

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

ЯКОВЕНЧУК ВЛАДИСЛАВ ВОЛОДИМИРОВИЧ

Допускається до захисту:

Завідувач кафедри

інформаційних технологій,

кандидат технічних наук, доцент

_____ Т.В. Нескородева

«__» _____ 20__р.

**РОЗРОБКА ВЕБ-ДОДАТКУ ВЕДЕННЯ ФІНАНСОВОГО ОБЛІКУ З
ФУНКЦІЯМИ ВІЗУАЛІЗАЦІЇ ДАНИХ**

Спеціальність 122 Комп'ютерні науки

Кваліфікаційна (бакалаврська) робота

Керівник:

Бабаков Р.М., доцент

кафедри інформаційних технологій

Оцінка: _____ / _____ / _____
(бали за шкалою СКТС/за національною шкалою)

Голова ЕК: _____
(підпис)

Вінниця - 2021

АНОТАЦІЯ

Яковенчук В. В. Розробка веб-додатку ведення фінансового обліку з функціями візуалізації даних. Спеціальність 122 «Комп'ютерні науки», спеціалізація «Інформаційні технології»

Донецький національний університет імені Василя Стуса, Вінниця, 2021.

У кваліфікаційній (бакалаврській) роботі досліджені сучасні методи побудови веб-додатку для керування особистими фінансами. Розгляд аналогів, пошук та вивчення представлених додатків для керування фінансами. Розробка додатку з ціллю надання зручного та швидкого рішення керування особистими фінансами.

Ключові слова: керування фінансами, фінанси, веб-додаток, сучасні технології, розробка.

50 с., 28 рис., 19 джерел.

ABSTRACT

Yakovenchuk V.V. Development of a web application for financial accounting with data visualization functions. Specialty 122 "Computer Science", specialization "Information Technology". Vasyl Stus Donetsk National University, Vinnytsia, 2021.

In the qualification (bachelor's) work the modern methods of building a web application for personal finance management are investigated. Consideration of analogues, search and study of the presented applications for financial management. Development of an application to provide a convenient and fast solution for personal finance management.

Key words: financial management, finance, web application, modern technologies, development.

50 p., 28 draw., 19 source.

ЗМІСТ

ВСТУП	6
Розділ 1.	7
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ, ПОСТАНОВКА ЗАДАЧІ ТА ОГЛЯД ІСНУЮЧИХ АНАЛОГІВ.....	7
1.1. Актуальність теми.....	7
1.2. Розгляд аналогів.....	8
1.3 Постановка задачі	13
Розділ 2.	14
АНАЛІЗ ТА ВИБІР АКТУАЛЬНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	14
2.1 Вибір програмних засобів розробки	14
Розділ 3.	34
ОПИС РОЗРОБКИ ТА ПРОЕКТУВАННЯ ПОСТАВЛЕНОЇ ЗАДАЧІ.....	34
3.1 Загальна структура веб-додатку.....	34
3.2 Розробка веб-інтерфейсу.....	35
3.3 Процес розробки	38
3.4 Взаємодія користувача та веб-додатку.....	44
ВИСНОВОК	48
СПИСОК ВИКОРИСТАНИХ ПОСИЛАНЬ.....	49

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

- JS – JavaScript (Мова програмування JavaScript);
- DOM – Document Object Model (Об’єктна модель документа)
- HTML – HyperText Markup Language (Мова розмітки)
- CSS – Cascading Style Sheets (Таблиці стилів)
- API – Application programming interface (Програмний інтерфейс)
- RTA – Real Time Application (Додаток, що працює в режимі реального часу)
- JSON – JavaScript Object Notation (JavaScript об’єкти)
- GIT – Система для розподіленого керування версіями файлів
- WS – WebStorm (Інтегроване середовище розробки)

ВСТУП

Контроль фінансів необхідна річ у житті сучасної людини. Адже важливо не тільки те, скільки ви заробляєте, але й скільки витрачаєте. Фінанси передбачають процес управління грошима, їх оптимізацію та належне розподілення. Ведення особистої бухгалтерії, планування доходів та розходів, не є обов'язковим для кожної людини, але дозволяє більш раціонально використовувати наявні ресурси.

Здається, що управління фінансами це, складне заняття, яке потребує певних навиків та знань. Частково це правда, керувати фінансами навіть малого підприємства неможливо без відповідних знань. Але контролювати та оптимізувати особисті фінанси можливо і з мінімумом знань у даній сфері.

Тема бакалаврської роботи – «Розробка веб-додатку ведення фінансового обліку з функціями візуалізації даних». Дана тема є актуальною оскільки, в сучасному світі необхідно:

1. Чітко розуміти власні доходи та витрати (фінансовий облік).
2. Навчитись планувати власні доходи та витрати (фінансове планування).
3. Навчитись накопичувати капітал (інвестиції).

Мета бакалаврської роботи – створення веб-додатку для фінансового обліку. За допомогою даного додатку користувач матиме можливість ввести облік фінансів:

1. Вносити, зберігати та переглядати інформацію про доходи та витрати.
2. Відслідковувати заощадження.
3. Планувати бюджет.

Розділ 1.

АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ, ПОСТАНОВКА ЗАДАЧІ ТА ОГЛЯД ІСНУЮЧИХ АНАЛОГІВ

1.1. Актуальність теми

Фінансовий облік – це нудна, монотонна, кропітка, але необхідна робота. Її основна задача – ми повинні розуміти звідки та скільки грошей отримуємо, куди та скільки коштів витрачаємо. І далі, аналізуючи цю інформацію потрібно навчитись оптимізувати витрати.

Є декілька способів ведення обліку, наприклад – на папері, в зошиті. Цей спосіб не потребує наявності комп'ютера, чи будь яких інших девайсів, не потрібно вивчати інтерфейс програми. Але даний спосіб не зручний, велика вірогідність помилок, потребує великої кількості час для ведення обліку та аналізу.

Облік у таблицях MS Excel дозволяє не купувати і не встановлювати спеціалізовані програми. Можна швидко налаштувати під свої потреби. Але складно різнобічно підійти до обліку особистих фінансів, налаштовувати залежності між таблицями, легко допустити помилку. Крім того, складно аналізувати дані. Цей спосіб найбільш актуальний для ведення спрощеного обліку особистих фінансів.

Облік у спеціалізованих програмах дозволяє проводити різнобічний облік і аналіз особистих фінансів. Цей спосіб потребує мінімальної кількості часу, від вас потрібно тільки вчасно вносити необхідні дані та аналізувати результати. Вся логіка уже продумана розробниками.

Використання будь якого з цих способів набагато краще, ніж не слідкувати за фінансами взагалі. Але все ж найкраще, швидше та ефективніше використовувати спеціальні програми.

1.2. Розгляд аналогів

Сьогодні на ринку достатньо багато програмних продуктів, які можуть допомогти у обліку та контролі власних фінансів, в цьому розділі, я наведу найвідоміші з них.

1.2.1 CoinKeeper

Цікаве та одне з найпопулярніших рішень для контролю сімейного бюджету, управління доходами, витратами і фінансами.

Додаток веде аналіз фінансів на підставі покупок і доходів користувача. Ліміти витрат по місяцях, тижнях або кварталах допомагають вчасно зупинитися, ініціюючи відправку періодичних повідомлень. Дозволяє отримувати розширену статистику витрат і доходів, залишати коментарі, мітки і складати графіки. Додавання категорій і підкатегорій витрат відбувається без обмежень.

Розширена версія дозволяє вести спільний облік з різних пристроїв - своєрідний аналог сімейної підписки.

Переваги Coinkeeper:

1. можливість настройки лімітів за категоріями витрат;
2. можливість введення будь-якої валюти;
3. автоматичний імпорт банківських операцій;
4. контроль витрат декількох користувачів;
5. кроссплатформенність.

Серед недоліків Coinkeeper можна відмітити що, у безкоштовній версії багато реклами, та значно урізаний функціонал.

