних даних;
- вийти зі свого облікового запису.

Веб-додаток повинен мати зручний інтерфейс. Чітко відображати фінансові ресурси. Додаток повинен максимально полегшити взаємодію з біржами для зручності користувача. Відображення всієї інформації, необхідної для контролю коштів.

2.2. Опис застосованих математичних методів

Під час кваліфікаційної роботи не використовувалися математичні алгоритми на стороні клієнта. Всі математичні алгоритми були застосовані на стороні сервера через API [5]. Математичні методи не використовуються на стороні клієнта.

2.3. Опис використаної архітектури та шаблонів проектування

Щоб створити цей додаток, було вибрано модульну архітектуру додатку (рис. 2.1).

Модуль - це функціональна частина програми, яка відповідає за певну функціональність.

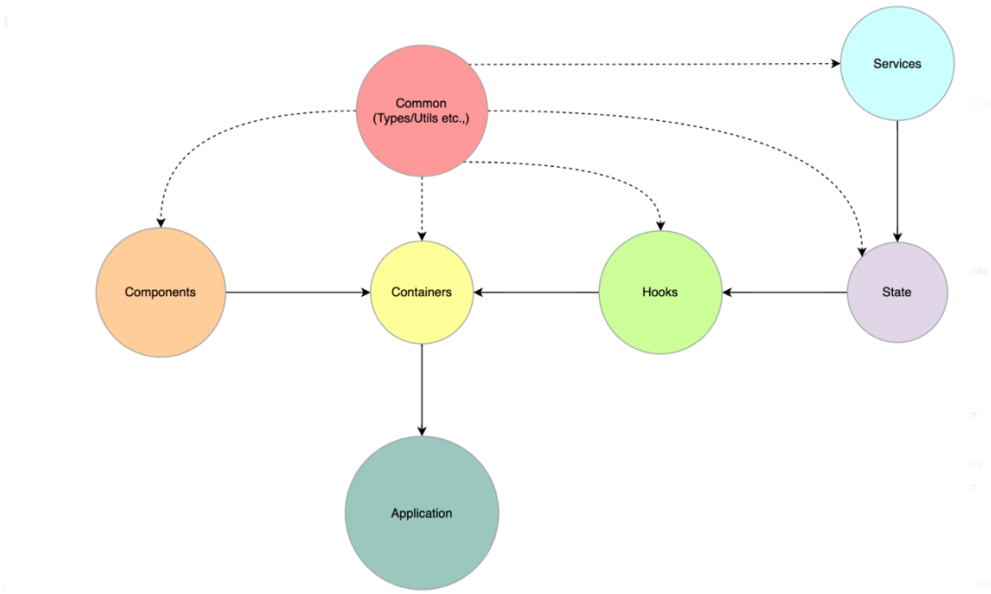

Рис. 2.1. Графічне представлення шаблону modules

Модульна архітектура відповідає за наступні вимоги:

- весь код модуля повинен знаходитися в одній папці. Модуль є незалежною частиною веб-додатока, яка не залежить від іншого

модуля, і не повинно бути архітектурної залежності від модуля при видаленні модуля;

- модулі не залежать один від одного. Зміни в модулі не повинні впливати на інші модулі. Модуль може залежати лише від основних компонентів або ядра;
- ядро системи містить загальнодоступний API [5], який надає модулі з інструментами вводу/виводу та набір компонентів для створення інтерфейсу користувача.

Програмний/корневий модуль – це просто колекція/композиція контейнерів (він же розумні компоненти). Він відповідає за завантаження програми за допомогою react-suspense , межі помилок і будь-яких постачальників верхнього рівня . Загалом, він повинен містити якомога мінімальний код.

Модуль контейнерів — це набір розумних компонентів, які підключаються до стану через гачки. Модулі контейнерів імпортують лише загальні модулі, компоненти та модулі гачків. Це допомагає нам масштабувати модуль хуків, не впливаючи на стани.

Модуль hooks [6] постачає дані в контейнери через стан, а також оновлює стан на основі взаємодії користувача з компонентами контейнера.

Модуль стану відповідає за заповнення стану на стороні клієнта за допомогою служби та загальних модулів. Це дозволяє нам уникнути забруднення контейнерів і гачків за допомогою API [5] сервісів.

Незалежні модулі підключаються лише до звичайних модулів , хоча це не так незалежно, як нам хотілося б, все ж простіше передати його зовнішній бібліотеці.

Модуль компонентів — це просто набір суто функціональних компонентів, які не піклуються про стан програми.

Хоча це може бути абсолютно незалежним, як зовнішня бібліотека, ви все одно можете створити спеціальні компоненти програми , які дуже нагадують типи даних із загального модуля. Тому часто корисно

виділити/позначити загальні та специфічні компоненти.

У загальних компонентах не використовуються типи із загальних модулів, що дозволяє нам перемістити їх до зовнішньої бібліотеки компонентів.

Модулі служб утворюють набір API [5], які взаємодіють із серверною частиною, і отримана відповідь має відповідати основним типам даних, присутнім у наших загальних модулях.

2.4. Опис використаних технологій та мов програмування

Кваліфікаційна робота була створення з використанням наступних технологій:

- react.js;
- next.js;
- typescript;
- javascript;
- axios;
- styled components;
- redux toolkit.

React - це бібліотека Javascript [7, 8, 9], створена Facebook для розробки інтерфейсів користувача. Бібліотека React стала однією з найпопулярніших бібліотек для створення інтерфейсу користувача , через що багато розробників прийшли до думки, що React [2] - це фреймворк, але це так. Facebook вдалося створити неймовірно швидкий інструмент для розробки інтерфейсів, а також оптимізував роботу з браузером, завдяки цьому додатки, які написані на React, стали працювати швидше.

Virtual DOM - це точна копія справжнього DOM [9] дерева в браузері, яка зберігається в пам'яті і синхронізується з реальним DOM [9], не посередні зміни відбуваються у Virtual DOM [10], після чого він порівнюються з реальним DOM [9], коли порівняння пройшли, зміни відбуваються в реальному DOM [9]. Таким чином, браузеру не потрібно повторно

відтворювати всю сторінку, йому потрібно лише змінити місця, де це потрібно.

React отримав свою популярність не тільки завдяки Virtual DOM [10], але і завдяки синтаксису JSX [10]. Синтаксис JSX - це особливий синтаксис, який дозволяє писати звичайні HTML-теги [3] безпосередньо у Javascript [7, 8, 9], після чого він скомпільовується за допомогою Bable в звичайний Javascript [7, 8, 9] код, тим самим роблячи розробку більш зручною.

Бібліотека React [10] ідеально підходить для створення SPA [3] - "односторінкового додатка". SPA [3] - це веб-додаток або веб-сайт, який використовує лише одну сторінку HTML-документа [3] як обгортку для відображення інтерфейсу користувача, який динамічно завантажується через Javascript [7, 8, 9] безпосередньо через AJAX [11, 12, 13].

Next.js — це фреймворк, заснований на бібліотеці React.js, має свою спеціальну маршрутизацію, а також можливість генерувати статичні і динамічні сторінки на стороні сервера, а також давати генерацію сторінок на стороні клієнта. Next.js [1] вирішує одну з найбільших проблем бібліотеки React [2], а саме SEO. React.js бібліотека створює SPA-додаток [3] на стороні користувача, він отримує звичайний HTML-документ [4] з порожнім вмістом, а вміст відображається посереднім на пристрої клієнта, через цю функцію SPA-додатків Google і Yandex пошукові системи не можуть сканувати вміст сайту, коли пошуковий робот заходить на сайт, він бачить порожній HTML Документ [4] – Next.js [1] вирішує цю проблему за допомогою рендерінгу сайту на сервера.

Javascript - це мова програмування, яка була створена для інтерактивних елементів у браузері.

Javascript був задуманий як звичайна мова програмування для з’єднання HTML [3] і CSS [14], що допоможе додати на сайт більше інтерактивних елементів та можливостей взаємодії з користувачем.

Оскільки раніше сайти були досить простими його функціоналу вистачало для типових завдань, але пройшло багато часу і за короткий час сайти почали рости і набувати масштабного функціоналу і потім з'ясувалося,

що JavaScript не зручний для великих додатків.

Вся справа в тому, що, JavaScript - це мова, яка була створена для невеликих завдань в браузері і не мала великого функціоналу, а також не була типізовою мовою і допомагала швидко писати невеликі документи JavaScript [7, 8, 9], які додавали інтерактиву на наш сайт. Але бізнес ставить нові цілі для розробників. Кожен раз доводилося переписувати документи JavaScript для вирішення цих самих проблем, і кожен раз, коли документи переписувалися і код ставав все більшим і менш зрозумілим, через те, що JavaScript [7, 8, 9] не призначався, для таких великих проектів виникали проблеми з підтримкою, після чого розробники JavaScript [7, 8, 9] вирішили його вдосконалити.

Розробники почали впроваджувати нові можливості в JavaScript і все повинно було бути добре, але були проблеми з підтримкою браузерів. Так як кожен браузер мав певне ядро, а також свої особливості, стало зрозуміло, що нові можливості мови просто не будуть підтримуватися і тоді розробники прийшли до певних стандартів, які отримали назву ECMAScript [9]. Це рішення допомогло прийти до загального стандарту і налагодити роботу з браузерами, але це був тільки перший крок.

Через деякий час виникла нова проблема, справа в тому, що JavaScript розроблявся дуже швидко і розробники браузерів просто не встигали доповнювати нові можливості мови.

Розробники JavaScript [7, 8, 9] вирішили, що якщо браузери не встигають за розвитком мови, їм потрібен якийсь проміжний варіант, який дозволив би розробникам писати зручний код за допомогою сучасного синтаксису мови і при цьому не морочитися з сумісністю з браузерами. Вони вирішили придумати нову технологію, яка перекладатиме один сучасний стандарт мови в інший, який зможуть розуміти браузери і назвали його Bable.

Typescript - це надмножина JavaScript [7, 8, 9], яка розширює його можливості. Він був розроблений Корпорацією Майкрософт для покриття недоліків мови JavaScript [7, 8, 9], а саме шляхом автододавання тексту, типів,

інтерфейсів, нові можливості для класів, узагальнених типів тощо. Коли Typescript [15] прийшов на допомогу розробнику, стало набагато простіше розробляти великий інтерфейс користувача. Виявлення помилок на етапі написання коду, а також чудове автозаповнення, яке дозволяє іншому розробнику швидко зрозуміти код.

Typescript [15] не підтримується браузерами, тому при необхідності він компілюється в звичайний JavaScript код.

Сьогодні Typescript [15] є стандартом розробки веб-додатків, операційної підтримки або інших завдань. Typescript [15] допомагає поліпшити структуру коду і полегшує його обслуговування.

Axios - це звичайна бібліотека AJAX [11, 12], яка допомагає відправляти запити на сервер. За допомогою Axios ви можете гнучко налаштовувати запити на отримання на сервер і не турбуватися про базову конфігурацію запитів, а також Axios [11] може використовуватися на стороні сервера. Axios [13] підтримує Promise [7, 8, 9], завдяки цьому ви можете використовувати сучасний синтаксис async/await.

Redux Toolkit - це глобальний менеджер, який допомагає вам контролювати стан вашого додатка, а також передавати дані в улюблену точку нашого додатка. Redux [16, 17] використовує архітектуру FLUX [16, 17] , яка базується на дії, стані, перегляді.

Дії – це подія, яка сповіщає Store про те, що вам потрібно змінити, видалити, оновити або маніпулювати даними в нашому глобальному Store [16, 17].

Store - це глобальне сховище, де зберігаються всі дані програми і до них можна отримати доступ з будь-якого місця нашого коду.

View - це готовий інтерфейс користувача, який чекає і отримує дані для відображення на сторінці нашого додатка.

Redux Toolkit - це передова бібліотека над Redux [16, 17], яка допомогла вирішити проблему з великою кількістю кодування.

Styled-components - це сучасна бібліотека, створена для React [2] і має

стилізований компонентний підхід, який допомагає стилізувати залежності компонентів від props, які потрапляють в компонент. Стилізовані компоненти вирішують проблему з вендарнимі префіксами, темами, прохідними стилями і зручністю написання коду стилю [18].

2.5. Опис структури програми та алгоритмів її функціонування

Діаграма прецедентів — відображає випадки використання між програмою та акторами, які входять до складу програми (рис. 2.2).

Актори з'являються як людина на діаграмі і описують якусь роль, яку він повинен виконати.

У нашому випадку акторами можуть бути вже зареєстровані користувачі в системі або тільки нові користувачі.

Незареєстрований користувач має доступ тільки до головної сторінки сайту і можливість потрапити на вступні сторінки.

Щоб отримати повний функціонал програми, потрібно зареєструватися в додатку через свою пошту, а також заповнити форму з необхідними даними.

Після авторизації користувач має можливість використовувати весь функціонал програми. Він забезпечений персональними даними, які є тільки у нього під своїм власним профілем, наприклад: гаманець USDT, гаманець SWH, таблиці витрат, поповнення, дохід і багато іншого.

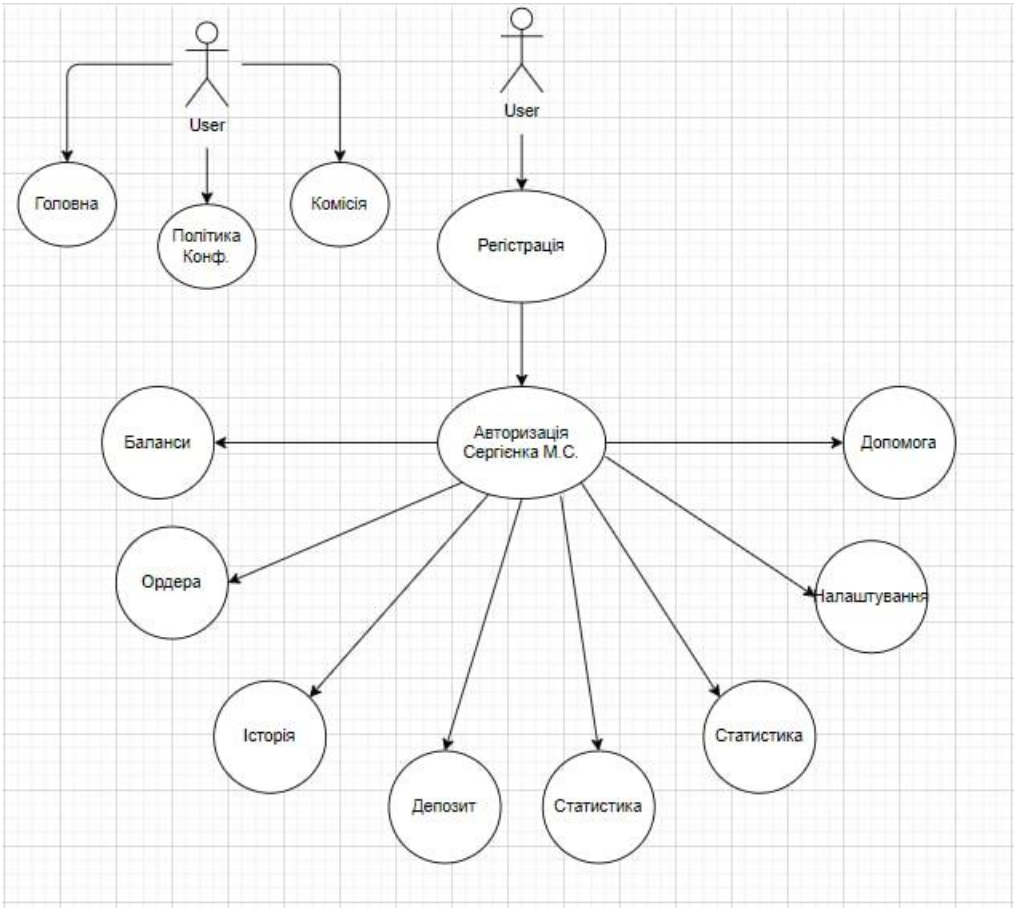

Рис. 2.2. Діаграма прецедентів

Перелік особливостей після реєстрації та вступу в додаток:

- поповнюйте свій гаманець, який створюється при реєстрації в системі;
- створення замовлення на покупку;
- створення депозиту;
- відстежування свої доходи і витрати;
- перегляд поточного балансу;
- звернутися до технічної підтримки;
- отримувати сповіщення, коли відбувається подія;
- перевірити історію ваших поповнень;
- перевірити історію своїх висновків;
- перевірити список закритих замовлень;
- перевірити витрати;

- зміна свої особисті дані;
- вихід зі свого облікового запису.

Для реєстрації використовується спеціальна сторінка, на якій є реєстраційна форма. Реєстрація відбувається в два етапи. Перший етап полягає в тому, щоб заповнити форму даними, після чого відбувається перевірка на стороні клієнта, а потім на стороні сервера, якщо перевірка пройде успішно, для користувача вже створюється профіль, але його потрібно буде активувати. Для активації профілю на адресу електронної пошти, яка була вказана при реєстрації, буде відправлено повідомлення. Після натискання на посилання профіль стає активним і ви можете увійти в додаток.

Для авторизації користувачеві необхідно зберегти свою пошту, а також особистий пароль, який він вказав під час реєстрації, після чого на його пошту повинен прийти код підтвердження, який він повинен ввести в форму авторизації, після чого користувач отримує ключ авторизації і можливість користуватися додатком.

Веб-застосунок створюється на основі компонентів. Компонентний підхід полягає у створенні невеликих елементів інтерфейсу користувача, які можуть співіснувати окремо або бути зібрані у великі блоки інтерфейсу користувача, які будуть неодноразово використовуватися в додатку.

