affairs, transport, manufacturing, where people and robotic systems cooperate, hazardous industries, energy, social management, legal institutions and more.

Currently, there is no regulatory and legal framework for the use of artificial intelligence, so its implementation is spontaneous, which leads to unpredictable results and accidents. Artificial intelligence used in critical infrastructures, in areas related to human health and life, belongs to the category of high risk. Based on the analysis and due to the impossibility of ensuring the absolute reliability of computer systems and their software, the authors do not recommend the use of artificial intelligence in areas related to safety, health and human life, especially large human teams. Devices using artificial intelligence systems should be marked with messages about its use with a clear warning about the partial reliability of the device in terms of safety and consumer responsibility for the use of such a device. The authors strongly discourage the use of artificial intelligence in responsible decision-making in areas related to the security of large groups of people.

information processing, computer systems, computer systems, algorithm, software, reliability, artificial intelligence, risks, safety of life, labor protection

Одержано (Received) 08.03.2022 *Прорецензовано (Reviewed) 17.03.2022*
Прийнято до друку (Approved) 31.03.2022

УДК 004.4 DOI: https://doi.org/10.32515/2664-262X.2022.5(36).1.124-134

В.В. Босько, доц., канд. техн. наук, **Л.В. Константинова,** викл., **О.К. Конопліцька-Слободенюк,** викл., **Д.В Фесечко,** магістр
Центральноукраїнський національний технічний університет, м. Кропивницький, Україна
e-mail: victorvv2@ukr.net, liliyashel1976@gmail.com, ksuha80@gmail.com

Аналіз та дослідження фреймворку AngularJS як засобу розробки вебсайтів

Наведено аналіз фреймворку AngularJS на підтримку використання повноцінних класів, на наявність модульної архітектури, зв'язування даних, компонентів, що пришвидшують роботу та спрощують налагодження програм, а також сильних сторін в порівнянні з іншими фреймворками.

Також в роботі проаналізовано можливості розробки вебсайтів засобами фреймворку AngularJS. Для цього було проведенно дослідження та програмну реалізацію різних типів вебсайтів засобами фреймворку AngularJS. Розглянуто його недоліки й переваги. Результатом аналізу є обгрунтування вибору фреймворку при розробці вебсайтів в залежності від поставлених задач.

комп'ютерна інженерія, вебсайт, фреймворк, AngularJS

Постановка проблеми. Фронтенд розробка є дуже важливим напрямком в IT-індустрії, так як вона є «фасадом» вебдодатку. Головними критеріями оцінок при розробці вебдодатку є його економічна складова, трудоємкість інтеграції, UI, UX, а також при розробці важливим враховується наявність документації по API. Застосування фреймворку може прискорити та спростити процес розробки вебдодатків. Тому дослідження фреймворку AngularJS, як засобу розробки вебсайтів та аналіз результатів є актуальним сьогодні.

AngularJS була випущена компанією Google у 2010 році, як новаторська технологія у світі веброзробки. Вона впровадила нові стандарти у розробку вебсайтів, та випустила ряд потужних інноваційних технологій. Ця платформа стала кроком уперед у створенні прогресивних вебдодатків. Але з часом команда Google, побачила недоліки в AngularJS, які неможливо було вирішити шляхом еволюції технологій та оновлень, і використала отриманий досвід, щоб переробити фреймворк з нуля.

Це стало переломним моментом у розвитку фреймворку. Враховуючи всі недоліки, команда Google презентувала абсолютно новий фреймворк у 2016 році, назвала його Angular 2, який увібрав в себе всі найкращі досягнення AngularJS. Тому всі версії 1.xx називаються AngularJS, а 2 та пізніші версії - Angular.

Різниця Angular і AngularJS не стосується назви або просто нових версій фреймворку. Технології відрізняються за своєю суттю. Перш за все, AngularJS заснований на JavaScript, тоді як Angular написаний на TypeScript. По-друге, також змінилася базова архітектура та підходи до прив'язки даних. Нарешті, Angular покращив продуктивність, та сумісність браузерів і впровадив підтримку розробки мобільних додатків у кількох наступних випусках.

Ключовим моментом стало оголошення Google про остаточну версію AngularJS версії 1.7 без намірів подальших випусків. Так, в 2018 році AngularJS вступила у трирічний довгостроковий період підтримки, який пізніше був продовжений на шість місяців через пандемію.

Отже, головним питанням є те, чого очікувати від припинення підтримки AngularJS після 31 грудня 2021 року. За останні роки AngularJS не отримав багато нових впроваджень, але більшість помилок були виявлені та виправлені. Проекти написані на ньому досі працюють, тому не має необхідності переходити на нові технології, якщо цього не вимагають поточні чи майбутні вимоги.

