МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет комп'ютерних наук, фізики та математики
Кафедра комп'ютерних наук та програмної інженерії

МОДЕЛЮВАННЯ ТА РОЗРОБЛЕННЯ SPA WEB-ДОДАТКУ «MY CASH&TIME» ЗАСОБАМИ ФРЕЙМВОРКУ REACT.JS

Кваліфікаційна робота (проєкт)
на здобуття ступеня вищої освіти «бакалавр»

Виконав: здобувач спеціальності: 122
Комп'ютерні науки
Освітньо-професійної програми:
Комп'ютерні науки
Жуков Андрій Сергійович
Керівник: старший викладач Черненко І.Є.
Рецензент:
Орлов Володимир Андрійович,
ARTJOKER OÜ, CTO (Chief Technology Officer)

Івано-Франківськ – 2023

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

СУБД ..... Система управління базами даних

API .....Application Programming Interface

CSS .....Cascading Style Sheets

DNS.....Domain Name System

DOM ..... Document Object Mod

DRY .....Don’t Repeat Yourself

FTP .....File Transfer Protocol

HTML..... HyperText Markup Language

HTTP .....HyperText Transfer Protocol

HTTPS.....HyperText Transfer Protocol Secure

JS ..... JavaScript

JSON .....JavaScript Object Notation

MPA ..... Multi Page Application

MVC ..... Model View Controller

NoSQL ..... Not Only Structured Query Language

RAD .....Rapid Application Development

React.js ..... React JavaScript

SASS .....Syntactically Awesome Stylesheet

SCSS ..... Sassy Cascading Style Sheets

SOLID:

S Single Responsibility Principle

O Open-Closed Principle

L Liskov Substitution Principle

I Interface Segregation Principle

D Dependency Inversion Principle

SPA ..... Single Page Application

SQL ..... Structured Query Language

SSH ..... Secure Shell

TS ..... TypeScript

UML ..... Unified Modeling Language

WebML ..... Web Modeling Language

WSDM ..... Web Site Design Method

XML ..... eXtensible Markup Language

XHTML..... Extended Hypertext Markup Language

ВСТУП

Актуальність теми. Інформаційні технології вже давно стали невід'ємною частиною сучасного світу. Саме вони визначають подальший економічний та суспільний розвиток людства. Не є виключенням і фінансова сфера.

Майже у кожного є смартфон або ноутбук. Саме тому діджиталізація всіх сфер людської діяльності є лише питанням часу. Не є виключенням і фінансова сфера. У нашій державі, а також в інших країнах процес керування фінансами вже перейшов у комп'ютери, а частіше – навіть у смартфони. Адже набагато зручніше та надійніше зосередити процес керування власними фінансами у своєму смартфоні чи ПК аніж у блокноті. Наразі актуальним є розроблення універсального інструменту для роботи з власними фінансами, що зможе полегшити життя студентам, робітникам, програмістам та просто звичайним людям. Цей інструмент має структурно поділятися на:

– *клієнтську частину* – web-додаток, за допомогою якого користувач може переглядати свої фінанси – активні та пасивні статті доходів та витрат. Також можна буде керувати часом – користувач зможе записувати, скільки годин на тиждень він витрачає на роботу, відпочинок та інші види проведення дозвілля;

– *серверну частину* – всі дані, які користувач буде вносити, будуть запам'ятовуватись в базу даних, звідки потім буде можливість вивести на екран все, що вносив користувач.

В даній роботі описується процес моделювання та розробки web-додатку, підбір інструментів для роботи, бази даних, створення та постановка на сервер готового web-додатку.

Об'єктом дослідження кваліфікаційної роботи є технології розроблення web-додатків.

Предметом дослідження є web-додаток «My Cash&Time».

Метою кваліфікаційної роботи є моделювання та розроблення web-додатку «My Cash&Time» засобами фреймворку react.js.

Виходячи з поставленої мети, були визначенні наступні з**авдання дослідження:**

- Огляд особливостей моделювання front-end та back-end.
- Порівняння програмних середовищ для розроблення.
- Огляд особливостей постановки web-додатку на сервер.
- Розроблення web-додатку «My Cash&Time».
- Написання документації до продукту.

Структура роботи складається з вступу, трьох розділів, висновку і списку літератури.

РОЗДІЛ 1
МОДЕЛЮВАННЯ WEB-ДОДАТКУ

Моделювання – це метод наукового дослідження об’єкта, або об’єктів, пізнання. Моделювання в широкому сенсі – це метод теоретичного та практичного пізнання, коли суб’єкт замість безпосереднього об’єкта пізнання обирає, або створює схожий з ним, допоміжний об’єкт, а здобуту інформацію переносить на реальний предмет вивчення.[1]

1.1 Візуальне моделювання

В програмуванні моделювання – це процес що використовується для вивчення й аналізу вимог до даних, необхідних для підтримки бізнес-процесів у межах відповідних інформаційних систем.[2]

Візуальне моделювання – це спосіб створення програм шляхом маніпулювання графічними об’єктами замість написання програмного коду в текстовому вигляді.[3]

Метод візуального моделювання здобув достатньо широкого розповсюдження завдяки своїй відносній простоті, адже значна частина візуальних мов базується на ідеї «фігур і ліній», де фігури розглядаються як суб’єкти і з’єднуються лініями які являють собою відношення.[3]

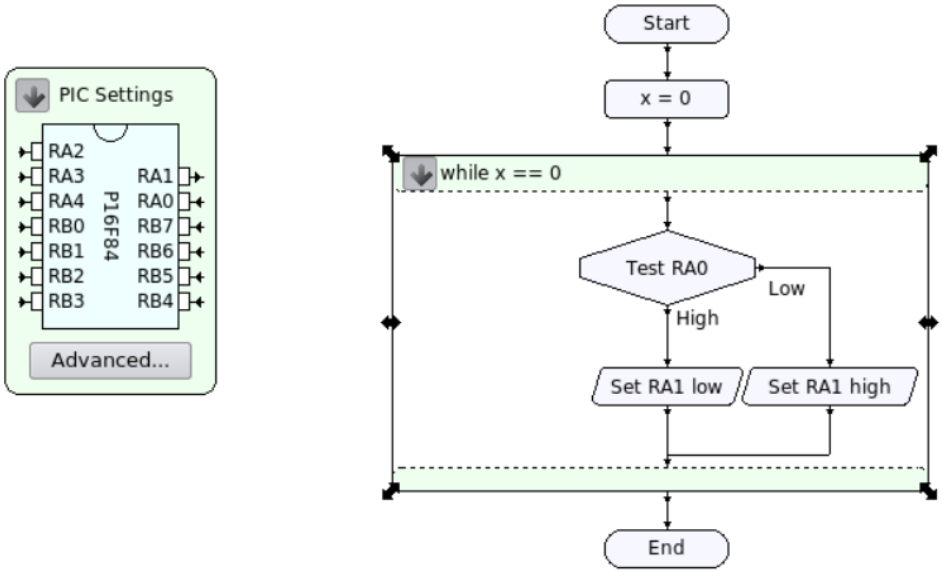

Рисунок 1.1 Приклад візуального програмування

До візуального програмування також відносять і Rapid Application Development (RAD) – швидку розробку програм. Ця технологія забезпечує прискорену розробку та модифікацію додатків за рахунок використання ООП та візуального програмування.[4]

Засобами візуального програмування зазвичай вирішують завдання побудови інтерфейсу користувача і спрощення розробки програми шляхом заміни методу «написання програми» на метод візуального конструювання. Цей засіб краще відповідає природі людського сприйняття ніж текстове програмування, адже складна структура представлена наочно.

Найбільш відомими та поширеними сучасними методологіями проєктування і розробки web-додатків є WebML та WSDM. В даній роботі я буду розглядати методологію WebML.

Методологія WebML є підходом до розробки web-додатків на основі моделі.[5] Дана методологія об’єднує традиційні прийоми, які добре відомі розробникам, такі як сценарії на мові UML та концептуальне проєктування даних за допомогою моделі Entity-Relationship, з новими

поняттями і методами програмування web-сторінок, які є важливими для web-додатків.

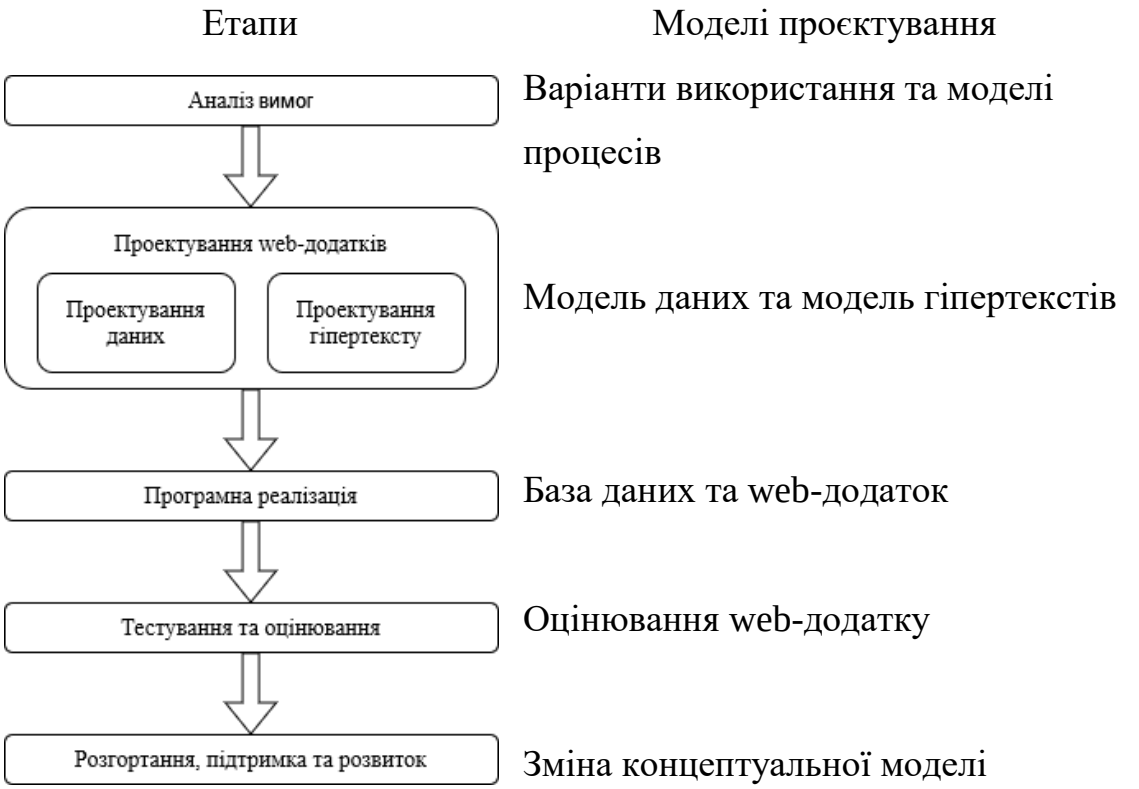

Рисунок 1.2 Етапи розроблення в WebML

Для моделювання в WebML необхідно виконати наступні дії:

- Виявити групи користувачів для яких розробляється додаток;
- Специфікація функціональних вимог які пов’язані з функціями, які надаються користувачам;
- Виявити базові інформаційні об’єкти – основні данні до яких буде надано доступ користувачам;
- Декомпозиція web-додатку – створення попереднього бачення додатку так, щоб задовольнити набору функціональних вимог та вимог користувачів.

Для моделювання web-додатків використовують UML діаграми.

UML – уніфікована мова моделювання, що використовується розробниками програмного забезпечення для візуалізації процесів роботи систем. UML-діаграми загалом поділяються на 14 типів:

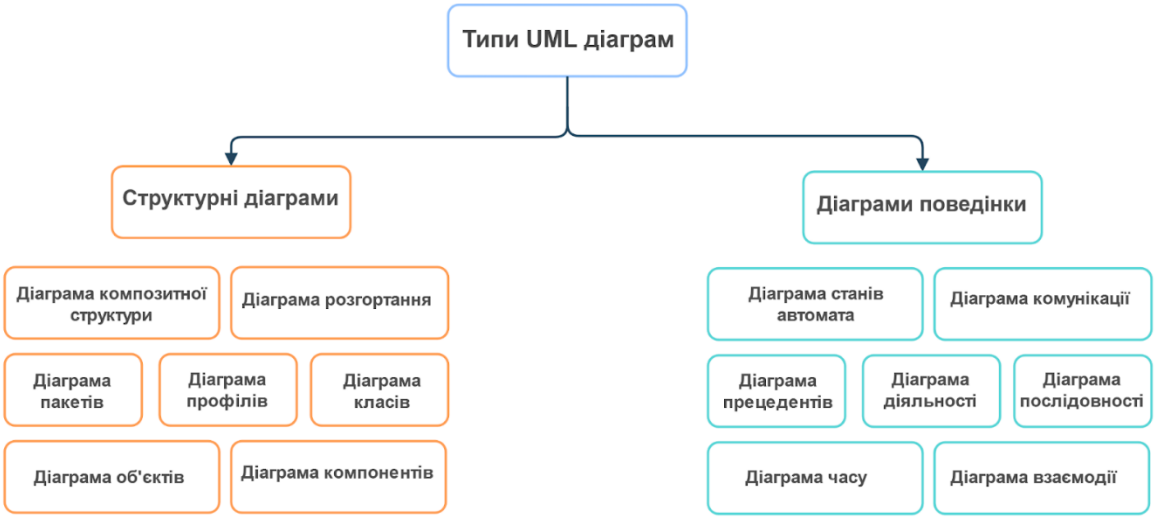

Рисунок 1.3 Типи UML-діаграм

1.2 Моделювання front-end частини додатку

1.2.1 Вимоги до front-end частини додатку

Front-end, або як його ще називають, користувацький інтерфейс, має свої вимоги та повинен відповідати їм. Інтерфейс сучасного web-додатку повинен містити в собі такі якості:

- простота;
- інтуїтивна зрозумілість;
- впізнаваний дизайн.

Простіше кажучи, якщо ці якості не будуть присутніми в одному додатку – користувачам прийдеться самим здогадуватись де і що за що відповідає, що відлякує користувачів та потенційних клієнтів.

Іншою особливістю є відсутність стандартів на оформлення візуальних елементів інтерфейсу. Якщо ми візьмемо елемент кнопка, або поле введення, або меню, або будь-який інший візуальний елемент, то в рамках операційної системи цей елемент виглядає однаково і поводиться однаково в будь-якій програмі. Відповідно, користувач звикає до розташування, зовнішнього вигляду і поведінки візуальних елементів даної операційної системи. Меню у будь-якій програмі розташовані вгорі

вікна (Windows) або в спеціальному рядку меню (MacOs). Кнопка завжди має колір, визначений поточною палітрою користувача. Зовнішній вигляд кнопки однаковий у різних програмах тощо.

Описане вище не є справедливим повною мірою для веб-додатків. Стандартами мови HTML передбачено візуальні елементи кнопка, поле введення і так далі. Однак як у силу порівняльної бідності цих елементів, так і в силу наявних розвинених програмних засобів для їх модифікації щоразу робітник розфарбовує елементи і змінює їхню поведінку на свій розсуд. Можна спробувати сформулювати деяке правило доброго тону для розробки інтерфейсу web-додатку: не варто надмірно модифікувати зовнішній вигляд і поведінку візуальних елементів у порівнянні з найбільш поширеним, наявним на існуючих широко відомих сайтах. Є сенс спочатку подивитися, як та ж сама функція реалізована найбільш значущими, поширеними web-додатками. В іншому випадку є суттєвий ризик дезорієнтувати користувача. Якщо намалювати круглу кнопку, користувачі можуть просто не знайти її на екрані, тому що звичка, що раніше сформувалася, змушувала шукати очима кнопку у формі квадрата.

Звичайно, вищесказане — це лише правило, а зовсім не закон, адже цілком можливо, що нестандартний інтерфейс якраз і є фішкою розробника сайту. Наведений перелік особливостей не є вичерпним, але є достатнім для демонстрації специфічності інтернет-додатків, їхньої несхожості на завдання класичного прикладного програмування.

1.2.2 Технології створення front-end частини додатку

Всесвітня павутина Інтернет існує вже багато років та постійно оновлюється. Front-end частина додатків пройшла шлях від гіпертекстової розмітки (HTML) зовсім без стилів до розробки всього інтерфейсу за допомогою фреймворків мови JavaScript. З’явився розподіл на клієнтську частину та серверну частину web-додатків та сайтів. Тому для створення користувацького інтерфейсу сьогодні вже замало володіти

основами HTML+CSS, необхідно добре володіти та розуміти принципи та паттерни програмування мови програмування JavaScript. На сьогодні стек технологій, які необхідно знати та використовувати виглядає так:

- HTML5;
- CSS3;
- JavaScript;
- Фреймворк JS (Angular, Vue.js, React.js);
- Node.js.

Фреймворки розширюють можливості JavaScript та надають змогу використовувати особливості даного фреймворку. Для React.js, наприклад, такими особливостями є:

- Компонент – частина коду, який відповідає за зовнішній вигляд одного з елементів сайту або додатку, такі шматочки можуть бути вкладеними;
- Стан – вся інформація про елемент, у тому числі і про його відображення. [6,7]

1.2.3 Розроблення UML діаграми поведінки front-end частини

UML діаграма поведінки повинна цілком відтворювати поведінку front-end частини при різних ситуаціях життєвого циклу web-додатку.