На (рис. 2.3) показано зберігання невеликих компонентів інтерфейсу користувача , з яких складається наш додаток та розташування папок в проекті, а також основних документі (рис. 2.4).

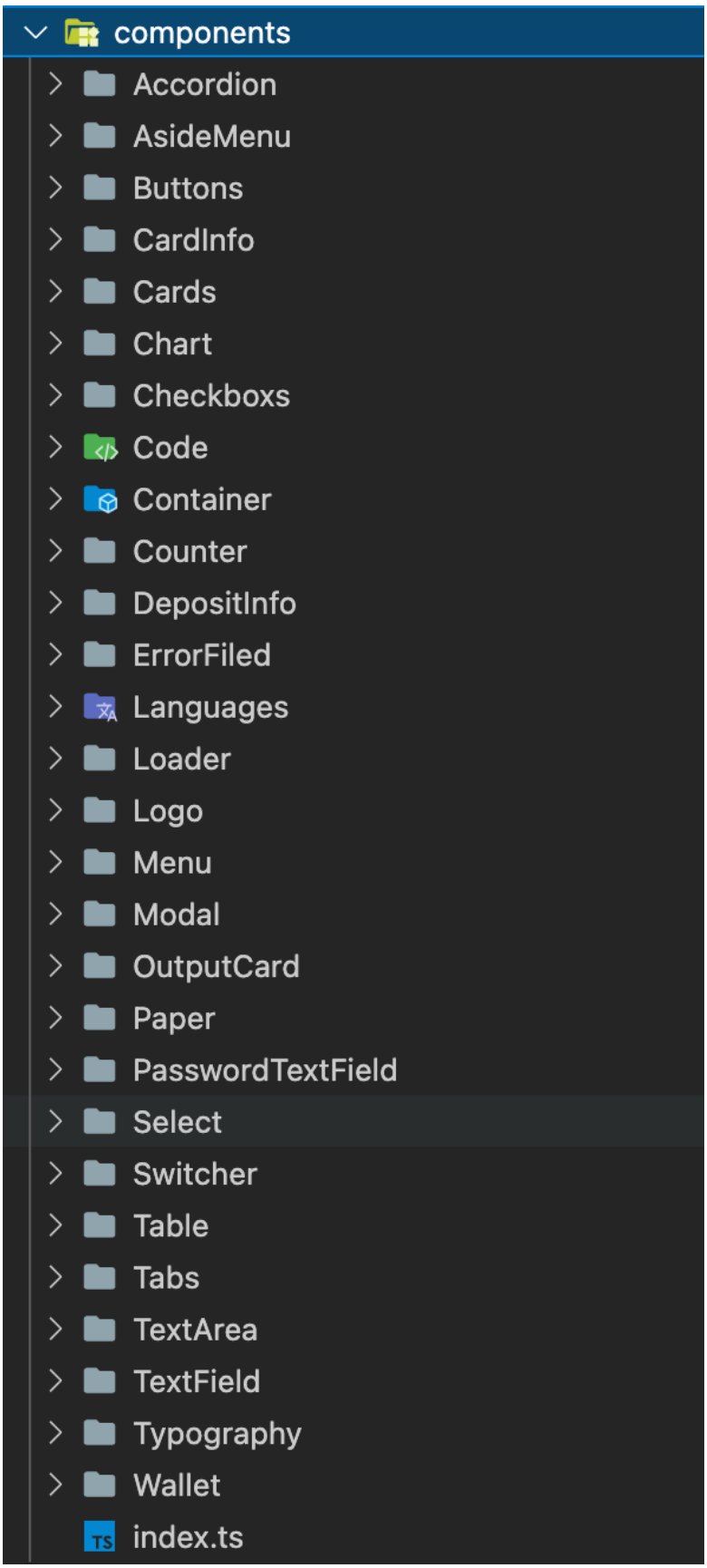

Рис. 2.3. Каталог з компонентами

Основна структура каталогів нашого додатка складається з:

- у каталозі «auth» є документ, який відповідає за авторизацію нашого користувача;
- у каталозі "components" знаходяться наші невеликі компоненти інтерфейсу користувача, які використовуються в нашому додатку;
- каталог "core" містить ядро нашої програми;
- у каталозі «hooks» є спеціальні гачки, які зберігають логіку нашого додатка, а також додаткові функції для взаємодії з компонентами;
- каталог "layouts" зберігає всі наші компоненти макетів;
- каталог "modules" зберігає всі наші модульні компоненти, які не залежать від інших модулів;
- у каталозі "pageComponents" сторінки є компонентами;
- каталог "page" - це каталог, де зберігається наша маршрутизація сторінок;
- "public" каталог зберігає всі наші статичні дані, які будуть доступні через Next.js [1];
- каталог "redux" відповідає за розташування файлів, які відповідають за глобальний стан програми;
- "service" відповідає за розташування файлів, які описують запит на сервер;
- каталог "styles" відповідає за файли, які відповідають за глобальні стилі програми;
- каталог "types" відповідає за введення даних, включених до нашої програми;
- каталог "utils" відповідає за незначні функції підтримки;
- node_modules відповідає за залежності додатку.

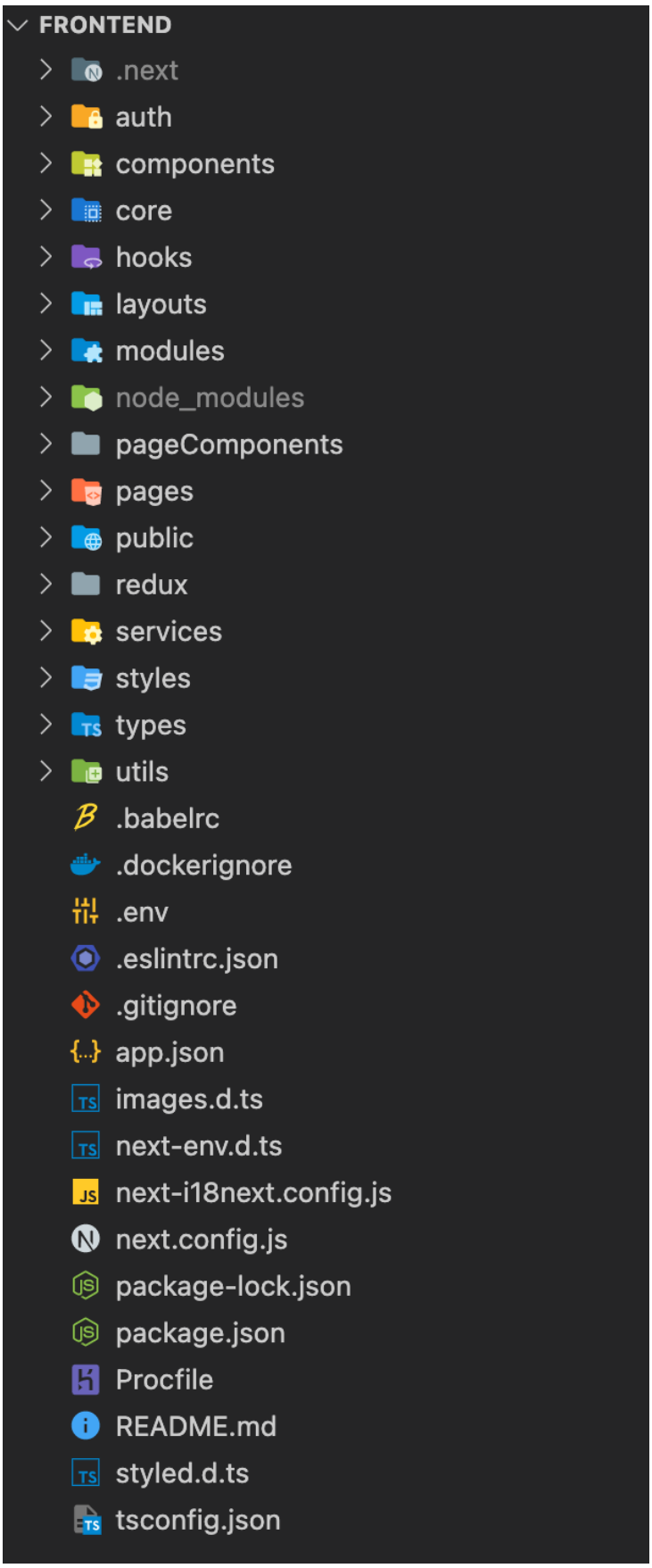

Рис. 2.4. Розташування папок в проекті і основних документів

2.6. Обґрунтування та організація вхідних та вихідних даних програми

Вихідними даними є представлення даних які зберігаються у базі даних, у зручному для графічного представлення вигляді, які, наприклад, дозволяють побудувати графік валют, або заповнити різні таблиці.

Обмін даних між сервером та клієнтським за стосунком відбувається у форматі JSON [7, 8, 9]. Так як такий тип даних зручний як для обробки на сервері, так і для роботи на клієнті.

2.7. Опис роботи розробленого програмного продукту

2.7.1. Використані технічні засоби

Для користувача є важливим мати систему, яка може запускатися та працювати з сучасними браузерами, тож необхідно мати такі мінімальні параметри ЕОМ які рекомендують розробники більшості браузерів:

- ЦП: Intel Core 4-го покоління або вище;
- Відеоадаптер: 3D адаптер nVidia, Intel, AMD/ATI;
- Відеопам’ять: 2 гб;
- Накопичувач: 128 Гб;
- Оперативна пам’ять: 4 ГБ.

2.7.2. Використані програмні засоби

Під час роботи були використані такі програмні засоби, як:

- VS Code;
- Node.js;
- Git, GitHub.

Visual Studio Code - це простий зручний редактор коду, який безкоштовний і простий у вивченні, розроблений корпорацією Microsoft.

Оскільки Microsoft займається створенням і просуванням мови TypeScript [15], vs-code має відмінну сумісність та підтримку TypeScript [15].

VS Code має безліч різних доповнень у вигляді розширень, які були розроблені іншими розробниками, завдяки яким вони охоплюють більшість недоліків IDE.

Node.js - це кросплатформний бібліотека з відкритим вихідним кодом, яка використовується для запуску веб-додатків за межами браузера клієнта [19, 20].

Він використовується для програмування на стороні сервера і в основному розгортається для серверів, керованих подіями, таких як традиційні веб-сайти та фонові служби API [5], але спочатку був розроблений з урахуванням архітектури push-сповіщень у режимі реального часу. Кожен браузер має свою версію движка JS [7, 8, 9], а node.js [16, 17] побудований на движку Google Chrome V8 JavaScript [8].

Git є найбільш часто використовуваною системою керування версіями. Git [21] відстежує зміни, які ви вносите у файли, тому у вас є запис про те, що було зроблено, і ви можете повернутися до певних версій, якщо вам це коли-небудь знадобиться. Git [21] також спрощує співпрацю, дозволяючи вам об'єднувати зміни, внесені кількома людьми, в одне джерело.

GitHub [21] є одним з найбільших ресурсів, що забезпечують можливості віддаленого сховища для спільної розробки програмного забезпечення та співпраці.

2.7.3. Виклик та завантаження програми

Веб-додаток завантажується, натиснувши на посилання, коли користувач натискає на нього. Додаток відображається в браузері будь-якого пристрою, головною умовою є встановлений браузер на пристрої користувача, а також нова версія браузера, так як старі браузери можуть не підтримувати сучасні технології, які використовуються на сайті. Додаток можна запустити

на будь-якій платформі ОС Windows, Linux, Mac OS на таких браузерах Google Chrome, Firefox, Yandex, Opera і багатьох інших.

Додаток працює на двох останніх версіях браузера, але чим новіша версія, тим краще, це захистить від небажаних збоїв в системі або помилок.

2.7.4. Опис інтерфейсу користувача

Коли користувач потрапляє в наш додаток, його зустрічає головна сторінка, де він може ознайомитися з додатком, а також прочитати інформацію про криптовалюте.

З головної сторінки здійснюється вся можлива навігація по додатку, але користувач має два рівні доступу. Перший - авторизований користувач, який має доступ тільки до головної сторінки і публічних новин, а другий - авторизований, де користувач отримує можливість користуватися додатком в повну силу і використовувати криптовалютні кошти.

Головна сторінка розповість вам про переваги веб-додатка, а також можливості. Мета цієї сторінки познайомить користувача з криптовалютою і всіма правилами сервісу (рис. 2.5, рис. 2.6, рис. 2.7, рис. 2.8, рис. 2.9, рис.2.10).