Аналіз останніх досліджень і публікацій. При аналізі останніх досліджень і публікацій [1-11] було проаналізовано переваги та недоліки розробок вебсайтів за допомогою фреймворку AngularJS.

Постановка завдання. Таким чином метою дослідження є аналіз можливостей реалізацій вебсайтів засобами фреймворку AngularJS.

Для вирішення поставленого завдання була визначена програма дослідження, що складається з наступних завдань:

1. Огляд існуючих систем для розробки вебсайту засобами фреймворку AngularJS.

2. - Дослідження системи та бібліотек для розробки вебсайту засобами фреймворку AngularJS.

3. - Визначення можливостей та функціоналу AngularJS відповідно до сучасних вимог у веброзробці.

Виклад основного матеріалу. AngularJS - це програма з відкритим кодом JavaScript від Google для розробки прикладних програм. Наступні його версії більш відомі як Angular 2 і т.д. Бувши частиною екосистеми JavaScript, AngularJS негайно налагодив зв'язок із веб-розробниками, це був перший фреймворк, який дозволяв розробляти інтерактивні вебсайти. Це призвело до розробки односторінкових додатків [4], які чудово реагували та мали чудовий користувальницький інтерфейс.

Головна область застосування AngularJS створення односторінкових додатків. Його головними перевагами на старті було те що додатки на AngularJS були більш структурованими, що полегшувало процес тестування та розробки. Під час виходу його сильними сторонами була активна підтримка Google, двостороннє зв'язування даних, взаємодія з іншими бібліотеками, та декларативне програмування, яке полегшувало підтримку та читання коду.

Що стосується AngularJS у якості SPA фреймворку, він використовує патерн проектування MVC, який добре зарекомендував себе і тепер став доступний для JavaScript світу. Він лежить в більшості сучасних фреймворків. Звичайно, особливої новизни й революції в цій події немає. Розробник з досвідом, може без жодних фреймворків спроектувати додаток, дотримуючись цього патерну. Однак фреймворк

зобов'язує розробника дотримуватися правильної архітектури. AngularJS - це структурна основа для динамічних вебдодатків. Це дозволяє використовувати HTML як мову шаблонів та розширювати синтаксис HTML, щоб чітко та лаконічно виражати компоненти програми. Зв'язування даних AngularJS та ін'єкція залежностей зменшують кількість коду і роблять його більш читабельним.

З одного боку, AngularJS має досить низький поріг входження в порівнянні з багатьма подібними рішеннями. З іншого - документація носить злегка суперечливий характер. Вона добре структурована, є приклади коду, але деякі речі висвітлені вкрай слабо. З ними доведеться розбиратися самостійно, вивчаючи вихідний код або запитуючи коментарі від колег по цеху.

AngularJS в першу чергу створений для спрощення складних процесів при створенні та керуванні JS додатків. В його основі лежить структура MVC, що робить цей фреймворк особливо корисним для створення односторінкових сайтів. Бібліотека заснована на звичайному JS і HTML, тому AngularJS автоматично піклується про маніпуляції з DOM і AJAX запити, які в іншому випадку розробникам довелося б писати самим. AngularJS можна швидко додати на будь-яку HTML сторінку за допомогою простого тегу, оскільки цей інструмент надає модульні будівельні блоки коду JS, які можна поєднувати й тестувати.

Всі ці можливості фреймворку призвели до того, що його досі активно використовують. Він був одним з перших SPA фреймворків, тому нові фреймворки багато що з нього перейняли та хоча для нових проектів краще використовувати більш сучасні рішення. AngularJS зробив великий внесок у розвиток односторінкових додатків.

Фреймворки дуже допомагають при проектуванні архітектури ПЗ, оскільки в концепціях фреймворків закладані кращі практики розробки, тому просто дотримуючись цих правил можна уникнути багатьох проблем і помилок при проектуванні додатку. Як правило фреймворки являють собою набір класів та функцій, які активно взаємодіють між собою і підтримують головну методологію фреймворку, а також надають свій функціонал для розширення можливостей додатку використовуючи свої концепції.

Також фреймворки впливають на швидкість розробки, та слідкують, щоб розробник не зробив помилок при проектуванні додатку. Оскільки вибір фреймворку такий важливий, треба звертати увагу на його підтримку, а також на кількість розробників які його використовують. Щоб при виявленні якоїсь помилки, можна було її швидко усунути та повідомити ком'юніті.

