Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерних наук_____

Центр післядипломної освіти

Кафедра_____Програмної інженерії_____

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

_____другий (магістерський)_____

Дослідження методів автоматичного складання веб-сайтів з використанням штучного інтелекту

Виконав: студент 2 курсу, групи ІПЗ мзд-17-1
спеціальності 121- Інженерія програмного забезпечення

Освітньо-наукової програми
Інженерія програмного забезпечення_____

_____Терела М.А._____

Керівник _____доц. Назаров О.С._____

Допускається до захисту

Зав. кафедри, проф. _____ З.В.Дудар

2019 р.

Харківський національний університет радіоелектроніки

Факультет_Комп'ютерних наук_____

Кафедра_Програмної інженерії_____

Рівень вищої освіти_____другий (магістерський)_____

Спеціальність _121-Інженерія програмного забезпечення_____

освітньо-наукова програма _Інженерія програмного забезпечення_____

освітньо-професійна програма _Програмне забезпечення систем_____

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____»_____ 20 ___ р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові _Терелі Максиму Анатолійовича_____

1. Тема роботи _ Дослідження методів автоматичного складання веб-сайтів з використанням штучного інтелекту _____
2. Термін подання студентом роботи до екзаменаційної комісії 20 червня 2019 р.
3. Вихідні дані до роботи: *синтез інструментів із набору стандартних засобів розробки для автоматизації розробки веб-сайтів, розширюваний веб-додаток. Використовувати ОС MAC OS, Python, TensorFlow, Pix2code.*
4. Перелік питань, що потрібно опрацювати в роботі *мета роботи, аналіз проблемної галузі і постановка задачі, огляд особливостей штучного інтелекту, огляд існуючих рішень по генерації веб-сайтів, структура бази даних, опис розробленої програмної системи, аналіз можливих застосувань.*
5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) *Мета, завдання, обґрунтування доцільності розроблення, постановка задачі, об'єктна модель системи, базові моделі, методи й алгоритми, структура бази даних, структурно-логічна схема взаємодії даних, план захисту інформації (за необхідністю), інтерфейс програмної системи, результати тестування програмної системи.*

Зворотний бік бланку завдання

6. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім’я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Спецчастина	доц. Назаров О.С.		

Календарний план

№	Назва етапів дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Об'єктний аналіз поставленої задачі	11 квітня 2019р.	
2	Розробка моделі взаємодії даних	19 квітня 2019р.	
3	Розробка структури зберігання даних	27 квітня 2019р.	
4	Створення коду програми		
5	Тестування і налагодження програми	25 травня 2019р.	
6	Підготовка пояснювальної записки.	26 травня 2019р.	
7	Спецчастина	28 травня 2019р.	
8	Підготовка презентації та доповіді	30 травня 2019р.	
9	Нормоконтроль, рецензування	15 червня 2019р.	
10	Занесення диплома в електронний архів	18 червня 2019р.	
11	Попередній захист	18 червня 2019р.	
12	Допуск до захисту у зав. кафедри	червня 2019р.	

Дата видачі завдання _____ _____2019 р.

Студент _____

Керівник роботи _____ __доц. Назаров О.С.__

.

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до атестаційної роботи: 107 с., 3 табл., 34 рис., 5 додатків, 35 джерел.

ШТУЧНИЙ ІНТЕЛЕКТ, МАШИННЕ НАВЧАННЯ, ШТУЧНІ НЕЙРОННІ МЕРЕЖІ, СТВОРЕННЯ ВЕБ-САЙТІВ, ВЕБ-САЙТ, ВЕБ-ДИЗАЙН, PYTHON, TENSORFLOW.

Об’єктом дослідження є сучасні методи автоматичного складання веб- сайтів з використанням штучного інтелекту. Основним завданням є реалізація дизайну та структури веб-додатку, що використовує штучні нейронні мережі.

Метою роботи є створення автоматизованої системи розробки веб-сайтів з використанням штучного інтелекту. Для досягнення поставленої мети потрібно створити алгоритм побудови веб-додатку, за допомогою якого будуть генеруватися готові рішення в реальному часі.

Методи розробки базуються на технології Python, програмна бібліотека для машинного навчання TensorFlow.

У результаті роботи був проведений аналіз предметної галузі, виявлено проблемні місця та поставлено задачу розробки готового рішення. Були сформовані вимоги та реалізовано дизайн веб-додатку та досліджено методи автоматичної розробки веб-сайтів.

ARTIFICIAL INTELLIGENCE, MACHINE EDUCATION, ARTIFICIAL NEURAL NETWORKS, CREATING WEB SITES, WEB SITE, WEB DESIGN, PYTHON, TENSORFLOW.

The object of research is the modern methods of automatic compilation of websites using artificial intelligence. The main task is to implement the design and structure of a web application that uses artificial neural networks.

The purpose of the work is to create an automated system for developing websites using artificial intelligence. To achieve this goal you need to create an algorithm for building a web application that will generate ready-made solutions in real time.

Development methods are based on Python technology, a software library for TensorFlow machine learning.

As a result of the work, the analysis of the subject area was carried out, the problem areas were identified and the task of developing a ready-made solution was set. Requirements were formulated and the Web application design was implemented and the methods for the automatic development of web sites were explored.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

АС	автоматизована система
БД	база даних
КАТ	комп’ютерне адаптивне тестування
МН	машинне навчання
МНМ	модульні нейронні мережі
МП	модульне проектування
ООП	об’єктно-орієнтоване програмування
СП	системне проектування
ШІ	штучний інтелект
ШНМ	штучна нейронна мережа

ВСТУП

Створення веб-сайтів, а також напрямки удосконалення і спрощення процесу їх розробки набувають усе більшого значення. Автоматизація розробки є основним пріоритетом для швидкого і якісного отримання результату. У зв'язку з цим питання автоматизації розробки веб-сайтів за допомогою штучного інтелекту стоять на першому місці. Актуальність теми полягає і в тому, що безпосереднє застосування штучного інтелекту (далі − ШІ) в усіх сферах розробки програмного забезпечення вимагає принципово нових рішень у створенні веб-додатків. Зокрема, існує необхідність заміни застарілої технології WEB 3.0 більш сучасним стандартом WEB 4.0.

Питання дослідження методів автоматичного складання веб-сайтів з використанням ШІ знаходять відображення в роботах таких провідних фахівців, як Дрожжинов Володимир Іванович, Райков Олександр Миколайович, Зуенко Олександр Анатолійович, Алмаматов Олександр Анатолійович, Анкіт Гупта, Кріс Констанс, Девід Космайер, Діпак Агарвал, Дмитро Суслопаров, Фернандо Алмейда, Джон Брандон, Кая Ісмаїл, Ларрі Рід, Пітер Вукчевич, Стів Джонс та інших.

Основною метою та завданням цієї роботи є дослідження особливостей створення автоматизованої системи розробки веб-сайтів з використанням штучного інтелекту. Для досягнення поставленої мети потрібно дослідити технології побудови веб-додатку, за допомогою яких відбувається процес генерації готових рішень в реальному часі. Прикладна цінність цього дослідження базується на еволюції стандарту web, внаслідок чого змінився традиційний спосіб розробки веб-сайтів. Він став більш сучасним внаслідок використання штучного інтелекту, машинного навчання та штучних нейронних мереж.

Предметом дослідження є методи автоматичного складання веб-сайтів з використанням штучного інтелекту. У рамках цього дослідження буде оглянуто базові складові розробки веб-сайтів за допомогою штучного інтелекту, проаналізовано його основні компоненти та які основні бібліотеки допоможуть

вирішити проблеми, покращити рішення, що не були реалізовані в існуючих донедавна. Окремо буде виділено нові компоненти, які дозволять в майбутньому створювати нові рішення в сфері дизайну без допомоги фахівців в області графічного дизайну.

Об'єктом дослідження є веб-сайти та підходи до їх створення в рамках програмування та графічного дизайну. В рамках дослідження буде проаналізовано сучасні тенденції в сфері Web, що дозволить дослідити появу першого стандарту та його подальшу еволюцію. В рамках аналізу буде виділено горизонт проблем та запитів користувачів, що виникають через цю тенденцію.

При вирішенні поставлених завдань застосовувалися такі методи, як метод спостереження, порівняння, узагальнення, експерименту, аналізу та інші.

Наукова новизна дослідження полягає у створенні методики розробки веб-додатків з використанням штучного інтелекту, та обґрунтуванні як прямих, так і альтернативних шляхів використання цих інструментів задля покращення існуючих рішень. З практичної точки зору буде розглянуто створення алгоритму перенесення вихідного коду в середовище систем керування вмістом (від англ. CMS), що допоможе вирішити проблему захищеності вихідних даних та їх подальшої автоматизації користувачами.

Практичне значення цієї роботи полягає у створенні нового рішення по генерації веб-сайтів та покращенні існуючих рішень. Серед основних можливостей додатку − автоматична генерація новоствореного дизайну сайту в повноцінне рішення на основі CMS, та розв’язання супутніх проблем, пов’язаних із новими можливостями. Набір інструментів, що буде отримано на цьому прикладі, може бути використано для подальшого розширення можливостей в створенні сайтів, або перенесення функцій в існуючі рішення з генерації веб-додатків.

Теоретичні та практичні результати дослідження дозволяють покращити існуючі методи генерації веб-сайтів. Результати дипломної роботи можуть бути використані підприємствами, фізичними особами, розробниками для проектування веб-додатків.

За результатами атестаційної роботи освітньо-кваліфікаційного рівня «магістр» опубліковано 3 наукові праці:

– Терела М.А. Дослідження технологій автоматичного складання веб-сайтів з використанням штучного інтелекту // Ukrainian Journal of Educational Studies and Information Technology. Vol. 7. 2019.

– Терела М.А. Порівняння розробки на основі ШІ та веб-розробника // Альманах України. №6 (27). 2019.