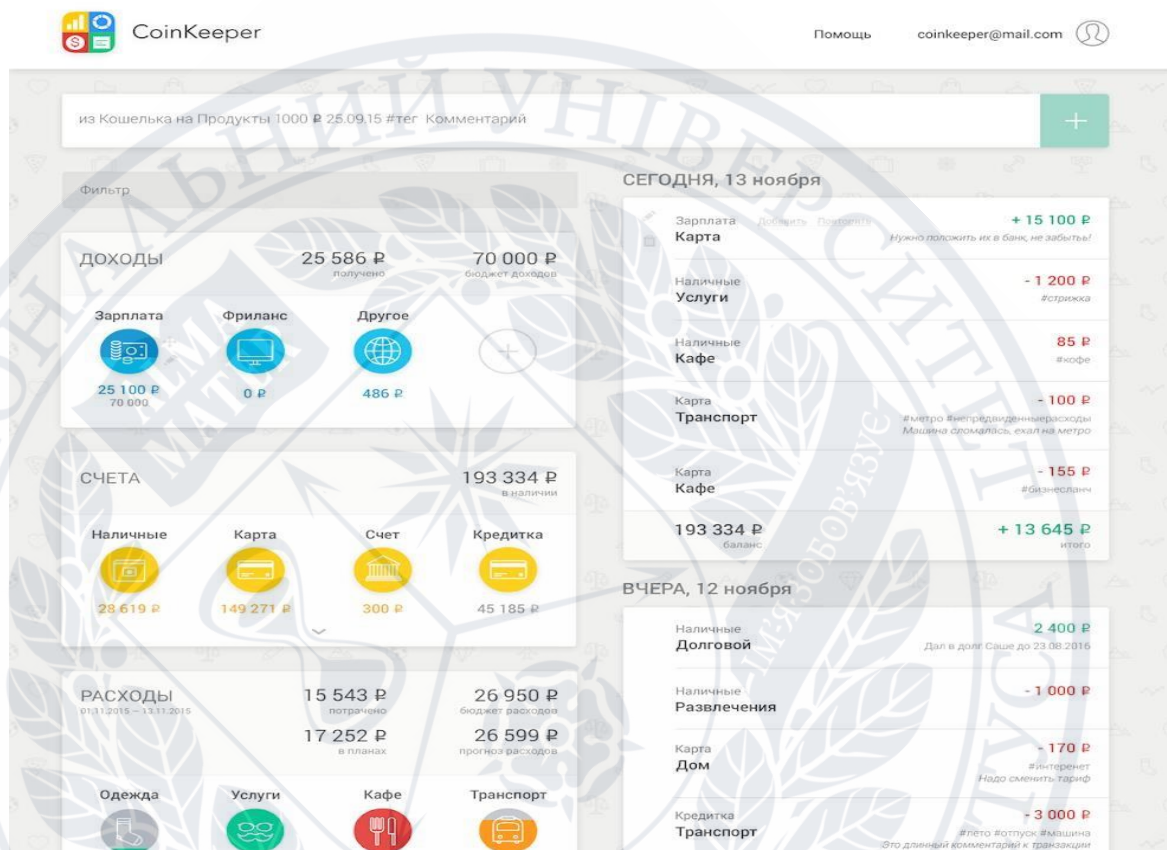


Рисунок. 1.1 – головна сторінка Coinkeeper.

1.2.2 Money Lover

Додаток фіксує боргові зобов'язання, регулярні виплати і своєчасно нагадує про необхідність здійснити черговий платіж.

В налаштуваннях зазначаються бажані фінансові результати, до яких ви будете прагнути. Це дозволяє контролювати свій бюджет, ощадні рахунки і уникати всіляких фінансових проблем. У преміум-акаунті є опція експорту даних в Excel і відключена реклама.

Переваги Money Lover:

1. безшовна інтеграція між пристроями, платформами;
2. нагадування про оплату рахунків;
3. відстеження боргових зобов'язань;
4. наявність різних режимів (наприклад, туристичний для мандрівників);
5. зручна візуалізація;
6. налаштування повідомлень про перевищення лімітів.

Недоліки Money Lover:

1. всі дрібні операції вносяться вручну;
2. висока вартість преміум-версії.(550 грн)



Рисунок. 1.2 – інтерфейс Money Lover.

1.2.3 Finkee

Додаток досить зручний, хоча введення транзакції краще в CoinKeeper. До нього легко адаптуватися, якщо регулярно хоча б днів 5-10 вносити дані. Головний плюс - повністю безкоштовний річний облік, без будь-яких обмежень. Інший плюс програми - можливість входу через соц.мережі.

Важлива особливість, якої немає в аналогічних програмах, - можна вести облік не тільки в будь-якій світовій валюті (рублі, євро, долари, гривні, тощо), але і в криптовалюті. На українському ринку додатків для обліку фінансів такого не було.

Інша корисна функція - регулярні транзакції. Ви вибираєте типи платежів, які проходять регулярно через фіксований інтервал часу, і програма автоматично їх спише, коли потрібно.

У Finkee основна робоча версія - безкоштовна і в ній доступні всі важливі функції, крім синхронізації карт. Синхронізацію можна підключити, сплативши підписку вартістю 2,5 долара на місяць. Чи пробний період - 30 днів. Поки що доступна інтеграція тільки з найбільшим банком України - ПриватБанком. Причина проста - послугами цієї компанії користується більшість українців. І це важливо, тому що в інших додатках додати карту українського банку не завжди можливо. Незабаром розробники випустять оновлення, де список доступних банківських установ буде більше.

Інші переваги:

1. Можливість в безкоштовній версії створювати необмежену кількість фінансових акаунтів, категорій і тегів . Це дозволяє дуже сильно деталізувати облік, навіть без покупки платної підписки.
2. На відміну від більшості додатків, в безкоштовній версії ви можете використовувати додаток з будь-якої кількості пристроїв одночасно .
Всі пристрої будуть синхронізовані між собою.
3. Алгоритм для визначення категорій витрат, який підлаштовується під користувача (програма запам'ятовує які категорії ви вказуєте для транзакцій).
4. Можливість гнучко створювати бюджети , не витрачаючи на це багато часу (інтерфейс ще потребує доопрацювання).
5. Інформативний графік для визначення динаміки витрат і доходів. А також кругова діаграма, яка показує куди йдуть гроші.

Звичайно, порівнювати гігантів ринку CoinKeeper і Spendee з молодим українським проектом не зовсім коректно, але Finkee має великий потенціал для розвитку.

Якщо ж порівнювати її з іншими додатками популярними в Україні, то варто зауважити, що Nomemoney (колись популярний додаток) перестало активно розвиватися. Українці також користуються Monefy (дуже схоже на оновлену версію Spendee) і YNAB (шикарний функціонал, але мало поширене і немає банківських інтеграцій).

Finkee однозначно варто приділити увагу , особливо якщо ви тільки починаєте вести облік своїх фінансів.

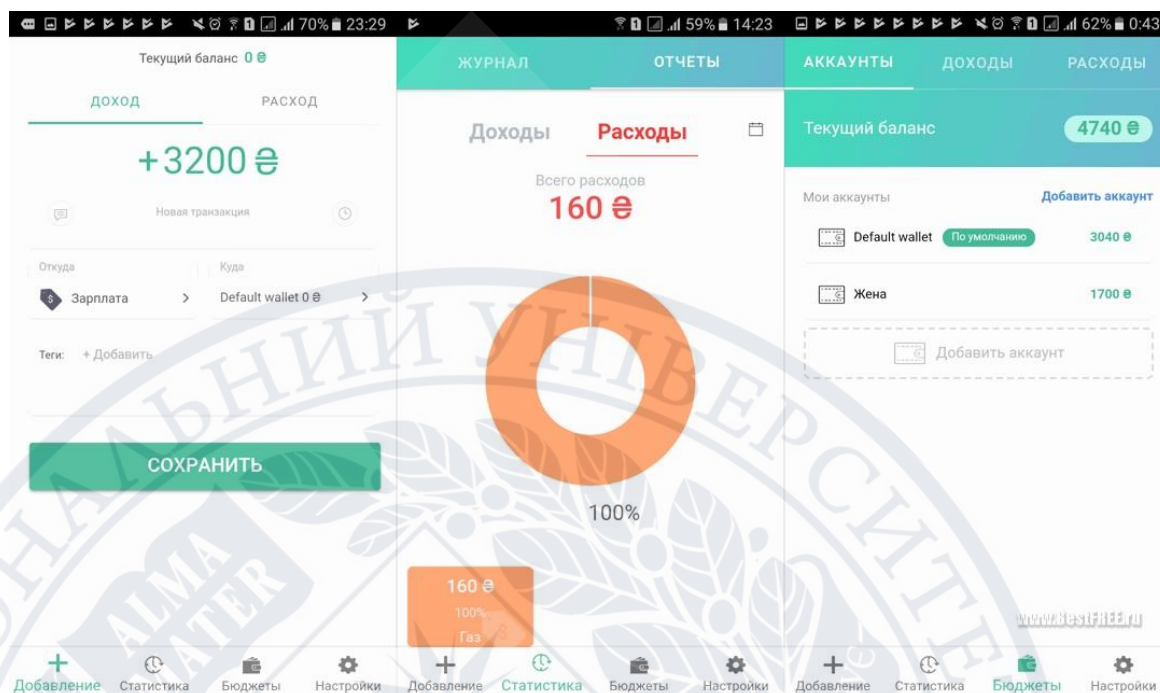


Рисунок. 1.3 – інтерфейс Finkee.

1.3 Постановка задачі

Мета бакалаврської роботи полягає в аналізі існуючих аналогів програм для особистого фінансового обліку, пошуку та вирішення їх проблем та створення веб-додатку “Ведення фінансового обліку з функціями візуалізації даних”.