AngularJS надає дуже хороший контроль над даними на стороні клієнта. Не потрібно думати про те, щоб вибрати елементи з DOM та заповнити їх значеннями. Завдяки доступній прив'язці даних є можливість оновити дані в частині JavaScript і побачити зміни в частині HTML. Це справедливо і для зворотних дій. Як тільки користувач щось змінить в частині HTML, він одразу отримає нові значення в частині JavaScript. Також фреймворк має потужну інжекторну залежність. Існують заздалегідь визначені класи для виконання запитів AJAX та керування маршрутами.

Є ряд причин через які AngularJS виділяється серед конкурентів:

- Спрощена двостороння прив'язка даних. AngularJS дозволяє прив'язувати дані до HTML за допомогою виразів, а директиви AngularJS дозволяють розробникам розширювати функціонал HTML і створювати нові конструкції. Маніпуляції з DOM і код прив'язки даних обгорнуті в прості елементи, які можна швидко і просто вставити в HTML шаблони.

- AngularJS спроектований універсальним фреймворком, тому з його допомогою можна створити веб-додаток майже будь-якого типу. Особливо він корисний при створенні односторінкових вебдодатків.

- AngularJS входить в пакет ПЗ MEAN, який також включає MongoDB, Express.js і Node.js. Тому клієнтська та серверна частини проекту виконується за допомогою JS. У якості альтернативи для сервера можна використовувати Ruby on Rails. З AngularJS також добре стикується ASP.NET і C #.

- AngularJS побудований по техніці functionality-first, тому фреймворк найбільше підходить для розробки зверху вниз. Модульна концепція AngularJS дозволяє спростити поділ роботи на різні команди в великих проектах. У таких командах в пріоритеті мінімальна кількість коду, тому додатки AngularJS, як правило, компактні й легкі в редагуванні.

Проте є моменти, які необхідно знати при виборі AngularJS для свого проекту, вони допоможуть правильно спроектувати логіку додатку і розв'язувати поставлені задачі:

- Перше і найголовніше, AngularJS нав'язує свою структуру розробникам. AngularJS спроектований максимально дружньо, тому його інструменти інтуїтивно зрозумілі. Однак розробникам, яким потрібні гнучкі рішення, доведеться шукати обхідні шляхи.

- Для деяких проектів AngularJS надто зайвий. У таких випадках більше підійдуть легкі фреймворки для створення статичних сайтів. AngularJS також не пристосований обробляти маніпуляції з DOM з великою кількістю даних, тому, що покладається на перевірки для управління змінними DOM (будь-яка зміна змінних тягне оновлення DOM). Для більшості сайтів це не буде проблемою, але для GUI редакторів або відеоігор це може викликати проблеми.

- AngularJS також не підтримує високонавантажені галереї фото. Подібні проблеми з продуктивністю можна вирішити через форми з високим рівнем користувальницької взаємодії.

Існує ряд порад, які варто дотримуватися, щоб поліпшити швидкість додатків на AngularJS:

- AngularJS використовує так званий “дайджест цикл”, який займається оновленням DOM. Дайджест цикл додатків AngularJS - хороший індикатор його продуктивності. Його можна представити, як цикл, який перевіряє наявність змін в змінних, та займається їх оновленням. Чим коротше дайджест цикл, тим швидше додаток працює.

- Швидкість додатку залежить від кількості відстежень. Кожен раз при введенні прив'язки даних створюється все більше змінних $$watchers і $scopes, що подовжують дайджест цикл. Занадто багато $$watchers можуть викликати затримки. Тому треба використовувати їх по мінімуму.

- По можливості треба використовувати одноразові прив'язки в старих версіях.

- Доступ до DOM виконується з затратами, тому треба зберігати маленький розмір дерев DOM. Не змінювати DOM без реальної на те необхідності та не встановлювати вбудовані стилі.

- Треба стежити за областями видимості. Не треба давати змінним занадто велику область видимості, щоб збирач сміття JS міг вивільнити пам'ять.

Також варто зауважити, що AngularJS відмінно працює з Node.js. Як асинхронне подієве JavaScript–оточення, Node.js спроектований для побудови масштабованих

мережевих додатків. Де для кожного з'єднання викликається функція зворотного виклику, проте коли з'єднань немає Node.js засинає.

Це контрастує з більш загальною моделлю в якій використовуються паралельні OS потоки. Такий підхід є відносно неефективним та дуже важким у використанні. Також користувачі Node.js можуть не турбуватись про блокування процесів, оскільки немає жодних блокувань. Майже жодна з функцій у Node.js не працює напряму з I/O, тому процес не блокується ніколи. Оскільки нічого не блокується на Node.js легко розробляти масштабовані системи.