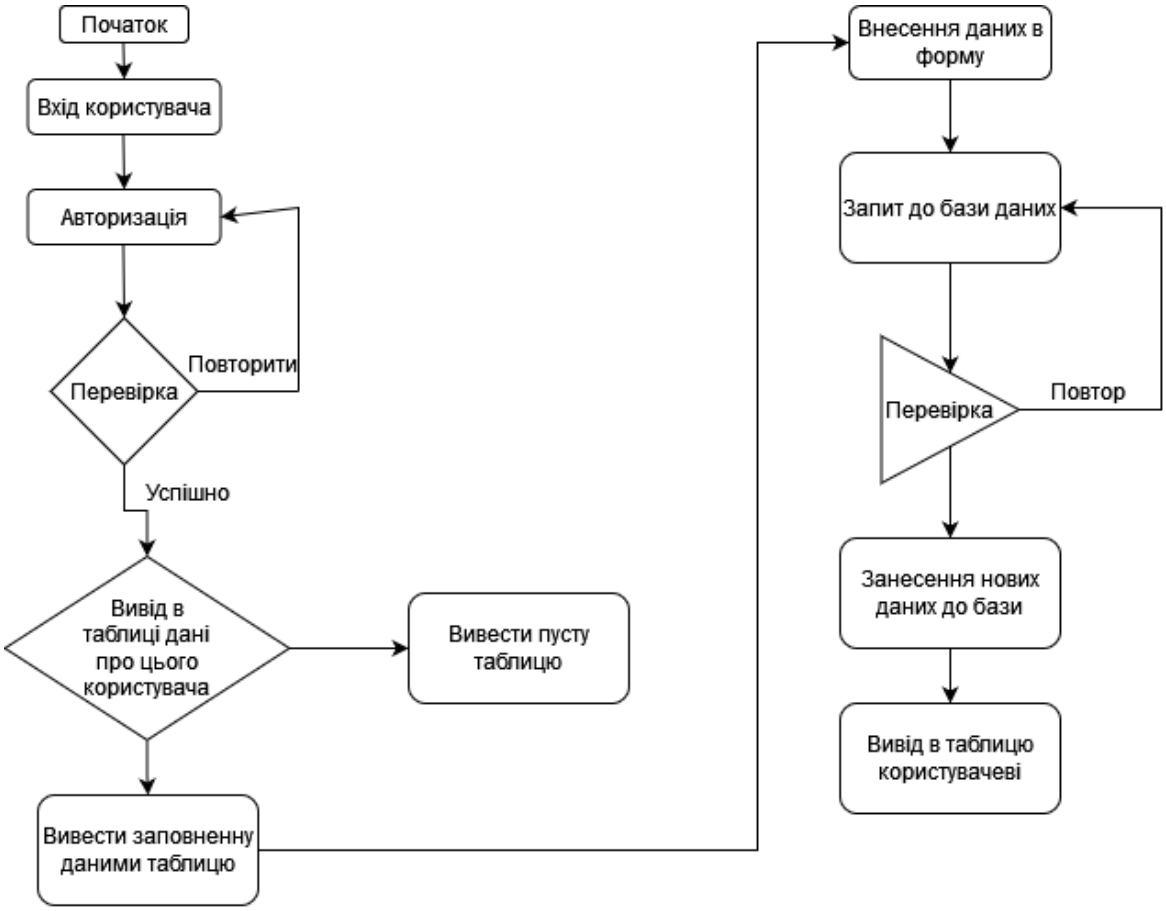

Рисунок 1.4 UML діаграма поведінки

1.3 Моделювання back-end частини додатку

1.3.1 Вимоги до back-end частини додатку

Back-end частина додатку є ключовою складовою будь-якого сучасного web-додатку. Ось деякі вимоги, які повинні бути враховані при розробці back-end частини:

– *Масштабованість*. Додаток повинен бути здатним відповідати на зростаючу кількість запитів, трафіку та обсягів даних. Для цього можна використовувати горизонтальне та вертикальне масштабування.

– *Безпека*. Back-end частина має забезпечувати безпеку додатку та даних користувачів. Це можна досягти шляхом використання шифрування, аутентифікації та авторизації, захисту від атак типу SQL Injection та Cross-Site Scripting.

– *Швидкість*. Back-end частина повинна бути оптимізована для швидкої відповіді на запити та обробки даних. Цього можна досягти шляхом використання швидких алгоритмів, хешування даних та зменшення кількості запитів до бази даних.

– *Надійність*. Back-end частина має бути надійною та міцною, щоб запобігти відмовам у роботі додатку. Це можна досягти шляхом використання відповідного архітектурного підходу, наприклад, мікро сервісної архітектури, резервування, контролю помилок та резервного копіювання даних.

– *Розширюваність*. Back-end частина повинна бути здатною до легкої розширюваності, якщо потрібно додати новий функціонал. Це можна досягти шляхом використання принципів SOLID та патернів проєктування, таких як Dependency Injection та Factory.

– *Інтегрованість*. Back-end частина має бути здатною інтегруватися з іншими системами та службами.

1.3.2 Технології створення back-end частини додатку

Сьогодні існує багато технологій, які можна використовувати для створення back-end частини додатку. Ось деякі з них:

– *Java та Spring Framework*. Java є дуже популярною мовою програмування для back-end розробки, а Spring Framework - однією з найбільш використовуваних технологій для розробки Java-додатків. Spring Framework надає багато функціональних можливостей, таких як інверсія керування, вбудовані контейнери Servlet та підтримка REST API.

– *Python та Flask/Django*. Python також є дуже популярною мовою програмування для back-end розробки. Flask та Django - це дві

найпопулярніші технології для розробки back-end додатків на Python. Flask є легким та простим у використанні фреймворком, а Django - повноцінним фреймворком, який надає багато готового функціоналу.

– *Node.js ma Express.js*: Node.js - це платформа, яка дозволяє виконувати JavaScript на стороні сервера. Express.js - це легкий та простий у використанні фреймворк для створення back-end додатків на Node.js. Він надає багато функціоналу для розробки REST API.

– *Ruby ma Ruby on Rails*: Ruby є дуже елегантною мовою програмування, а Ruby on Rails - це повноцінний фреймворк для розробки back-end додатків на Ruby. Rails надає багато готового функціоналу, який дозволяє швидко створювати додатки.

– *PHP ma Laravel*: PHP є дуже популярною мовою програмування для back-end розробки. Laravel - це один з найпопулярніших фреймворків для розробки back-end додатків на PHP. Він надає багато готового функціоналу, такого як автентифікація, маршрутизація та шаблонізація.

1.3.3 Розроблення UML діаграми поведінки back-end

UML діаграма поведінки back-end частини показує як back-end повинен реагувати на запити від клієнта, що робити з запитом та які етапи обробки проходить запит від моменту натискання клавіші користувачем до отриманні відповіді. Ось така діаграма повністю показує роботу back-end для додатку «My Cash&Time»:

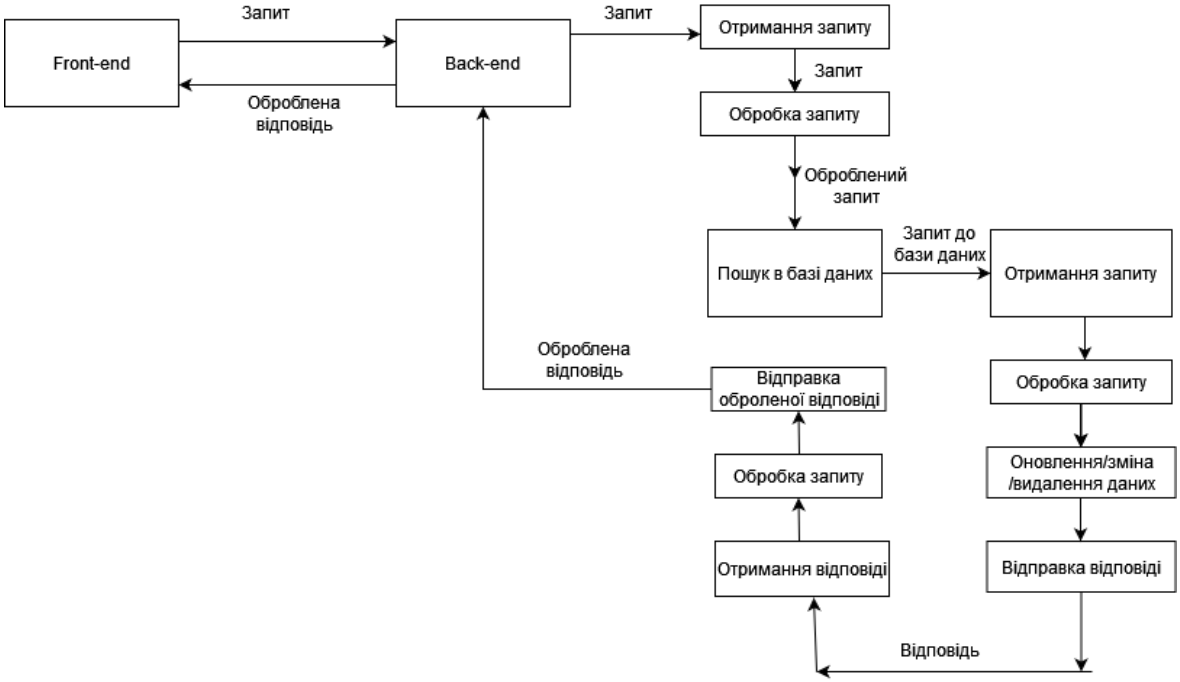

Рисунок 1.5 UML-діаграма роботи back-end частини додатку

1.4 Моделювання бази даних

База даних сучасного web-додатку повинна відповідати таким критеріям:

- швидка взаємодія;
- доступність;
- підтримка всіх необхідних зв’язків;
- цілодобова робота.

Також необхідно, щоб була взаємодія бази даних з React.js. Для коректної роботи необхідно розробити структуру таблиць бази даних, їх поля, доступи та можливість зв’язування. Згідно вимог, база даних додатку «My Cash&Time» має складатися з 5 таблиць: User, Time, Cash, Active, Passive.

Таблиця «User» повинна мати такі поля:

- *Id* – унікальний номер користувача в системі;
- *Username* – ім’я користувача;
- *Userpass* – пароль користувача.

Таблиця «Time» повинна мати такі поля:

- *Id* – унікальний номер в базі;
- *User_id* – унікальний номер користувача в системі для зв’язування таблиць;
- *spendTime* – витрата часу;
- *allTime* – загальна кількість часу на тиждень;
- *timeLeft* – залишок часу.

Таблиця «Cash» повинна мати такі поля:

- *Id* – унікальний номер в базі;
- *User_id* – унікальний номер користувача в системі, для зв’язування таблиць;
- *activeName* – назва активу;
- *activePrice* - вартість активу;
- *pasiveName* – назва пасиву;
- *pasivePrice* – вартість пасиву;
- *allIncome* – загальна сума надходжень;
- *AllExpencec* – загальна сума витрат.

Таблиця «Active» повинна мати такі поля:

- *Id* – унікальний номер в базі;
- *activeName* – назва активу;
- *activePrice* - вартість активу.

Таблиця «Passive» повинна мати такі поля:

- *Id* – унікальний номер в базі;
- *pasiveName* – назва пасиву;
- *pasivePrice* – вартість пасиву.

Для створення бази даних використовується СУБД – система управління базами даних.

В даній роботі використовується СУБД «phpMyAdmin». Вона використовує діалект мови SQL, а саме MySQL.

З базами працюють за допомогою команд мови SQL, такими як:

- CREATE DATABASE – для створення бази;
- USE DATABASE – для вибору бази;
- CREATE TABLE – для створення таблиці в обраній базі;
- SELECT * FROM – для вибору всіх даних з потрібної таблиці;
- SELECT * FROM WHERE – для вибору всіх даних з потрібної таблиці за заданими параметрами.

Наступна UML діаграма показує зв’язки та залежності між таблицями бази даних.