– Терела М.А. Мета і роль штучного інтелекту в стандарті Web 4.0 // Сучасні світові тенденції розвитку науки та інформаційних технологій: Матеріали II Міжнародної науково-практичної конференції (м. Одеса, 24–25 травня 2019 р.) / ГО «Інститут інноваційної освіти»; Науково-навчальний центр прикладної інформатики НАН України. – Одеса : ГО «Інститут інноваційної освіти», 2019.

1 АКТУАЛЬНІСТЬ ПРОБЛЕМИ РОЗВИТКУ СТВОРЕННЯ WEB-САЙТІВ ТА ПОДАЛЬША ЇХ АВТОМАТИЗАЦІЯ

1.1 Аналіз існуючих стандартів Web

Всесвітня павутина (від англ. World Wide Web) − це найбільша інформаційна мережа, за допомогою якої користувачі можуть обмінюватися інформацією, читати і записувати дані через комп'ютер та мобільні пристрої, приєднанні до Інтернету. Однак велика кількість користувачів не відрізняють Інтернет від web, які є двома окремими, але в той же час спорідненими поняттями. Інтернет – це глобальна мережа, що пов'язує комп'ютери між собою для подальшої взаємодії. Веб – це стандарт, який з кожною новою ревізією надає користувачам нові можливості взаємодії з глобальною мережею. В цій главі буде розглянуто виникнення та еволюційний розвиток стандарту web.

1.1.1 Передумови створення та поява web 1.0

Наприкінці XX − на початку XXI століття ми спостерігали появу нового стандарту Web 1.0, який допомагав, насамперед, у перенесенні друкованих засобів масової інформації, книг, музикальних, відео-композиції та ін. в мережу Інтернет.

Засновником web вважається Тім Бернерс-Лі, британський вчений і колишній співробітник CERN. 12 березня 1989 р. Бернерс-Лі написав пропозицію про створення в CERN локальної комунікаційної системи, яка в кінцевому рахунку стала всесвітньою мережею. Система призначалася для спілкування між співробітниками CERN, але в своїй пропозиції Бернерс-Лі вказав на можливість розширення мережі на весь світ [1].

У 1990 р. Бернерс-Лі і бельгійській вчений Роберт Кайо запропонували використовувати гіпертекст для зв'язку і доступу до різного роду інформації, що розміщується в веб-вузлах (веб-сайтах), які користувач може переглядати за своїм

бажанням. Бернерс-Лі закінчив створення свого першого веб-сайту та в грудні того ж року, а перший тест сайту був виконаний близько 20 грудня 1990 року, після чого Бернерс-Лі повідомив про проект на телеконференції 7 серпня 1991 року [1].

Новостворені сторінки були виключно статичні, і користувачі не могли вносити до них змін, а тільки переглядали надану. У мережі почали з'являтися особисті веб-сторінки користувачів, які переважно були розміщені на безкоштовних хостингових серверах, таких як GeoCities. Компанії почали використовувати нові можливості Web для реклами і розширення своєї продукції [2]. Головною метою веб-сайтів було оприлюднення інформації, яка буде доступна для будь-кого і в будь-який час. Наглядно принципи роботи за стандарту Web 1.0 відображено на рисунку 1.1.

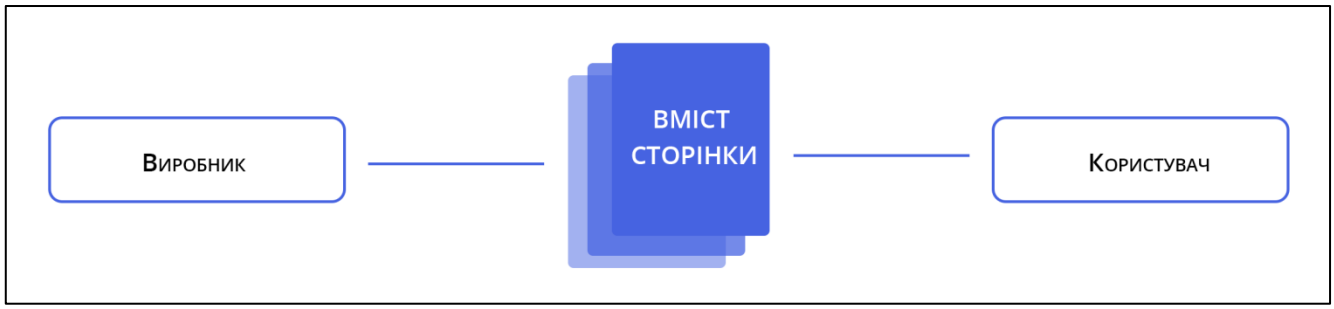

Рисунок 1.1 – Принци роботи Web 1.0

Основні веб-протоколи, що використовуються в web 1.0 – це HTML, HTTP, URI. З часом до переліку протоколів додалися XML, XHTML і CSS та ін. Важливо відзначити, що за часів появи нової технології почали активно використовувати серверні та клієнтські сценарії, так як ASP, PHP, JSP, CGI, PERL (на стороні сервера) і JavaScript, VBscript, Flash (на стороні клієнта) [3].

1.1.2 Соціальна революція через призму стандарту web 2.0

У 2004 році термін Web 2.0 офіційно був визнаний Дейлом Догерті, віце-президентом O'Reilly Media, на конференції, проведеній O'Reilly і MediaLive International. Новий стандарт було оцінено як бізнес-революцію у комп'ютерній

індустрії, викликаній переходом до більш досконалої платформи в Інтернеті [4]. Головною особливістю став перехід від статичних до динамічних веб-сторінок. На рисунку 1.2 зображена схема взаємодії користувачів між собою та з мережею.

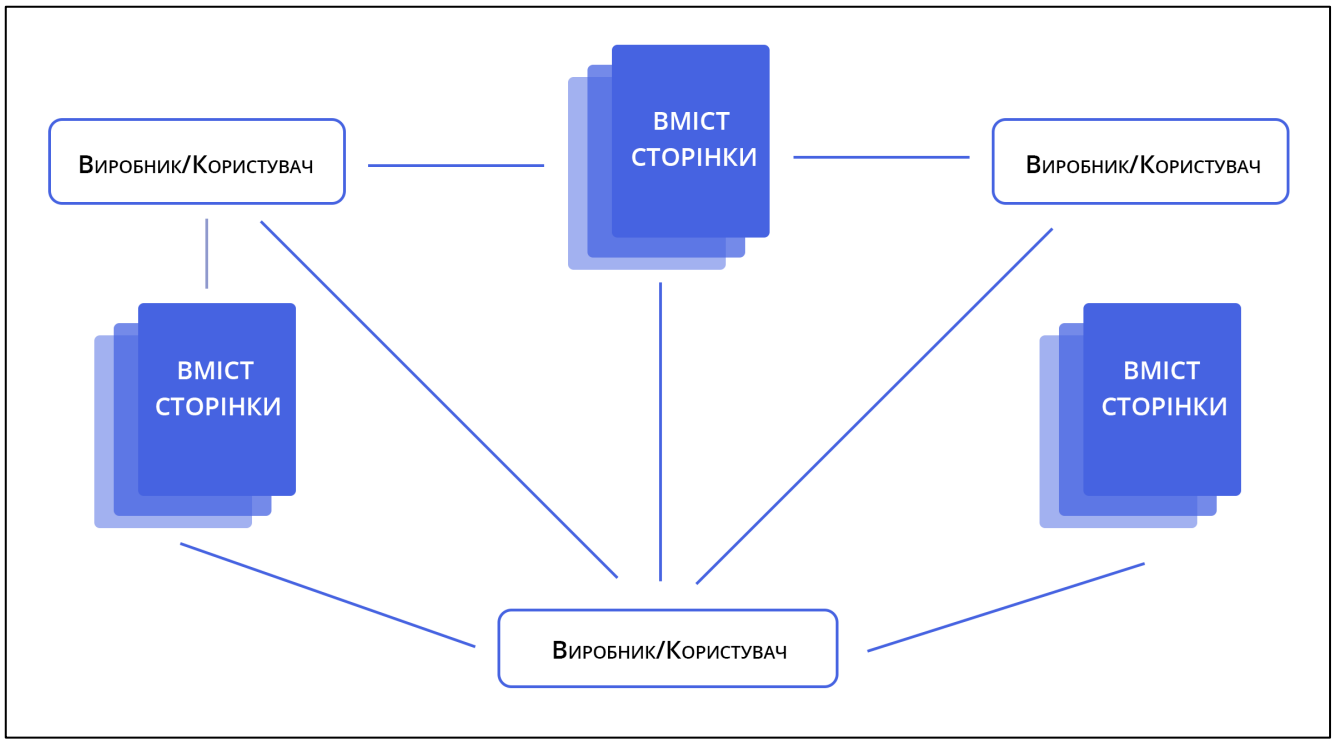

Рисунок 1.2 – Взаємодія користувачів між собою та з мережею

Орієнтований на користувача дизайн (User-centered design (UCD). Філософія Веб 2.0 повністю побудована навколо кінцевого користувача, його потреб і переваг. У цих умовах важливе місце займав дизайн проектів. Йдеться про відповідність принципів проектування веб-проектів стандарту ISO 13407: 1999 р. «Людино-орієнтовані процеси проектування інтерактивних систем». Цей стандарт визначає орієнтований на користувача дизайн (від англ. Human Centered Design, HCD), який характеризується активним залученням користувача до процесу розробки програмної системи, для досягнення прозорого розуміння призначених вимог і відповідного розподілу функцій між користувачами і технологіями, а також залучення до роботи фахівців. Під останнім мається на увазі залучення до розробки проекту різних категорій фахівців: керівників, кінцевих користувачів, програмістів, різних категорій дизайнерів, сервісних фахівців, консультантів, інструкторів та інших. При цьому важливо, щоб розробкою займалися як технічні

фахівці, так і ті, хто володіє комп'ютерними технологіями лише на рівні користувачів [5].

Web 2.0 підтримують такі веб-технології та послуги, як блоги, сайти, соціальні мережі, вікі, інструменти комунікації та ін. На думку В. І. Дрожжнікова та О. М. Райкова, також важливим є можливість розробки інтерактивних додатків нового типу [1].