Для досягнення визначеної мети необхідно:

- проаналізувати сучасні аналоги;
- визначити їх недоліки;
- пошук рішення актуальних проблем додатків фін.обліку;
- визначити набір технологій для створення додатку;
- аргументувати вибір технологій;
- розробити повноцінний веб-додаток з базою даних.

Розділ 2.

АНАЛІЗ ТА ВИБІР АКТУАЛЬНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Вибір програмних засобів розробки

В даному розділі переглянуті програмні засоби, за допомогою яких розроблявся додаток: JavaScript, Vue.js, Firebase, HTML, CSS, Git. В якості середовища розробки використовувався WebStorm.

2.1.1. JavaScript

JavaScript (JS) — динамічна, прототипна, об'єктно-орієнтована мова програмування. Одна з реалізацій стандарту ECMAScript.

JavaScript найчастіше використовується для створення сценаріїв вебсторінок. Це надає можливість на боці клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, асинхронно обмінюватися даними з сервером, керувати браузером, змінювати зовнішній вигляд та структуру вебсторінки.

JavaScript класифікують як скриптову мову програмування з динамічною типізацією. Також вона є прототипною (підмножина об'єктно-орієнтованої). JavaScript також підтримує інші парадигми програмування (частково функціональну та імперативну) і деякі архітектурні властивості. Зокрема:

автоматичне керування пам'яттю, прототипне наслідування, динамічна та слабка типізація, функції як об'єкти першого класу.

JavaScript має певні властивості об'єктно-орієнтованої мови. Але завдяки концепції прототипів, має іншу підтримку об'єктів, ніж традиційні мови ООП.

Також, JavaScript має деякі властивості, притаманні функціональним мовам, зокрема: об'єкти як списки, каррінг (currying), функції як об'єкти першого рівня, замикання (closures), анонімні функції — це надає мові додаткову гнучкість.

Мова програмування JavaScript має С-подібний синтаксис, але є корінні відмінності з мовою С:

1. об'єкти мають можливість динамічної зміни типу, завдяки механізму прототипів.
2. функції, це об'єкти першого класу.
3. можливість обробки винятків.
4. можливість автоматичного приведення типів.
5. можливість інтроспекції .
6. анонімні функції.

На даний момент JavaScript це єдина високорівнева, динамічна мова програмування, доступна практично скрізь. В першу чергу на веб-сторінках. Завдяки цьому є можливість вивчити всього одну мову, частково використати напрацювання для клієнта та сервера, для інтернету речей, машинного навчання та в робото-будуванні.

Мова JavaScript використовується для:

1. написання сценаріїв вебсторінок, надання їм інтерактивності.
2. створення односторінкових та прогресивних вебзастосунків (React, AngularJS, Vue.js).
3. серверного програмування (Node.js).
4. стаціонарних додатків (Electron, NW.js).
5. мобільних додатків (React Native).
6. в прикладних програмах (наприклад Adobe Creative Suite).
7. в PDF-документах.

```
1 module.exports = class ArrayService {  
2   static binarySearch(array, target) {  
3     let left = 0  
4     let right = array.length - 1  
5  
6     while (left <= right) {  
7       const middle = Math.floor((right + left) / 2)  
8       if (array[middle] > target) {  
9         right = middle - 1  
10      } else if (array[middle] < target) {  
11        left = middle + 1  
12      } else {  
13        return middle  
14      }  
15    }  
16  
17    return null  
18  }  
19 }
```

Рис. 2.1 – Приклад JavaScript коду

JavaScript відомий великою кількістю бібліотек (фреймворків).

Фреймворк (каркас) – програмна платформа, яка визначає структуру програмної системи. Спрощує розробку та об'єднання різних компонентів великого програмного проекту. Основною різницею між фреймворком та бібліотекою є те, що бібліотека використовується в програмі, як набір підпрограм схожої функціональності, не впливає на архітектуру програмного продукту і не

накладає на неї ніяких обмежень. Фреймворк диктує свої правила побудови архітектури додатку, встановлюючи на початковому етапі розробки поведінку за замовченням – каркас, який потрібно розширювати та змінювати згідно з вказаними вимогами.

На відміну від бібліотеки, яка об'єднує в собі набір схожої функціональності, фреймворк містить велику кількість різних по тематиці бібліотек. Більшість веб сайтів використовують сторонні бібліотеки JavaScript.

У даній роботі я буду використовувати один з найпоширеніших фреймворків – Vue.js

У практичній частині цієї роботи JavaScript використовується єдиною мовою програмування. Завдяки розвитку JavaScript ми маємо змогу однією мовою програмування за допомогою різних бібліотек розробляти як клієнтську, так і серверну частину веб-додатку.

2.1.2. Vue.js

Vue.js - JavaScript фреймворк, з відкритим вихідним кодом, для створення користувацьких інтерфейсів. Даний фреймворк легко інтегрується в проекти з паралельним використанням інших JavaScript-бібліотек.

Vue.js може використовуватись як фреймворк для розробки односторінкових веб-додатків в реактивному стилі. На даний момент проект підтримує творець Еваном Ю та інші активні члени основної команди з різних компаній, наприклад: Netlify, Netguru , Baidu , Livestorm. За результатами

опитування, проведеного в 2016 році для JavaScript, показав, що Vue має 89% задоволеності розробників.

В порівнянні з іншими веб-фреймворками, розробники називають Vue.js прогресивним і поступово адаптованим.

Завдяки цьому, в розробника є можливість налаштувати структуру програми відповідно до власних вимог. Vue.js вважається простішим в освоєнні, ніж AngularJS, оскільки його API є набагато простішим в освоєнні. Це дозволяє використовувати тільки знання JavaScript і HTML. Можливе застосування Typescript.

У Vue.js є власна офіційна досить багата документація на багатьох мовах, викладена на vuejs.org, яка може послужити прикладом в поясненні проектування і розробки в браузері. У Vue.js реалізується MVVM шаблон.

MVVM – шаблон програмування, що полегшує відокремлення розробки графічного інтерфейсу від розробки бізнес логіки (бек-енд). Модель представлення це частина, що відповідає за перетворення даних та їх подальшу підтримку і використання.

Завдяки цьому, модель представлення більше схожа на модель, ніж на представлення і оброблює більшість логіки відображення даних. Модель представлення також реалізовує патерн медіатор, організовує доступ до бекенд логіки, навколо множини правил використання.

Vue.js пропонує можливість прив'язки даних на Javascript, так що введення і виведення даних сполучається одразу з джерелом даних. Тому, режим ручного визначення даних (як приклад, через jQuery) з DOM-HTML не потрібен. При

цьому, немає необхідності в жодних додаткових анотаціях, як в Knockout.js, оголошені в Vue-Element змінні JavaScript відносяться до реактивних елементів.

Особливості Vue.js:

Шаблони Vue.js

У Vue використовується синтаксис шаблонів на основі HTML, що дозволяє декларативно зв'язати рендеринг DOM, з основними екземплярами даних у Vue. Усі Vue HTML шаблони валідні, і можуть бути парситись браузером та HTML парсерами. Vue компілює шаблони в рендерингові функції віртуального DOM. В поєднанні з реактивністю, Vue має можливість розумно обчислювати кількість компонентів для ре-рендингу та використовувати мінімальну кількість маніпуляцій з DOM, коли стан застосунку змінюється.

В Vue ви маєте можливість напряду писати рендерингові функції використовуючи JSX або використовувати синтаксис шаблонів. Для того, щоб це виконати, вам потрібно замінити шаблон на рендерингову функцію. Завдяки рендеринговій функції є можливість для потужних патернів базованих на компонентах - наприклад, нова транзитна система тепер повністю базується на компонентах, що використовує рендерингові функції.

Реактивність Vue.js

Серед найвиразніших особливостей Vue є ненав'язлива реактивна система. Моделі це звичайні плоскі об'єкти JavaScript. Завдяки цьому, керування станами є дуже простим та інтуїтивним. Vue надає можливість оптимізованого ре-рендерингу з коробки, без потреби щось робити додатково. Кожен компонент відслідковує свої реактивні залежності під час рендерингу, тому система точно знає коли має відбуватись ре-рендеринг і які компоненти потрібно задіяні.

Ефекти переходу

У Vue.js присутні різноманітні шляхи застосування ефектів переходу, коли елемент оновлюють, додають або видаляють з DOM.

Наприклад:

1. автоматичне застосування класів для CSS переходів та анімацій
2. інтеграція сторонніх бібліотек CSS анімацій, наприклад, Animate.css
3. використання JavaScript для маніпуляцій з DOM напряму, під час переходів
4. інтеграція сторонніх JavaScript бібліотек для анімацій, наприклад, Velocity.js

Коли згорнутий перехідний елемент компоненту видаляють чи вставляють, відбувається наступне:

1. Автоматична перевірка чи має застосовуватись до даного елемента CSS анімації або переходи. Якщо мають, то CSS класи для переходів будуть додані або видалені у потрібний час
2. Якщо компонент має JavaScript зачіпки, вони зачіпки будуть викликані у потрібний час
3. Якщо CSS переходи/анімації були відсутні та ніяких JavaScript не було надано, операції DOM для додавання і видалення відбудеться одразу ж, в наступному фреймі.