Якщо говорити про нові версії Angular, вони є більш сучасними та використовуються частіше. Це спричинено тим, що вони більш стабільні, мають більшу кількість модулів, має систему CLI, зручні в тестуванні й мають кращу структуру коду завдяки використанню TypeScript. Також варто звернути увагу на бібліотеки які використовує Angular, вони значно полегшують розробку. Серед основних переваг Angular у порівнянні з іншими фреймворками:

- Angular використовує бібліотеку RxJS, яка використовує концепції реактивного програмування та observables, для заміни функцій зворотного виклику в написанні асинхронного коду.

- Angular використовує Dependency Injection — патерн проектування, який оголошує сервіси в конструкторі компонента, що допомагає розділити структуру коду.

- Angular CLI — дуже зручний інструмент, для створення та підтримки додатку, який на початку проекту генерує все необхідне.

Серед недоліків Angular можна віднести велику кількість кода, а також складний поріг входу в фреймворк.

З випуском AngularJS або Angular 1 отримав негайну популярність та підтримку, оскільки тепер була можливість перетворювати статичні HTML-сторінки на інтерактивні, використовуючи AngularJS. Однак незабаром були випущені інші фреймворки, які почали висвітлювати недоліки AngularJS. Через жорстку конкуренцію з боку ReactJS та VueJS, Google пішов на повне перезавантаження з Angular 1 на Angular 2, використовуючи TypeScript як нову мову. Рішення перейти з JavaScript на TypeScript було прийнято для того, щоб уникнути помилок JavaScript та запровадити невелику кількість статичних типів, особливості, яку вимагали багато наявних веброзробників.

Було багато чого змінено, починаючи з Angular 2. Компанія змінила парадигму, яку використовував AngularJS, була змінена не тільки мова, а й архітектура, та підхід до прив'язки даних. Однак і AngularJS, і Angular продовжують використовувати програмісти та веб-розробники відповідно до їхніх вимог. Основні фактори, які відрізняються в Angular та AngularJS:

- AngularJS використовує архітектуру MVC або Model View Controller. Розробник розміщує бізнес-логіку в моделі, представленні й у контролері, а AngularJS виконує всю необхідну обробку для отримання результату. Основні блоки побудовані за допомогою компонентів та директив в AngularJS. Компоненти - це не що інше, як директиви із заздалегідь заданим шаблоном. Вони надають сучасну структуру додаткам, що полегшує створення та підтримку великих програм.

- AngularJS з самого початку використовував JavaScript, але для Angular 2 та пізніших версій почали використовувати TypeScript. TypeScript - це розширення JavaScript, яке забезпечує статичні типи в процесі розробки. Статичні типи дозволяють уникнути багатьох помилок, при розробці програмного забезпечення, а також покращують продуктивність, на відміну від динамічних типів, які ускладнюють використання AngularJS для великих та складних додатків.

- Ін'єкції залежностей (DI). Як AngularJS, так і Angular використовують залежність від ін'єкцій, але спосіб їх здійснення дуже відрізняється. У AngularJS DI вводиться в різні функції зв'язку, функції контролера та визначення директив. З іншого боку, Angular реалізує ієрархічну систему введення залежностей, використовуючи декларації, функції конструктора та постачальників.

- В Angular 2 існує власний інтерфейс командного рядка або CLI. Його використовують для генерації компонентів, служб тощо і навіть для швидкого та ефективного завершення проектів. Він дозволяє легко генерувати різні версії одного і того ж проекту для різних платформ з динамічною перевіркою типів. Angular CLI спрощує розробку додатків, та їх компіляцію. У AngularJS немає власного CLI.

- Використовуючи технологію прив'язки даних, Angular є більш інтуїтивно зрозумілим, ніж AngularJS. Розробник AngularJS повинен пам’ятати правильну директиву ng для прив’язки властивості чи події.

- Angular набагато швидший, ніж AngularJS. Двостороння прив'язка, яка зробила оригінальний Angular JS популярним серед веброзробників, виявила його складності, оскільки з його допомогою важко розробляти складні програми. Щоб забезпечити та здійснити двостороннє прив'язування, AngularJS продовжує перевіряти кожну масштабовану змінну за її попереднім значенням, використовуючи цикл. Angular має потокову архітектуру, де виявлення змін здійснюється за допомогою однонапрямного потоку даних, що робить програми набагато швидшими.