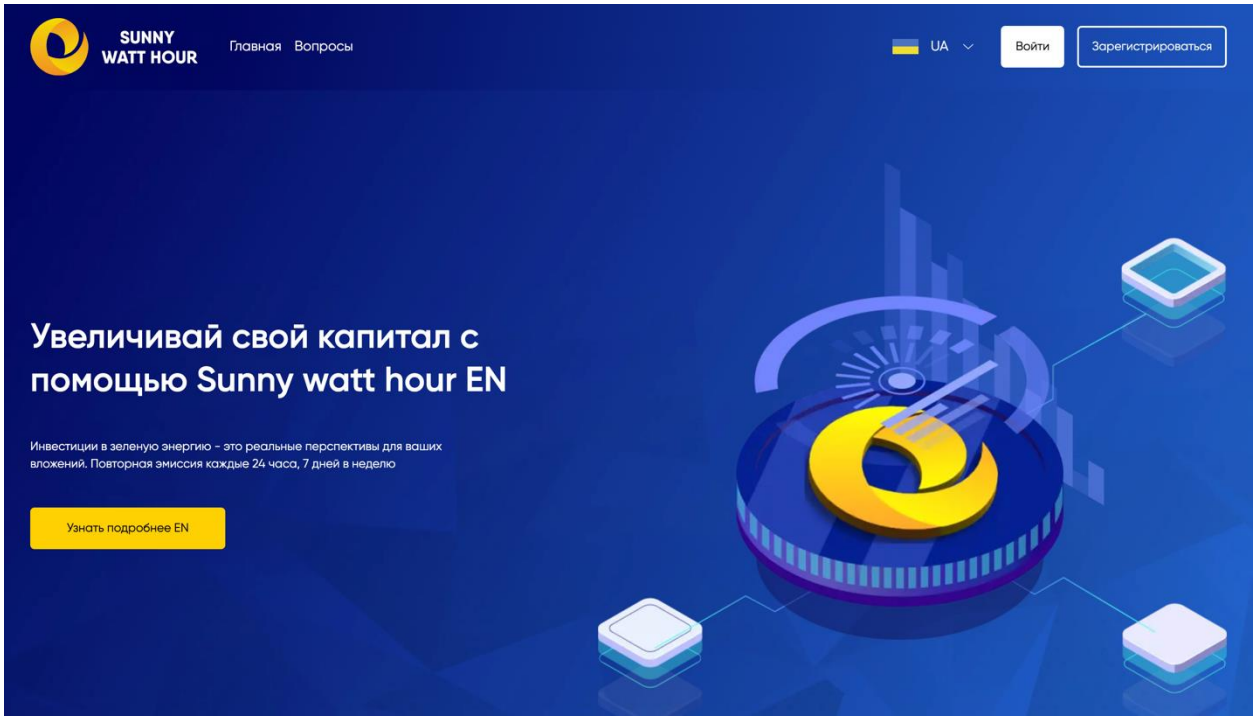

Рис. 2.5. Головна сторінка веб-додатка

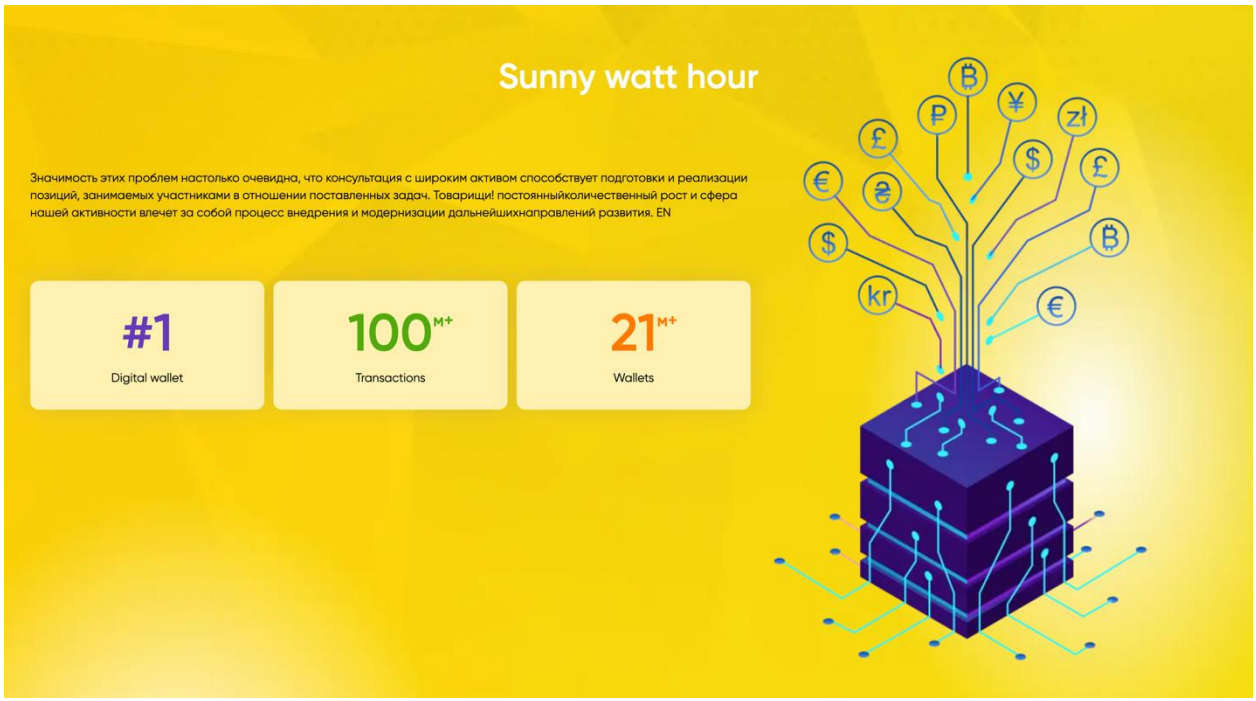

Рис. 2.6. Блок з інформацією про Sunny watt hour

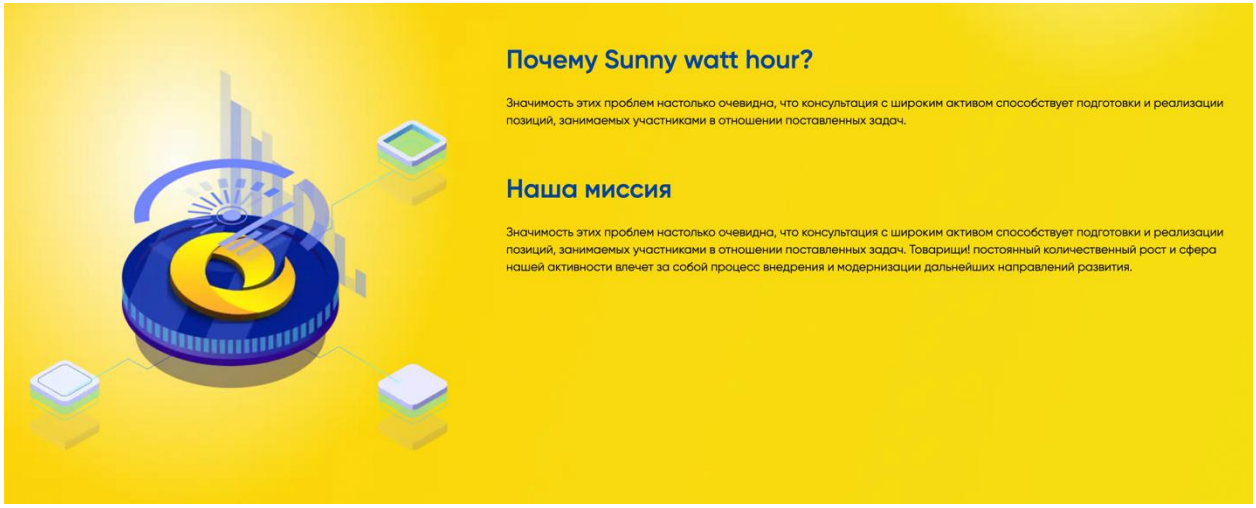

Рис. 2.7. Продовження рис. 2.6

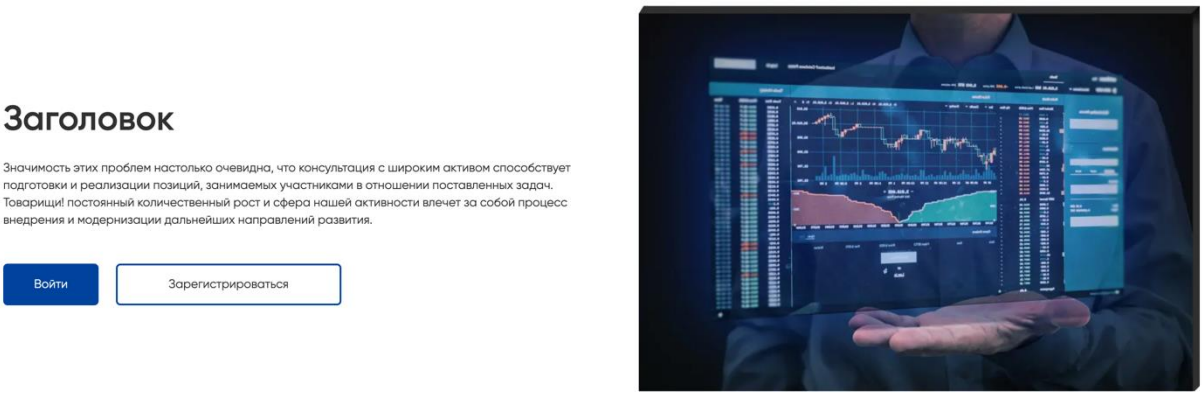

Рис. 2.8. Інформаційний блок

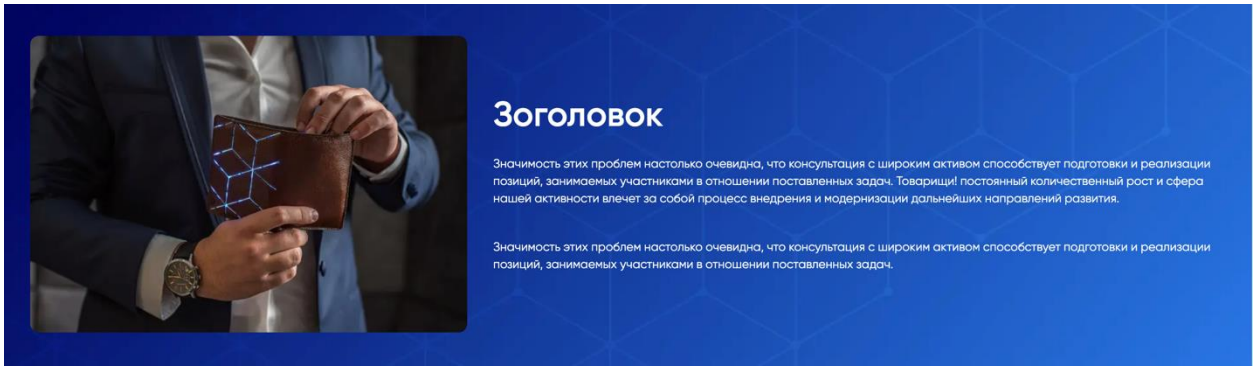

Рис. 2.9. Інформаційний блок

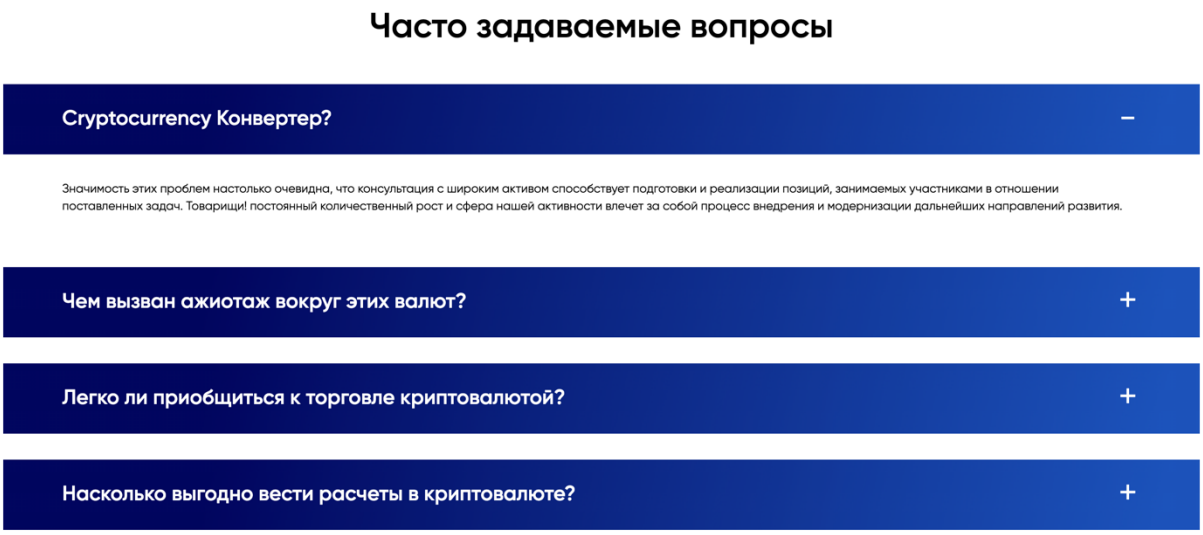

Рис. 2.10. Часті питання

Сторінка реєстрації дозволяє користувачеві зареєструватися в додатку, а також отримати індивідуальний рахунок в додатку з унікальним ніком і криптовалютними гаманцями (рис. 2.11).

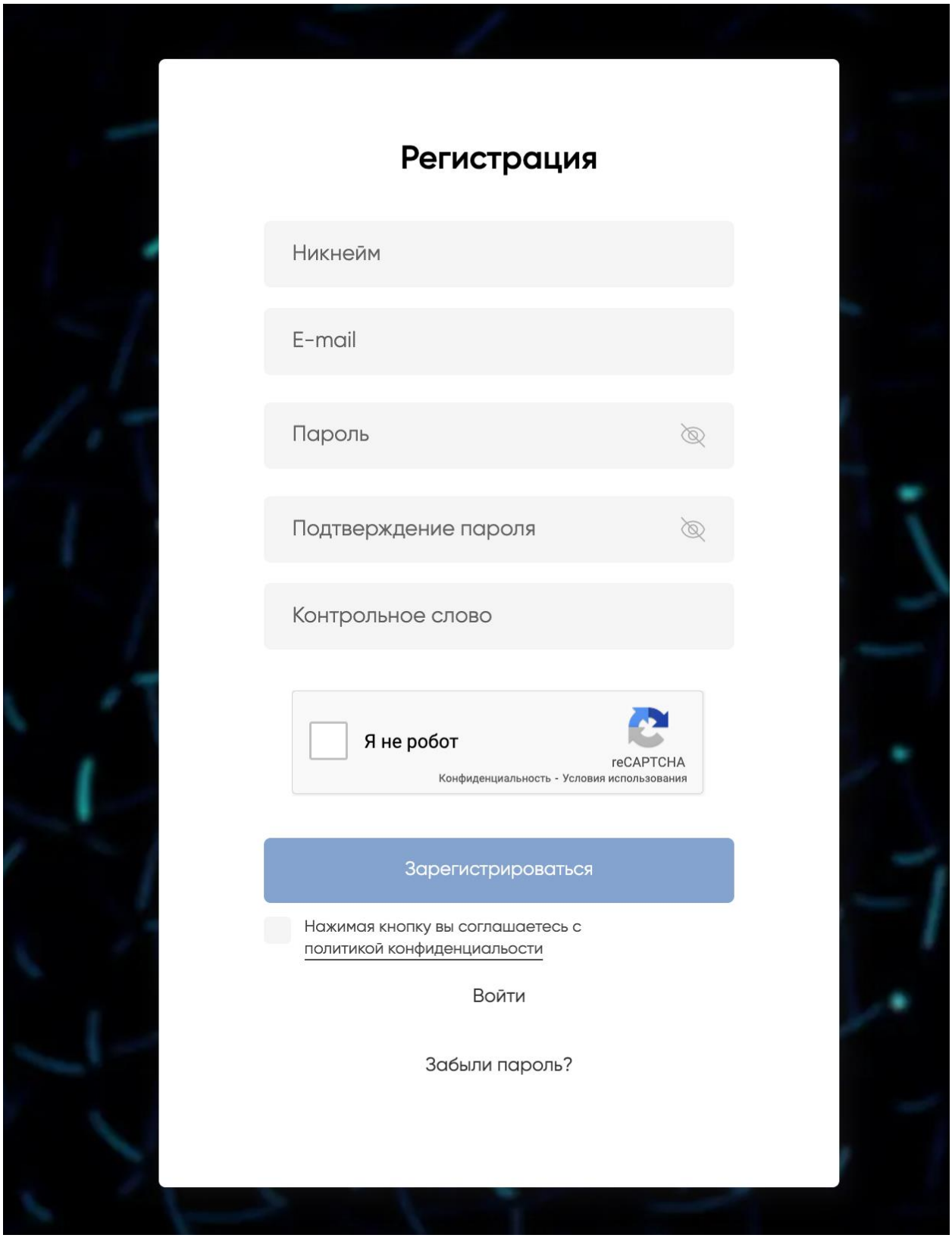

Рис. 2.11. Реєстраційна форма

Система реєструє в два етапи. Першим кроком є введення особистої інформації з профілю (рис. 2.12). Другий етап - ввести код, який надійшов на особисту пошту, який був вказаний при реєстрації (рис. 2.13).

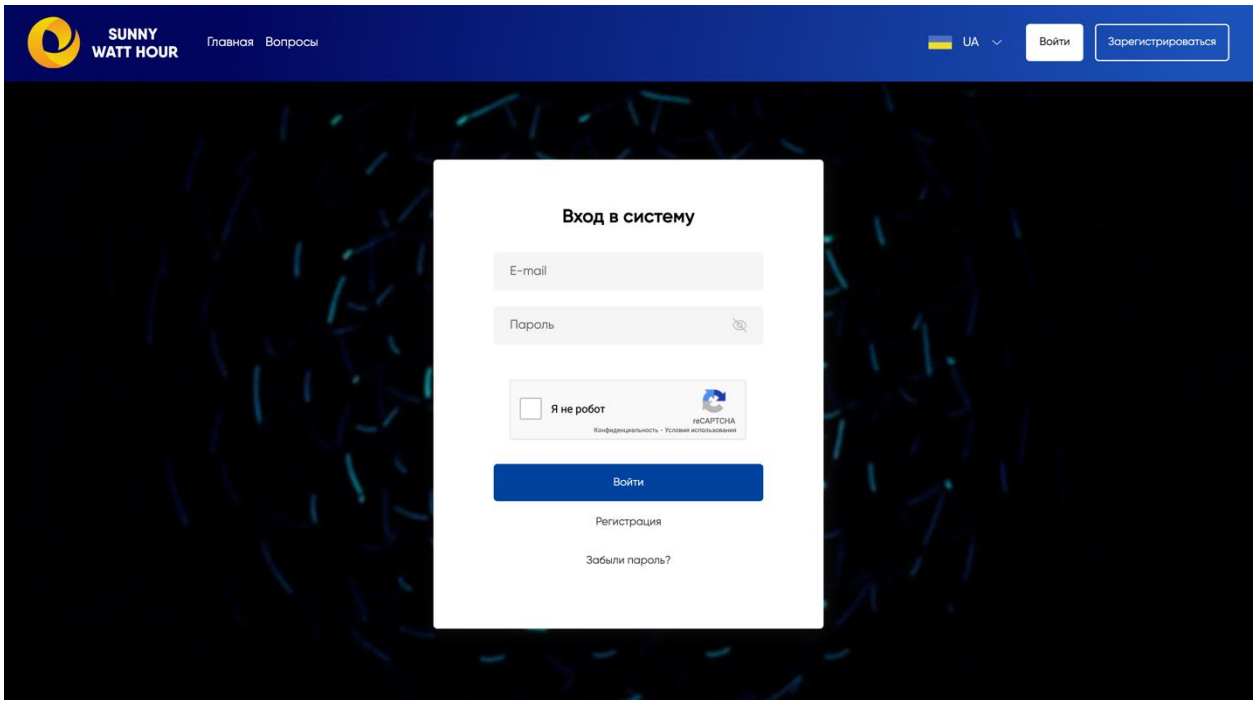

Рис. 2.12. Форма авторизації першої частини

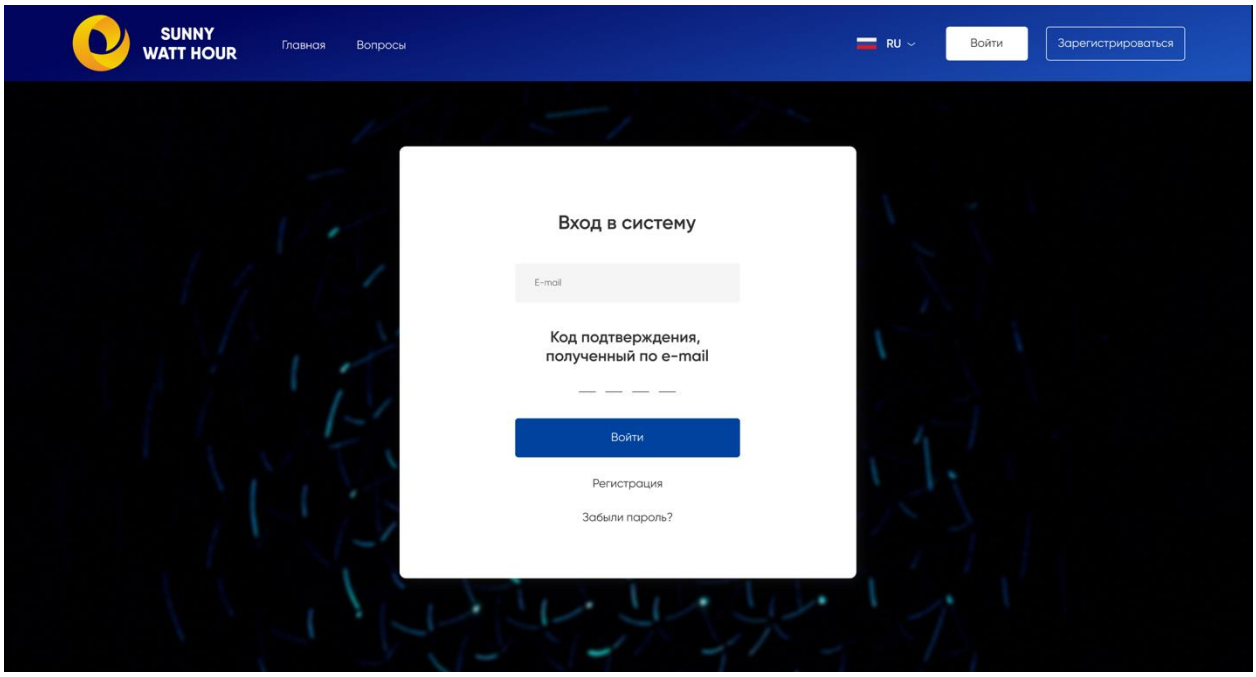

Рис. 2.13. Форма авторизації другої частини

Якщо користувач забув пароль для облікового запису, він може відновити пароль за допомогою спеціальної сторінки «Відновлення пароля» (рис. 2.14).