Роутинг Vue.js

Vue не включає роутингу, але є пакет vue-router, який дозволяє вирішити це питання. vue-router містить підтримку зв'язування вкладених шляхів з вкладеними компонентами та пропонує деталізований контроль переходів. Vue дозволяє створювати додатки завдяки компонентам. Якщо додати vuерouter, все що потрібно буде зробити це зв'язати ваші компоненти та роути. Також дозволити vue-router вирішувати де їх рендерити.



```
<script>
import Home from '~/views/Home';
import About from '~/views/About';

const { VUE_APP_MODE } = process.env;

export default {
  data() {
    return {
      navbarTitle: `App.vue`,
    };
  },
  methods: {
    goToHomePage() {
      this.$navigateTo(Home);
    },
    goToAboutPage() {
      VUE_APP_MODE == 'web' ? this.$router.push('about') :
      this.$navigateTo(About);
    }
  }
};
</script>
```

Рисунок. 2.2 – приклад Vue.js коду.

В розробці веб-додатку Vue.js використовується основним фреймворком для створення користувацького інтерфейсу, який легко масштабується. Також

будуть використані наступні бібліотеки даного фреймворку: `vue-cli`, `vue-router`, `vue-validate`, `vue-chartjs`.

Vue CLI

Vue CLI є повноцінною системою для швидкої розробки на Vue.js, та надає:

1. Графічний користувацький інтерфейс для повноцінного створення та управління проектами.
2. Інтерактивне створення проекту через `@vue/cli`.
3. Прискорене прототипування через `@vue/cli+ @vue/cli-service-global`, без зміни конфігурацій.
4. Велика колекція офіційних плагінів, інтегруючих кращі інструменти екосистеми фронтенда.

Vue CLI сповнений прагненням стати стандартним інструментарієм в екосистемі Vue. Завдяки йому забезпечується безперебійна робота різних інструментів збірки, встановлюються розумні значення за замовчуванням. Це дає можливість зосередитися на розробці програми, а не витратити час на його налаштування. У той же час, присутня гнучкість налаштування конфігурації кожного з інструментів без необхідності створення конфігурації в окремому файлі.

Vue Router

Vue Router це офіційна бібліотека маршрутизації, яка глибоко інтегрується з Vue.js і надає можливість, легко створювати SPA-додатки.

Включає такі можливості:

1. Вкладені маршрути / представлення
2. Модульна конфігурація маршрутизатора
3. Доступ до параметрів маршруту, запиту, символів підстановки
4. Анімація переходів представлень на основі Vue.js
5. Зручний контроль навігації
6. Автоматичне проставлення активного класу CSS для посилань

2.1.3. Firebase

Firebase - це американська компанія яка спеціалізується на постачанні хмарних послуг. Заснована в 2011 році Ендрю Лі і Джеймсом Темпліном. В 2014 році була поглинена корпорацією Google .

Основний сервіс Firebase - це хмарна СУБД NoSQL класу, яка дозволяє розробникам додатків синхронізувати і зберігати дані між декількома клієнтами. Підтримуються інтеграції з додатками під операційні системи Android і iOS, також, реалізовано API для додатків на JavaScript, Java, ObjectiveC і Node.js, є можливість працювати безпосередньо з базою даних в стилі REST з ряду JavaScript-фреймворків, таких як, AngularJS , React , Vue.js , Ember.js і Backbone.js. Також, передбачено API для шифрування даних.

Серед інших послуг компанії - запусканий 13 травня 2014 року хостинг, який дозволяє зберігання статичних файлів (наприклад, CSS , HTML , JavaScript),

забезпечує доставку через CDN та сервіс аутентифікації клієнта з використанням коду винятково на стороні клієнта та підтримкою входу через Facebook, GitHub, Google (Firebase Simple Login).

Також, компанією був випущений, під ліцензією MIT, веб-редактор коду Firepad, який забезпечує спільну одночасну роботу декільком користувачам з одним документом, який став основою для редакторів Stash Realtime Editor фірми Atlassian та Koding. Ще серед вільних проєктів компанії присутній безкоштовний месенджер Firechat.

Служби і рішення для розробки:

1. Firebase Analytics — безкоштовне рішення для аналітики, що надає змогу ознайомитись із використанням додатків та залученням користувачів.
2. Firebase Cloud Messaging — крос-платформове рішення для надсилання повідомлень і нотифікацій в Android, iOS та вебзастосунках..
3. Firebase Authentication — служба, що дозволяє аутентифікувати користувачів, використовуючи код лиш на стороні клієнта. Підтримує соціальні логін-провайдери Twitter, GitHub і Google (та Google Play Games). Окрім того, включає в себе систему управління користувачами, за допомогою якої розробники мають можливість увімкнути автентифікацію користувача за допомогою електронної пошти та пароля, які зберігаються в Firebase.
4. FRD - Firebase Realtime Database – служба, що надає в режимі реального часу базу даних та бекенд. Ця служба надає розробникам

застосунків API, яке дозволяє синхронізовувати дані додатків між клієнтами та зберігати їх у Firebase хмарі. Також компанія надає клієнтські бібліотеки, які дозволяють інтеграцію із застосунками Android, iOS, JavaScript. Також доступна база даних через REST API та прив'язки до сценаріїв JavaScript, таких як AngularJS, React та Backbone.js. REST API використовує протокол подій з сервером, який є інтерфейсом для створення з'єднань HTTP, отримання pushповідомлень. Розробники, які використовують Firebase Realtime Database, мають можливість захищати свої дані за допомогою правил безпеки, які застосовуються на сервері.

5. Firebase Hosting — статичний і динамічний веб-хостинг. Було запуснено 2014 року 13 травня. Хостинг має підтримку статичних файлів, наприклад: CSS, HTML, JavaScript та інші, також має динамічну підтримку Node.js, через Cloud Functions.

Хостинг передає файли через (CDN) за допомогою HTTPS протоколу та SSL шифрування.

6. Firebase Storage надає можливість надійного завантаження та вивантаження файлів в застосунках Firebase, незалежно від якості інтернет мережі. Ви можете використовувати його для зберігання аудіо-, відео-, зображень чи будь якого іншого вмісту, створеного користувачами. Підтримується Google Cloud Storage.

7. Kit ML — мобільна система для машинного навчання, яка була запуснена 8 травня 2018 року в режимі бета-тестування. Kit ML API містить різні інструменти, серед яких: сканування баркодів, розпізнавання облич, розпізнавання тексту, створення опису для зображень та розпізнавання наземних об'єктів. На даний момент вона

доступна для Android та iOS розробників. Має можливість імпорту власних моделей TensorFlow. API можна використовувати у хмарі або пристрої.

Firebase Authentication

В більшості додатків є необхідність знати особистість користувача. Знання цього дозволяє додатку безпечно зберігати призначені для користувача дані і забезпечувати однакову, синхронізовану роботу на всіх пристроях користувача.

Authentication надає серверні служби, які прості у використанні SDK та готові бібліотеки призначені для користувача інтерфейсу і аутентифікації користувачів у додатку. Firebase Authentication підтримує аутентифікацію з використанням номерів телефонів, паролів, популярних постачальників посвідчень, таких як Facebook, Google та інших.

Authentication інтегрується з іншими сервісами Firebase та використовує галузеві стандарти, такі як OAuth 2.0 і OpenID Connect, завдяки цьому є можливість легко інтегрувати з вашим призначенням для користувача сервером.

Основні переваги:

1. Легкий вхід з будь-якої платформи

Firebase Authentication створена для спрощення створення безпечних систем аутентифікації. При цьому покращуючи вхід в систему та адаптацію для кінцевих користувачів. Забезпечує наскрізне рішення ідентифікації, аутентифікацію по телефону, підтримує облікові записи електронної пошти та паролів, а також вхід в Google, Twitter, GitHub та інші.

2. Гнучкий, оперативний призначений для користувача інтерфейс

FirebaseUI надає рішення для аутентифікації яке обробляє потоки, призначеного для користувача інтерфейсу входу користувачів.

FirebaseUI Auth реалізовує передові методи аутентифікації на мобільних пристроях та веб-сайтах, що збільшує конверсію входу і реєстрації для вашого додатку.

3. Комплексна безпека

Firebase Security, створена тією ж командою, яка розробила Google Sign-in, Chrome Password Manager і Smart Lock. Використовує внутрішній досвід Google для управління однієї з найбільших баз даних облікових записів.

4. Швидка реалізація

Визначення власної системи аутентифікації займає місяці, і для її обслуговування в майбутньому буде необхідна команда інженерів. Завдяки даному сервісу є можливість налаштування всієї системи аутентифікації вашої програми, використовуючи менше 10 рядків коду, навіть під час обробки складних випадків, наприклад, злиття облікових записів.