Розробники використовують три основні підходи до розробки додатків в сфері Web 2.0: асинхронний JavaScript і XML (AJAX), Flex і Google Web Toolkit [2].

– асинхронний JavaScript і XML-AJAX − це підхід веб-розробки, який використовується для розробки більшості інтерактивних веб-сайтів шляхом вилучення невеликого обсягу даних з веб-сервера і відображення його в веб-додатку без перезавантаження всієї сторінки. AJAX включає кілька технологій: XHTML або HTML, каскадні таблиці стилів (CSS), JavaScript і XML.

– Flex − це набір засобів для розробки програмного забезпечення (SDK), призначений для створення міжплатформених, багатофункціональних інтернет-додатків (RIA) в Інтернеті. Flex заснований на Flash і підтримує загальні шаблони проектування.

– Google Web Toolkit − це середовище розробки Java з відкритим вихідним кодом, яка спрощує створення додатків AJAX. Це дозволяє веб-розробникам налагоджувати програми AJAX на мові Java, використовуючи засоби розробки Java за своїм вибором. GMT надає компілятор і спеціальний веб-браузер, який допомагає розробникам налагоджувати додатки [4].

1.1.3 Семантичний веб та роль високоякісних сервісів в його розвитку

У 2006 році Джон Марков в статті, опублікованій в газеті The New York Times, назвав наступний етап еволюції глобальної мережі − Web 3.0. Також цей етап називається семантичним Інтернетом, що підкреслює допомогу машин людям

в розумінні інформації в інтернеті, її агрегування і отримання з неї неочевидних висновків [1].

Основна ідея Web 3.0 − визначити структурні дані і зв’язати їх для більш ефективного виявлення, автоматизації, інтеграції та повторного використання в різних додатках. Стандарт здатний поліпшити управління даними, підтримувати доступність мобільного інтернету, моделювати креативність та інновації, підвищувати зацікавленість клієнтів і допомагати організовувати співпрацю у соціальних мережах. Особливу увагу заслуговує процес введення Штучного інтелекту. Сьогодні можна дедалі частіше зустріти нові продукти з використанням машинного навчання, нейронних мереж, що значно спрощують і полегшують процеси розробки. На рисунку 1.3 зображена архітектура семантичного вебу.

В наші дні пошукові системи можуть набагато точніше надавати інформацію, користувачі більш тісно пов’язані зі своїми колегами та друзями за допомогою веб та мобільних додатків, надані більш широкі можливості для зберігання та оброблення інформації, що робить web 3.0 корисним для користувачів мережі. Прийнято виділяти наступні ключові елементи для кращого розуміння нового стандарту:

– соціальні мережі. В останні кілька десятиліть соціальні мережі користуються величезною популярністю серед однодумців і громадських груп. За допомогою них особи виражають свої почуття, обмінюються думками та ідеями. Соціальні мережі вважаються ефективним і привабливим способом об’єднання людей по всьому світу.

– семантична мережа. Семантична мережа покликана допомагати користувачам у знаходженні нової інформації, розуміти сенс знайденого матеріалу та його контексту, в якому вони використовуються. Інформація структурована таким чином, що ШІ розуміє інформацію так само, як і людина.

– web 3D. В останні кілька років віртуальний тривимірний світ, такий як Second life, Red Light та ін. набрали величезну популярність в громадських місцях. Web 3D надає можливість людям жити у віртуальному світі, досліджувати його, знайомитися з іншими учасниками, брати участь в індивідуальній та колективній діяльності тощо [7].

Рисунок 1.3 – Архітектура семантичного вебу

Для кращого розуміння відмінностей у версіях web, потрібно дослідити ключові відмінності між ними. В таблиці 1.1 відображено порівняння основних параметрів стандартів Web перших трьох поколінь.

Таблиця 1.1 – Порівняння параметрів стандартів Web перших трьох поколінь

	Web 1.0	**Web 2.0**	**Web 3.0**
Статистика	45 мільйонів користувачів, в основному в Північній Америці та Європі (1996)	1,2 мільярда користувачів, глобальний Інтернет (2006)	4 мільярди підключено до Інтернету (2016)
Тип взаємодії	Лише для читання (статитичні сторінки)	Читання / запис	Портативний / особистий

Продовження таблиці 1.1

Джерела	Енциклопедія Britannica Online	Wikipedia	DBpedia + Linked Data Web
Фокус	Підприємства та компанії	Групи та спільноти	Lifestreams & Legations
Присутність	Домашні сторінки	Блоги	Автоматизовані послуги агрегації
Вміст	Власні дані	Спільне використання даних	Інтеграція даних
Пошукова система	Google, Yahoo	Google, Bing	Google, Bing, Yahoo та ін.
Браузер	Netscape, IE	Chrome .vs. Firefox	Операційна система, Хмара
Реклама	Медійні оголошення	Word-of-Click	Нагороди
Вимірювання	Переглядів сторінки (CPM)	Ціна за клік (CPC)	Залучення (CPA)
Бізнес-модель	PPV/Продажі	Членство / підписки	Crowdfunding / Мікроплатежі
Додавання технологій	1. Веб-сканер 2. Портал 3. Веб-браузер 4. Формат файлу 5. Файли та папки 6. iframe 7. Мова обробки даних Shell, формати даних HTML і XHTML, зберігання СУБД 8. Довідник 9. Настільні програми (JAVA, C ++ та VB) 10. Сайти 11. Веб-сервери 12. Домашні сторінки та редактори HTML	1. Пошукові системи 2. Пошук ключових слів 3. Мобільний веб 4. Метадані 5. Теги та хмари тегів 6. Віджет 7. Мова обробки даних SQL, формати даних JSON і XML і реляційне зберігання RDBMS 8. Схема 9. Багаті Інтернет-додатки (AJAX, FLEX і RIA) 11. Веб-сервіси (API для SOA, ESB & Mashups 12. i18n & L10N	1. Рекомендації двигунів 2. Семантичний пошук 3. Семантичний Веб 4. НЛП 5. Машинне навчання та AI (класифікація, кластери тощо) 6. Зв’язані дані 7. Мова маніпуляції даними SPARQL, формати даних RDF і OWL & Triple-магазини 8. Онтологія 9. Віртуальна реальність і доповнена реальність 10. Віртуальні світи 11. Електронне навчання

Кінець таблиці 1.1

	13. Переклад повного тексту 14. Онлайн-радіо та чати 15. Електронна комерція	13. Блоги та вміст, створений користувачами 14. Онлайн-відео та відеоконференції 15. Електронний бізнес	12. Адаптивний / Гібридний WebApps (HTML5, JavaScript2 і CSS3) і Мобільні програми (iOS / Android) 13. Опис логіки та правил 14. Хмарні обчислення, SaaS & IaaS

Отже, виходячи з вищезазначених даних слід проаналізувати автоматичну генерацію веб-сайтів в стандартів Web 4.0. Окрему увагу слід зосередити на ролі інтелектуальних агентів, та їх взаємодію між собою.

1.2 Автоматична генерація веб-додатків та розвиток інтелектуальних агентів в процесі еволюції Web технологій

Учені Парвата і Маріселька вважають, що поява Web 4.0 відбудеться в 2020-2030 рр. і пов’язують з його концепцією чотири технології: (I) мистецтво інтелекту; (II) нанотехнології; (III) телекомунікації; і (IV) контрольовані інтерфейси. З іншого боку, науковці Недева і Динева пов’язують концепцію Web 4.0 з наступними технологіями: (I) інтелектуальні агенти; (II) мобільні технології; і (III) послуги хмарових обчислень. Муругесан, Россі, Уілбанкс і Джаваншир стверджують, що стандарт Web 4.0 буде більш розумним і більш загальним, заснованим на агентсько-орієнтованій парадигмі. Ще одне бачення проблеми розгортає Солтісік-Пьорнукевич, який заявляє, що веб буде розвиватися в напрямку кількісного збільшення знань у поєднанні із системним підходом. Нат і Ізварі передбачають, що Web 4.0 буде ґрунтуватися на трьох основних концепціях: (I) методика розуміння природної мови; (II) нова модель зв’язку між машиною та машиною (M2M) і (III) нова модель інтерфейсу [8].

Таким чином, основу Web 4.0 складають такі концепції:

– нова модель взаємодії між машинами (англ. Machine-to-Machine –M2M). Відповідно до постулатів концепції мережа складається з інтелектуальних агентів,

розміщених в хмарному сервері. При цьому мережа здатна спілкуватися між агентами і делегувати їм повноваження.

– методика розуміння природної мови (англ. natural language understanding − NLU) − підтема обробки природної мови в штучному інтелекті, що займається розумінням прочитаного тексту машиною.

– нова модель інтерфейсу (англ. user interface – UI) – оновлена модель взаємодії користувача з інформаційною системою. Наприклад, якщо задати питання: «Як я хочу, щоб таксі приїхало за мною?», то мій телефон повинен мати можливість автоматично зв'язатися із найближчої компанією таксі без мого прямого втручання.

На рисунку 1.4 представлена концептуальна карта, побудована на парадигмі Web 4.0 з використанням методу SODA. Були визначені п'ять вимірів: (I) симбіотична мережа; (II) мережа речей; (III) соціальні мережі; (IV) поширення мережі; і (V) розподілені обчислення.