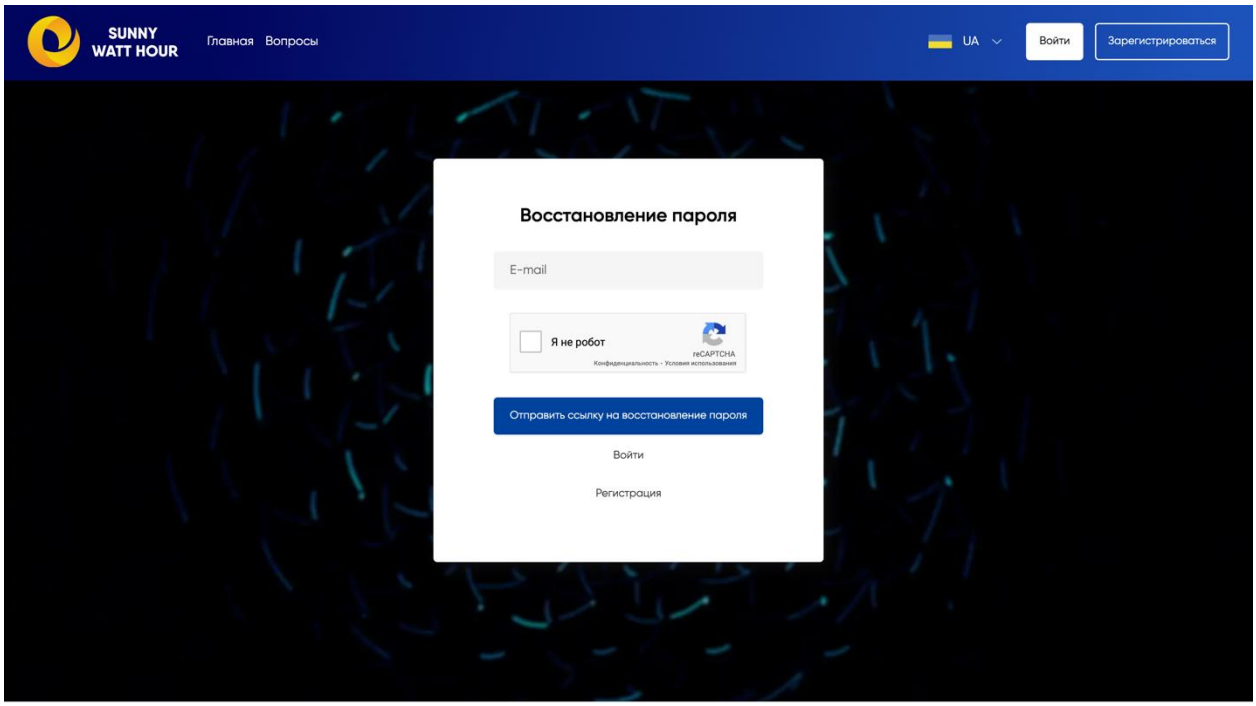

Рис. 2.14. Форма відновлення пароля

При вході в додаток користувач знаходиться на сторінці залишків, де він може дізнатися поточний стан коштів (рис. 2.15, рис. 2.16).

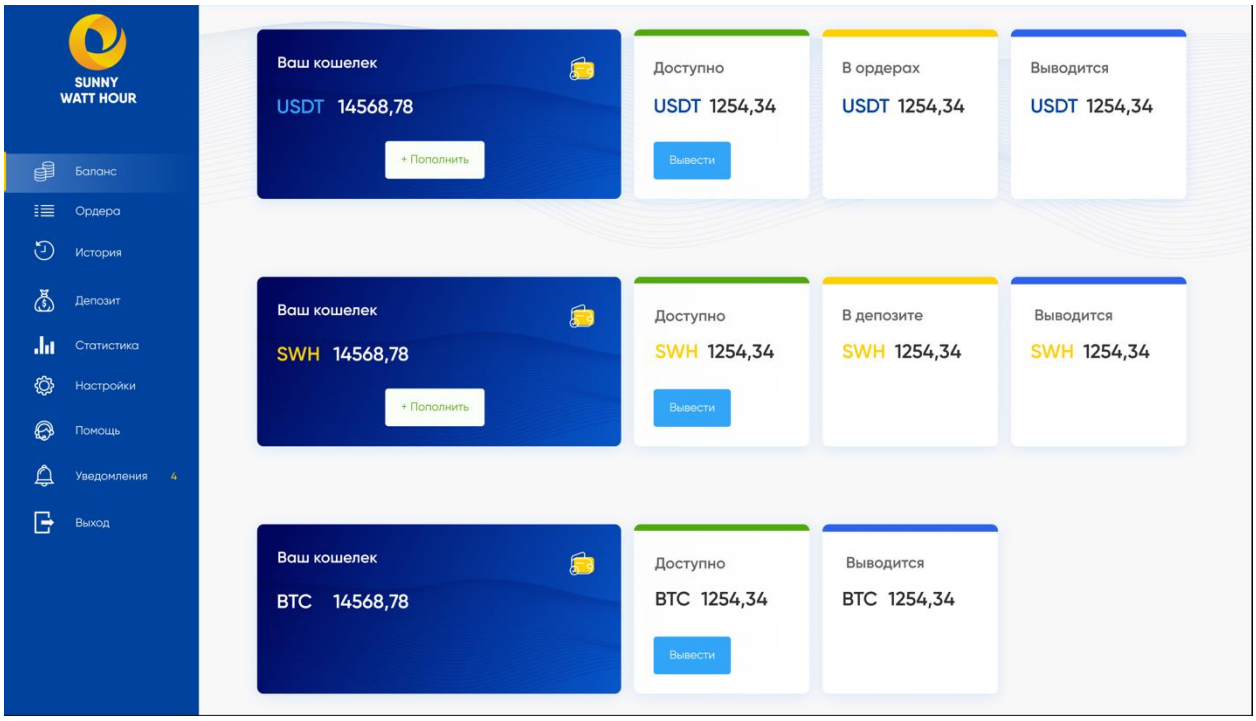

Рис. 2.15. Сторінка залишків

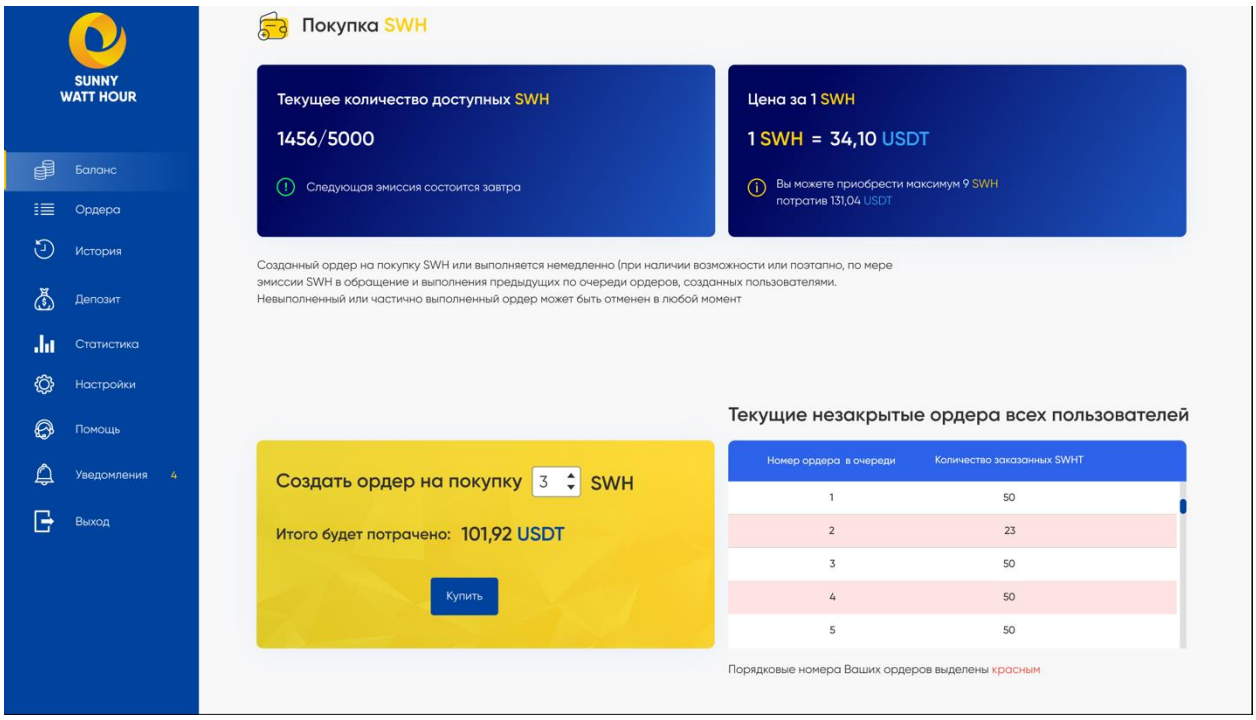

Рис. 2.16. продовження сторінки балансу

На сторінці замовлень можна побачити список замовлень, а також скасувати замовлення, якщо воно не потрібно (рис. 2.17).

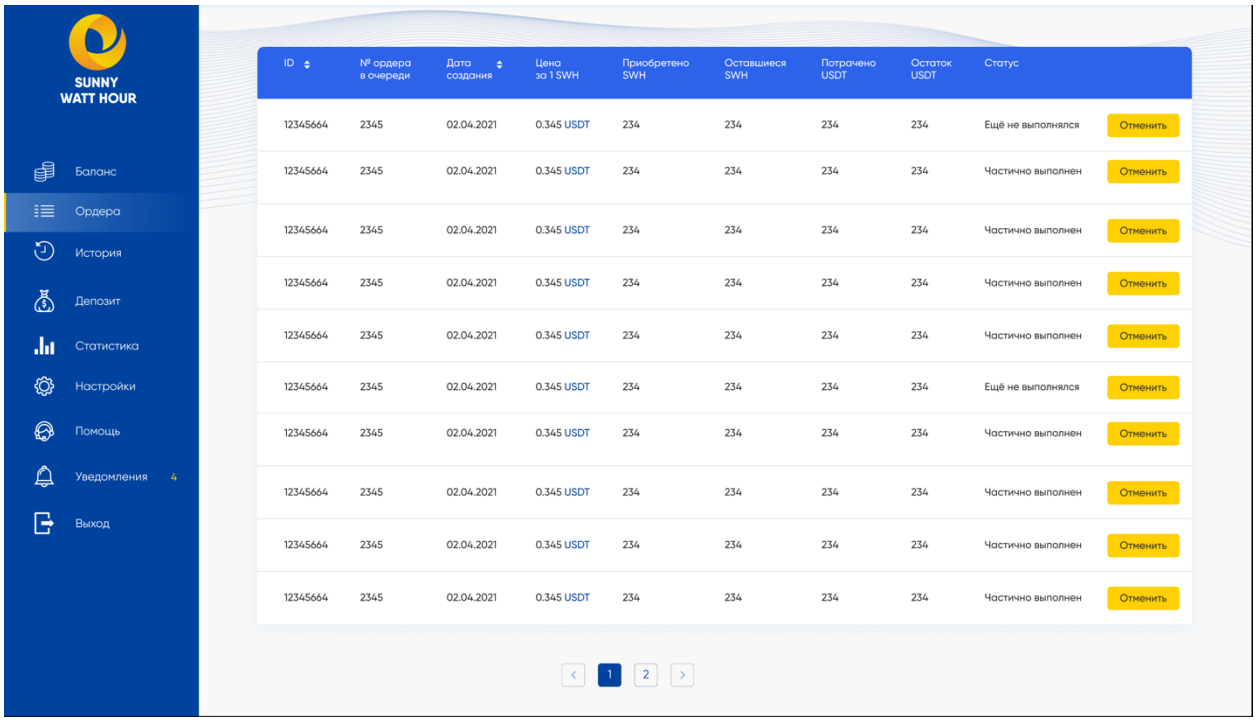

Рис. 2.17. Сторінка замовлення

Сторінка історії несе відповідальність за всі дії, які були зроблені готівкою (рис. 2.18).

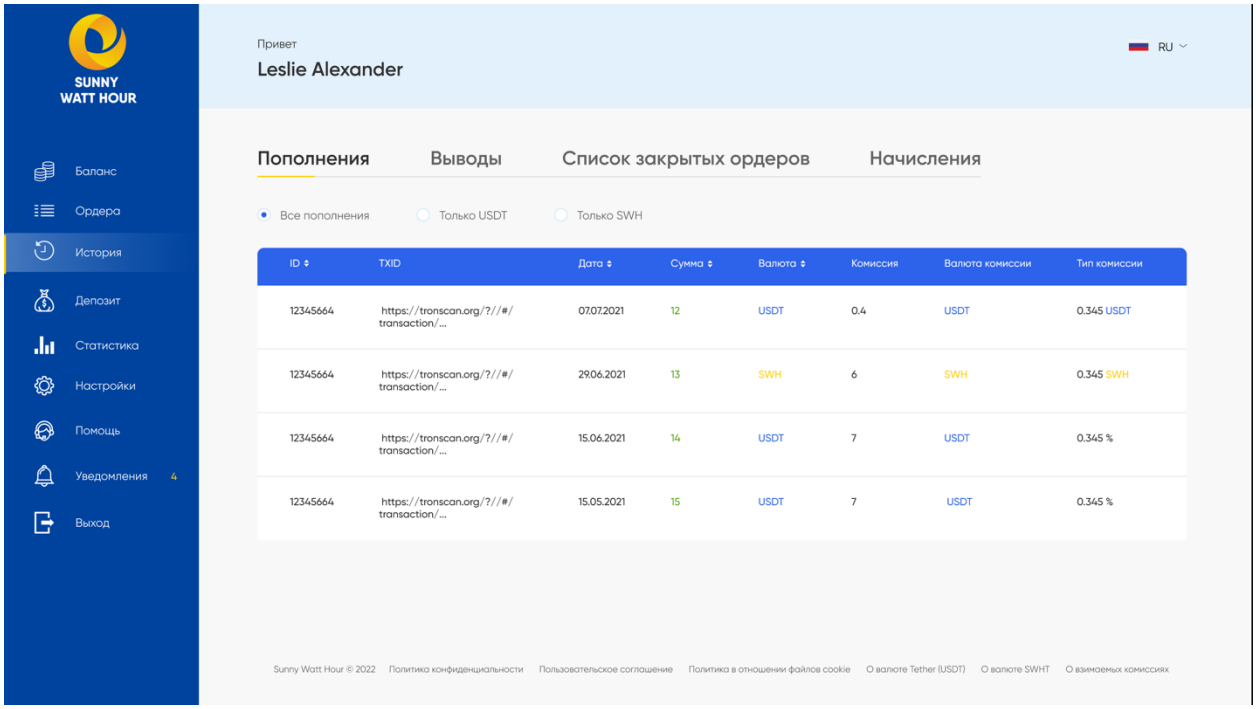

Рис. 2.18. Сторінка історії

На сторінці депозиту можна створити депозит, а також переглянути баланс і історію вкладів (рис. 2.19, рис. 2.20).