Firestore Hosting

Firestore Hosting надає швидкий та безпечний хостинг для вашого вебдодатку, статичного і динамічного контенту і мікросервісів.

Firestore Hosting - це хостинг веб-контенту виробничого рівня для розробників. За допомогою однієї команди ви можете швидко розгорнути вебдодатки і обслуговувати як статичний, так і динамічний контент в глобальній CDN (мережі доставки контенту). Також можливо пов'язати хостинг Firestore з хмарними функціями або Cloud Run для створення і розміщення мікросервісів в Firestore.

Ключові можливості:

1. Обслуговування контенту через безпечне з'єднання.
2. Розміщення статичного і динамічного контенту, а також мікросервісів.
3. Є можливість емулювати та навіть ділитися своїми змінами перед запуском.
4. Розгортається за допомогою однієї команди.

2.1.4. Materialize.css

Materialize - це бібліотека компонентів інтерфейсу, створена за допомогою CSS, JavaScript та HTML. Матеріалізація компонентів інтерфейсу допомагає створювати функціональні, привабливі та послідовні веб-сторінки, а також веб-програми, дотримуючись сучасних принципів веб-дизайну, наприклад як, портативність браузера, незалежність пристроїв. Це допомагає створювати швидші та красиві веб-сайти. Він натхненний Google Material Design.

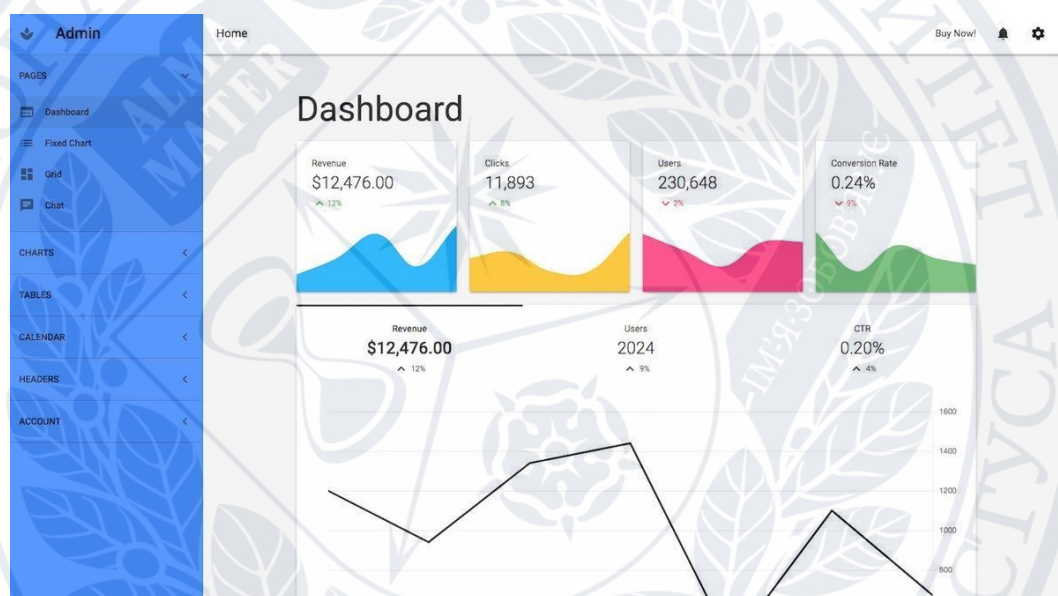


Рисунок. 2.3 – приклад макету та дизайну Materialize.css

2.1.5. GitHub

GitHub - один з найбільших веб-сервісів для хостингу ІТ-проектів та їх спільної розробки.

GitHub оснований на системі контролю версій Git, та розроблений на Ruby on Rails. Сервіс є безкоштовним для ІТ-проектів з відкритим вихідним кодом і невеликих приватних, надаючи їм всі можливості. Для великих корпоративних проектів пропонуються різні тарифні плани.

Окрім розміщення коду, в середовищі GitHub учасники можуть коментувати правки один одного, спілкуватися а також стежити за новинами знайомих.

Завдяки широким можливостям Git, програмісти мають можливість об'єднувати свої репозиторії. GitHub пропонує зручний інтерфейс для цього і відображає внесок кожного учасника у вигляді дерева .

Для проектів є невеликі Вікі, особисті сторінки та система стеження за помилками.

Прямо на сайті є можливість переглядати файли проектів з підсвічуванням синтаксису.

- Є можливість створювати приватні репозиторії. Вони будуть доступні тільки вам і вибраним користувачам. Раніше ця можливість була платною.
- Можна додавати нові файли напряму, через веб-інтерфейс сервісу .
- Код проектів можна копіювати через Git або скачати у вигляді архіва з сайту.

- Також, сервіс підтримує редагування та отримання коду через SVN і Mercurial.
- На сайті є сервіс pastebin для швидкої публікації коду.

Алгоритм початку роботи з GitHub:

1. Необхідно встановити git та створити обліковий запис GitHub;
2. Створити локальне сховище;
3. Додати новий файл в репозиторій;
4. Створити нову гілку git;
5. В GitHub створити нове сховище;
6. Вставити гілку в GitHub;
7. Створити запит для витягу коду;
8. Тепер отримайте зміни з GitHub назад до свого комп'ютера.

При розробці GitHub зазвичай використовується для зручного зберігання коду та перегляду будь-якої інформації про зміну коду.

2.1.6. WebStorm

WebStorm від JetBrains — це інтегроване середовище для розробки, створене для мови програмування JavaScript, від компанії JetBrains. Є розробленою на основі платформи IntelliJ IDEA. WS є спеціалізованою версією PhpStorm, надаючи частину його можливостей. WS постачається з одразу встановленими плагінами JavaScript (такі як для Node.js), які є доступні для PhpStorm безкоштовно.

WebStorm підтримує наступні мови програмування: JavaScript, TypeScript, CoffeeScript, та Dart.

WS забезпечує аналіз коду на льоту, автодоповнення, рефакторинг, інтеграцію з системами управління версіями, навігацію по коду. Дуже важливою перевагою даного середовища розробки є робота з проектами. Підтримується множинна вкладеність — в таких конструкціях підтримується коректний рефакторинг.

Серед основних можливостей:

1. Можливість модифікації файлів .css, .html, .js з одночасним переглядом результатів (Live Edit)
2. підтримка HTML5.
3. підтримка Node.js.
4. Можливості Zen Coding і Emmet.
5. Налаштування коду на JavaScript.
6. Віддалене розгортання по протоколам FTP, SFTP, на монтованих мережних дисках і т. Д. З можливістю автоматичної синхронізації.

7. Інтеграція з системами управління версіями: Subversion , Git , GitHub, Perforce , Mercurial , CVS підтримуються з коробки з можливістю створення списків змін і відкладених змін.
8. Інтеграція з системами стеження за помилками.



Розділ 3.

ОПИС РОЗРОБКИ ТА ПРОЕКТУВАННЯ ПОСТАВЛЕНОЇ ЗАДАЧІ

3.1 Загальна структура веб-додатку

В загальному веб-додаток виглядає наступним чином.



Рисунок. 3.1 – головна сторінка додатку

На головній сторінці знаходиться «шапка» сайту. В шапці вказано актуальний час, ім'я користувача та два впливаючих меню.

З правого боку знаходиться меню, за допомогою якого ми можемо здійснювати навігацію по додатку.

За замовченням, головною сторінкою є розділ Рахунок. В даному розділі відображається інформація про поточний стан рахунку у трьох валютах та актуальний курс валют

Також на головній сторінці є можливість перейти у профіль користувача, для налаштувань, або вийти з облікового запису.

3.2 Розробка веб-інтерфейсу

Весь макет та дизайн створений за допомогою Vue.js та бібліотеки Materialize. Використовуючи фреймворк Vue, є можливість розділити частини користувацького інтерфейсу на компоненти. Завдяки цьому, з'являється можливість легко керувати, редагувати та масштабувати в цілому весь додаток.

```

1  <template>
2    <div class="col s12 m6 l8">
3      <div class="card orange darken-3 bill-card">
4        <div class="card-content white-text">
5          <div class="card-header">
6            <span class="card-title">{{ 'CurrencyAmountTitle' | localize }}</span>
7          </div>
8          <table>
9            <thead>
10             <tr>
11               <th>{{ 'Currency' | localize }}</th>
12               <th>{{ 'CurrencyType' | localize }}</th>
13               <th>{{ 'Date' | localize }}</th>
14             </tr>
15            </thead>
16            <tbody>
17              <tr v-for="cur in currencies" :key="cur">
18                <td>{{ cur }}</td>
19                <td>{{ rates[cur].toFixed(4) }}</td>
20                <td>{{ date | date('date') }}</td>
21              </tr>
22            </tbody>
23          </table>
24        </div>
25      </div>
26    </div>
27  </div>
28 </template>

```

Активуй
Чтобы акти
раздел "Пі

Рисунок. 3.2 – код шаблону головної сторінки створеного за допомогою Materialize

Завдяки Vue.js та Materialize можна значно спростити розробку, адже цей набір технологій дозволяє створити шаблони кожної сторінки, та використовувати їх пізніше як компоненти.

```

30 <script>
31 export default {
32   props: ['rates', 'date'],
33   data: () => ({
34     currencies: ['UAH', 'USD', 'EUR']
35   })
36 }
37 </script>

```

Рисунок. 3.3 – Приклад використання коду Vue в HTML файлі

За допомогою фреймворку Vue.js ми можемо оновлювати дані через стан додатку. Це дуже зручно, так як можна оновити елемент, не оновлюючи сторінку в цілому. Завдяки цьому збільшується швидкодія веб-додатку, та економиться трафік користувачів.