Angular та AngularJS це вебфреймворки, які вимагають від розробника реалізовувати MVC патерн у своїй програмі [5]. Патерн MVC (Model-View-Controller) це один з найрозповсюдженіших шаблонів проектування, який ділить логіку програми на окремі компоненти, що якісно впливає на розробку продукту, та її подальшу підтримку. Особливо зручно коли одночасно з клієнтською частиною, цей шаблон використовується на серверній частині, наприклад він є основним в багатьох серверних фреймворків: Spring MVC, NestJS, SailsJS, ASP.NET MVC Framework та інших.

При використанні шаблону MVC, програма ділиться на окремі компонента: модель, представлення і контролер. Отже, у разі внесення змін або додавання нової модифікації у компонент, він повинен мінімально впливати на інші компоненти.

Головні завдання компонентів:

- Model (модель) - відповідає за зберігання даних користувача, і надання необхідного інтерфейсу до цих даних.

- View (представлення) - відповідає за дані, які будуть надаватися користувачу, а також за форму у якій вони будуть представлені.

- Controller (контролер) - містить бізнес-логіку додатка, реагуючи на події в браузері, та повідомляючи модель про оновлені дані.

За допомогою такої внутрішньої структури, система ділиться на самостійні частини й розподіляє логіку роботи програми між компонентами для того, щоб введення даних, обробка даних, та виведення цих даних користувачу не залежали один від одного.

Якщо простежити як працює шаблон MVC більш детально, то Controller відстежує дії користувача, наприклад, введення даних, або натискання клавіші. Після чого викликається запит або функція, яка звертається до компонента Model, щоб отримати необхідні дані, наприклад, з бази даних і за допомогою компонента View, ці дані надаються користувачу у потрібній йому формі. Архітектура такого роду робить компоненти незалежними та коли користувач використає контролер, це призведе до змін в базі даних, та оновлення цих даних для користувача.

Існують також шаблони, схожі на архітектуру MVC, наприклад MVP (Model-

View-Presenter) [10] та архітектура N–Tier. Різниця між MVC та MVP, в тому, що дані у MVC, передаються з Model у View, а у MVP проходять через View. Перевагою MVC є чітке розділення логіки представлення даних і логіки програми.

Найчастіше у реалізації MVC для взаємодії між представленням та моделлю, використовують спеціальні протоколи взаємодії, використовуючи апарат події (підписка/оповіщення). Коли внутрішні дані в моделі змінюються, відсилається сигнал всім залежним шаблонам, що дані оновились. Для цього зазвичай використовуються Observer, Strategy або Factory Method, щоб обрати правильний контролер, та взаємодіяти з моделлю.

Існують різні модифікації моделі MVC, найрозповсюдженіші серед них: пасивна та активна модель. Пасивна модель — це коли контролер стежить за змінами в моделі та у разі оновлення даних, відповідає за їх зображення користувачу. Активна модель — це коли сама модель повідомляє шаблон, що її дані оновилися за допомогою системи повідомлень. Такого виду система є більш незалежна між моделлю, контролером і шаблонами.

За допомогою сучасних технології, таких як SPA та PWA, розробник може зробити взаємодію з користувачем більш інформативною та зручною. Від цього залежить кількість відвідувачів а також популярність сайту. У цьому можуть допомогти такі фреймворки як Angular, AngularJS, React та інші.

AngularJS вимагає від розробника реалізації оригінального MVC у своїх вебдодатках. Для цього він надає всі необхідні сервіси та технологію впровадження залежностей. Це допомагає Angular-додаткам мати чітку структуру, якісну підтримку, а також легко тестувати додаток на помилки.

AngularJS має великий технологічний функціонал для створення динамічних сторінок, у цьому йому допомагають: сервіси, модулі, фільтри, директиви, область застосування.

Особливу увагу слід приділити директивам в AngularJS. Оскільки вони використовуються для надання HTML нових можливостей. У процесі компіляції, директиви які знаходяться в HTML модифікують DOM, або додають йому нову поведінку. Також є можливість розробки своїх директив, як узвичаєно, їх називають за допомогою lowerCamelCase стилю.

В AngularJS також існує ієрархічна структура - scope, представлена у вигляді об'єктів. Вони використовуються для передачі даних між контролером і представленням. Вони схожі на DOM, але вони наслідують властивості батьківських scopes. Головним є $rootScope, він є батьківським стосовно всіх інших об'єктів $scope. За допомогою директиви $watch можна встановити спостереження за виразами в рамках scope, в процесі виконання фази зв'язування шаблону. Директива $Watch слідкує за даними, та оновлює їх за необхідністю.

Сервіси - призначені для виконання окремої задачі. Наприклад $http сервіс призначений для надсилання HTTP запитів і є обгорткою над стандартним XMLHttpRequest. Щоб використовувати сервіс необхідно під’єднати його до контролера, директиви, сервісу і т.д. Для створення сервісу необхідно використовувати фабричний метод factory. В AngularJS сервіси реалізуються за допомогою патерну singelton, щоб для проекту існував тільки один екземпляр, який можна викликати з кожного місця програми.