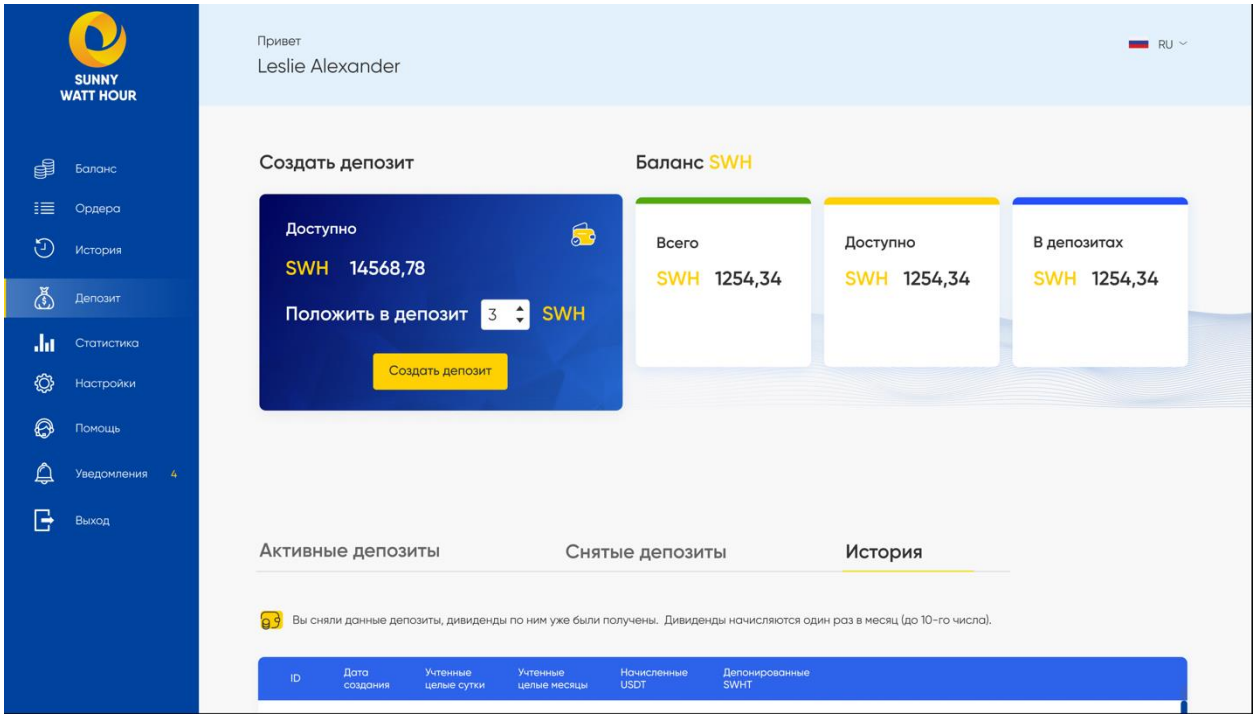

Рис. 2.19. Сторінка депозиту

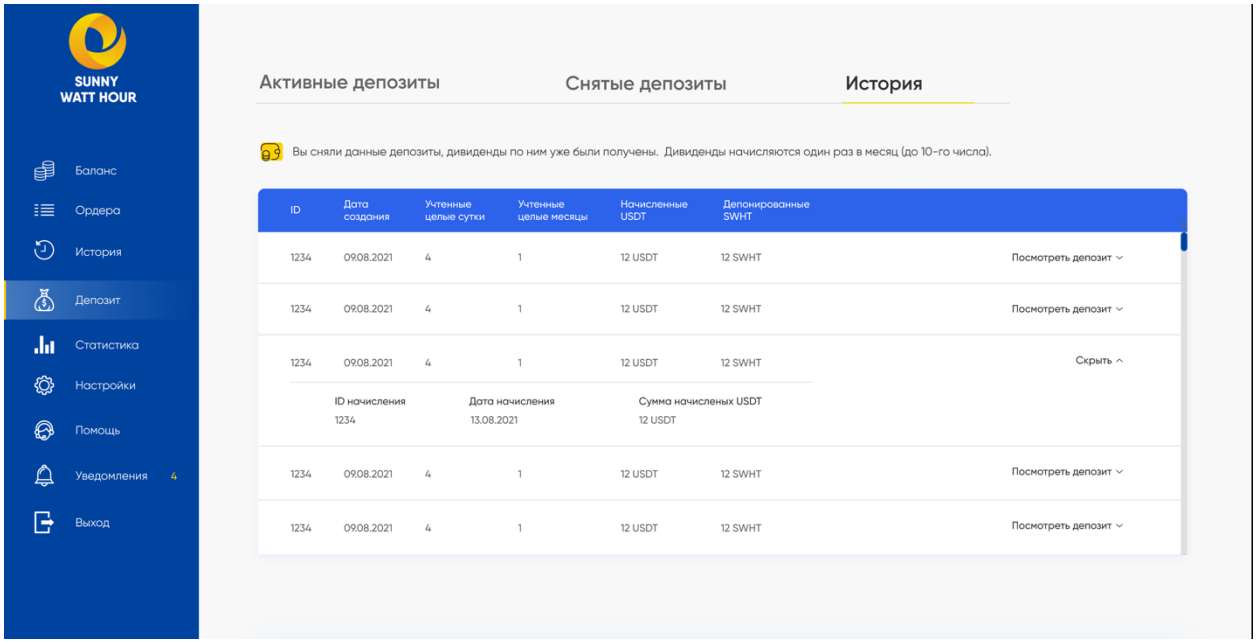

Рис. 2.20. Продовження сторінки депозиту

На сторінці статистики користувач може відстежувати прибуток або витрати на всі доступні криптовалюти за допомогою таблиці або явного графіку (рис. 2.21, рис. 2.22).

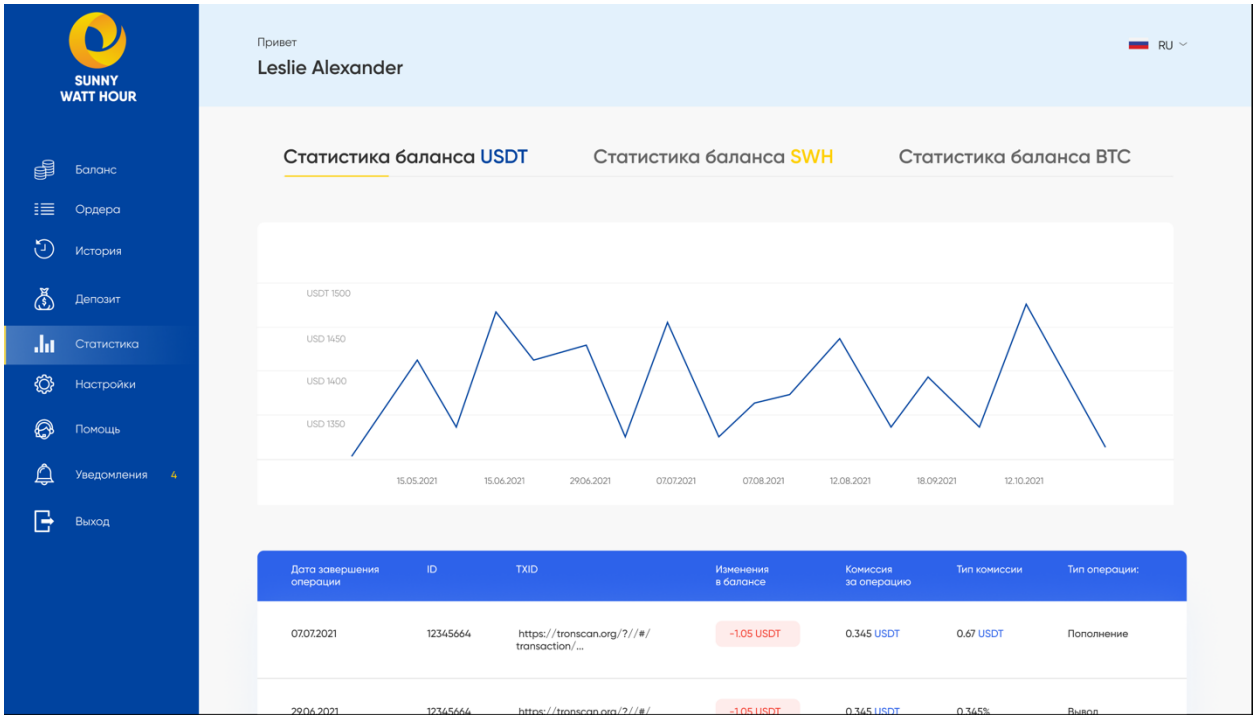

Рис. 2.21. Сторінка статистики

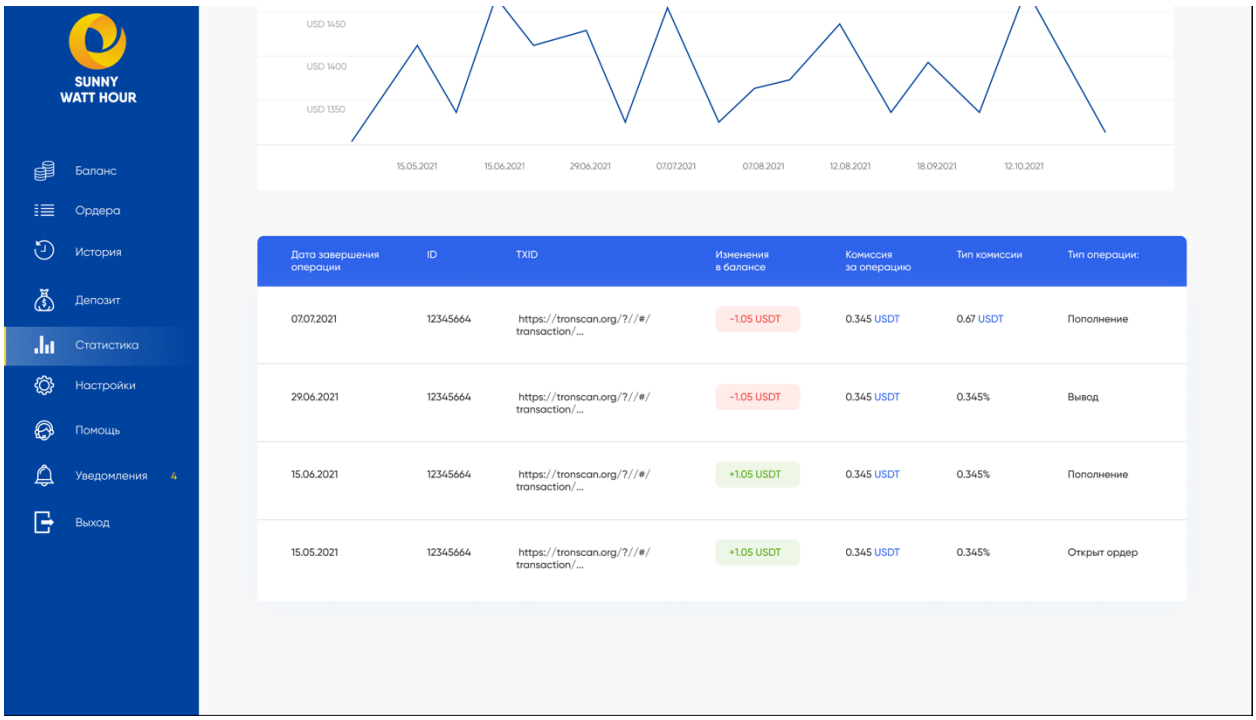

Рис. 2.22. Продовження сторінки статистики

На сторінці налаштувань користувач може змінити свої персональні дані: нікнейм, пароль, пошту (рис. 2.23).

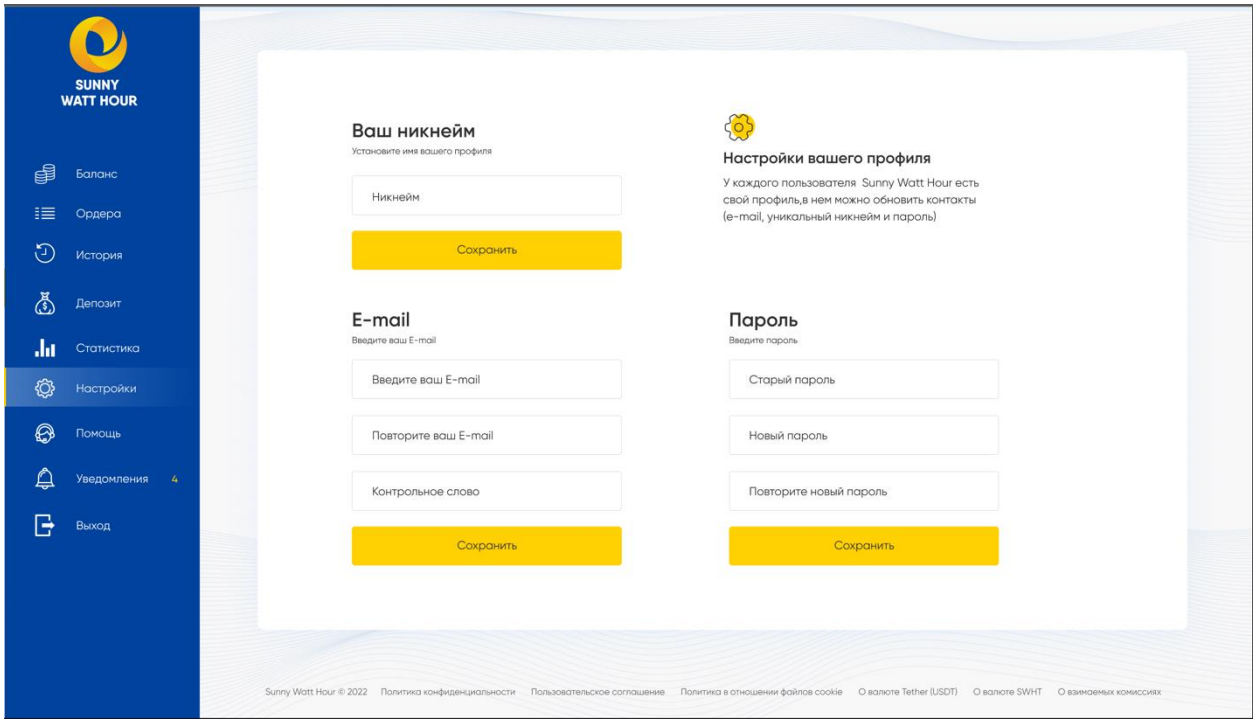

Рис. 2.23. Сторінка налаштувань

Користувач може отримати допомогу в додатку через довідкову форму, для цього потрібно заповнити її і вибрати причину звернення (рис. 2.23).

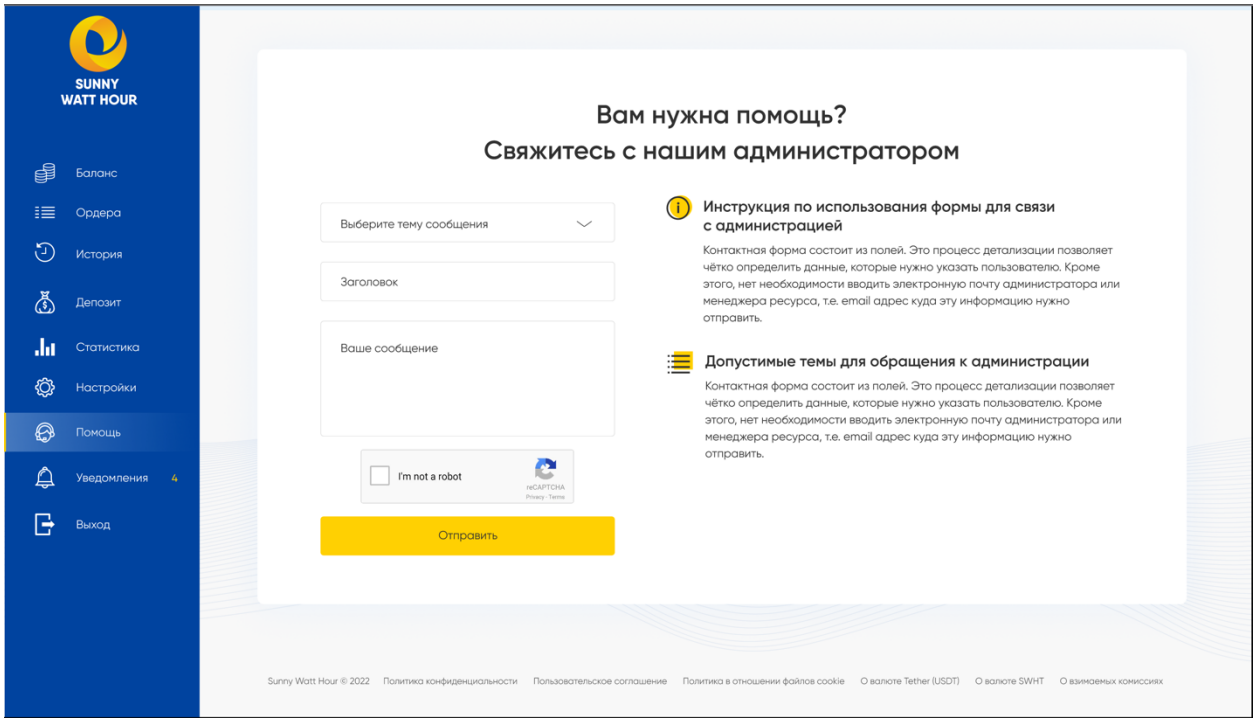

Рис. 2.24. Сторінка довідки

На сторінці сповіщень користувач отримує всі повідомлення про події, що відбулися (рис. 2.25).

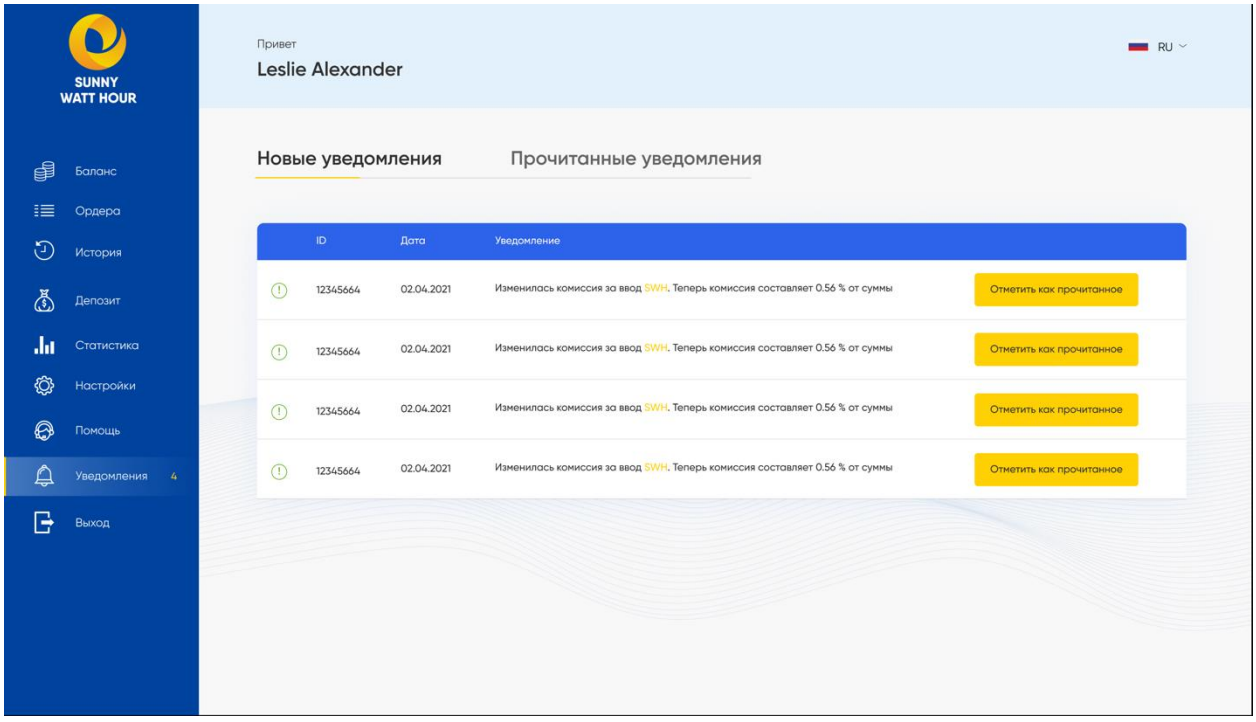

Рис. 2.25. Сторінка сповіщень

Для поповнення балансу користувачеві дається його особистий гаманець, який був створений при реєстрації. Завдяки побудованим сервісам він може поповнити рахунок (рис. 2.26).