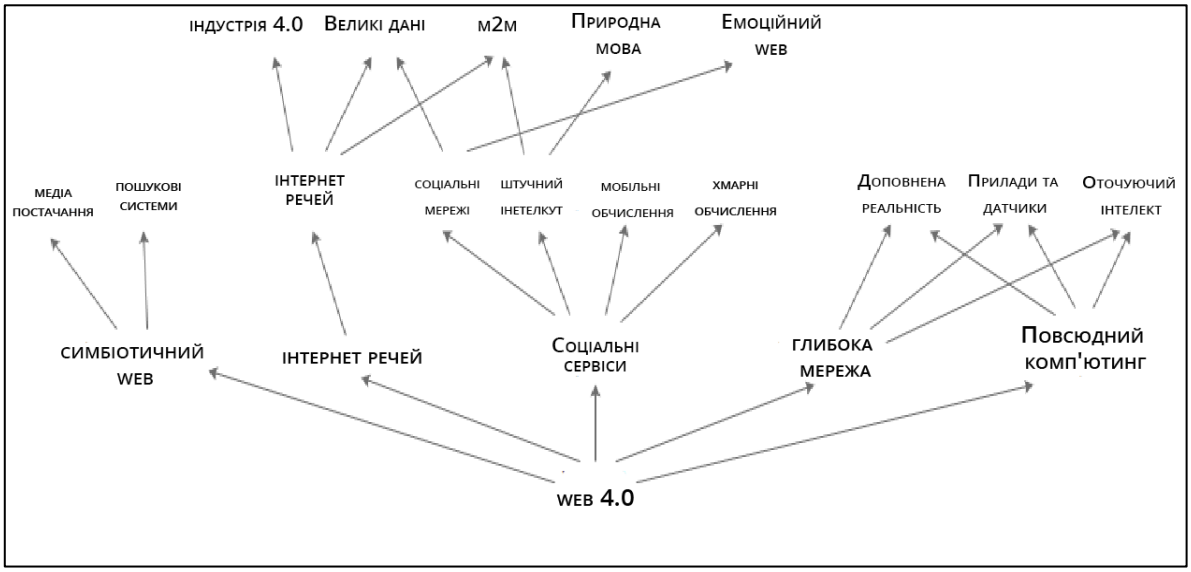

Рисунок 1.4 – Парадигма Web 4.0 з використанням методу SODA

Кожен вимір має кілька спільних елементів. Розподілені обчислення і поширення мережі мають однакові елементи, тому ці два терміни можна розуміти як синоніми. Інші загальні елементи розділяються між «Мережею речей» і «Соціальними мережами», а саме необхідність застосування алгоритмів великих даних і поява M2M-комунікацій, які походять від використання Інтернету речей і штучного інтелекту [8].

1.2.1 Розумні агенти

Як слушно зазначають В. І. Дрожжніков та О. М. Райков, під час формування нового стандарту WEB 4.0 особлива увага приділялася процесам переходу від семантичної веб-технології до метавебу. Так, у додатку на основі Web 4.0 ваші програмні агенти перебувають в мережі Інтернет чи працюють стаціонарно на вашому комп'ютері. Ці агенти можуть міркувати і спілкуватися з іншими подібними агентами і системами, працювати разом з ними для вирішення конкретного завдання. Технологію Web 4.0 слід розуміти, в першу чергу, як «інтелектуальний веб» (Intelligent Web) або «розумний веб» (Smart Web) [1].

Термін «Агент» в теорії сучасних систем штучного інтелекту − це узагальнююче поняття. Воно може позначати будь-який з природних або штучних «елементів», що володіє автономністю і здатністю рецепції конкретних аспектів свого оточення і безпосередньої взаємодії з ним. З точки зору класичного штучного інтелекту агент (інтелектуальна система) повинен був володіти глобальним баченням проблеми, мати всі необхідні здібності, ресурси і знання для її вирішення. У концепції розподіленого штучного інтелекту реалізований інший підхід. Всі агенти децентралізовані, тобто поведінка моделі в цілому не визначається якимось одним з агентів, а є результатом їх незалежної активності. По суті, агенти служать механізмами, що забезпечують співорганізацію колективної діяльності, підпорядкованої завданням вибору адекватного цільового ситуативного вирішення завдань, що мають складну управлінську структуру. Інтелектуальним агентам притаманні такі основні властивості:

– товариськість – взаємодія і комунікація з іншими агентами;

– ситуативна реактивність – адекватне сприйняття стану середовища і реакція на його зміну;

– цілеспрямованість [10].

Розглянемо основні аспекти онтології агента. Агенти можуть бути автономними або напівавтономними. Вони взаємодіють, формуючи структуровані спільноти для вирішення конкретних завдань. У агентів є певне коло завдань, вони

мають у своєму розпорядженні часткові знання про методи і функції інших агентів. Вони є «впровадженими» об'єктами, тобто володіють чутливістю лише до свого навколишнього середовища і не мають уявлень про стан всієї області існування агентів. Як і в людському суспільстві, знання, вміння і обов'язки агентів індивідуальні. Агентний підхід включає в себе різноманітність агентних типів, тому в літературі можна зустріти інше визначення – «Мультиагентне моделювання». Мультиагентні системи – це нова парадигма інформаційно-комунікаційної технології [9].

Коли справа доходить до розробки веб-дизайну, інтелектуальний агент зможе створювати першокласні веб-сайти, оскільки матиме складні знання про те, що потрібно для високорозвиненої сторінки. Виходячи з наведених даних, потрібно виділити основну модель принципи роботи інтелектуальних агентів, їх взаємодію з навколишнім середовищем та між собою. На рисунку 1.5 зображений принцип роботи інтелектуального агента [10].

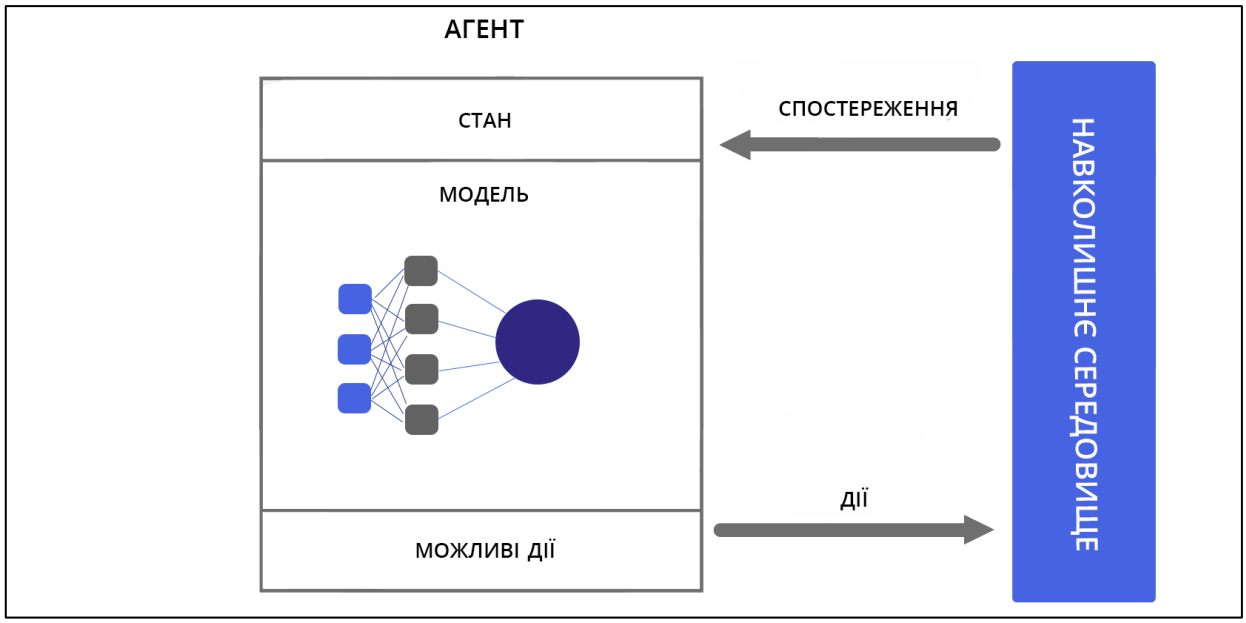

Рисунок 1.5 – Принцип роботи інтелектуального агента

Не буде «поганих» або «хороших» веб-сайтів після того, як план «найкращого веб-сайту» буде створений ШІ. Ще одна цікава річ про інтелектуальних агентів полягає у тому, що вони працюють за тими ж принципами, що й люди. Вони аналізують ситуацію, створюють план, вживають заходів, досягають певних цілей і отримують нагороди.

1.2.2 Існуючі рішення автоматичної генерації веб-сайтів

Вперше спроби розробити платформу для автоматичної генерації веб-додатків намагалася команда Grid. У 2014 році талановита команда дизайнерів і розробників запустила компанію на Kickstarter, яка була успішно профінансована. Основною метою, яка стояла перед командою, була розробка системи автоматичної генерації сайтів за допомогою ШІ, яка б враховувала цілі та призначення додатка [10].

Під час проходження рекламної компанії на Kickstarter на новітню технологію звернули увагу засоби масової інформації. Відомі технічні блоги (Mashable, Wired) та портали новин почали писати про вплив технології на ринок B2B і B2C. Один із перших інвесторів Grid Джеррі Янг описав проект, як керовану систему на базі ШІ, і додав: «Хмарний штучний інтелект Grid буде заново створювати веб-дизайн, усуваючи непотрібні, які віднімають багато часу, частини веб-розробки і створюючи елегантні веб-сайти» [11].

На думку Кая Ісмаїла, технологія штучного інтелекту Grid більше схожа на систему, яка застосовує до декількох стандартних шаблонів різноманітні кольори, додає зображення і не використовує потенціалу, про який розробники зазначали в своїй рекламній компанії. Окрім цього, велика кількість користувачів Reddit публічно заявили претензії до компанії. Одна частина користувачів просила відшкодувати кошти, а інші звинуватили компанію в шахрайстві [13].

Однак є більш вдалі платформи, такі як Wix ADI, Bookmark, Jimdo Dolphin, Firedrop. Алгоритм покладений в основу ШІ, який використовується в названих платформах, покликаний створити веб-додаток з готових численних макетів, кольорових схем, шрифтів, зображень, навігації, чатів та інших готових компонентів індивідуального веб-сайту. Під час їх створення система задає декілька ключових питань, і потім ШІ обирає ті компоненти, які найкраще комбінуються з заготовленими зразками, виходячи з потреб користувача.

Перш за все, здатність ШІ створювати веб-сайт з нуля залежить від наповнення бази даних. По-друге, передбачається, що існує простий набір

інструкцій та шаблонів, часто використовуваних в веб-дизайні. Той факт, що у Wix більше 100 мільйонів користувачів, говорить на користь всієї цієї ідеї − це швидкий, простий і ефективний спосіб створення сайту [14].