Рисунок 1.6 Структура бази даних

РОЗДІЛ 2
ПІДГОТОВКА ДО РОЗРОБКИ SPA WEB-ДОДАТКУ

2.1 Вибір програмного середовища для розробки

Visual Studio Code та WebStorm – це два популярних редактори коду, які використовуються для розробки web-додатків. Їх можна порівняти за такими ознаками:

– *Відкритість*. VS Code – це відкрите програмне забезпечення та безкоштовний редактор коду, тоді як WebStorm – це пропрієтарне програмне забезпечення з платою за користування.

– *Підтримка мов*. VS Code підтримує велику кількість мов програмування, таких як JavaScript, TypeScript, Python, C++, PHP та інші, тоді як WebStorm націлений саме на web-розробку, та підтримує HTML, CSS, JavaScript з усіма його багаточисленними фреймворками.

– *Редагування коду*. Обидва редактори мають потужні можливості редагування коду, включаючи автодоповнення, налагоджування коду тощо.

– *Налаштування*. VS Code має велику кількість налаштувань та розширень, що дозволяє користувачам налаштовувати редактор на свій смак. WebStorm має менше розширень, але його можна налаштовувати на рівні проєкту, або групи проєктів. [7, 8]

Зважаючи на всі переваги та недоліки обох редакторів, для розробки даного web-додатку я обрав редактор WebStorm.

2.2 Система контролю версій Git

Git – це безкоштовна система контролю версій, яка дозволяє розробникам вести історію змін коду, керувати версіями програмного забезпечення та співпрацювати у роботі над проєктами. Ядро Git представляє собою набір утиліт командного рядка. [9]

Основні переваги використання системи Git:

– *Розподілена архітектура*. Git дозволяє розробникам зберігати локальну копію репозиторію та працювати з ним офлайн. Кожен розробник може змінювати свою локальну копію та синхронізувати з центральним репозиторієм.

– *Ефективність*. Git дуже ефективний у відстеженні змін та керуванні версіями коду. Він дозволяє швидко знаходити історію змін, повертатись назад до попередніх версій та об’єднувати різні гілки розробки.

– *Гнучкість*. Git дозволяє розробникам використовувати різні стратегії розробки, включаючи такі особливості як гілки розробки, злиття гілок, та перенесення змін з одного репозиторію до іншого.

– *Ком’юніті*. Git має широку активну спільноту розробників, яка допомагає користувачам знаходити відповіді на питання та розв’язувати проблеми. [9]

Підсумовуючи можна сказати що Git є потужним та корисним інструментом для розробки програмного забезпечення. Він дозволяє зручно та ефективно керувати версіями коду, співпрацювати з іншими розробниками та швидко виправляти проблеми. Використання системи контролю версій Git дозволяє працювати більш професійно та підвищити безпеку від втрати файлів.

GitHub та GitLab – це два з найпопулярніших онлайн сервісів для роботи з Git-репозиторіями.

GitHub був створений в квітні 2008 року, в січні того ж року був створений перший приватний репозиторій, а до кінця 2011 року в проєкті

було зареєстровано більше мільйона користувачів та більше двох мільйонів репозиторіїв. На GitHub розміщена копія вихідного коду ядра Linux, а також розміщені офіційні репозиторії таких великих IT-гігантів як Facebook(Meta), Twitter, Google, Microsoft, Apple, Valve та інших. [10]

GitLab був створений у жовтні 2011 року українськими програмістами для персональних потреб. Пізніше GitLab перетворився в інтегроване рішення, яке охоплює весь життєвий цикл розробки програмного забезпечення, а потім і весь життєвий цикл DevOps. Систему використовують такі компанії як: IBM, Sony, NASA, CERN, фонд GNOME та інші.[11]

Щоб вибрати між GitHub та GitLab необхідно знати їх відмінності, ось основні з них:

– *Хмарне рішення*. GitHub є хмарним рішенням, яке пропонує безкоштовне та платне розміщення відкритого та приватного програмного забезпечення на їх серверах. З іншого боку, GitLab може бути встановлений локально на власному сервері або використовуватися як хмарне рішення.

– *Рівень доступу*. GitHub пропонує безкоштовні та приватні репозиторії для відкритого та закритого програмного забезпечення, а GitLab пропонує безкоштовні та приватні репозиторії для відкритого та закритого програмного забезпечення, але для безкоштовних репозиторіїв є обмеження в розмірі файлів та місці на сервері.

– *Функціональність*. GitHub та GitLab пропонують різні функції, такі як засоби збирання, тестування, автоматизації робочих процесів, огляду коду та інші. GitLab має більш широкий функціонал у плані CI/CD та DevOps, а GitHub більш поширений серед спільноти відкритого програмного забезпечення.

– *Ціна*. GitHub та GitLab пропонують різні пакети тарифів для підприємств та індивідуальних користувачів. GitHub дозволяє безкоштовно розміщувати відкрите програмне забезпечення, тоді як

GitLab пропонує безкоштовні та приватні репозиторії з обмеженнями в розмірі та місці.

– *Масштабованість*. GitLab зазвичай вважається більш масштабованим рішенням для підприємств, оскільки воно може бути встановлено локально, тоді як працювати з GitHub вийде лише онлайн. [10, 11]

Для розробки даного web-додатку достатньо безкоштовних можливостей GitLab.

Інтегрування системи контролю версій в розробці web-додатків є важливою та розповсюдженою практикою для збереження ресурсів проєкту в безпечному та структурованому вигляді. Для роботи з системою контролю версій Git необхідно:

– *Ініціювати новий репозиторій*: створювати новий репозиторій на сайті GitHub, або GitLab.

– *Клонувати репозиторій*: клонувати репозиторій на локальний комп’ютер за допомогою команди «git clone».

– *Коміти*: додавати локальні файли в хмару Git за допомогою команд «git add» (додавання файлів до репозиторію) та «git commit» (створення коміту).

– *Відправка*: відправляти локальний репозиторій у хмару за допомогою команди «git push».

Також багато інтегрованих середовищ розробки (IDE) мають підтримку системи контролю версій і надають зручний інтерфейс для роботи з репозиторіями. Так, наприклад, у WebStorm вже інтегрована робота з Git-репозиторіями без сторонніх плагінів.

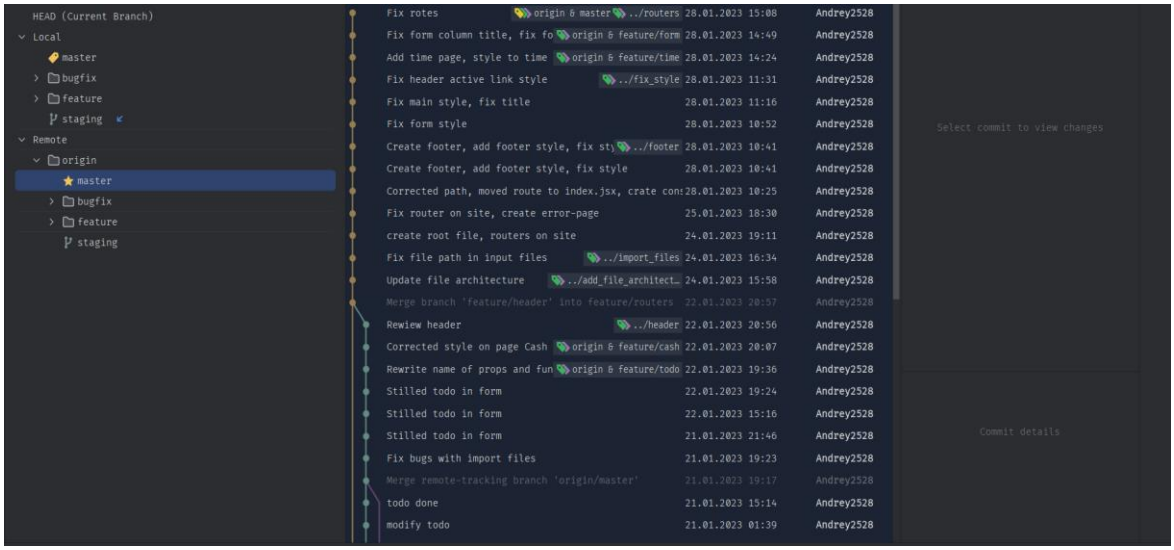

Рисунок 2.1 Приклад інтерфейсу Git в WebStorm

РОЗДІЛ 3
РОЗРОБЛЕННЯ WEB-ДОДАТКУ ЗА ДОПОМОГОЮ ФРЕЙМВОРКУ REACT.JS

3.1 Розроблення front-end складової додатку

Розроблення front-end складової web-додатку включає в себе такі етапи:

– *Аналіз вимог та дизайн*. На цьому етапі проводиться дослідження потреб користувачів, визначаються вимоги до функціональності та зовнішнього вигляду додатку, розробляється дизайн інтерфейсу.

– *Вибір технологій та інструментів*: вибираються необхідні технології (HTML, CSS, JS), інструменти (IDE), а також бібліотеки та фреймворки (React.js).

– *Розроблення структури та компонентів*: розробляється структура проєкту, проєкт розбивається на компоненти та розробляються самі компоненти, які будуть використовуватись в додатку. Кожен компонент повинен мати визначену структуру, зовнішній вигляд, логіку та взаємодію з іншими компонентами.

– *Розроблення інтерфейсу взаємодії з користувачем*: розробляються інтерфейс та логіка взаємодії з користувачем. Додаток повинен мати зручний та інтуїтивно зрозумілий інтерфейс та ефективну систему взаємодії з користувачем.

– *Розроблення функціональності*: додається функціонал – можливість реєстрації користувача, робота за даними, взаємодія з API та інше.

– *Тестування*: проводиться тестування додатку, яке допомагає виявити недоліки та допрацювати їх.