Фільтри використовуються для перебору даних перед зображенням їх користувачу, а також для фільтрації колекцій, масивів та інших типів даних.

Звдяки тому, що додатки на AngularJS завантажуються у вигляді декларативного опису, це дає змогу зручно підключати сторонні модулі, вказувати нові налаштування,

а також зручно тестувати додаток. За обробку подій, які виконують користувачі відповідає директива ng-click, яка схожа на виклик onclick. AngularJS замінює стандартний потік JavaScript на власний цикл обробки подій, за допомогою чого виконується зв'язування даних, обробки виключень, відстеження властивостей та інші операції.

Angular також використовує шаблон MVC, але його структура відрізняється. Сам фреймворк розділений на окремі модулі, кожен з яких реалізує певний функціонал. AppModule є кореневим модулем, який має набір елементів:

- Component — містить web-сторінки з HTML, CSS та основною логікою.
- Service — надає необхідні дані для компонентів.
- Directive — перетворює DOM елементи.

Компонент — надає інтерфейс та відповідає за логіку певної частини вебсторінки. Створюється за допомогою декоратора @Component(). Сервіси - надають всі необхідні дані компоненту, це можуть бути запити до сервера, або виконання певних алгоритмів. Директиви схожі на компоненти, але мають HTML-шаблони, створюються за допомогою декоратора @Directive(). Потім Angular компілює код для браузера за допомогою компілятора Angular lvy.

Представлений Angular 2.0 у 2016 виправив більшість недоліків архітектури AngularJS, та впровадив багато нових технологій, які ускладнили розробку, але зробили її надійнішою. Одним з нововведень стала підтримка розробки під мобільні додатки, нативні та вебдодатки.

Розробка структурної схеми.

Структурна схема надає інформацію про взаємодію майбутніх частин програми, за який функціонал вони будуть відповідати, та як будуть взаємодіяти між собою. Не можна сказати що вони є інформативними, оскільки на них не можна відстежити, як дані передаються та змінюються. Тому в залежності від технічного завдання, структурні схеми розробляють для окремих частин програми.

Для розробки структурної схеми зазвичай використовують метод покрокової деталізації, щоб вказати всі компоненти з яких складається схема, а також набори підпрограм та підключені бібліотеки.

Структурними компонентами програми можуть називатися підсистеми, бібліотеки, бази даних і т.д. А за допомогою зв'язків можна продемонструвати, як вони взаємодіють між собою, обмінюються інформацією, підключати нові бібліотеки. При розробці структурної схеми програмного забезпечення були розглянуті основні модулі програми, та їх взаємодія.

Дуже важливо під час розробки, особливо якщо проект великий і має складну архітектуру, поділити програму на структури, щоб показувати логіку програми, показати місця, де підключаються бібліотеки, та за що вони відповідають.

В наведеній схемі взаємодія модулів, показана в схемі ієрархії. Де до головного модуля підключаються розроблені модулі, однак схема не відбиває порядок функціонування системи. Схема доповнюється розшифровкою функцій, які виконують підключення модулів.

У структурній схемі визначені зовнішні специфікації програми, які дають розуміння функціональній частині програми, за що вони відповідають, та як вони взаємодіють з вхідними та вихідними даними. Особливо важливо розуміти тип структури даних.

Розподіл програми на окремі модулі відбувався за принципом від загального до конкретного, де окремі модулі виступали у ролі сервісів, які виконували окремі задачі. Хоча є інші варіанти створення ієрархії, перебираючи декілька варіантів та

підтримуючи реалізацію шаблону MVC, такий розподіл надає найзручніший порядок введення частин в експлуатацію, які будуть легко підтримуватися та тестуватися.

Структурна схема показує головний принцип роботи програми, її основні та другорядні блоки, їх призначення, та їх взаємодію між собою. Це допомагає у розумінні роботи додатку, що полегшує подальшу його підтримку.