Отже, як саме працюють конструктори сайтів на основі ШІ? Хоча процес збору даних від користувача може відрізнятися, алгоритм роботи в них однаковий. Після збору інформації про користувача, задаючи прості питання про задачі та цілі користувача, ШІ починає створювати веб-сайт. Ґрунтуючись на своїх рішеннях виключно на основі цих відповідей, він може створити унікальний і повністю налаштований веб-сайт [15].

Алгоритм, покладений в систему генерації веб-додатків на основі ШІ, має доступ до багатьох макетів сайтів, дизайнів, контенту і опцій навігації. Ґрунтуючись на рішеннях про потреби та цілі користувача, а також про те, що він раніше дізнався про вас, ШІ вибирає найкращу можливу комбінацію, створюючи унікальний веб-сайт з нуля. Окремо слід відзначити, що системи здатні створювати не просто готовий дизайн, а також впроваджувати заходи безпеки для майбутнього захисту від злому [15].

Також конструктори веб-додатків на основі ШІ можуть контролювати аналітику сайту, вносити корективи або ділитися поліпшеннями і відправляти їх прямо в поштову скриньку користувача. Дуже просто вносити зміни і отриманий результат побачити безпосередньо на веб-сайті. Окрім цього, важливою функцією є проведення A/B тестування на сайті, на основі якого буде запропоновано список поліпшень, які могли б значно покращити функціональні та візуальні можливості, які ви встановили для веб-сайту [20]. На рисунку 1.6 зображено взаємозв’язок між дизайном продукту, системами штучного інтелекту та тестуванням A/B [16].

Так, під час користування Wix ADI, Джон Брендон створив веб-сайт за допомогою ключових слів «книга» і «автор» і виявив, що портал зробив велику роботу з форматування сайту, і виглядав приблизно так, як хотів автор. Портал згенерував сайт, у середині якого розміщено великий розділ для обкладинки книги або зображення, була надана можливість обрати кілька розділів сайту, додати сторінку магазину і контактну інформацію. Окрім цього, була надана можливість

додати чат і форму електронної розсилки, відстежувати реєстрації та створювати інформаційний бюлетень [17].

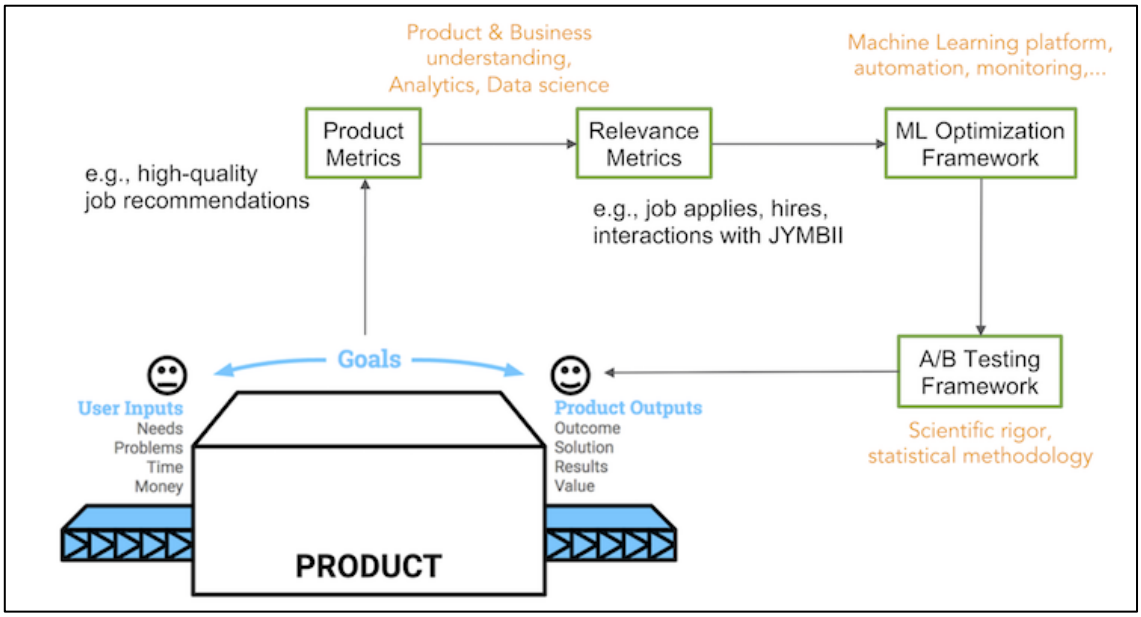

Рисунок 1.6 – Ілюстрація взаємозв'язку між дизайном продукту, системами штучного інтелекту та тестуванням A/B

Окрему увагу слід приділити функціям, на основі яких ШІ впливатиме на принципи компонування веб-додатку:

– динамічна модульна система компонентів, яка буде автоматично вибирати кращі з них (заголовки, логотипи, навігацію) на основі вибору, який задає користувач;

– генератор контенту і галузевого макета, який автоматично відображає кілька варіантів макета на основі заданого або завантаженого контенту і галузі;

– автоматичний режим реагування, який адаптує веб-дизайн до планшетного, мобільного та варіанту для смарт-годинників або, навпаки, коли користувач вирішує почати з розробки дизайну, орієнтованого на мобільні пристрої та смарт-годинники;

– аналіз і впровадження відомих веб-сайтів, які дадуть користувачеві можливість вибирати з різних макетів на основі веб-сайтів, які їм до вподоби;

– функція автоматичної генерації, яка надасть користувачеві сотні альтернативних варіантів для кращого або вибраного шаблону [12].

1.2.3 Порівняння розробки на основі ШІ та веб-розробника

Порівняємо роботу, виконану веб-розробниками і ШІ за декількома характеристиками, такими як витрати, дизайн та програмний код, а також за ефективністю використання часу.

– Економічна ефективність – це одна з найголовніших ознак, яка цікавить більшість користувачів. За допомогою систем на основі ШІ, можна створити і запустити свій веб-сайт за мінімальною ціною. Це більш доступне рішення, в порівнянні з професійними командами веб-розробників.

– Коли справа доходить до ефективності часу, немає нічого, що могло б перемогти творця сайту на основі ШІ. Він перевершує навіть найдосвідченіших фахівців у цій галузі. Це просто ще один інструмент, який наближає створення сайтів до повністю автоматизованого процесу.

– Першість з дизайну і програмного коду, як і раніше, дістається розробникам. Системи на базі ШІ не можуть створювати код так само, як професійні програмісти. ШІ пише код без урахування семантичної розмітки, що важливо для структурування даних і посилення їх внутрішнього значення.

– Те ж саме стосується і проектування. ШІ насправді не розробляє ваш сайт, а вибирає дизайн зі зразків у базі даних. Це означає, що веб-дизайнери повинні створювати нові шаблони і наповнювати базу даних, які будуть коректно застосовуватися до будь-якого контенту [10].

З приводу автоматичної генерації веб-додатків висловив свою думку генеральний директор Cortica Ігаль Райхельгауз, а саме: генератори на основі ШІ порушують загальні правила побудови веб-додатків. Однак він відзначає, що ШІ можна навчити цим правилам, утім є випадки, коли правила потрібно порушувати для отримання унікального результату, щоб сайт ставав унікальним і неповторним. Програмне забезпечення можна навчити порушувати правила час від часу, але для цього потрібно постійно вдосконалювати систему. Однак навчити порушувати типові правила за допомогою несподіваних новацій, які використовують в своїй роботі провідні дизайнери, доволі важко відтворити [1].

Отже, виходячи з вищезазначених постулатів, немає сумнівів у тому, що сайти розроблені за допомогою ШІ, надзвичайно корисні і прості в створенні. Кількість підприємств, які бажають створити свої веб-сайти, зростає. Згідно з дослідженням, проведеним Clutch, в 2016 році 46 відсотків малих підприємств не мали веб-сайтів, але це число скоротилося до 29 відсотків всього за один рік [1]. Такий величезний попит змусить людей, які керують цим бізнесом, шукати дешеві і швидкі рішення. Слід підкреслити, що веб-сайти, розроблені на основі ШІ, обтяжені великою кількістю проблем, таких як неструктурований програмний код, який буде доволі складно в подальшому змінювати. Утім, для більшості користувачів такий варіант буде прийнятним.

Це також означає, що штучний інтелект в основному здатний справлятися із заданими завданнями так само, як початкові веб-розробники, і що робота останніх може виявитися під загрозою в недалекому майбутньому. Але великі відомі бренди, ймовірно, не будуть задоволені дешевим посереднім зовнішнім виглядом і обмеженим інтерфейсом. Таким чином, у майбутньому масове виробництво веб-додатків буде здійснюватися за допомогою ШІ. Водночас, першокласні, зроблені за індивідуальним замовленням сайти будуть замовляти компанії виключно з високим рейтингом.

2 ОГЛЯД ОСНОВНИХ ПОНЯТЬ ШТУЧНОГО ІНТЕЛЕКТУ ТА ПРОГРАММНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗРОБКИ

Автоматизація різноманітних робіт відбувалася упродовж багатьох століть, але за декілька останніх десятиліть вона набула величезних масштабів. Дійсно, Нік Срнічек і Алекс Вільямс − автори книги «Inventing the Future», передбачають, що від 47 до 80 відсотків робіт, які зазвичай виконують люди, можуть бути автоматизовані протягом наступних двадцяти років [1].

Зазвичай під автоматизацією розуміють спрощення простих фізичних завдань, які можна розкласти на кілька простих кроків. Але якщо взяти сучасні процеси автоматизації в будь-якій сфері, можна переконатися, що це далеко від істини. Відтоді, як компанія Apple продемонструвала світу Macintosh у 1984 році, ця сфера діяльності значно змінилася. Нові досягнення у сфері інформаційних технологій, такі як веб-сайти, програмне забезпечення стали результатом порівняно нових технологічних розробок. Так, наразі програмісти використовують усі сучасні «ноу-хау» від розробників програмного забезпечення. Але виникає логічне запитання: чи можна процес розробки веб-додатків еволюціонувати? Чи зможе ШІ повністю замінити роботу програмістів та веб-дизайнерів?