3.1.1 Аналіз вимог та дизайн

Додаток «MY CASH&TIME» розробляється для полегшення підрахунку фінансів та часу, а отже повинен мати такий функціонал:

- Авторизація користувача;
- Відображення імені користувача;
- Відображення двох сторінок: фінанси (cash) та час (time);
- Внесення даних про користувача в базу даних;
- Внесення даних про активи/пасиви користувача в таблицю бази;
- Підрахунок загальних фінансів та відображення їх на сторінці додатку;
- Бути доступним 24/7;
- Мати інтерфейс зручний та інтуїтивно зрозумілий для користувача;
- Приємний зовнішній вигляд;
- Можливість для масштабування.

Задля цього необхідно виконати деякі умови:

- Продумати архітектуру файлів додатку, яка дозволить з легкістю масштабувати додаток;
- Обрати базу даних, що буде відповідати всім вимогам;
- Обрати веб-сервер для забезпечення цілодобової роботи додатку.

Дизайн додатку обирався та розроблявся з урахуванням новітнього бачення web-додатків, зручного інтерфейсу та функціоналу додатку.

Рисунок 3.1 Головне вікно додатку

3.1.2 Вибір технологій та інструментів

Порівнявши всі технології для розробки web-додатків, зрозумівши плюси та мінуси кожного, був обраний такий стек технологій:

– HTML.

– SCSS - Sassy Cascading Style Sheets –можна вважати що це «діалект» Sass. Синтаксис SCSS більш схожий на класичний CSS, але має свої переваги.

– JS: а саме фреймворк під назваою React.js - це JavaScript-бібліотека для зручної розробки інтерфейсів, тобто зовнішньої частини сайтів та додатків, з якими взаємодіє користувач.

При розробці додатку буде використовуватись система контролю версій Git, яка інтегрована в середовище розробки WebStorm, а сам проєкт буде зберігатись на GitLab.

В якості інструментів розробки обрано WebStorm, так як він краще підходить для розробки web-додатків, та має інтегрований Git для роботи.

3.1.3 Розроблення структури компонентів

Розроблення структури компонентів – це процес планування та організації компонентів, які будуть використовуватися для створення програмного продукту або системи.

Компоненти – це повторно використовувані блоки коду, які можуть бути використані для виконання певних функцій в програмному продукті.

Структура компонентів визначає, які компоненти необхідні для побудови системи та як вони будуть взаємодіяти між собою.

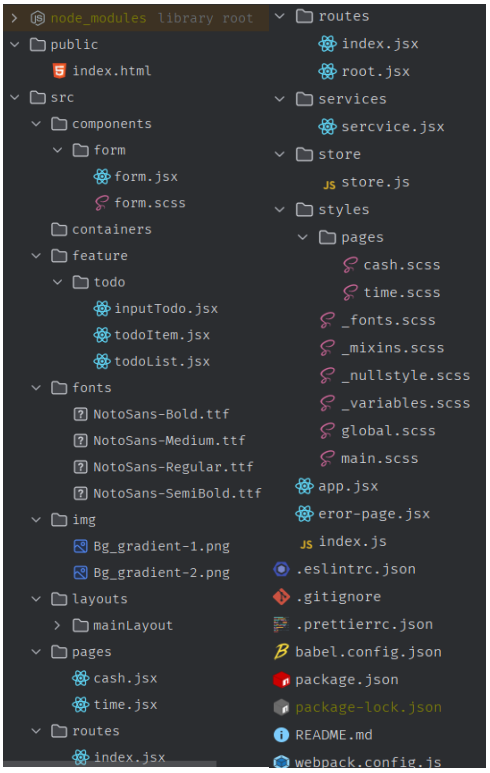

Рисунок 3.2 Структура додатку

– Node_modules – папка яка зберігає в собі файли які необхідні для роботи на локальному комп’ютері.

– Public – папка в якій містяться файли для відображення на сторінці такі як: index.html.

– Src – папка яка містить в собі всі необхідні файли для відображення додатку.

– Components – папка в якій лежать папки з файлами, що становлять собою компоненти для React.

- Features – папка в якій знаходяться окремі модулі додатку.
- Fonts – папка з шрифтами.
- Img – папка з картинками.
- Layouts – слої, тобто обгортка для модулів.
- Pages – папка з файлами сторінок.
- Routes – папка з навігацією по додатку.
- Service - папка з функціями запитів до бази даних або роботи з іншим видом бізнес логіки.
- Store – робота з Redux.
- Styles – папка з файлами стилів, які в свою чергу розбиті на підпапки для упорядкування стилю для блоків (header.js) та файли зі змінними (_variables.scss) та загальні стилі (_general.scss).
- App.js – головний файл нашого додатку, в якому відбувається рендер елементів;
- Index.js – файл який збирає всі компоненти, та передає їх для відображення в index.html;
- Package.json – файл в якому вказані вся інформація про проєкт: назва, версія, автор, модулі які використовуються.

3.1.4 Розроблення інтерфейсу взаємодії з користувачем

Розроблення інтерфейсу взаємодії з користувачем це процес створення інтерфейсу, за допомогою якого користувач може взаємодіяти з додатком. Ефективний інтерфейс взаємодії з користувачем дозволяє забезпечити зручну та логічну навігацію, швидкий та точний доступ до інформації, та ефективне використання можливостей програми або сайту.

Інтерфейс взаємодії додатку з користувачем буде будуватись за допомогою React.js. Для цього необхідно виконати основні кроки:

- *Налаштувати середовища розробки*: необхідно встановити Node.js та пакетний менеджер npm.

– *Створення компонентів*: компонент є основним елементом React.js, який відповідає за відображення частини інтерфейсу. Кожен компонент містить властивості та методи, що дозволяють його настроювати та взаємодіяти з іншими компонентами.

– *Налаштування маршрутизації*: налаштування маршрутизації необхідно в web-додатках з однією або декількома сторінками. Для цього використовують бібліотеку react-router.

– *Розроблення дизайну*: створення зручного інтерфейсу по розробленому раніше дизайну.

– *Тестування*: для перевірки ефективності та правильності роботи інтерфейсу.

Після виконання основних кроків по розробці зовнішнього вигляду, переходимо до розробки функціональності.

3.1.5 Розроблення функціональності

Web-додаток є складним програмним продуктом, що містить багато функціональних можливостей та інтерфейсів. Можна виділити такі основні пункти для розробки функціональності web-додатку:

– *Проєктування бази даних*: в залежності від функціональності додатку необхідно створити схему бази даних, яка буде використовуватися для зберігання та отримання інформації.

– *Розроблення серверної сторони*: на серверній стороні додатку розробляється програмний код, який взаємодіє з базою даних та надає потрібні сервіси та можливості. Цей код може бути написаний на різних мовах програмування, таких як Python, PHP, Node.js тощо.

– *Розроблення клієнтської сторони*: на клієнтській стороні додатку розробляється інтерфейс, який забезпечує зручну взаємодію користувача з додатком. Для цього можна використовувати HTML, CSS, JavaScript, React.js та інші технології.

3.1.6 Тестування

Тестування є важливим етапом в розробці програмних засобів в цілому та web-додатку зокрема. В даній роботі я лише коротко пройдусь по основним видам тестування. Тестування буває:

- *Функціональне* - перевіряє, чи працює функціонал додатку так, як очікувалося. Тестування функціональності додатку можна проводити вручну або за допомогою автоматизованих тестів.
- *Тестування інтеграції* - перевіряє, як добре різні компоненти додатку працюють разом. Цей вид тестування дозволяє переконатися, що всі компоненти додатку правильно взаємодіють.
- *Тестування продуктивності* - перевіряє, як швидко та ефективно працює додаток. Цей вид тестування дозволяє переконатися, що додаток може працювати з великою кількістю користувачів та великим обсягом даних.
- *Тестування безпеки* - перевіряє, наскільки добре додаток захищений від різних видів атак та зломів. Тестування безпеки може включати перевірку захисту від SQL-ін'єкцій, Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF) та інших видів атак.
- *Тестування користувацького інтерфейсу* - перевіряє, наскільки зручним та легким у використанні є інтерфейс користувача. Цей вид тестування дозволяє переконатися, що користувачі можуть легко взаємодіяти з додатком та зрозуміти, як він працює.

Додаток пройшов тестування всіх систем і показав себе на відмінно.

3.2 Розроблення back-end складової додатку

Розроблення Back-end або серверної частини, складається з таких етапів:

- *Аналіз вимог до back-end частини*: перед тим, як розпочати розроблення, необхідно ретельно проаналізувати вимоги до додатку.

– *Вибір технологій*: на основі аналізу вимог до додатку можна визначити технології, які найкраще підходять для розроблення back-end. Наприклад, ви можете використовувати мови програмування, такі як Java, JS, Python або PHP, або фреймворки, такі як Ruby on Rails, Django, Nest.js.

– *Розроблення API*: API є важливою складовою back-end складової додатку. Ви можете розробляти API вручну або використовувати фреймворки, такі як Swagger, для автоматизації цього процесу.

– *Розроблення бази даних*: для зберігання даних back-end складової додатку необхідно розробити базу даних. Ви можете використовувати реляційні бази даних, такі як MySQL або PostgreSQL, або нереляційні бази даних, такі як MongoDB або Cassandra.

– *Розроблення бізнес-логіки*: бізнес-логіка визначає, як back-end складова додатку взаємодіє з базою даних та іншими складовими додатку. Розроблення бізнес-логіки може включати в себе визначення правил обробки даних, алгоритмів обчислень тощо.

3.2.1 Аналіз вимог до додатку

Для back-end частини web-додатку «My Cash&Time» виділимо такі *основні вимоги*:

– *Доступність*: серверна частина повинна бути доступна 24/7 для забезпечення постійної роботи.

– *Стабільність*: серверна частина повинна витримувати навантаження та швидко обробляти данні користувачів;

– *Захищеність*: данні користувачів повинні бути надійно захищені.

– *Швидкість*: back-end повинен приймати та швидко обробляти запити.