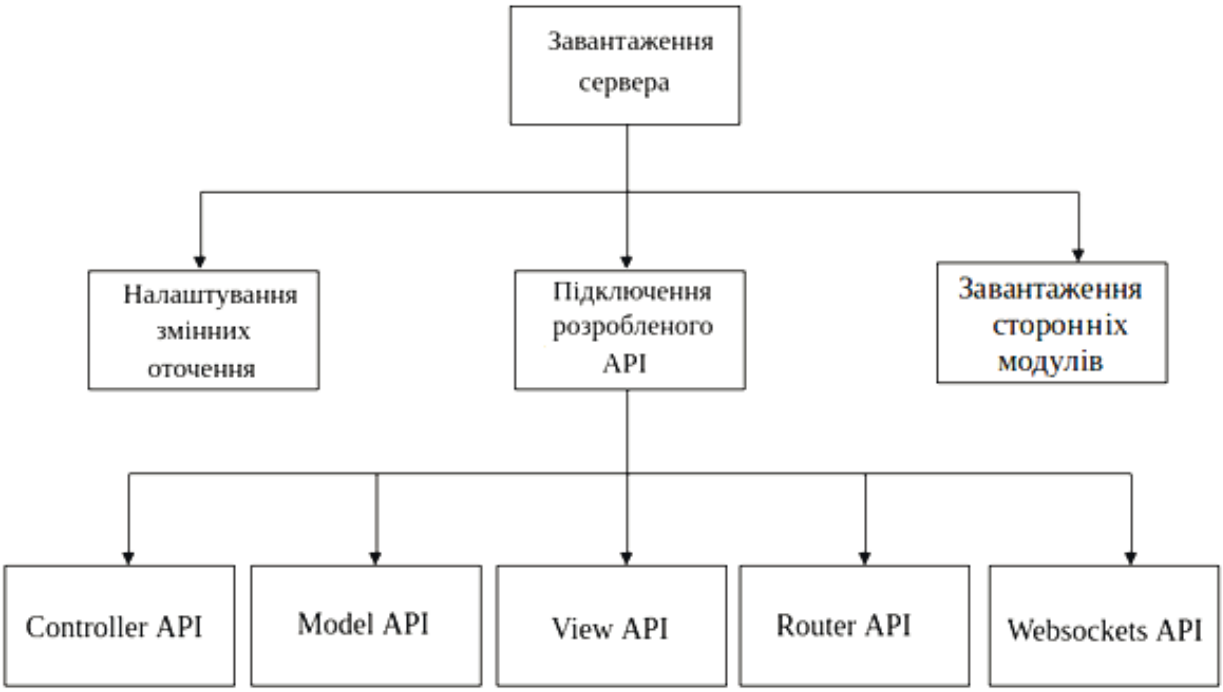

Рисунок 1 – Структурна схема

Джерело: [11]

Використовуючи всі вищезгадані технології, за допомогою AngularJS можна реалізувати вебсайт, використовуючи REST API та сокети, який буде задовільняти всім сучасним вимогам. Така архітектура та структура краще підходить для розробки малих та середніх додатків, які потрібно розробити за короткий проміжок часу, та які буде легко підтримувати.

Висновки. У дослідженні наведено теоретичне узагальнення для порівняння та виявлення кращих технологій відповідно до поставлених сучасних вимог у галузі веброзробки, та використання фреймворку AngularJS. Представлено дані, з яких можна зробити висновки, які технології сьогодні використовуються, у яких сферах, та для якого функціоналу призначені сучасні фреймворки. Які можливості та для яких проектів краще використовувати Angular та AngularJS. Рішення даного завдання полягало у вирішенні наступних задач: огляд фреймворків Angular та AngularJS, та порівняння їх функціональних можливостей для створення вебсайтів, огляд серверних технологій для розробки вебсайтів, сучасні архітектурні рішення для побудови вебсайтів. Розроблені додатки підтримують функціонал, який використовується на більшості сайтів в Інтернеті, серед основних можливостей це додавання медіафайлів, можливість надсилання e-mail повідомлень, використання сокетів для спілкування в чаті, авторизація та динамічний інтерфейс. За допомогою фреймворку AngularJS була продемонстрована технологія односторінкових додатків, яка надає можливість розробити швидкий та зручний для сприйняття інтерфейс.

Список літератури

1. Node.js v14.0.0 Documentation. URL: https://nodejs.org/api/ (accessed 5 February 2022)
2. AngularJS. URL: https://uk.wikipedia.org/wiki/AngularJS (accessed 6 February 2022)
3. AngularJS. URL: https://angularjs.org/ (accessed 4 February 2022)
4. Односторінковий застосунок. URL: https://uk.wikipedia.org/wiki/Односторінковий_застосунок (дата звернення 6.02.2022).
5. AngularJS MVC. URL: https://www.tutorialspoint.com/angularjs/angularjs_mvc_architecture.htm (accessed 5 February 2022)
6. Introduction to Node.js . URL: https://nodejs.dev/ (accessed 6 February 2022)
7. Серверное программирование веб-сайтов. URL : https://developer.mozilla.org/ru/docs/Learn/Server-side (дата обращения: 5.02.2022)
8. Янг А., Мек Б., Кантелон М. Node.js в действии. 2-е издание. СПб.: Питер, 2018. 432 с.
9. John Resig, Bear Bibeault, Josip Maras Secrets of theJavaScript Ninja : Manning, 2016. 464c.
10. Что такое MVC. URL : https://ru.hexlet.io/blog/posts/chto-takoe-mvc-rasskazyvaem-prostymi-slovami (дата обращения: 5.02.2022)
11. Создаем приложение Art-pixel на Angular и Nest.js. Часть 1. URL: https://habr.com/ru/post/647411/ (дата обращения: 4.02.2022)