Для початку потрібно розглянути алгоритм роботи виконання програмним забезпеченням генерування веб-сайтів на основі ШІ. По-перше, потрібно визначити загальну ціль для системи ШІ, наприклад, «надати нові можливості працевлаштування для членів, які відповідають їх навичкам і інтересам», або «надати рекрутерам список кандидатів, які відповідають заданим критеріям пошуку і, ймовірно, приведуть до успішного влаштування на роботу».

По-друге, слід мати набір проміжних метрик, які використовуються в якості проксі для визначення того, наскільки добре система досягає своєї мети. Це є необхідним тому, що вихідні метрики продукту (наприклад, успішний найм) не є чимось, що алгоритм машинного навчання може легко оптимізувати. Як результат, є створення алгоритму, який покращує початковий спосіб отримання результатів з вихідних даних [2].

2.1 Теоретичні положення штучного інтелекту та його основних компонентів

2.1.1 Штучний інтелект

Згідно з тлумаченням Оксфордського словника англійської мови, «штучний інтелект – це теорія та розвиток комп'ютерних систем, спроможних виконувати завдання, які зазвичай потребують людського інтелекту, такі як візуальне сприйняття, розпізнавання мови, прийняття рішень та переклад іноземних мов» [21].

Потрібно відзначити, що на основі 22-ї Національної конференції зі штучного інтелекту веб-аналітики сформулювали список областей дослідження штучного інтелекту у сфері web:

– ШІ для веб-сервісів: семантичні описи, планування, зіставлення і координація;

– ШІ для веб-співробітництва та співпраці;

– агенти і багатоагентні системи в Інтернеті;

– мовні технології для веб-системи, включаючи вилучення інформації, відповіді на питання, узагальнення тексту, машинний переклад;

– інтеграція інформації в мережі;

– інтелектуальні інтерфейси для веб-систем;

– мови, інструменти і методології для подання, управління та візуалізація даних семантичної мережі;

– аналіз посилань і аналіз графів в мережі;

– машинне навчання та Інтернет;

– розпізнавання веб-спаму;

– пошук, запит, візуалізація та інтерпретація семантичної павутини;

– соціальні мережі та ідентифікація спільноти та ін. [19].

Перед поглибленим розглядом технології з генерування веб-додатків за допомогою ШІ, важливо зробити крок назад і подивитися, як працюють ці алгоритми.

По-перше, потрібно визначити загальну мету для системи ШІ, наприклад, «надати нові можливості працевлаштування для членів, які відповідають їх навичкам і інтересам» або «надати рекрутерам список кандидатів, які обидва відповідають заданим критеріям пошуку і, ймовірно, приведуть до успішного влаштування на роботу».

По-друге, слід мати набір проміжних метрик, які використовуються в якості проксі для визначення того, наскільки добре система досягає своєї мети. Це є необхідним тому, що вихідні метрики продукту (наприклад: успішний найм) не є чимось, що алгоритм машинного навчання може легко оптимізувати. У нашому прикладі ці показники можуть включати в себе кількість учасників, які відносяться до наданих їм вакансій, кількість підтверджених співробітників, кількість учасників, які натискають на списки вакансій і та ін.

Як результат, є створення алгоритму, який покращує (відповідно до ваших показників релевантності) існуючий метод отримання результатів з вихідних даних [13].

Слід також зазначити, що окрім розробки рішень введення штучного інтелекту для генерації веб-додатків, такі корпорації як Google, Facebook і інші подібні компанії розробили набір інструментів для штучного інтелекту, що включають в себе готові функції машинного навчання в веб-додатках [17].

Це надає в першу чергу розробникам веб-додатків змогу інтегрувати готові рішення в веб-платформи і мобільні технології. Розробникам не потрібно вивчати інші мови програмування, щоб скористатися перевагами цієї технології. Замість цього вони працюють з API і інструментами, наприклад, Python, Ruby, C++, Java, .Net, Node.js, JavaScript, CSS, HTML [20].

Послідовність в розробці веб-сайтів на основі HTML стала інтеграцією алгоритмів на основі ШІ. Фактично, робота над платформою і підтримка ШІ дозволяють веб-розробникам приймати рішення, пов’язані з дизайном, версткою і контентом. Це дозволяє створювати веб-інтерфейси, які можуть ефективно охоплювати всю аудиторію [18]. Також, веб-розробники, ймовірно, можуть попрощатися з деякими з найважчих аспектів своєї роботи. Алгоритми ШІ незабаром зможуть самостійно управляти операціями тестування і забезпечення

якості, наприклад, звільняючи співробітників-людей для більш ефективного використання свого часу і творчої енергії [21].

2.1.2 Машинне навчання

В сучасному світі постають важливі питання щодо методів навчання. З розвитком технологій термін «навчання» постійно розширюється, знаходячи застосування не лише серед навчання людей, а й серед навчання машин. Машини, обладнані штучним інтелектом, можуть замінити людину у виконанні небезпечних та складних завдань, де необхідна велика сила, швидка увага, або обробка великої кількості схожої інформації [23].

Щоб вирішити задачу за допомогою комп'ютера, необхідно використати певний алгоритм – набір інструкцій, що виконуються та обробляють вхідні дані для отримання вихідних. Є задачі, для вирішення яких алгоритму немає. Це задачі, в яких програміст не може описати ланцюжок дій та умов, при яких отримуються правильні вихідні дані, за умови будь-яких вхідних. Наведемо простий приклад з фільтруванням спаму в електронній скриньці. Листи, що повинні потрапити під категорію спаму, мають різний характер. Крім того, це залежить від власних уподобань особи – власника поштової адреси. Щоб машина – комп'ютер – могла відрізняти корисну кореспонденцію від сміття, її треба навчити. Найпростіший спосіб – це дати на аналіз велику кількість листів, відібраних користувачами як спам, і позначити їх. Таким чином машина буде знаходити певні закономірності та правила, виробляючи алгоритм самостійно. Таким чином працює багато прикладного програмного забезпечення, що не має чіткого визначеного алгоритму. Отже, машинне навчання (Machine Learning) – це підрозділ комп'ютерних наук, що забезпечує комп'ютери можливістю самостійно вчитися і виконувати певні дії без чітко вказаних програм [23].

Машинне навчання є новою тенденцією в наші дні і являє собою застосування штучного інтелекту. Воно використовує певні статистичні

алгоритми, щоб змусити комп’ютери працювати певним чином без явного програмування. Алгоритми отримують вхідне значення і прогнозують вихід для цього з використанням певних статистичних методів. Основною метою машинного навчання є створення інтелектуальних машин, які можуть мислити і працювати, як люди [20].

Метод глибокого машинного навчання отримує все більше поширення в порівнянні з іншими методами машинного навчання. Така ситуація пов’язана з тим, що в «традиційному» машинному навчанні людині необхідно самостійно вибирати з усього об’єму інформації. Такі дані називаються ознаками. Якщо такі ознаки виявити не вдалося, то машинне навчання працює погано. Неякісні мережі автоматично відключають необхідні дані від незначних і уміють визначати правильні ознаки. До недоліків можна віднести необхідність у обробці великого обсягу даних.

2.1.3 Штучні нейронні мережі

Нейронні мережі можуть використовуватись для вирішення різних задач, таких як розпізнавання образів, апроксимація функцій, задач управління, обробка зашумлених даних та інших. Оскільки для навчання нейронних мереж потрібні лише тренувальні дані, характер взаємозв’язків між входами та виходами не повинен бути відомий. Це є значною перевагою особливо в випадках, коли такий взаємозв’язок є складним. Також нейронні мережі здатні до узагальнення – у разі успішного навчання нейронна мережа поверне правильний результат даних, що не входять до навчальної вибірки [22].

Іншими словами, штучні нейронні мережі – це обчислювальні парадигми, які реалізують спрощені моделі біологічних нейронних мереж (БНМ). Під БНМ будемо розуміти локальні ансамблі нейронів, які об’єднані синаптичними зв’язками. Сукупність таких ансамблів формує мозок із його різноманітними функціональних можливостями [23].

Сьогодні відома велика кількість нейронних структур та їх модифікацій, що орієнтовані на вирішення конкретного типу задач. Найбільш відомі типи таких структур показані на рисунку 2.1 [23].