3.2.2 Вибір технологій

Технологій написання серверної частини, так само як і мов програмування, існує велика кількість. Вибір технологій тісно пов’язаний

та залежить від вимог до проєкту та технічних навичок розробника. Технології можна вибирати виходячи з таких чинників:

– *Логіка*: якщо додаток потребує складної логіки та операцій з базами даних, потрібно використовувати мову з відповідними бібліотеками або фреймворками.

– *Продуктивність*: додаток повинен обробляти великий потік запитів, тому вибір технології повинен базуватись на їх продуктивності та масштабованості.

– *Сумісність з іншими технологіями*: технології повинні забезпечувати сумісність з front-end та базами даних.

В ході пошуків технологій розробки серверної частини стало зрозуміло, що для найкращої сумісності з front-end та обраним стеком технологій є Node.js.

Node.js – це середовище виконання JavaScript коду на серверному боці, яке дозволяє розробникам використовувати JS для створення back-end частини для додатків. [13,14]

В чому же його переваги:

– *Швидкість*: Node.js використовує JavaScript V8 engine, що дозволяє обробляти великий потік запитів з високою продуктивністю.

– *Незалежність*: працює на багатьох операційних системах, детальніше про це буде в розділі про сервери.

– *Масштабованість*: однією з головних вимог до технології розробки була масштабованість. Node.js має вбудовану підтримку для асинхронної обробки запитів та подій, що дозволяє легко масштабувати додатки.

– *Сумісність*: використання JavaScript на back-end та front-end забезпечує гарну сумісність та надійну роботу.

Ці переваги роблять Node.js досить популярним вибором для розробки back-end частини додатків, а особливо для додатків з великою кількістю запитів.[14]

Розроблення back-end частини додатків також не стоїть на місці, тому як новий крок еволюції технологій серверної частини з’явився Nest.js.

Nest.js – це фреймворк який базується на Node.js для створення ефективних, масштабованих програм на стороні сервера.[15, 16]

За своєю специфікою Node та Nest дуже схожі, але Nest має ряд переваг:

– *Модульність*: Nest.js базується на модульній структурі, що дозволяє розділяти функціональність додатку та використовувати різні модулі для різних задач.

– *Підтримка TS*: підтримка TypeScript дозволяє використовувати статичну типізацію та знижує кількість помилок в коді.

– *Підтримка WebSocket*: дозволяє створювати додатки в режимі реального часу.

– *Легка інтеграція*: завдяки тому що Nest.js використовує JavaScript та повністю підтримує TypeScript, дуже легко інтегруватись з іншими бібліотеками та фреймворками.[16,17]

Порівняємо два популярні фреймворки для розробки back-end, які базуються на Node.js - Express та Nest.js.

– *Модульність*: Nest базується на модульній структурі, що забезпечує гнучкість розробки. Express такої підтримки не має.

– *Підтримка TS:* Nest підтримує TypeScript «з коробки», а в Express для цього необхідно налаштовувати та встановлювати плагіни.

– *Продуктивність*: Nest «з коробки» має високу продуктивність, за рахунок використання можливостей асинхронного програмування. Express також має високу продуктивність, але вимагає більше ручного налаштування для досягнення цієї продуктивності.

– *Підтримка WebSocket*: в Nest вбудована підтримка WebSocket, що дозволяє розробляти back-end в режимі реального часу. В Express

немає вбудованої підтримки, але його можна використовувати з бібліотеками для WebSocket, такими як Socket.IO.[18]

– *Інтеграція*: обидва фреймворка добре підтримують інтеграції з іншими бібліотеками та фреймворками.

Порівнюючи всі переваги та недоліки одного та іншого фреймворка, для розробки back-end частини додатку «My Cash&Time» обрано Nest.js.

3.2.3 Розроблення API

API – це опис способів взаємодії однієї програми з іншою. Розроблення API передбачає створення інтерфейсу, який дозволяє взаємодіяти з додатком.[19[]

Для розроблення робочого API треба визначити такі кроки:

– *Визначення ресурсів*: визначення які об’єкти будуть доступні через API.

– *Визначення методів*: які методи будуть використовуватися для роботи з ресурсами, такі як GET, POST, PUT, DELETE та інші.

– *Визначення параметрів*: які параметри будуть передаватись в методах для отримання, створення, редагування або видалення ресурсів.

– *Визначення формату*: в якому форматі буде приходити відповідь – JSON або XML.

– *Безпека*: методи захисту API від несанкціонованого доступу.

Документація: має містити повний опис функціональності та опис архітектури додатку.

Під час розробки API back-end частини додатку важливо врахувати потреби користувачів та всі прописані вище вимоги.

3.2.4 Розроблення бізнес-логіки

Бізнес логіка сучасного web-додатку це сукупність правил, принципів, залежностей поведінки об’єктів предметної області.[20]

Інакше кажучи, бізнес логіка – це реалізація правил та обмежень автоматизованих операцій.

Під час розроблення бізнес-логіки важливо пам’ятати про масштабованість додатку та можливість розширення функціоналу в майбутньому. Наприклад, використовувати паттерни програмування та принципи SOLID.

Додаток «My Cash&Time» будується на основі шаблона MVC, що дозволяє відділити бізнес-логіку від користувацького інтерфейсу. В результату такого розділення додаток краще масштабується, тестується та реалізовується.

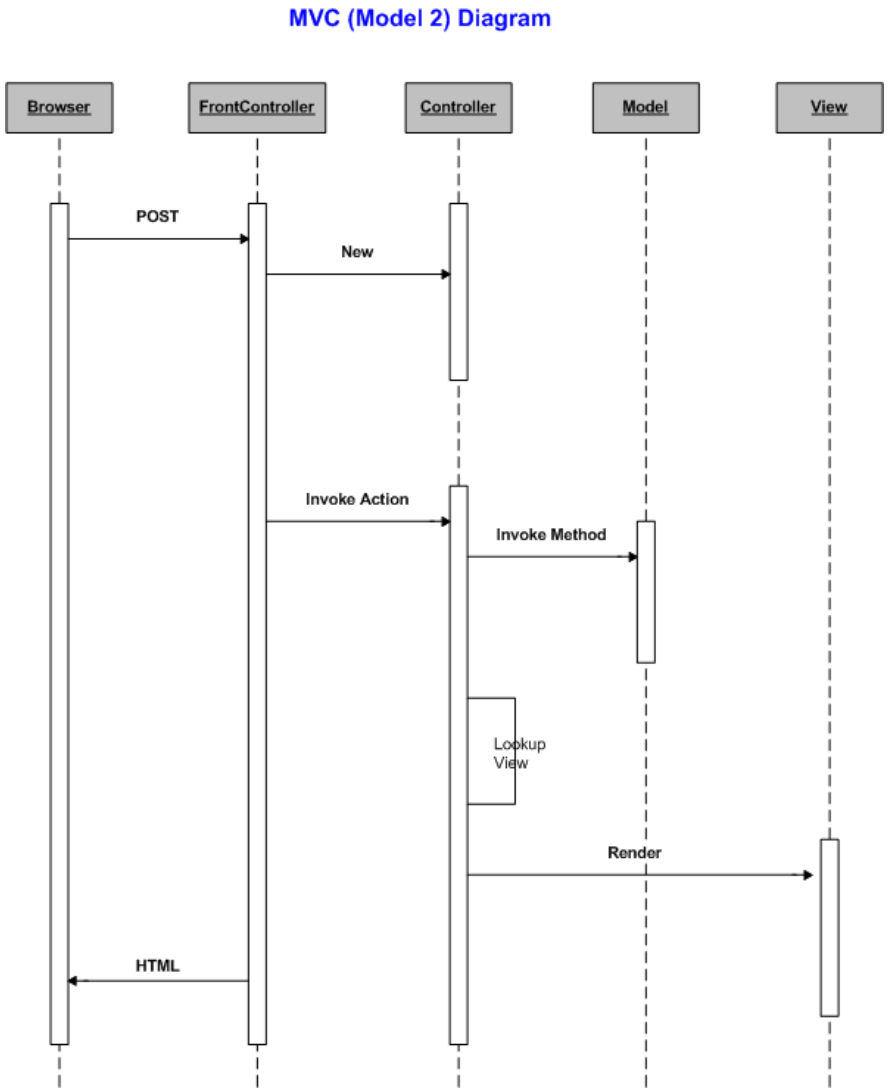

Рисунок 3.3 Відображення принципу роботи додатку за шаблоном MVC

3.3 Розроблення бази даних

Розроблення бази даних для web-додатку складається з таких етапів:

− *Аналіз вимог до бази даних*: Визначення сутностей, взаємозв'язків між ними, типів даних, що потрібно зберігати, та інших важливих параметрів.

− *Проєктування схеми бази даних*: Схема бази даних описує структуру таблиць, взаємозв'язки між ними та правила для зберігання та взаємодії з даними.

− *Розроблення моделей даних*: моделі даних - це сутності, які зберігаються в базі даних. Кожна модель даних описує тип даних, що міститься в таблиці, та зв'язки між таблицями.

− *Вибір бази даних*: в залежності від вимог до додатку та його масштабу можна вибрати різні типи баз даних. Наприклад, реляційні бази даних (MySQL, PostgreSQL, Oracle) підходять для зберігання структурованих даних, тоді як нереляційні бази даних (MongoDB, CouchDB, Cassandra) підходять для зберігання неструктурованих даних.

− *Розроблення запитів*: запити - це команди, які дозволяють вибирати, вставляти, оновлювати та видаляти дані з бази даних. Для розробки запитів можна використовувати мови запитів, такі як SQL, або ORM-бібліотеки, які дозволяють працювати з базою даних у зручний спосіб.

− *Тестування бази даних*: перед використанням бази даних в робочому середовищі необхідно переконатись що база даних відповідає всім поставленим вимогам.