Дизайн адаптивний, майже, для будь-яких пристроїв, наприклад:

- ПК;
- Смартфон;
- Планшет;
- Будь-які пристрої в яких встановлений браузер.



Рисунок. 3.4 – Вигляд додатку на планшеті

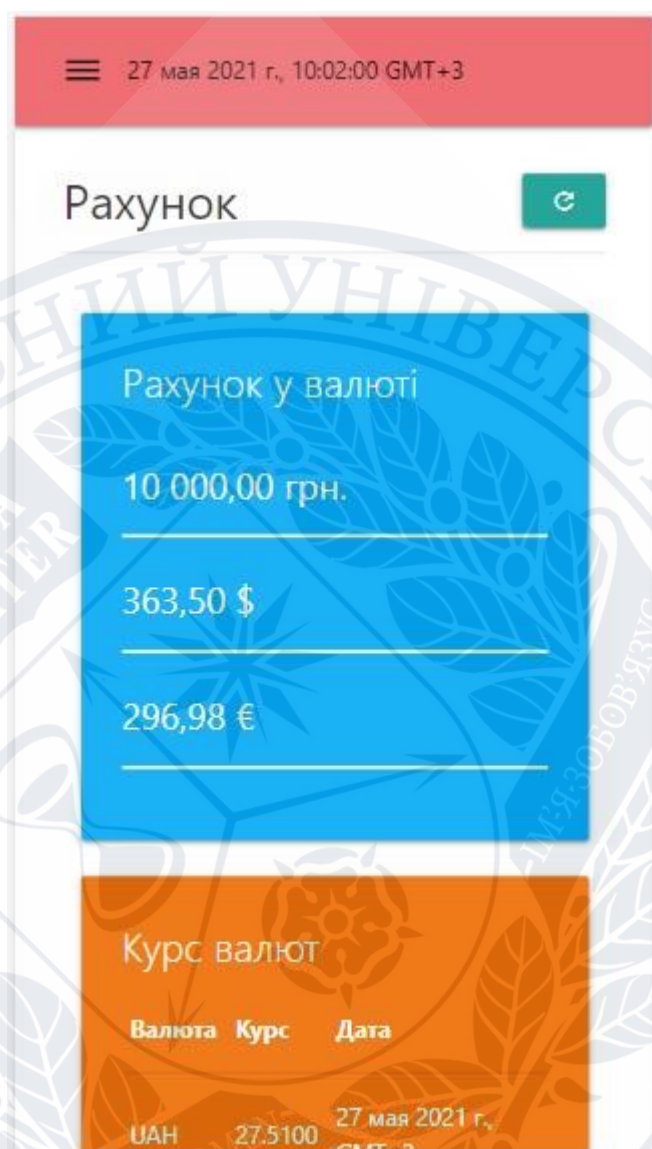


Рисунок. 3.5 – мобільний вигляд веб-додатку

3.3 Процес розробки

Для початку потрібно створити Vue.js проект за допомогою Vue CLI та внести необхідні налаштування.

Розглянемо структуру додатку.

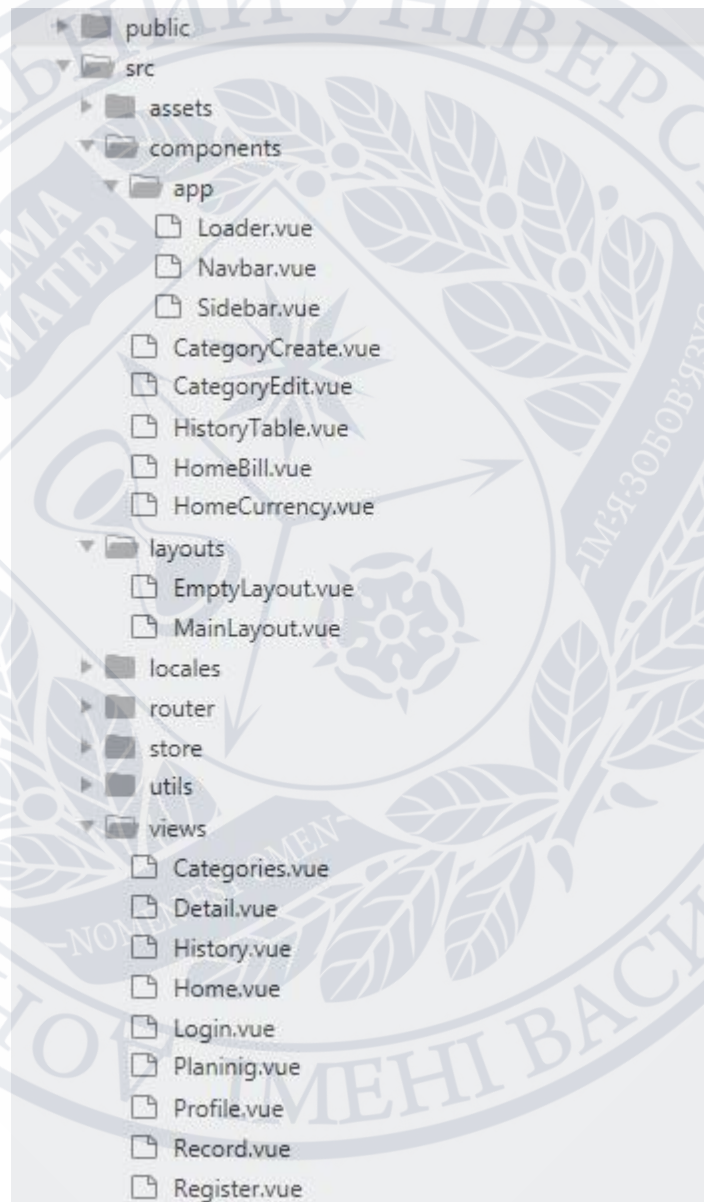


Рисунок. 3.6 – структура веб-додатку

Код додатку знаходиться у *src/* каталозі.

У каталозі *components/* знаходяться основні vue.js компоненти. Вони є важливою концепцією Vue. Дана абстракція надає можливість збирати великі програми з маленьких «шматочків». Ці «шматочки» являють собою придатні до повторного використання об'єкти. На даний момент, майже будь-який інтерфейс можна уявити як дерево компонентів:



Рисунок. 3.7 – дерево компонентів

У каталозі *views/* знаходяться HTML шаблони сторінок додатку, які під'єднуються до Vue.js компонентів.

Налаштування бази даних за допомогою Firebase Cloud Firestore.

Firebase використовується для оновлення в режимі реального часу, авторизації користувачів та хостингу. Налаштовувати внутрішню базу даних та інтегрувати її в проект Vue.js будемо за допомогою Cloud Firestore.

Імпортуємо Firebase та Cloud Firestore, а також необхідні для розробки модулі та компоненти.

```
19 import firebase from 'firebase/app'
20 import 'firebase/auth'
21 import 'firebase/database'
22
23 Vue.config.productionTip = false
24
25 Vue.use(VueMeta)
26 Vue.use(Vuelidate)
27 Vue.use(messagePlugin)
28 Vue.use(titlePlugin)
29 Vue.filter('date', dateFilter)
30 Vue.filter('currency', currencyFilter)
31 Vue.filter('localize', localizeFilter)
32 Vue.directive('tooltip', tooltipDirective)
33 Vue.component('loader', loader)
34 Vue.component('Paginate', Paginate)
35
36
37 const firebaseConfig = {
38   apiKey: "",
39   authDomain: "asp-ads.firebaseio.com",
40   databaseURL: "https://asp-ads.firebaseio.com",
41   projectId: "asp-ads",
42   storageBucket: "asp-ads.appspot.com",
43   messagingSenderId: "",
44   appId: ""
45 };
46
47 firebase.initializeApp(firebaseConfig);
48
49
50 let app
51 firebase.auth().onAuthStateChanged(() => {
52   if (!app) {
53     new Vue({
54       router,
55       store,
56       render: h => h(App)
57     }).$mount('#app')
58   }
59 })
60
```

Рисунок. 3.8 – імпорт Firebase

За допомогою vue-router налаштовуємо авторизацію та створюємо логіку входу.