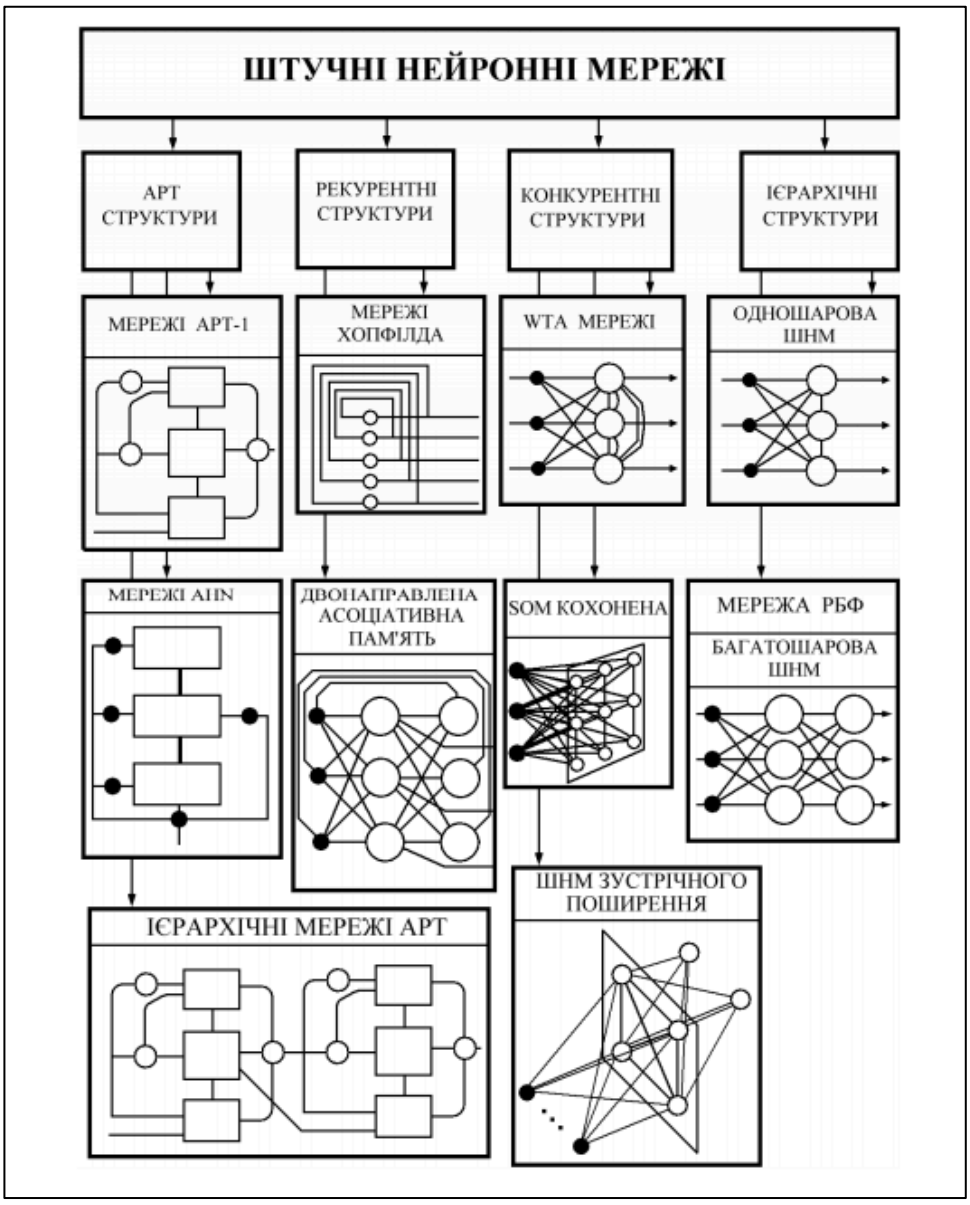

Рисунок 2.1 – Нейронні структури

Виділяють три парадигми навчання: з учителем, самонавчання і змішану. У першому способі відомі правильні відповіді до кожного вхідного прикладу, а ваги підлаштовуються так, щоб мінімізувати помилку. Навчання без вчителя дозволяє розподілити зразки по категоріях за рахунок розкриття внутрішньої структури і природи даних. При змішаному навчанні комбінуються два вищевикладених підходи. Існує велика кількість алгоритмів навчання, орієнтованих на вирішення різних завдань. Серед них виділяють алгоритм зворотного зменшення помилок,

який є одним з найбільш ефективних сучасних алгоритмів. Його основна ідея складається в тому, що зміна ваг синапсів відбувається з урахуванням локального градієнта функції помилки. Різниця між реальними і правильними відповідями нейронної мережі, які визначаються на вихідному шарі, поширюється в зворотному напрямку назустріч потоку сигналів (див. рис. 2.2).

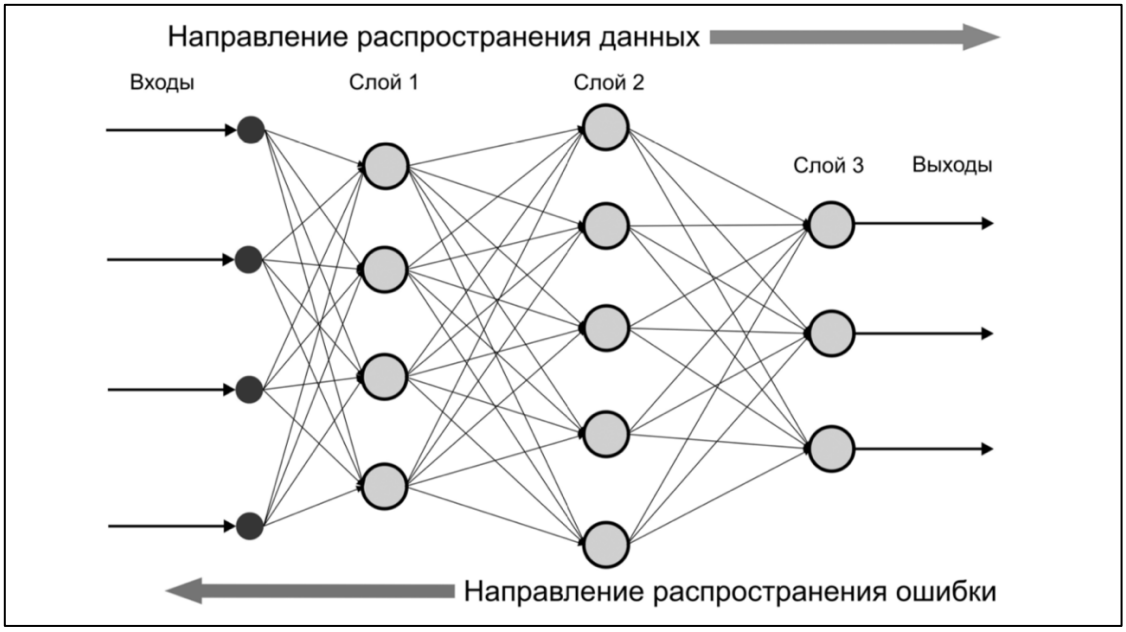

Рисунок 2.2 – Метод зворотного поширення помилки для багатошарової повно нейронної мережі

В підсумку кожен нейрон здатний визначити внесок кожної своєї ваги в сумарну помилку мережі. Це означає, що найпростіше правило навчання відповідає методу найшвидшого спуску. Тобто зміни синоптичних ваг пропорційні їхньому внеску в загальну помилку і надають можливість взаємодії між ними [34].

2.2 Програмне забезпечення

2.2.1 Python

У програмі на мові Python будь-яка сутність представлена спеціальним класом, і будь-який об’єкт, який використовується в програмі, представлений його примірником. Фактично є велика схожість з підходом, що застосовується в C ++.

Для роботи з програмами доступні кілька режимів роботи:

– інтерактивний режим (режим діалогу). В даному випадку інтерпретатор обробляє, як команди, рядки, що вводяться користувачем.

– виконання інструкцій з файлу. Інтерпретатор обробляє команди, що містяться в переданому файлі. Програми можуть бути як звичайними скриптами, так і працювати у фоновому режимі процесами.

У програмах на мові Python можна використовувати функції, написані на C, що виявляється корисним для прискорення низькорівневих операцій.

Python має засоби функціонального програмування: генератори, ітератори, вбудовані функції (map, reduce, filter, apply).

Існують бібліотеки для розподілених / паралельних обчислень з використанням мови Python. Також написано безліч фреймворків для роботи з кластерами, хмарними сервісами, grid-мережами.

Є бібліотеки і вбудовані засоби для здійснення мережевої взаємодії, віддаленого запуску і виклику функцій (RPC). За допомогою зазначених бібліотек існує можливість побудови RPC модели. Існують бібліотеки і фреймворки як для синхронного, так і для асинхронного режимів роботи. Окрім зазначених вимов слі також зазначити про використання сокетів. Підтримується взаємодія між клієнт-серверними додатками з використанням сокетів, інтерфейсів COM, є вбудована бібліотека з реалізацією RPC, велика кількість високорівневих протоколів на основі яких будується нейронна мережа [24].

2.2.2 Django

Django − найпопулярніший фреймворк мови Python з широкою аудиторією користувачів - розробників, що означає досить велику кількість готових рішень. Щороку випускаються нові версії фреймворка, зараз 1.9. Django має ряд переваг над іншими фреймворками, це:

– зрозуміла документація − безліч прикладів, пояснень і, найголовніше − відкритий вихідний код, який дуже добре написаний.

– вбудований ORM (Object-relational mapper) − існує безліч більш гнучких і потужних бібліотек, але Django ORM справляється зі своїм завданням дуже навіть добре. Великий плюс цього ORM − це повна відсутність SQL - синтаксису. Наприклад, замість: SELECT * FROM «Users» WHERE id = 2 Використовується: Users.objects.get (id = 2).

– автоматично генерується панель адміністратора − це є одним з найголовніших переваг фреймворка. Функціональність дозволяє значно скоротити час на написання потрібного інтерфейсу адміністратора, а також дозволяє відразу керувати базою даних.

– підтримка Model-Template-View. Схожий з класичним MVC і дозволяє добре відокремлювати бізнес-логіку від дизайну. Дуже проста і зручна технологія для виведення шаблонів на браузер.

– висока швидкість роботи. Незважаючи на те, що мова програмування це python, Django має вбудовані засоби кешування сторінок і розподілу навантаження. В цілому в фреймворку Django все строго розподілено. Методи можуть зберігатися тільки в одному файлі, моделі в іншому, форми в іншому і т. п. При розробці на ньому програмісту не потрібно думати про архітектуру проекту, все заздалегідь продумано розробниками фреймворка. Хоча для когось це може видатися перевагою, для іншого – навпаки, недоліком. Отже, потрібно використовувати зазначено конструкцію лише у виняткових випадках [25].

2.2.3 Pix2code

Всі компоненти в системі Pix2Code постачаються з графічним інтерфейсом, за допомогою якого відбувається перетворення тексту на доменно-специфічну мову (DSL). Code2Pix приймає текстові описи в DSL і генерує GUI. Таким чином,

Code2Pix по суті є «компілятором» глибокого навчання, здатним відображати інтерфейси (див. рис. 2.3) [26].