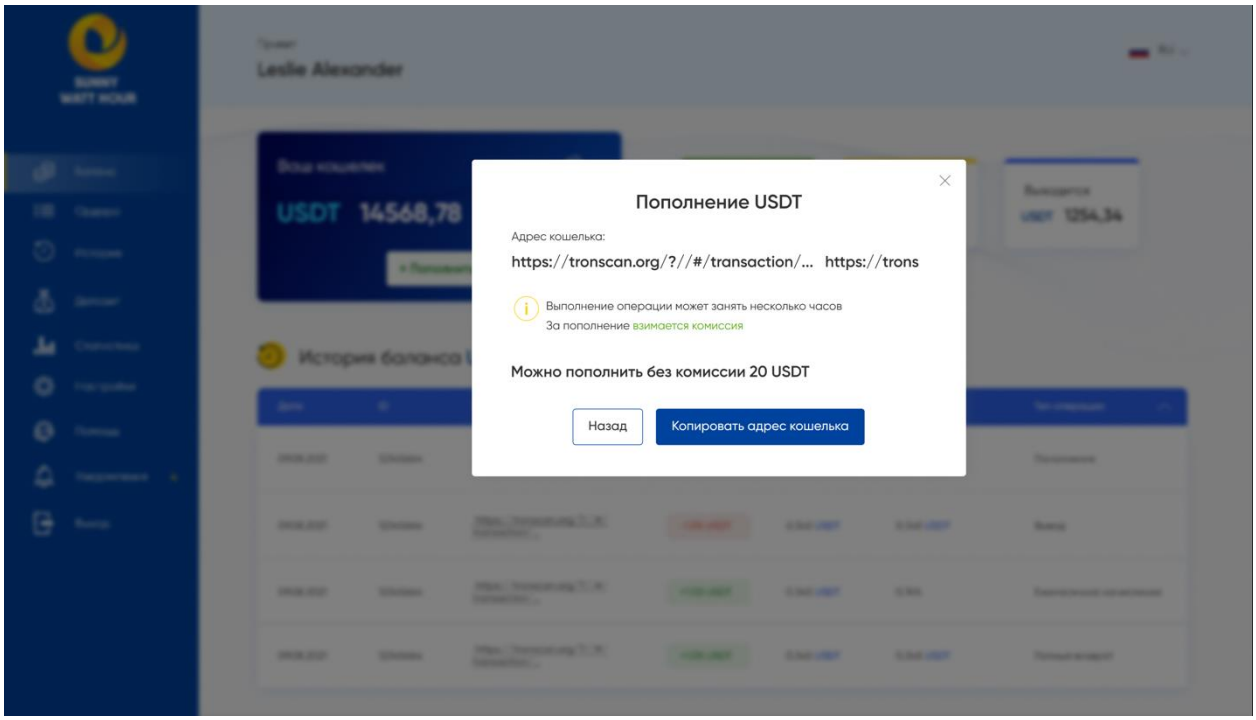

Рис. 2.26. Модальне поповнення балансу вікном

На сторінці виведення коштів користувач може зняти зароблені гроші або свої заощадження (рис. 2.27).

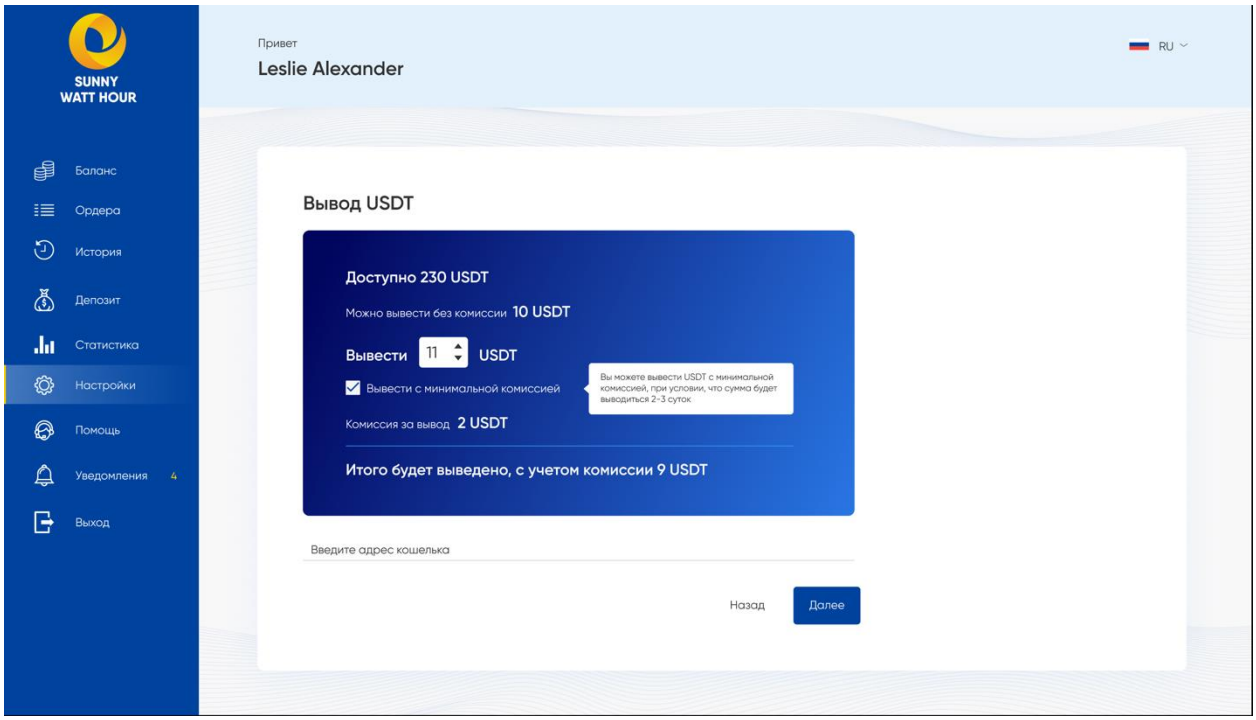

Рис. 2.27. Сторінка виведення валюти

Щоб зняти кошти, потрібно отримати спеціальний код, який приходить на пошту для підтвердження дій користувача і що він впевнений, що хоче зняти кошти (рис. 2.28, рис. 2.29).

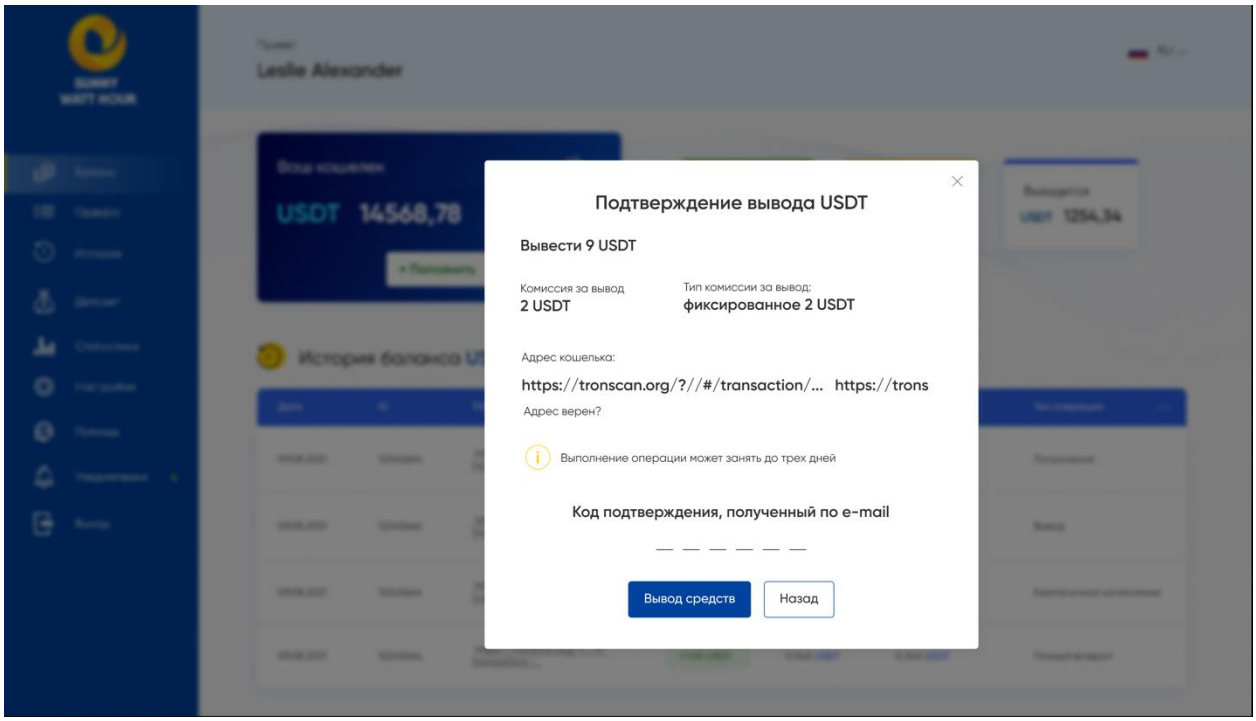

Рис. 2.28. Модальне вікно для підтвердження зняття коштів

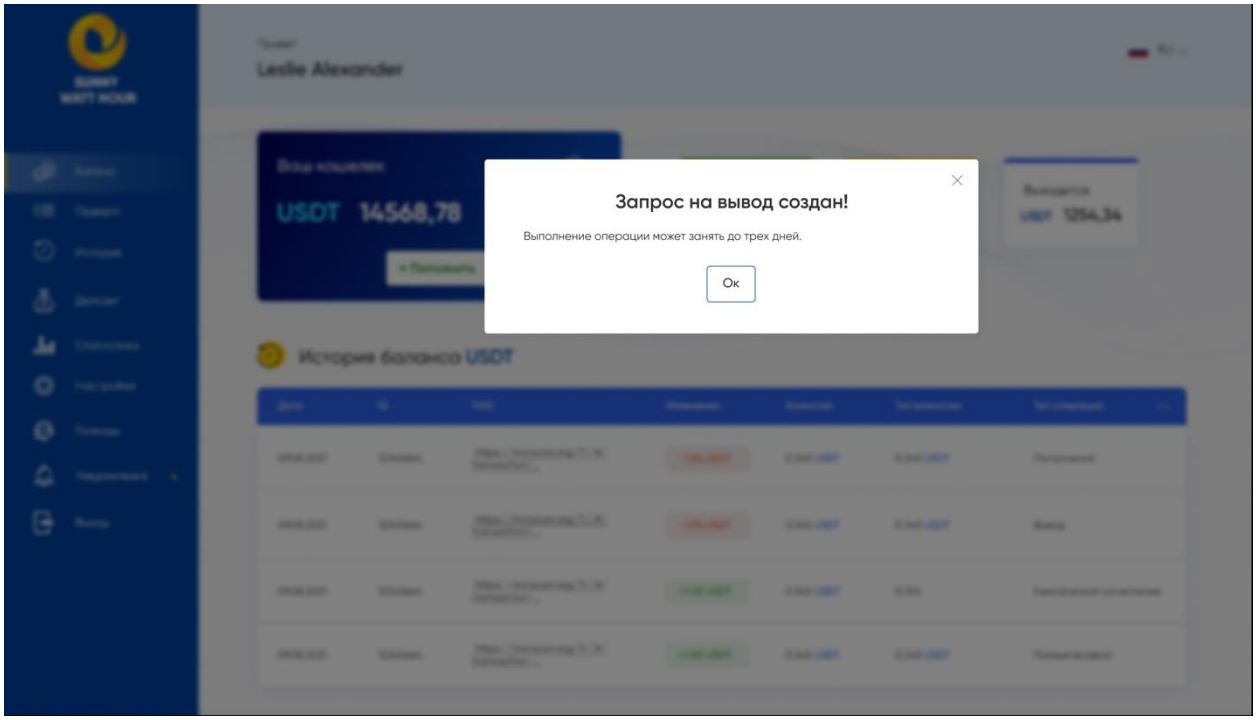

Рис. 2.29. Модальне вікно про успіх запиту на виведення коштів

Якщо користувач не перейде за існуючим посиланням, він викине сторінку 404, яку не знайдено (рис. 2.30).

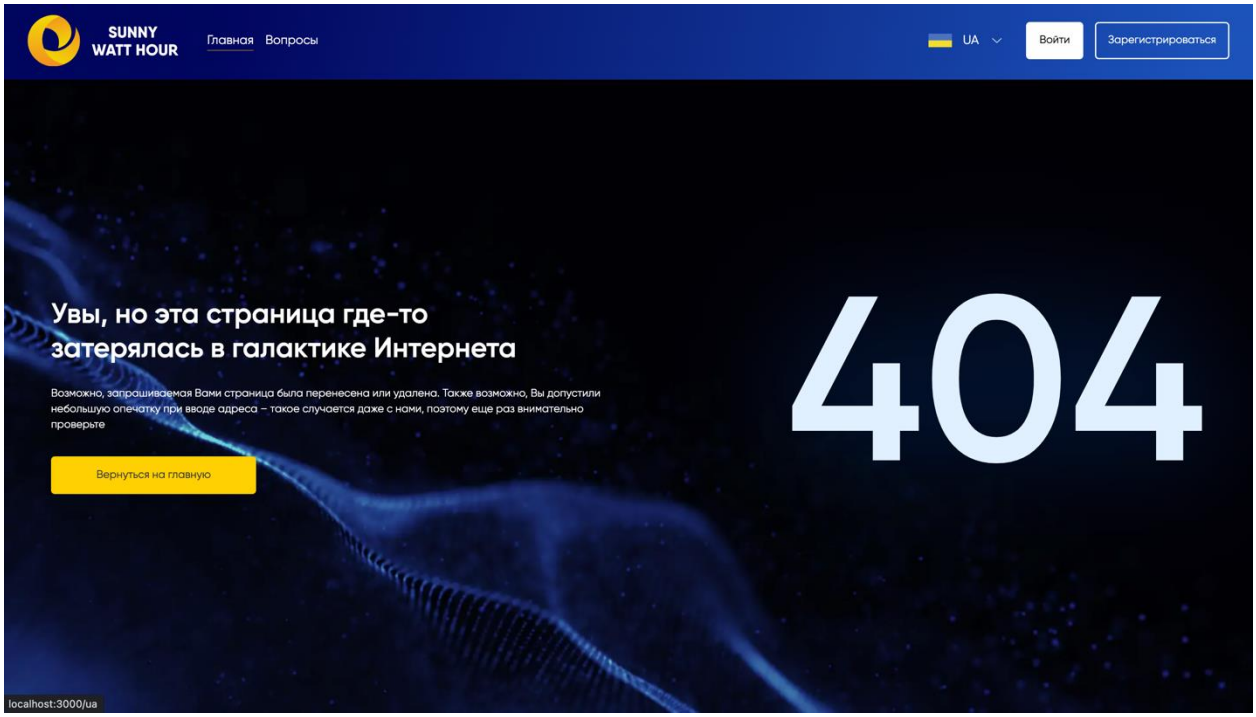

Рис. 2.30. Сторінка 404

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. передбачуване число операторів програми – 2000;

2. коефіцієнт складності програми – 1,3;

3. коефіцієнт корекції програми в ході її розробки – 0,05;

4. годинна заробітна плата програміста – 180 грн/год;

5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,2;

6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1;

7. вартість машино-години ЕОМ – 15 грн/год.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д, \qquad (3.1)$$

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 50);

t_u – витрати праці на дослідження алгоритму рішення задачі;

t_a – витрати праці на розробку блок-схеми алгоритму;

t_n – витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ – витрати праці на налагодження програми на ЕОМ;

$t_д$ – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p), \tag{3.2}$$

де q – передбачуване число операторів;

C – коефіцієнт складності програми;

p – коефіцієнт кореляції програми в ході її розробки.

$$Q = 2000 \cdot 1{,}3 \cdot (1 + 0{,}05) = 2730$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75 \ldots 85) \cdot K}, \tag{3.3}$$

де B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

K – коефіцієнт кваліфікації програміста, обумовлений стажем роботи з даної спеціальності;

$t_u = (2730 \cdot 1{,}2) / (83 \cdot 1) = 39{,}46$ людино-годин.

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20 \ldots 25) \cdot K}, \tag{3.4}$$

t_a = 2730 / (21 · 1) = 130 людино-годин.