3.3.1 Аналіз вимог до бази даних

База даних це та частина додатку, яка повинна лише зберігати та повертати дані користувача. Вона може мати в собі одну або декілька таблиць, тому необхідно визначити:

– *Типи даних*: визначити які дані яких типів будуть зберігатись в таблицях, буд-то string, bool, int, float та інші.

– *Взаємозв'язки*: види зв'язку таблиць в базі бувають різних типів, такі як, «один до багатьох», «багато до багатьох», «один до одного».

– *Правила для зберігання та взаємодії*.

3.3.2 Проєктування схеми бази даних

Процес проєктування бази даних і є процесом визначення структури та зв'язків між таблицями, які необхідні для зберігання та організації даних в системі.

Після аналізу вимог до таблиць, була розроблена така остаточна схема бази даних.

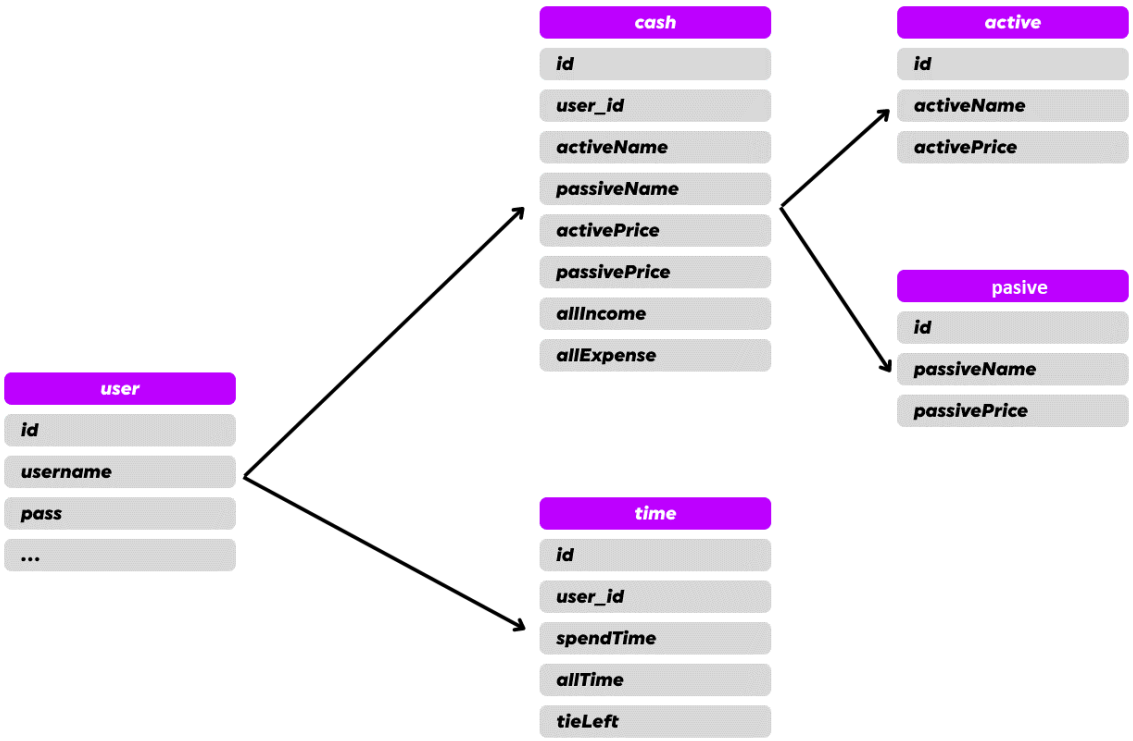

Рисунок 3.4 Остаточна схема бази даних

Дана база даних складається з декількох таблиць, а саме:

Таблиця User:

– Id - порядковий в таблиці, тип даних int.

– Username – ім'я користувача, тип даних text.

- Pass – пароль користувача, тип даних int

Таблиця Cash:

- Id - порядковий в таблиці, тип даних int.
- User_id – порядковий номер користувача, тип даних int.
- activeName – назва активу, тип даних text.
- pasiveName – назва пасиву, тип даних text.
- activePrice – вартість активу, тип даних float.
- pasivePrice – вартість пасиву, тип даних float.
- allIncome – загальна сума прибутку, тип даних float.
- allExpence – загальна сума витрат, тип даних float.

Таблиця Active:

- Id - порядковий в таблиці, тип даних int.
- activeName – назва активу, тип даних text.
- activePrice – вартість активу, тип даних float.

Таблиця Pasive:

- Id - порядковий в таблиці, тип даних int.
- pasiveName – назва пасиву, тип даних text.
- pasivePrice – вартість пасиву, тип даних float.

Таблиця Time:

- Id - порядковий в таблиці, тип даних int.
- User_id – порядковий номер користувача, тип даних int.
- SpendTime – витрачаємий час, тип даних float.
- allTime – загальний час, тип даних float.
- timeLeft – час який залишився, тип даних float.

В даній таблиці використовується тип зв’язків «один до багатьох».

3.3.3 Розроблення моделей даних

Розроблення моделей даних – це процес створення відображення даних, що зберігаються в базі даних, та відносини між ними.[21]

– *Модель сутність-відношення (ER-модель):* це найбільш поширена модель даних, яка використовується для проєктування реляційних баз даних. У моделі сутність-відношення сутності відображаються в таблиці, а відносини між ними - в зв'язки.[22]

– *Модель об'єктно-орієнтованої бази даних:* використовується для проєктування баз даних, які зберігають об'єкти. У цій моделі використовуються класи та об'єкти, які можуть мати атрибути та методи.[23]

– *Модель ієрархічної бази даних:* використовується для проєктування баз даних, які мають ієрархічну структуру. У моделі ієрархічної БД дані відображаються у вигляді дерева, де батьківський вузол може мати багато дочірніх вузлів.[24]

– *Модель мережевої бази даних:* використовується для проєктування баз даних, які мають складну структуру зв'язків між даними. У моделі мережевої БД дані відображаються у вигляді графа, де вузол може мати багато дочірніх та батьківських вузлів.[25]

В моделі сутність-відношення, або якщо коротше, ER-моделі, існує декілька основних концепцій:

– *Сутності* – це представники реальних об’єктів, які зберігаються в базі даних. Кожна сутність представлена в вигляді таблиць, де кожен запис відповідає окремому екземпляру сутності.

– *Атрибути* – це властивості сутностей, які зберігаються в базі даних. Кожан атрибут представлений у вигляді стовпця таблиці.

– *Відносини* – це зв’язки між сутностями, які відображають взаємозв’язки між даними. Відносини представляються у вигляді зв’язків між таблицями, де кожен зв’язок відображається у вигляді стовпців з зовнішніми ключами.

3.3.4 Розроблення запитів

Розроблення запитів – це процес створення запитів до бази даних для отримання потрібних даних. Запити можуть бути виконані з метою отримання інформації, аналізу даних або оновлення даних.

Запити бувають таких типів:

- *Запит на вибірку:* використовують для отримання даних з таблиці. Наприклад – SELECT * FROM TABLE.
- *Запит на оновлення:* використовується для оновлення даних в таблиці. Наприклад – UPDATE id SET username=”Name” WHERE ID=2.
- *Запит на вставку:* використовують для додавання нових даних до таблиці. Наприклад – INSERT INTO table (a,b) VALUES (c,d).
- *Запит на видалення:* використовується для видалення даних з таблиці. Наприклад – DELETE FROM table WHERE id = 2.
- *Запит на об'єднання:* використовується для об'єднання даних з двох або більше таблиць в один запит. Наприклад JOIN.
- *Запит на групування:* використовується для групування даних за певними стовпцями в таблиці. Наприклад – GROUP BY .

Наглядно процес обміну даними з базою можна зобразити так:

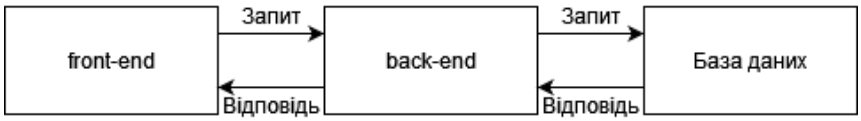

Рисунок 3.5 Схематичне зображення запитів в додатку

Якщо пояснювати всю систему запитів, то це можна охарактеризувати так:

Користувач вводить дані => front-end відправляє дані в back-end => back-end обробляє запит та відправляє в базу даних => в залежності від запиту база даних або змінює, або додає, або повертає дані в back-end =>

back-end обробляє відповідь та передає її на front => front-end відображає оброблену відповідь => користувач бачить оформлену відповідь.

3.4 Постановка на сервер

Постановка на сервер – це процес публікації додатку на сервері, щоб користувачі могли отримувати до нього доступ через мережу Інтернет. Щоб встановити web-додаток на сервер потрібно пройти декілька етапів:

- *Вибір сервера:* сервер повинен відповідати вимогам.
- *Налаштування сервера:* після вибору сервера сервер необхідно налаштувати для роботи з додатком. Це може включати в себе встановлення необхідного ПЗ, налаштування БД, налаштування безпеки та доступу.
- *Перенесення додатку:* за допомогою FTP або SSH протоколу перенести додаток на сервер.
- *Налаштування доменного імені:* для доступу користувачів до додатку необхідно налаштувати доменне ім’я через панель управління сервером або DNS-записів.

3.4.1 Вимоги до серверу

Перед завантаженням web-додатку на сервер, необхідно обрати сервер, зважаючи на такі вимоги:

- *Доступність***:** сервер має бути доступним для людей з усього світу.
- *Надійність:* сервер повинен витримувати навантаження.
- *Швидкодія:* сервер повинен швидко обробляти запити та надавати відповідь користувачеві.
- *Захищеність:* сервер повинен бути захищеним від різного роду атак, мати протокол шифрування HTTPS.

– *Місце розташування:* для кращого доступу користувачів до додатку необхідно вибирати більш нейтральне місце розташування.