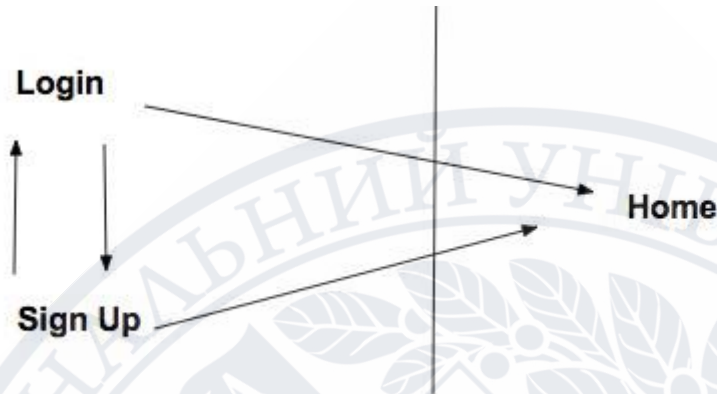


Рисунок. 3.6 – логіка входу

```

1  import firebase from 'firebase/app'
2
3  export default {
4
5    actions: {
6      async login({ commit }, { email, password }) {
7        try {
8          await firebase.auth().signInWithEmailAndPassword(email, password)
9        } catch (e) {
10         commit('setError', e)
11         throw e
12       }
13     },
14     async register({ dispatch, commit }, { email, password, name }) {
15       try {
16         await firebase.auth().createUserWithEmailAndPassword(email, password)
17         const uid = await dispatch('getUid')
18         await firebase.database().ref(`/crm-users/${uid}/info`).set({
19           bill: 10000,
20           name
21         })
22       } catch (e) {
23         commit('setError', e)
24         throw e
25       }
26     },
27     getUid() {
28       const user = firebase.auth().currentUser
29       return user ? user.uid : null
30     },
31     async logout({ commit }) {
32       await firebase.auth().signOut()
33       commit('clearInfo')
34     }
35   }
36 }
37
38

```

Рисунок. 3.9 – код авторизації

Перевіряємо авторизацію.

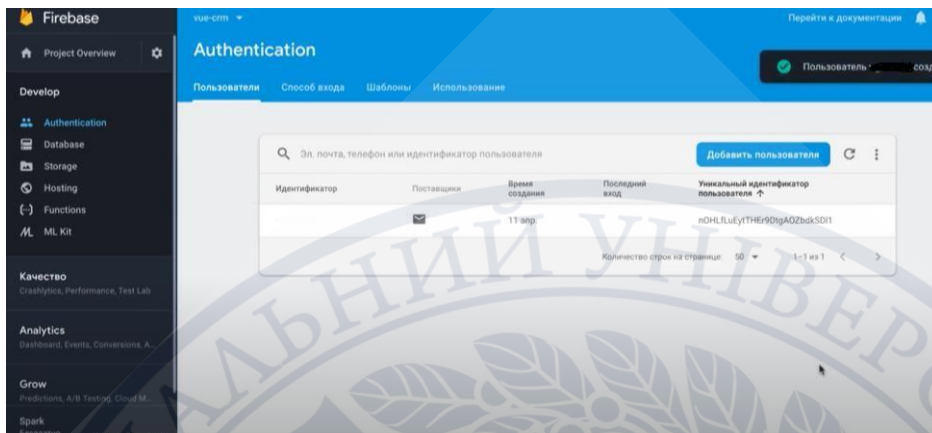


Рисунок. 3.10 – БД firebase

Створюємо навігацію за допомогою компонентів Vue.

```

88 <script>
89 export default {
90   props: ['value'],
91   data: () => ({
92     links: [
93       { title: ('Menu_Bill'), url: '/', exact: true },
94       { title: ('Menu_History'), url: '/history' },
95       { title: ('Menu_Planning'), url: '/planning' },
96       { title: ('Menu_NewRecord'), url: '/record' },
97       { title: ('Menu_Categories'), url: '/categories' }
98     ]
99   })
100 }
101 </script>

```

Рисунок. 3.12 – меню навігації

Також, за допомогою Firebase onAuthStateChanged додаємо обробку стану користувача для відстеження стану авторизації при перезавантаженні сторінки.

```

let app
auth.onAuthStateChanged(user => {
  if (!app) {
    app = new Vue({
      router,
      store,
      render: h => h(App)
    }).$mount('#app')
  }

  if (user) {
    store.dispatch('fetchUserProfile', user)
  }
})

```

Рисунок. 3.11 – обробка стану користувача

Створюємо HTML шаблони та під'єднаємо їх до проекту.

Приклад шаблону:

```

1  <template>
2    <div>
3      <div class="page-title">
4        <h3>{{ 'Categories' }}</h3>
5      </div>
6      <section>
7        <Loader v-if="loading" />
8        <div v-else class="row">
9          <CategoryCreate @created="addNewCategory" />
10         <CategoryEdit
11           v-if="categories.length"
12           :categories="categories"
13           :key="categories.length + updateCount"
14           @updated="updateCategories"
15         />
16         <p v-else class="center">{{ 'NoCategories' | localize }}</p>
17       </div>
18     </section>
19   </div>
20 </template>

```

Рисунок. 3.13 – приклад HTML шаблону

3.4 Взаємодія користувача та веб-додатку

Даним веб-додатком можливо користуватись лише авторизованим користувачам. Не авторизовані користувачі мають доступ до сторінки Login або Sing Up.

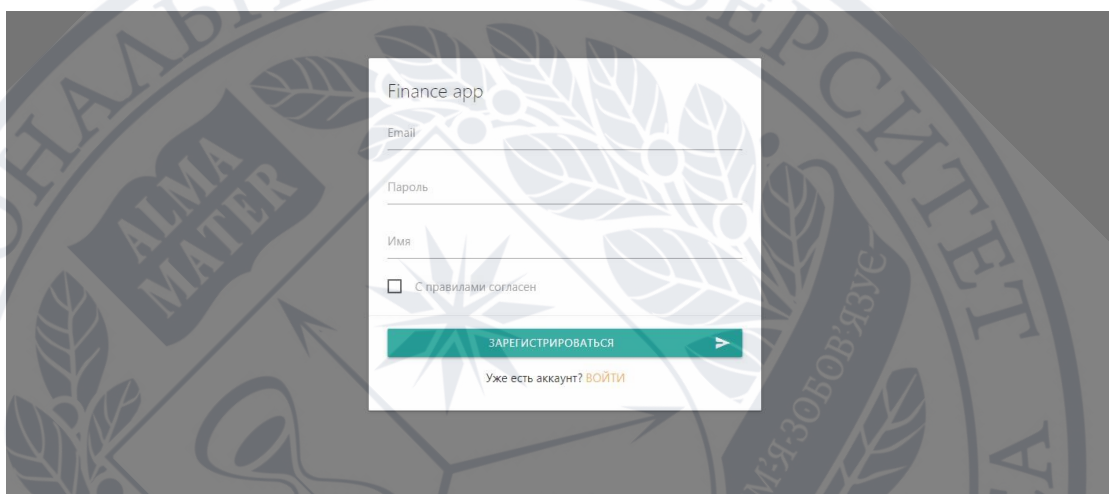


Рисунок. 3.14 – сторінка реєстрації

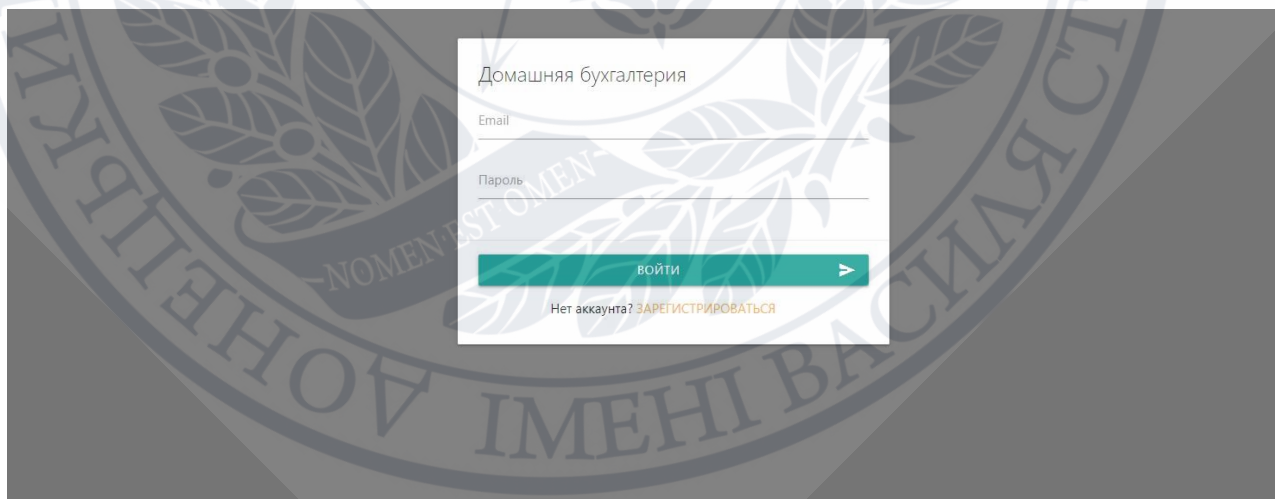


Рисунок. 3.15 – сторінка авторизації

Після успішного входу в систему або створення нового облікового запису, ми будемо перенаправлені на Домашню сторінку