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)\cdot K}, \quad (3.5)$$

t_n = 2730 / (23 · 1) = 118,69 людино-годин.

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4...5)\cdot K}, \quad (3.6)$$

$t_{отл}$ = 2730 / (4 · 1) = 682,5 людино-годин.

– за умови комплексного налагодження завдання:

$$t^{к}_{отл} = 1,2 \cdot t_{отл}, \quad (3.7)$$

$t_{отл}^{k}$ = 1,2 * 682,5 = 819 людино-годин.

Витрати праці на підготовку документації визначається за формулою:

$$t_д = t_{др} + t_{до}$$, людино-годин,

де $t_{др}$ – трудомісткість підготовки матеріалів і рукопису:

$$t_{\text{др}} = \frac{Q}{(15...20)*k}$$, людино-годин.

$t_{\text{до}}$ – трудомісткість редагування, печатки й оформлення документації:

$$t_{\text{до}} = 0{,}75 * t\text{д}$$, людино-годин.

Підставляючи відповідні значення, отримуємо:

$$t_{\text{др}} = \frac{2730}{17*1} = 160{,}5$$ людино-годин.

$t_{\text{до}} = 0{,}75 * 160{,}5 = 120{,}3$ людино-годин.

$t_{\text{д}} = 160{,}5 + 120{,}3 = 280{,}8$ людино-годин.

Повертаючись до формули (3.1), отримаємо повну оцінку трудомісткості розробки програмного забезпечення:

$t = 50 + 39{,}46 + 130 + 118{,}69 + 682{,}5 + 280{,}8 = 1301{,}45$ людино-годин.

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ включають витрати на заробітну плату виконавця програми З/п і витрат машинного часу, необхідного на налагодження програми на ЕОМ.

$$K_{ПО} = З_{ЗП} + З_{МВ} \text{ грн.,} \quad (3.11)$$

де $З_{ЗП}$ – заробітна плата виконавців, яка визначається за формулою:

$$З_{ЗП} = t \cdot C_{ПР} \text{ грн.,} \quad (3.12)$$

де t – загальна трудомісткість, людино-годин;

$C_{ПР}$ – середня годинна заробітна плата програміста, грн/година

$$З_{зп} = 1301{,}45 * 180 = 234261 \text{ грн.}$$

$З_{МВ}$ – Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} \cdot C_{М} \text{ грн.,} \qquad (3.13)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год.

$C_{МЧ}$ – вартість машино-години ЕОМ, грн/год.

$$З_{мв} = 682{,}5 * 15 = 10237{,}5 \text{ грн.}$$

Звідси витрати на створення програмного продукту:

$$К_{по} = 234261 + 10237{,}5 = 244498{,}5 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p} \text{ міс.,} \qquad (3.14)$$

де $В_к$- число виконавців;

$F_р$ – місячний фонд робочого часу (при 40 годинному робочому тижні $F_р$=176 годин).

$$T = \frac{1301{,}45}{1 \cdot 176} \approx 7{,}39 \text{ міс.}$$

Висновок: програмне забезпечення розроблено для продажу та купівлі криптовалюти. Вартість даного програмного забезпечення становить близько 244498,5 грн. Очікуваний час розробки становить 1301,45 годин, тобто 7,39 місяці при 40 годинному робочому тижні. Цей термін пов'язаний зі значним числом операторів, і включає час на дослідження і розробку алгоритму вирішення поставленого завдання, програмування по готовому алгоритму, налагодження програми і підготовку документації.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи був створений зручний веб-додаток з використанням сучасних веб-технологій SPA, який в даний час вважається одним з кращих.

Перевагами використання сучасних технологій є можливість масштабування програми, а також якість коду і швидкість написання програми, а також якість додатку.

Метою даного веб-додатка було створення зручного інтерфейсу для користувача, простого і зрозумілого, а також приховування важливих дій всередині програми.

Веб-додаток дозволяє контролювати свої доходи і витрати, здійснювати купівлю або продажу криптовалюти або створити депозит, замовлення. Користувач може отримати актуальну інформацію про обмінний курс або комісію.

В результаті роботи ми отримали зручний веб-додаток для взаємодії з біржами і купівлі-продажу нової криптовалюти. Актуальність програми полягає в тому, що вам не потрібно йти на біржі, щоб купити криптовалюту. Це можна зробити за допомогою зручного інтерфейсу користувача, який надсилає запит на BACKEND, де вже відбувається всі фінансові процеси.

Під час виконання даної кваліфікаційної роботи були виконані етапи створення програмного продукту:

- аналіз предметної галузі задачі;
- обрання раціональної архітектури та технології створення додатку;
- написання програмного коду веб-додатку;
- розробка рекомендацій щодо використання застосунку.

Під час виконання кваліфікаційної роботи також було визначено трудомісткість розробленого програмного продукту (1301, 45 людино-годин), проведений підрахунок вартості роботи по створенню програми (244498,5 грн) та розраховано час на його створення (7.39 міс).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Офіційна документація Next.js. https://nextjs.org/.
2. Офіційна документація React. https://uk.reactjs.org.
3. Стаття о SPA. https://wezom.com.ua/blog/chto-takoe-spa-prilozheniya
4. Інформаційне онлайн джерело по HTML https://developer.mozilla.org/ru/docs/Web/HTML.
5. Стаття о API. https://xn--90aamhd6acpq0s.xn--j1amh/teoriya/api-shcho-tse-take-prostymy-slovamy/
6. Стаття о hooks в React. https://dou.ua/lenta/articles/react-hooks-guide/
7. Онлайн-підручник з JavaScript. https://github.com/getify/You-Dont-Know-JS.
8. Нік Морган. JavaScript для маленьких, 2021. 368 с.
9. Онлайн-підручник з вивчення JavaScript. https://javascript.info. 1300 с.
10. Офіційна документація по роботі з формами в React. https://react-hook-form.com.
11. Офіційна документація Axios. https://axios-http.com/docs/intro.
12. Довідка по роботі з веб-технологіями. https://my-js.org/.
13. Онлайн джерело о веб-технологіям. https://code.mu/ru/markup/book/prime/
14. Офіційне джерело по CSS. https://developer.mozilla.org/en/docs/Web/CSS
15. Офіційна документація по TypeScript. https://www.typescriptlang.org/.
16. Офіційна документація по Redux. https://redux.js.org/
17. Офіційна документація по Redux-toolkit. https://redux-toolkit.js.org/
18. Офіційна документація Styled-component. https://styled-components.com
19. Офіційна документація npm. https://www.npmjs.com.
20. Офіційна документація з node.js. https://nodejs.org/en/.
21. Офіційна документація з Git. https://git-scm.com.

22. Офіційна документація по Webpack 5. https://webpack.js.org/concepts/.
23. Офіційна документація yup. https://github.com/jquense/yup.
24. Офіційна документація Frame motion. https://www.framer.com/motion/
25. React 16: огляд нової архітектури fiber. https://dou.ua/lenta/articles/react-fiber/

ДОДАТОК А

КОД ПРОГРАМИ

Лістинг page/index.ts:

```
import type { NextPage } from "next";
import { serverSideTranslations } from "next-i18next/serverSideTranslations";
import { HeaderMain, Footer } from "../modules";
import { SunnyWatt } from "../pageComponents";

const Home: NextPage = () => {
  return (
    <div>
      <HeaderMain />
      <SunnyWatt />
      <Footer />
    </div>
  );
};

export async function getStaticProps({ locale }: any) {
  return {
    props: {
      ...(await serverSideTranslations(locale, ["Home"])),
    },
  };
}

export default Home;
```

Лістинг pageComponents/SignIn/SignIn.component.ts:

```
import Link from "next/link";
import { useState } from "react";
import { ButtonForgoted, ButtonSignUp, SignInStyled, SignInTitle, Wrapper } from
"./SignIn.styles";

import { Container } from "../../components";
import { useTranslation } from "next-i18next";
import { SignInProps } from "./SignIn.types";

import { PreLogin } from "./PreLogin.component";
import { StepSigninSecond } from "./StepSigninSecond.component";

export const SignIn: React.FC<SignInProps> = () => {
  const { t } = useTranslation("SingIn");

  const [step, setStep] = useState<"first" | "second">("first");
```

```
  return (
    <Container variants="small">
      <SignInStyled>
        <Wrapper>
          <SignInTitle variants="headline2">{t("SignInTitle")}</SignInTitle>

          {step === "first" && <PreLogin t={t} changeStep={() => setStep("second")} />}
          {step === "second" && <StepSigninSecond t={t} />}

          <Link href="/signup">
            <ButtonSignUp as="a" color="#00439F" variants="text">
              {t("ButtonSignUp")}
            </ButtonSignUp>
          </Link>
          <Link href="/password-reset">
            <ButtonForgoted as="a" color="#00439F" variants="text">
              {t("ButtonForgoted")}
            </ButtonForgoted>
          </Link>
        </Wrapper>
      </SignInStyled>
    </Container>
  );
};
```

Лістинг pageComponents/SignIn/PreLogin.component.tsx:

```
import { useState } from "react";
import { useForm, Controller } from "react-hook-form";
import { yupResolver } from "@hookform/resolvers/yup";

import { FormSigninFirst, SigninFirstProps } from "./SignIn.types";

import { useAppDispatch } from "../../hooks/reactRedux";
import { schemePreLogin } from "./validate";

import { apiPreLogin } from "../../services/auth.service";
import { IPreLoginErrors } from "../../types/auth.types";

import {
  SignInInput,
  WrapperInputs,
  ReCAPTCHA,
  ButtonSignIn,
  ErrorFiledSignin,
  SignInPasswordInput,
} from "./SignIn.styles";

export const PreLogin: React.FC<SigninFirstProps> = ({ t, changeStep }) => {
  const dispatch = useAppDispatch();
  const [error, setError] = useState<string | null>(null);
```

```
  const {
    handleSubmit,
    register,
    control,
    formState: { errors },
  } = useForm<FormSigninFirst>({
    mode: "onChange",
    resolver: yupResolver(schemePreLogin),
    defaultValues: {
      recaptcha: "",
      login: "",
      password: "",
    },
  });

  const onSubmit = async ({ recaptcha, ...data }: FormSigninFirst) => {
    const response = await dispatch(apiPreLogin(data));

    if (response.meta.requestStatus === "rejected") {
      const payload = response.payload as IPreLoginErrors;
      setError(payload.Error?.[0]);
    }

    if (response.meta.requestStatus === "fulfilled") {
      changeStep();
    }
  };

  return (
    <>
      <WrapperInputs>
        <SignInInput
          isError={!!errors.login}
          errorMessage={errors.login?.message}
          {...register("login")}
          placeholder="E-mail"
        />
        <SignInPasswordInput
          isError={!!errors.password}
          errorMessage={errors.password?.message}
          {...register("password")}
          type="password"
          placeholder={t("SignInInputPasswordPlaceholder")}
        />
      </WrapperInputs>
      {error && <ErrorFiledSignin>{error}</ErrorFiledSignin>}

      <Controller
        name="recaptcha"
        control={control}
        render={({ field }) => <ReCAPTCHA {...field}
sitekey="6LdQwHceAAAAADb_wJVa8g2XalJ9g3YqDDeU1Ztz" />}
```

```
      />

      <ButtonSignIn type="button" as="button" onClick={handleSubmit(onSubmit)}
color="#00439F" variants="contained">
        {t("ButtonSignIn")}
      </ButtonSignIn>
    </>
  );
};
```

Лістинг pageComponents/SignIn/StepSigninSecond.component.tsx:

```
import { useState } from "react";
import { useRouter } from "next/router";

import { useForm, Controller } from "react-hook-form";
import { useAppDispatch } from "../../hooks/reactRedux";
import { FormSigninSecond } from "./SignIn.types";
import { apiLogin } from "../../services/auth.service";
import {
  SignInInput,
  WrapperInputs,
  ButtonSignIn,
  InputCodeWrapper,
  TitleCode,
  InputCode,
  ErrorFiledSignin,
} from "./SignIn.styles";
import { yupResolver } from "@hookform/resolvers/yup";
import { schemeLogin } from "./validate";
import { ILoginErrors } from "../../types/auth.types";
import { apiGetUser } from "../../services/user.service";

export const StepSigninSecond: React.FC<any> = ({ t }) => {
  const router = useRouter();
  const [error, setError] = useState<string | null>(null);
  const dispatch = useAppDispatch();

  const {
    handleSubmit,
    register,
    control,
    formState: { errors },
  } = useForm<FormSigninSecond>({
    mode: "onChange",
    resolver: yupResolver(schemeLogin),
    defaultValues: {
      email: "",
      code: "",
    },
  });
```

```
  const onSubmit = async (data: FormSigninSecond) => {
    const response = await dispatch(apiLogin(data));

    if (response.meta.requestStatus === "rejected") {
      const payload = response.payload as ILoginErrors;
      setError(payload.Error[0]);
    }
    if (response.meta.requestStatus === "fulfilled") {
      await dispatch(apiGetUser());
      router.push("/balance");
    }
  };

  return (
    <>
      <WrapperInputs>
        <SignInInput
          isError={!!errors.email}
          errorMessage={errors.email?.message}
          {...register("email")}
          placeholder="E-mail"
        />
        <InputCodeWrapper>
          <TitleCode variants="headline3">Код подтверждения, полученный по e-
mail</TitleCode>
          <Controller
            name="code"
            control={control}
            render={({ field: { onChange } }) => (
              <InputCode type="text" name="code" inputMode="numeric" onChange={onChange}
fields={4} />
            )}
          />
          {errors?.code && <ErrorFiledSignin>{errors.code.message}</ErrorFiledSignin>}
          {error && <ErrorFiledSignin>{error}</ErrorFiledSignin>}
        </InputCodeWrapper>
      </WrapperInputs>

      <ButtonSignIn type="button" as="button" onClick={handleSubmit(onSubmit)}
color="#00439F" variants="contained">
        {t("ButtonSignIn")}
      </ButtonSignIn>
    </>
  );
};
```

Лістинг pageComponents/SignIn/SignUp.component.tsx:

```
import Link from "next/link";
import { apiRegister } from "../../services/users.services";
import { registerValidate } from "./SignUp.validate";
import { yupResolver } from "@hookform/resolvers/yup";
```

```
import { useForm, Controller } from "react-hook-form";
import { ModalConfirm } from "../../modules";
import { FormRegister } from "./SignUp.types";
import {
  ButtonForgoted,
  ButtonSignIn,
  SignUpInput,
  SignUpStyled,
  SignUpTitle,
  Wrapper,
  WrapperInputs,
  ReCAPTCHA,
  SignUpCheckbox,
  PrivacyPolicy,
  AgreeText,
  Information,
  InformationText,
  SignUpPassword,
} from "./SignUp.styles";

import { Container, Modal } from "../../components";
import { ModalRef } from "../../components/Modal/Modal.types";

import { SignUpProps } from "./SignUp.types";
import { useTranslation } from "next-i18next";
import { AxiosResponse } from "axios";
import { useRef } from "react";
import { useRouter } from "next/router";

export const SignUp: React.FC<SignUpProps> = () => {
  const { t } = useTranslation("SingUp");
  const router = useRouter();
  const controller = useRef<ModalRef>(null);
  const {
    register,
    handleSubmit,
    control,
    setError,
    formState: { errors, isValid },
  } = useForm<FormRegister>({
    mode: "onChange",
    resolver: yupResolver(registerValidate),
    defaultValues: {
      email: "",
      keyword: "",
      password1: "",
      password2: "",
      username: "",
      checked: false,
    },
  });
```

```
  const onSubmit = async ({ email, keyword, password1, password2, username }: FormRegister)
=> {
    try {
      await apiRegister({ email, keyword, password1, password2, username });

      controller.current?.open();
    } catch (error: any) {
      const err = error?.response as AxiosResponse<{ email: string[] }>;

      Object.entries(err.data).map((t) => {
        setError(t[0] as any, {
          message: t[1][0],
        });
      });
    }
  };

  return (
    <Container variants="small">
      <SignUpStyled>
        <Wrapper onSubmit={handleSubmit(onSubmit)}>
          <SignUpTitle variants="headline2">{t("SignUpTitle")}</SignUpTitle>
          <WrapperInputs>
            <SignUpInput
              isError={!!errors.username}
              errorMessage={errors.username?.message}
              {...register("username")}
              placeholder={t("SignUpNickNamePlaceholder")}
            />
            <SignUpInput
              isError={!!errors.email}
              errorMessage={errors.email?.message}
              {...register("email")}
              placeholder="E-mail"
            />
            <SignUpPassword
              type="password"
              isError={!!errors.password1}
              errorMessage={errors.password1?.message}
              {...register("password1")}
              placeholder={t("SignUpPasswordPlaceholder")}
            />
            <SignUpPassword
              type="password"
              isError={!!errors.password2}
              errorMessage={errors.password2?.message}
              {...register("password2")}
              placeholder={t("SignUpConfirmPasswordPlaceholder")}
            />
            <SignUpInput
              isError={!!errors.keyword}
              errorMessage={errors.keyword?.message}
```

```
            {...register("keyword")}
            placeholder={t("SignUpControlWordPlaceholder")}
          />
        </WrapperInputs>
        <Controller
          name="recaptcha"
          control={control}
          render={({ field }) => <ReCAPTCHA {...field}
sitekey="6LdQwHceAAAAADb_wJVa8g2XalJ9g3YqDDeU1Ztz" />}
        />
        <ButtonSignIn disabled={!isValid} as="button" type="submit" color="#00439F"
variants="contained">
          {t("ButtonSignInRegister")}
        </ButtonSignIn>
        <Controller
          name="checked"
          control={control}
          render={({ field: { onChange } }) => (
            <SignUpCheckbox onChange={onChange} type="checkbox">
              <AgreeText variants="subtitle1">
                {t("AgreeText")}
                <Link href="/privacy-policy" passHref>
                  <PrivacyPolicy>{t("AgreeTextLink")}</PrivacyPolicy>
                </Link>
              </AgreeText>
            </SignUpCheckbox>
          )}
        />
        <Link href="/signin" passHref>
          <ButtonSignIn as="a" color="#00439F" variants="text">
            {t("ButtonSignIn")}
          </ButtonSignIn>
        </Link>
        <Link href="/password-reset" passHref>
          <ButtonForgoted color="#00439F" variants="text">
            {t("ButtonForgoted")}
          </ButtonForgoted>
        </Link>
      </Wrapper>
    </SignUpStyled>
    <Information>
      <InformationText variants="body1">{t("InformationText")}</InformationText>
    </Information>
    <Modal ref={controller} closeOverlaid={() => router.push("/signin")}>
      <ModalConfirm
        title="Аккаунт создан!"
        text="Подтвердите ваш аккаунт через свою почту "
        seconderyButtonText="OK"
        seconderyHandleClick={() => controller.current?.close()}
      />
    </Modal>
  </Container>
```

```
  );
};
```