3.4.2 Особливості комунікації

Особливості серверної комунікації залежать від типу та архітектури додатку. Найбільш розповсюдженим типом зв’язку є клієнт-серверна архітектура.

Клієнт-серверна архітектура – це один з архітектурних шаблонів програмного забезпечення та домінуюча концепція у створенні мережевих застосунків. Основною перевагою даної архітектури є відсутність жорсткої прив’язки до серверів, тобто запити обробляють різні сервери для одного клієнта.[26]

Для комунікації клієнт та сервер використовують протоколи комунікації.

Протоколи комунікації, або комунікаційний протокол - це обумовлені наперед правила передачі даних між двома пристроями.[27]

У контексті серверної комунікації існують різні протоколи які використовуються для обміну даними між клієнтом та сервером:

- HTTP – протокол передачі гіпертекстових документів через Інтернет.
- HTTPS (Hypertext Transfer Protocol Secure) - це захищена версія протоколу HTTP, яка використовує шифрування SSL/TLS для захисту передачі даних від несанкціонованого доступу.[28]
- WebSocket - це протокол, який забезпечує двосторонній обмін даними між браузером та сервером в режимі реального часу.[29]
- FTP (File Transfer Protocol) - протокол, що використовується для передачі файлів між клієнтом та сервером.[30]

Порівняємо протоколи зв’язку частин додатку HTTP та WebSocket:

- HTTP – це протокол запиту-відповіді, тобто користувач посилає запит, а сервер повертає відповідь. WebSocket – це двосторонній

протокол, тобто сервер може ініціювати передачу даних клієнту без запиту.

Дана особливість WebSocket чудово підходить до потреб додатку, адже при вході користувача, клієнт повинен відобразити дані про користувача без запиту самого користувача.

- HTTP передає дані у вигляді тексту, а WebSocket у вигляді бінарних даних.

Ця особливість додає переваг для використання WebSocket, адже бінарні дані набагато простіше опрацювати аніж текст.

- HTTP зазвичай використовується для запитів, які відбуваються рідко або з великим інтервалом, наприклад, форуми. WebSocket використовується для запитів в реальному часі, де необхідна швидкодія.

Спираючись на переваги WebSocket, в додатку використовується саме він.

Front-end та back-end – це дві частини одного додатку, які повинні взаємодіяти між собою. Front-end це клієнтська сторона, яка відповідає за те що бачить користувач. Back-end це серверна сторона, яка відповідає за обробку запитів від користувача та надання відповідей на ці запити.

Комунікація між частинами додатку, буде відбуватись за допомогою протоколу WebSocket як уже було зазначено раніше.

Від вибору формату даних, залежала все подальше розроблення, тому проаналізувавши всі вимоги до додатку, було обрано формат даних JSON.

JSON – легкий формат обміну даних, який дозволяє передавати структуровані дані.

Схематично спілкування частин додатку можна представити так:

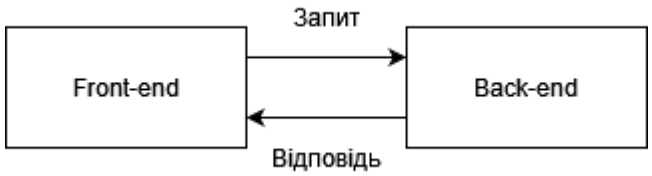

Рисунок 3.6 Схематичне зображення зв'язку Front-end та Back-end

Back-end взаємодіє з базою даних задля збереження, отримання, та оновлення даних. Так само як клієнт спілкується з сервером, так і сервер спілкується з базою даних за допомогою протоколів зв’язку.

Основні протоколи зв’язку це:

– SQL (Structured Query Language) є стандартною мовою запитів до реляційних баз даних. Запити виконуються через спеціальний сервер баз даних, який відповідає за збереження та доступ до даних.

– NoSQL (Not Only SQL) - це різновид баз даних, які не використовують SQL для запитів до даних. NoSQL бази даних зазвичай використовуються для збереження нереляційних даних, таких як документи, ключ-значення та графи.

Оскільки додаток «My Cash&Time» використовує реляційну базу даних MySQL, в додатку використовується SQL протокол зв’язку.

Для забезпечення безпеки та надійності даних, back-end має забезпечувати аутентифікацію та авторизацію користувачів, а також мати механізми для захисту від атак на базу даних, таких як SQL-ін’єкція та інших видів атак.

3.5 Написання документації до додатку

Написання документації такий же важливий процес, як і розроблення. Це необхідно робити задля збереження та передачі знань про проєкт, про його функціонал, архітектуру та інші аспекти проєктування.

Документація додатку має містити повний опис функціональності додатку включаючи опис основних функцій та їх параметрів. Чим

детальніше буде опис, тим більша аудиторія зможе користуватись документацією та додатком.

Також важливо частиною документації є опис архітектури. Опис архітектури містити:

- Детальний опис архітектури додатку.
- Опис структури.
- Розташування компонентів та взаємозв’язки між ними.

Ще одна важлива частина документації, це інструкції з використання. Вони повинні бути ясними та детальними, щоб будь який користувач зміг успішно користуватись додатком. [31]

ВИСНОВКИ

Поставлену мету – моделювання та розроблення web-додатку – досягнуто.

Розглянуто особливості моделювання front-end та back-end частин web-додатку. Виконано порівняння програмних середовищ для розроблення та огляд особливостей постановки web-додатку на сервер.

Результатом виконаної роботи є розроблений SPA web-додаток «My Cash&Time», який відповідає наступним вимогам:

- Доступність.
- Швидкодія.
- Безпека даних.
- Легка масштабованість.

Розроблений додаток відповідає всім визначеним вимогам. При цьому був досягнутий компроміс між складністю та швидкістю додатку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Наукове моделювання [Наукова стаття] URL: https://uk.wikipedia.org/wiki/%D0%9D%D0%B0%D1%83%D0%BA%D0%BE%D0%B2%D0%B5_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8E%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F
2. Моделювання даних [Наукова стаття] URL: https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8E%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85
3. Візуальне програмування [Наукова стаття] URL: https://uk.wikipedia.org/wiki/%D0%92%D1%96%D0%B7%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F
4. Система візуального проєктування та розробки додатків. Візуальне проєктування інтерфейсу. Угода про імена [Наукова стаття] URL: https://3ddroid.ru/there-are-some-advice/sistema-vizualnogo-proektirovaniya-i-razrabotki-prilozhenii-vizualnoe/
5. СУЧАСНІ МЕТОДОЛОГІЇ РОЗРОБКИ WEB-ДОДАТКІВ [Наукова стаття] URL: https://stud.com.ua/97681/informatika/suchasni_metodologiyi_rozrobki_dodatkiv
6. Учебный курс по React.js [Наукова стаття] URL: https://habr.com/ru/company/ruvds/blog/435468/
7. React.js [Документація МП] URL: https://ru.reactjs.org/docs/getting-started.html Visual Studio Code [Наукова стаття] URL: https://ru.wikipedia.org/wiki/Visual_Studio_Code

8. WebStorm [Наукова стаття] URL: https://ru.wikipedia.org/wiki/WebStorms
9. Git [Наукова стаття] URL: https://ru.wikipedia.org/wiki/Git
10. GitHub [Наукова стаття] URL: https://ru.wikipedia.org/wiki/GitHub
11. GitLab [Наукова стаття] URL: https://ru.wikipedia.org/wiki/GitLab
12. Что такое компонента и компонентная разработка? [Наукова стаття] URL: https://studfile.net/preview/9497985/page:12/
13. Node.js [Документація МП] URL: https://nodejs.org/uk/docs/
14. Node.js [Наукова стаття] URL: https://ru.wikipedia.org/wiki/Node.js
15. Nest.js [Документація МП] URL: https://docs.nestjs.com/
16. Nest.js [Наукова стаття] URL: https://uk.wikipedia.org/wiki/NestJS
17. TypeScript [Наукова стаття] URL: https://ru.wikipedia.org/wiki/TypeScript
18. Express [Документація МП] URL: https://expressjs.com/
19. API [Наукова стаття] URL: https://ru.wikipedia.org/wiki/API
20. Бизнес-логика [Наукова стаття] URL: https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B7%D0%BD%D0%B5%D1%81-%D0%BB%D0%BE%D0%B3%D0%B8%D0%BA%D0%B0
21. Модель данных [Наукова стаття] URL: https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85
22. ER-модель [Наукова стаття] URL: https://ru.wikipedia.org/wiki/ER-%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C
23. Объектно-ориентированная база данных [Наукова стаття] URL: https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0

%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85

24. Иерархическая модель данных [Наукова стаття] URL: https://ru.wikipedia.org/wiki/%D0%98%D0%B5%D1%80%D0%B0%D1%80%D1%85%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85
25. Сетевая модель данных [Наукова стаття] URL: https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85
26. Клієнт-серверна архітектура [Наукова стаття] URL: https://uk.wikipedia.org/wiki/%D0%9A%D0%BB%D1%96%D1%94%D0%BD%D1%82-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80%D0%BD%D0%B0_%D0%B0%D1%80%D1%85%D1%96%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0
27. Комунікаційний протокол [Наукова стаття] URL: https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D1%83%D0%BD%D1%96%D0%BA%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%B8%D0%B9_%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB
28. HTTPS [Наукова стаття] URL: https://ru.wikipedia.org/wiki/HTTPS
29. WebSocket [Наукова стаття] URL: https://ru.wikipedia.org/wiki/WebSocket
30. FTP [Наукова стаття] URL: https://ru.wikipedia.org/wiki/FTP
31. Как создавать и оформлять техническую документацию в IT: рекомендации для начинающих и подсказки для опытных [Наукова стаття] URL: https://techiia.com/ru/news/yak-stvoryuvati-ta-

oformlyuvati-tehnichnu-dokumentaciyu-v-it-rekomendaciyi-dlya-pochatkivciv-i-pidkazki-dlya-dosvidchenih