Головна сторінка додатку виглядає наступним чином:

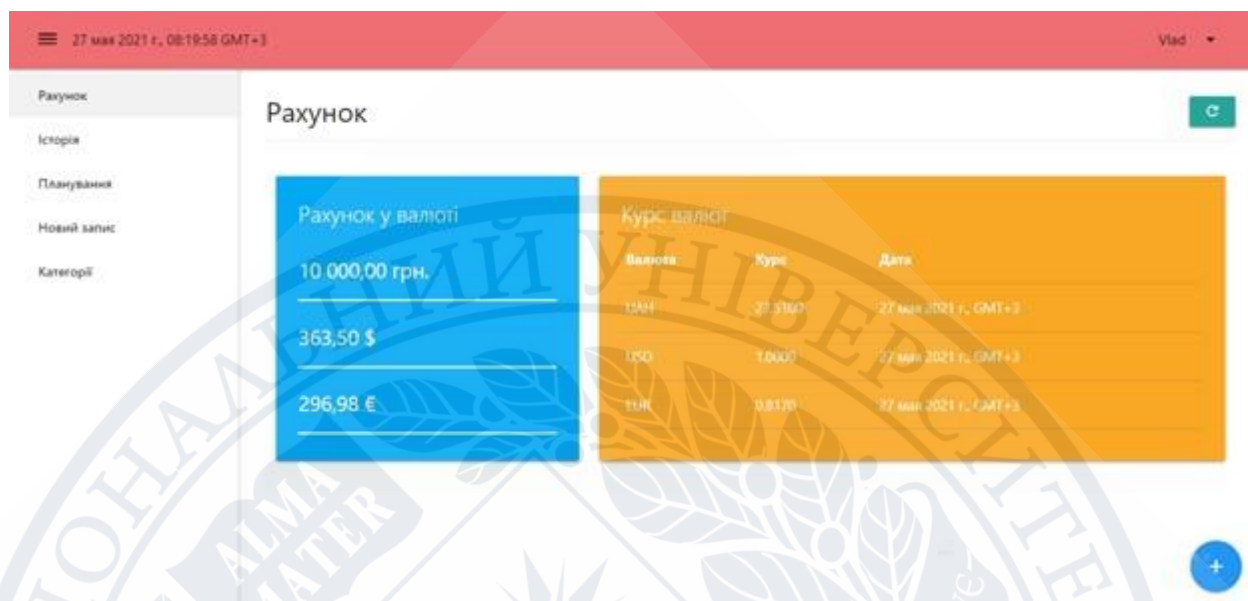


Рисунок. 3.16 – головна сторінка

На даній сторінці присутній поточний стан рахунку в трьох валютах, актуальний курс валют та меню навігації по додатку.

Для того, щоб почати користуватись додатком необхідно створити категорії витрат та заощаджень.



Рисунок. 3.18 – сторінка створення категорій

Після того, як категорії створенні, можна приступати до фінансових записів.

На сторінці Новий запис ми можемо обрати категорію, вказати дохід або витрати по даній категорії, внести суму та додати опис

Рисунок. 3.17 – сторінка створення записів

Після того, як запис створений, ми можемо переглянути його на сторінці Історія.

#	Сума	Дата	Категория	Тип	Открыть
1	20 000.00 грн.	07 июня 2021 г., 03:34:49 GMT+3	Зарплата	Дохід	
2	4 500.00 грн.	07 июня 2021 г., 03:36:50 GMT+3	Їжа	Расход	
3	500.00 грн.	07 июня 2021 г., 03:37:06 GMT+3	Розваги	Расход	
4	2 000.00 грн.	07 июня 2021 г., 03:37:32 GMT+3	Зарплата	Дохід	
5	300.00 грн.	07 июня 2021 г., 03:38:27 GMT+3	Транспорт	Расход	

Рисунок. 3.19 – сторінка історії записів

Також ми маємо можливість отримати повну інформацію про запис, натиснувши на нього.



Рисунок. 3.20 – сторінка історії записів

Також є сторінка планування, яка дозволяє переглядати бюджет у форматі прогрес бару.



Рисунок. 3.21 – сторінка планування

Можна підсумувати, що поставлені цілі були виконані, у практичній частині вийшло створити швидкий, стабільний та простий додаток для зручного керування особистими фінансами. Даний додаток реалізований у веб форматі, що дозволяє користуватись ним практично з будь-якого девайсу, що має доступ в мережу інтернет.

ВИСНОВОК

У бакалаврській роботі представлені результати, які відповідають поставленій меті та є вирішенням задачі проектування додатку для керування фінансами з функцією візуалізації даних.

Для вирішення було проаналізовано наявні аналоги, проведене їх порівняння, та поставлений вектор для написання практичної частини.

Був оглянутий спектр технологій та програмних застосунків для вирішення задачі, були обрані оптимальні на суб'єктивну думку рішення. Далі було представлено основний інструментарій для виконання роботи. Були оглянуті основні особливості кожної бібліотеки та фреймворку.

Для розробки додатку був використаний JavaScript фреймворк Vue.js, бібліотеки vue-cli, vue-router та Materialize.css для маршрутизації та інтерфейсу. В якості бази даних використовується Firebase Realtime Database, для авторизації користувачів - Firebase Auth. Додаток розгорнуто на хостингу Firebase Hosting.

Основні функції фінансового помічника, такі як створення категорій, внесення доходу та розходів, візуалізація даних, відображення рахунку у різних валютах були реалізовані. Надалі доцільно буде сконцентруватися на властивостях додатку таких як, покращення досвіду користувача при використанні системи.

Наприклад, покращення інтерфейсу, додавання нових функцій, нагадування про оплату рахунків, автоматичний імпорт банківських операцій тощо.

СПИСОК ВИКОРИСТАНИХ ПОСИЛАНЬ

1. Vue.js [Електронний ресурс] – Режим доступу до ресурсу: <https://cli.vuejs.org/ru/>
2. JavaScript Object Notation [Електронний ресурс] – Режим доступу до ресурсу: <https://www.json.org/json-en.html>
3. ViewModelView [Електронний ресурс] – Режим доступу до ресурсу: <https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93viewmodel>
4. GitHub [Електронний ресурс] – Режим доступу до ресурсу: <https://www.freecodecamp.org/news/the-beginners-guide-to-git-github/>
5. Tutorial for GitHub [Електронний ресурс] – Режим доступу до ресурсу: <https://product.hubspot.com/blog/git-and-github-tutorial-for-beginners>
6. Vue.js with Firebase [Електронний ресурс] – Режим доступу до ресурсу: <https://javascript.plainenglish.io/how-to-implement-a-showcase-web-app-in-vuejs-with-firebase-and-register-functionality-part-1-992089d17828>
7. Deploy Vue.js [Електронний ресурс] – Режим доступу до ресурсу: <https://dev.to/jesusrmendez/how-to-deploy-vue-js-firebase-1iic>
8. Firebase [Електронний ресурс] – Режим доступу до ресурсу: <https://firebase.google.com/>
9. Vue Router [Електронний ресурс] – Режим доступу до ресурсу: <https://router.vuejs.org/>
10. WebStorm [Електронний ресурс] – Режим доступу до ресурсу:

<https://itpro.ua/product/jetbrains-webstorm/?tab=description>

11. Materialize.css [Електронний ресурс] – Режим доступу до ресурсу:

<https://materializecss.com/about.html>

12. Materialize tutorial [Електронний ресурс] – Режим доступу до ресурсу:

<https://www.javatpoint.com/materialize-css-tutorial>

13. CoinKeeper [Електронний ресурс] – Режим доступу до ресурсу:

<https://about.coinkeeper.me/>

14. JavaScript [Електронний ресурс] – Режим доступу до ресурсу:

<https://developer.mozilla.org/en-US/docs/Web/JavaScript>

15. WebStorm [Електронний ресурс] – Режим доступу до ресурсу:

<https://uk.wikipedia.org/wiki/WebStorm>

16. Vuex [Електронний ресурс] – Режим доступу до ресурсу:

<https://vuex.vuejs.org/ru/guide/>

17. Firebase Realtime Database [Електронний ресурс] – Режим доступу до ресурсу:

<https://firebase.google.com/products/realtime-database>

18. Firebase Authentication [Електронний ресурс] – Режим доступу до ресурсу:

<https://firebase.google.com/products/auth>

19. Fixer [Електронний ресурс] – Режим доступу до ресурсу:

<https://fixer.io/documentation>