Лістинг pageComponents/SignIn/Balance.component.tsx:

```
import { useEffect, useRef, useState } from "react";
import { Container, Loader } from "../../components";
import { apiGetBalance } from "../../services/balance.services";
import {
  WafeSVG,
  FlexWrapper,
  BalanceStyled,
  BalanceCardInfo,
  BalanceWalletSWH,
  BalanceWalletUSDT,
  BalanceBuySwh,
  BalanceSwhPrice,
  BalanceBuyTitle,
  BalanceSWH,
  BalanceWrapperTitle,
  BuySVG,
  BalanceBuyText,
  BalanceCreateOrder,
  TableOrderBody,
  TableOrderHead,
  TableOrderRow,
  TableOrder,
  TableOrderBodyNumberOrder,
  TableOrderBodySWHT,
  TableOrderHeadNumberOrder,
  TableOrderHeadSWHT,
  BalanceInner,
  TableOrderTitle,
  TableOrderDesc,
  TableWrapper,
  BalanceFlexWrapper,
  SWH,
  USDT,
  Wrapper,
  CreateOrderWrapper,
  Empty,
} from "./Balance.styles";

import { BalanceProps } from "./Balance.types";

import { AddBalance } from "../../modules";
import { Modal } from "../../components";
import { ModalRef } from "../../components/Modal/Modal.types";
import { useTranslation } from "next-i18next";

import { useAppDispatch, useAppSelector } from "../../hooks/reactRedux";
import findWallet from "../../utils/findWallet";
```

```
export const Balance: React.FC<BalanceProps> = () => {
  const { t } = useTranslation("Balance");
  const controller = useRef<ModalRef>(null);
  const [wallet, setWallet] = useState<"SWH" | "USDT">("USDT");
  const dispatch = useAppDispatch();
  const user = useAppSelector(({ user }) => user.user);
  const queueOrders = useAppSelector(({ order }) => order.queueOrders);
  const commissionLimit = useAppSelector(({ balance }) => balance.commissionLimit);

  useEffect(() => {
    dispatch(apiGetBalance());
  }, []);

  const isQueueOrders = !!queueOrders.length;

  return (
    <BalanceStyled>
      <WafeSVG />
      <BalanceInner>
        <Container variants="auth">
          <BalanceFlexWrapper>
            <BalanceWalletUSDT
              title={t("balanceWalletUSDT")}
              balance={user?.sum_usdt}
              currency="USDT"
              colorCurrency="#32A5F9"
              buttonLabel={t("buttonLabel")}
              handleBtnClick={() => {
                setWallet("USDT");
                controller.current?.open();
              }}
            />
            <BalanceCardInfo
              title={t("balanceCardInfoTitleOne")}
              btnLabel={t("btnLabel")}
              currency="USDT"
              balance={user?.available_usdt}
              colorCurrency="#00439F"
              markColor="#52A90D"
              href="/withdrawal-usdt"
            />
            <BalanceCardInfo
              title={t("balanceCardInfoTitleTwo")}
              btnLabel={t("btnLabel")}
              currency="USDT"
              balance={user?.in_orders_usdt}
              colorCurrency="#00439F"
              markColor="#FFD004"
            />
            <BalanceCardInfo
              title={t("balanceCardInfoTitleThree")}
              btnLabel={t("btnLabel")}
```

```
    currency="USDT"
    balance={user?.withdraw_usdt}
    colorCurrency="#00439F"
    markColor="#2D62EA"
   />
  </BalanceFlexWrapper>
  <BalanceFlexWrapper>
   <BalanceWalletSWH
    title={t("balanceWalletUSDT")}
    currency="SWH"
    balance={user?.sum_swht}
    colorCurrency="#FFD004"
    buttonLabel={t("buttonLabel")}
    handleBtnClick={() => {
     setWallet("SWH");
     controller.current?.open();
    }}
   />
   <BalanceCardInfo
    title={t("balanceCardInfoTitleOne")}
    btnLabel={t("btnLabel")}
    currency="SWH"
    balance={user?.available_swht}
    colorCurrency="#FFD004"
    markColor="#52A90D"
    href="/withdrawal-swh"
   />
   <BalanceCardInfo
    title={t("balanceCardInfoTitleTwo")}
    btnLabel={t("btnLabel")}
    currency="SWH"
    balance={user?.in_deposit_swht}
    colorCurrency="#FFD004"
    markColor="#FFD004"
   />
   <BalanceCardInfo
    title={t("balanceCardInfoTitleThree")}
    btnLabel={t("btnLabel")}
    currency="SWH"
    balance={user?.withdraw_swht}
    colorCurrency="#FFD004"
    markColor="#2D62EA"
   />
  </BalanceFlexWrapper>
  <BalanceFlexWrapper>
   <BalanceWalletSWH
    title={t("balanceWalletUSDT")}
    balance={user?.available_btc}
    currency="BTC"
    colorCurrency="#fff"
   />
   <BalanceCardInfo
```

```
            title={t("balanceCardInfoTitleOne")}
            btnLabel={t("btnLabel")}
            currency="BTC"
            balance={user?.available_btc}
            colorCurrency="#313131"
            markColor="#52A90D"
            href="/withdrawal-btc"
          />
          <BalanceCardInfo
            title={t("balanceCardInfoTitleTwo")}
            btnLabel={t("btnLabel")}
            currency="BTC"
            balance="1254,34"
            colorCurrency="#313131"
            markColor="#FFD004"
          />
        </BalanceFlexWrapper>
        <BalanceWrapperTitle>
          <BuySVG />
          <BalanceBuyTitle as="h2" variants="headline2">
            {t("BalanceBuyTitle")} <BalanceSWH>SWH</BalanceSWH>
          </BalanceBuyTitle>
        </BalanceWrapperTitle>
        <FlexWrapper>
          <BalanceBuySwh
            title={t("BalanceBuySwhTitle")}
            currency="SWH"
            quantity="1456/5000"
            info={t("BalanceBuySwhInfo")}
          />
          <BalanceSwhPrice
            title={
              <>
                {t("BalanceSwhPriceTitle")} 1 {<SWH>SWH</SWH>}
              </>
            }
            rate={
              <>
                1 {<SWH>SWH</SWH>} = 34,10 {<USDT>USDT</USDT>}
              </>
            }
            info={
              <>
                {t("BalanceSwhPriceInfoOne")} 9 {<SWH>SWH</SWH>}
{t("BalanceSwhPriceInfoTwo")} 131,04{" "}
                {<USDT> USDT</USDT>}
              </>
            }
          />
          <BalanceBuyText variants="body1">{t("BalanceBuyText")}</BalanceBuyText>
        </FlexWrapper>
```

```
        <Wrapper>
          <TableOrderTitle variants="headline2">{t("TableOrderTitle")}</TableOrderTitle>

          <CreateOrderWrapper>
            <BalanceCreateOrder
              handleClick={() => console.log("click balance create order")}
              btnLabel={t("ButtonBuy")}
              currency="SWH"
              title={t("CreateOrderForBuy")}
              total="101,92"
              totalCurrency="USDT"
              totalText={t("TotalText")}
            />
            <TableWrapper>
              <TableOrder>
                <TableOrderHead>
                  <TableOrderHeadNumberOrder variants="subtitle2">
                    {t("TableOrderHeadNumberOrder")}
                  </TableOrderHeadNumberOrder>
                  <TableOrderHeadSWHT
variants="subtitle2">{t("TableOrderHeadSWHT")}</TableOrderHeadSWHT>
                </TableOrderHead>
                <TableOrderBody isEmpty={!isQueueOrders}>
                  {isQueueOrders ? (
                    queueOrders.map(({ amount_swh, number }) => (
                      <TableOrderRow bgColor="#FDE3E1">
                        <TableOrderBodyNumberOrder
variants="subtitle2">{number}</TableOrderBodyNumberOrder>
                        <TableOrderBodySWHT
variants="subtitle2">{amount_swh}</TableOrderBodySWHT>
                      </TableOrderRow>
                    ))
                  ) : (
                    <Empty variants="headline1"> Пусто </Empty>
                  )}
                </TableOrderBody>
              </TableOrder>
            </TableWrapper>
          </CreateOrderWrapper>
          <TableOrderDesc variants="body1">
            {t("TableOrderDesc")} <span>{t("TableOrderDescSpan")}</span>
          </TableOrderDesc>
        </Wrapper>
      </Container>
    </BalanceInner>
    <Modal ref={controller}>
      <AddBalance
        handleClick={() => controller.current?.close()}
        linkPurse={findWallet(user?.wallets, wallet as string) as string}
        currency={wallet}
        purseCommission={wallet === "USDT" ? commissionLimit?.no_percent_usdt :
commissionLimit?.no_percent_swh}
```

```
      />
    </Modal>
  </BalanceStyled>
 );
};
```

Лістинг pageComponents/SignIn/Commision.component.tsx:

```
import { Container, OutputCard } from "../../components";
import { Text, Title, CommisionWrapper, CommisionItem } from "./Commision.styles";
import { useTranslation } from "next-i18next";

import { data } from "./Commision.data";

export const Commision = () => {
 const { t } = useTranslation("Commision");
 return (
  <Container variants="main">
   <Title variants="headline1">{t("CommisionTitle")}</Title>

   {data.map((item) => {
    if (typeof item === "string") {
     return <Text variants="body1">{t(item)}</Text>;
    }

    if (Array.isArray(item)) {
     return (
      <CommisionWrapper>
       {item.map(({ commision, outputTitle, percent }) => (
        <CommisionItem>
         <OutputCard percent={t(percent)} outputTitle={t(outputTitle)}
commision={t(commision)} />
        </CommisionItem>
       ))}
      </CommisionWrapper>
     );
    }
   })}
  </Container>
 );
};
```

Лістинг pageComponents/SignIn/Deposit.component.tsx:

```
import { useEffect, useRef, useState } from "react";
import {
 DepositStyled,
 WafeSVG,
 DepositInner,
 DepositCardInfo,
 DepositCreate,
```

```
  WrapperFlex,
  DepositTab,
  DepositTabList,
  DepositTabPanel,
  DepositTabs,
  InfoDeposit,
  Icon,
  Text,
} from "./Deposit.styles";
import { DepositProps } from "./Deposit.types";
import { Container } from "../../components";
import { Deposits as TableDeposits } from "../../modules";
import { useTranslation } from "next-i18next";
import { useAppDispatch, useAppSelector } from "../../hooks/reactRedux";

import { apiGetDepositActive, apiDeleteDeposit } from "../../services/deposit.service";

import { Modal } from "../../components/Modal";
import { ModalRef } from "../../components/Modal/Modal.types";
import { ModalConfirm } from "../../modules";
import Link from "next/link";

const DepositActive: React.FC = () => {
  const dispatch = useAppDispatch();
  const [variantModal, setVariantModal] = useState<"success" | "error" | "confirm">();
  const [id, setId] = useState<number>(0);
  const controller = useRef<ModalRef>(null);

  const depositActive = useAppSelector(({ deposit }) => deposit.depositActive);
  useEffect(() => {
    dispatch(apiGetDepositActive());
  }, []);

  const handleDelete = (id: number) => {
    setVariantModal("confirm");
    controller.current?.open();
    setId(id);
  };

  const handleDeleteDeposit = (id: number) => {
    try {
      dispatch(apiDeleteDeposit({ id }));
      setVariantModal("success");
    } catch (error) {
      setVariantModal("error");
      controller.current?.open();
    }
  };

  return (
    <>
      <TableDeposits handleClick={handleDelete} deposits={depositActive} />
```

```
<Modal ref={controller} closeOverlaid={() => setVariantModal("confirm")}>
  {variantModal === "confirm" && (
    <ModalConfirm
      title="Подтвержения снятия депозита"
      text="Депонированные SWH будут немедленно изъяты из депозита и станут доступны для выполнения любых операций. Запись о депозите будет  помещена в раздел Снятые депозиты, где будет находиться до момента получения последних дивидендов. Дивиденды начисляются один раз в месяц (10-го числа). После получения последних дивидендов (за данный месяц, согласно учтенному количеству целых суток) запись о депозите будут убрана из списка Снятые депозиты"
      primaryButtonText="Снять депозит"
      seconderyButtonText="Назад"
      primaryHandleClick={() => handleDeleteDeposit(id)}
      seconderyHandleClick={() => controller.current?.close()}
    />
  )}

  {variantModal === "success" && (
    <ModalConfirm
      title="Депозит снят!"
      text="Депонированные SWH будут немедленно изъяты из депозита и станут доступны для выполнения любых операций. Запись о депозите будет  помещена в раздел Снятые депозиты, где будет находиться до момента получения последних дивидендов. Дивиденды начисляются один раз в месяц (10-го числа). После получения последних дивидендов (за данный месяц, согласно учтенному количеству целых суток) запись о депозите будут убрана из списка Снятые депозиты"
      seconderyButtonText="Ок"
      seconderyHandleClick={() => controller.current?.close()}
    />
  )}

  {variantModal === "error" && (
    <ModalConfirm
      title="Ошибка!"
      text={
        <>
          Операция не выполнена по техническим причинам. Попробуйте обновить страницу и повторить операцию
          несколько позднее. В случае повторения ошибки свяжитесь с администрацией при помощи 
          {
            <Link href="/support" passHref>
              <a>формы обратной связи</a>
            </Link>
          }
        </>
      }
      seconderyButtonText="Ок"
      seconderyHandleClick={() => controller.current?.close()}
    />
  )}
</Modal>
```

```
    </>
  );
};

const DepositRemove: React.FC = () => {
  const dispatch = useAppDispatch();

  const depositActive = useAppSelector(({ deposit }) => deposit.depositActive);
  useEffect(() => {
    dispatch(apiGetDepositActive());
  }, []);

  return <TableDeposits deposits={depositActive} />;
};

export const Deposit: React.FC<DepositProps> = () => {
  const { t } = useTranslation("Deposit");
  const user = useAppSelector(({ user }) => user.user);
  return (
    <DepositStyled>
      <WafeSVG />
      <Container variants="auth">
        <DepositInner>
          <WrapperFlex>
            <DepositCreate />
            <DepositCardInfo
              title={"Всего"}
              markColor="#52A90D"
              currency="SWH"
              colorCurrency="#FFD004"
              balance={user?.sum_swht}
            />

            <DepositCardInfo
              title={"Доступно"}
              markColor="#FFD004"
              currency="SWH"
              colorCurrency="#FFD004"
              balance={user?.available_swht}
            />

            <DepositCardInfo
              title={"В депозитах"}
              markColor="#2D62EA"
              currency="SWH"
              colorCurrency="#FFD004"
              balance={user?.in_deposit_swht}
            />
          </WrapperFlex>

          <DepositTabs>
            <DepositTabList>
```

```
            <DepositTab>{t("DepositTabOne")}</DepositTab>
            <DepositTab>{t("DepositTabTwo")}</DepositTab>
          </DepositTabList>
          <DepositTabPanel>
            <DepositActive />
          </DepositTabPanel>
          <DepositTabPanel>
            <Text variants="body1">
              <Icon />
              {t("DepositTabTextTwo")}
            </Text>
            <DepositRemove />
          </DepositTabPanel>
        </DepositTabs>
        <InfoDeposit />
      </DepositInner>
    </Container>
  </DepositStyled>
 );
};
```

Всі інші файли з програмним кодом можна буде переглянути на магнітному носії.

ДОДАТОК Б

ВІДГУК
керівника економічного розділу
на кваліфікаційну роботу бакалавра
на тему:
" Створення фронтент частини професійного UI інтерфейсу криптовалютного додатку SPA на мові Javascript з використання фреймворку Nextjs "
студента групи 121-18-2 Сергієнка Микити Ігоровича

Керівник економічного розділу **Л. В. Касьяненко**

доцент каф. ПЕП та ПУ, к.е.н

ДОДАТОК В

Перелік файлів на диску

ПЕРЕЛІК ДОКУМЕНТІВ НА МАГНІТНОМУ НОСІЇ

Ім’я файлу	Опис
Пояснювальні документи	
Кваліфікаційна робота Сергієнко М.І.docx	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Кваліфікаційна робота Сергієнко М.І.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Сергієнко М.І.rar	Архів. Містить коди програми і скомпільовану програму
Презентація	
Сергієнко М.І.ppt	Презентація кваліфікаційної роботи