References

1. Node.js v14.0.0 Documentation. *nodejs.org.* Retrieved from https://nodejs.org/api/ [in English].
2. AngularJS. *uk.wikipedia.org.* Retrieved from https://uk.wikipedia.org/wiki/AngularJS [in English].
3. AngularJS. *angularjs.org.* Retrieved from: https://angularjs.org/ [in English].
4. Odnostorinkovyi zastosunok [One-page application]. *uk.wikipedia.org.* Retrieved from https://uk.wikipedia.org/wiki/Односторінковий_застосунок [in Ukrainian]
5. AngularJS MVC. *tutorialspoint.com.* Retrieved from https://www.tutorialspoint.com/angularjs/angularjs_mvc_architecture.htm [in English].
6. Introduction to Node.js. *nodejs.dev* . Retrieved from https://nodejs.dev [in English].
7. Servernoe programmirovanie veb-sajtov [Website Server Programming]. *developer.mozilla.org* . Retrieved from https://developer.mozilla.org/ru/docs/Learn/Server-side [in Russian]
8. Yang, A., Mek, B., & Kantelon, M. (2018). *Node.js in action.* (2d ed.). SPb.: Piter. [in Russian]
9. Resig, J, Bibeault, B, & Maras, J. (2016). Secrets of theJavaScript Ninja : Manning [in English].
10. CHto takoe MVC [What is MVC]. *ru.hexlet.io.* Retrieved from https://ru.hexlet.io/blog/posts/chto-takoe-mvc-rasskazyvaem-prostymi-slovami [in Russian]
11. Sozdaem prilozhenie Art-pixel na Angular i Nest.js. Chast 1 [Create an Art-pixel app with Angular and Nest.js. Part 1]. *habr.com.* Retrieved from https://habr.com/ru/post/647411/ [in Russian]

Virtor Bosko, Assoc. Prof., PhD tech. sci., **Liliia Konstantynova,** lecturer, **Oksana Konoplitska-Slobodeniuk,** lecturer, **Denis Fesechko,** master
Central Ukrainian National Technical University, Kropyvnytskyi, Ukraine

Analysis and Research of the angularJS Framework as a Website Development Tool

The growing popularity of web application development is reaching not only developers but also entrepreneurs looking for effective business solutions. The main evaluation criteria when developing a web application are its economic component, complexity of integration, UI, UX, the ability to solve a wide range of tasks, as well as the availability of documentation from the API. Using the framework can speed up and simplify the process of developing web applications. Therefore, the study of the AngularJS framework as a tool for website development and analysis of results is relevant today.

To solve this problem, a research program was identified, consisting of the following tasks: review of existing systems for website development using the AngularJS framework; research of the system and libraries for website development by means of the AngularJS framework; defining the capabilities and functionality of AngularJS in accordance with modern requirements in web development.

It has been found that some developers think that it is better not to use third-party developments and develop web applications from scratch, while others use frameworks because it greatly simplifies and speeds up development, this technology also reduces duplication of code, simplifies tuning and speed of website development. Using HTML as a template language in Angular.js and directives allow focusing on logic processing and being more productive. They can be reused, which also increases the readability of the code. Parts of the program are located inside Angular.js modules, which are easy to handle. This breakdown allows downloading only the necessary services and performing automatic testing effectively.

The work presents an analysis of the AngularJS framework to support the use of full-fledged classes, modular architecture, data binding, components that speed up and simplify program debugging, as well as strong points compared to other frameworks.

The work also analyzes the possibilities of developing websites using the AngularJS framework. For this, research and software implementation of various types of websites using the AngularJS framework was carried out. The framework disadvantages and advantages are considered. The analysis results in the justification for the choice of the framework for the development of websites depending on the objectives.

computer engineering, website, framework, AngularJ

Одержано (Received) 07.02.2022 *Прорецензовано (Reviewed) 18.02.2022*
Прийнято до друку (Approved) 31.03.2022