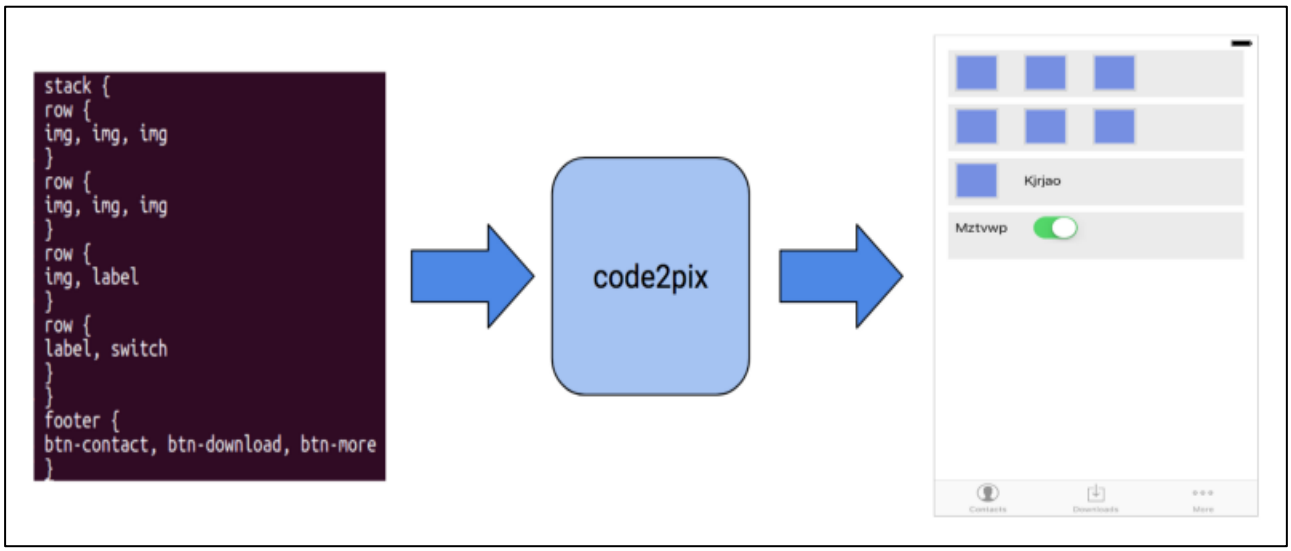

Рисунок 2.3 – Алгоритм генерації готового рішення за допомогою бібліотеки code2pix

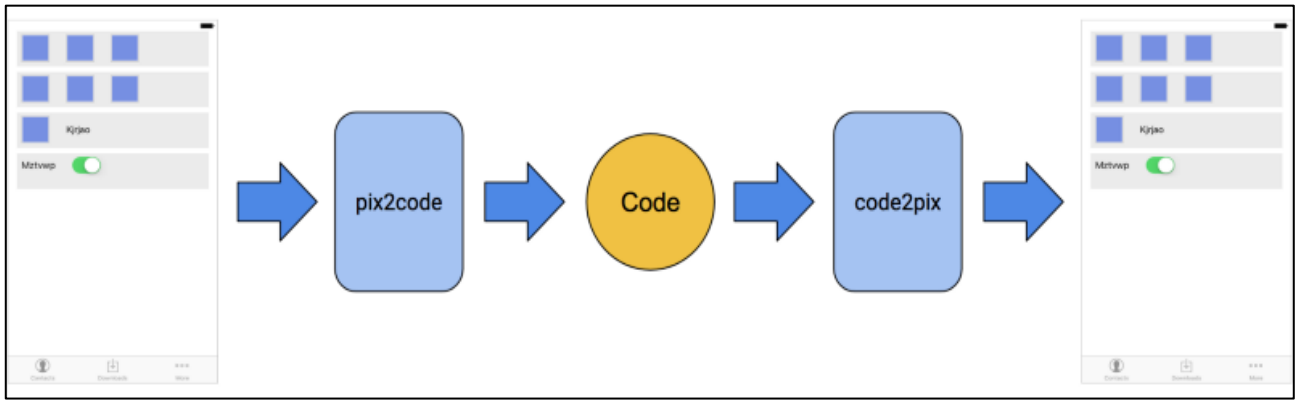

Рисунок 2.4 – Pix2Code архітектура

Code2Pix виконує ту ж саму роботу, що і будь-який звичайний рендер. Окремо слід відзначити, момент генерування макету до стандарту W3C. Фактично, будь-який засіб візуалізації, який обробляє код призначеного для користувача інтерфейсу, є системою «програмного коду в піксель» розробленою компанією Mozilla. Однак Code2Pix відрізняється від конкурентної технології тим, що вона диференційована. Це означає, що ми можемо використовувати Code2Pix як інструмент глибокого навчання для навчання інших ідентичних моделей шляхом поширення сигналів про помилки (див. рис. 2.4 та 2.5) [26, 27].

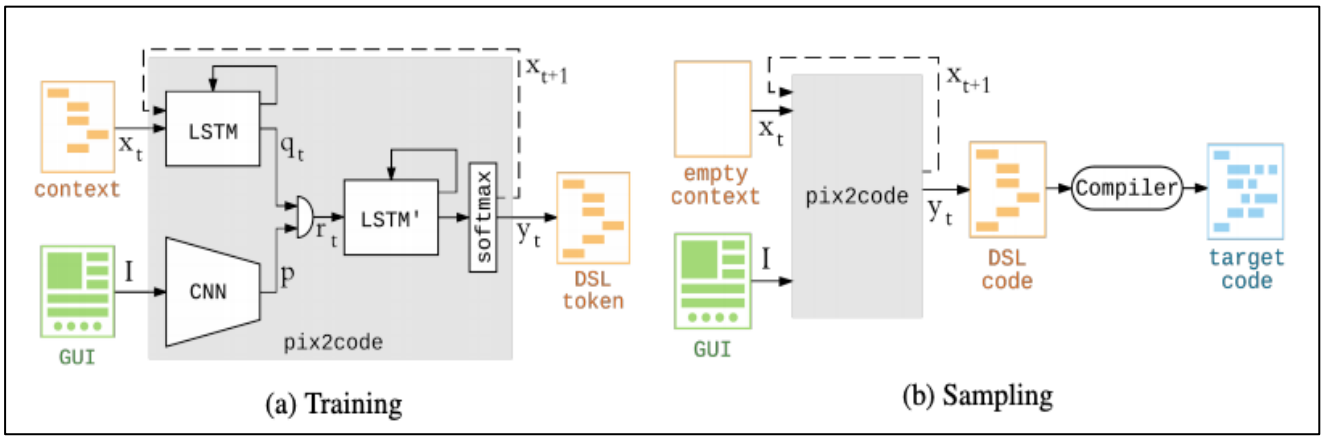

Рисунок 2.5 – Огляд архітектури моделі pix2code

Під час навчання зображення GUI кодується моделлю зору на основі CNN; контекст (тобто послідовність токенів з гарячим кодуванням, відповідних коду DSL) кодується мовної моделлю, що складається з стека рівнів LSTM. Два результуючих вектора ознак потім об'єднуються і подаються в другий стек рівнів LSTM, що діють як декодер. Нарешті, шар softmax використовується для вибірки одного токена за раз; розмір виведення шару softmax, який відповідає розміру словника DSL. Вихідне зображення і послідовності токенів, модель (тобто міститься в сірому квадраті) є диференційованою, таким чином, може бути оптимізована наскрізним градієнтним спуском, щоб передбачити наступний токен в послідовності. Отримана послідовність токенів DSL компілюється в необхідну мову перекладу з використанням традиційних методів проектування компілятора [27].

2.2.4 TensorFlow

TensorFlow – це бібліотека програмного забезпечення з відкритим програмним кодом, яка слугує для вираження алгоритмів машинного навчання і реалізацію для виконання таких алгоритмів. Обчислення, виражене з використанням TensorFlow, може бути виконано практично без змін на найрізноманітніших системах − від мобільних пристроїв, таких як телефони та

планшети − до великомасштабних розподілених систем, що складаються з сотень машин і тисяч обчислювальних пристроїв, таких як відео карти GPU [35].

Клієнтські програми взаємодіють з системою TensorFlow шляхом створення сеансу. Для створення графи обчислень інтерфейс Session підтримує метод Extend для доповнення поточної графи, керованого сеансом, додатковими вузлами і ребрами (початковий графік при створенні сеансу порожній). Іншою основною операцією, яку підтримує інтерфейс сеансу, є Run, яка приймає набір вихідних імен, які необхідно обчислити, а також необов'язковий набір тензорів, які потрібно подавати в графік замість певних вихідних даних вузлів. Використовуючи аргументи Run, TensorFlow може обчислити транзитивне закриття всіх вузлів, які повинні бути виконані для обчислення запитаних вихідних даних, а потім може організувати виконання відповідних вузлів у порядку, який враховує їх залежності (див. рис. 2.6 та табл. 2.1).

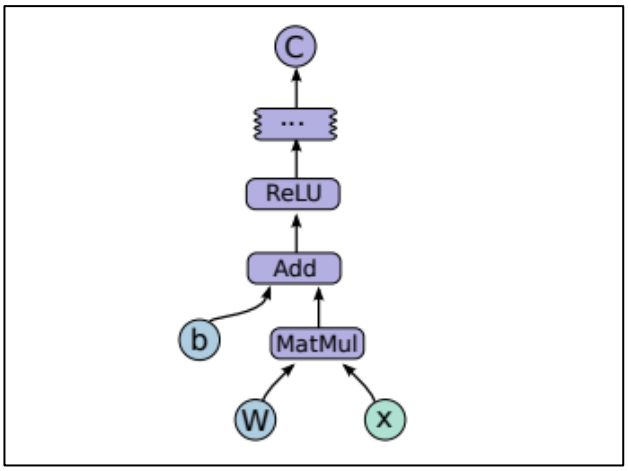

Рисунок 2.6 – Відповідний графік обчислень

```
    import tensorflow as tf
    b = tf.Variable(tf.zeros([100]))      # 100-й вектор,
ініціалізація
    # нулями
    W = tf.Variable(tf.random_uniform([784,100],-1,1))
    # 784x100 матриця з RNAL
    # значеннями
    x = tf.placeholder(name="x")          # Заповнювач для
введення
    relu = tf.nn.relu(tf.matmul(W, x) + b)
    # Relu(Wx+b)
    C = [...]    # Вартість розраховується як
```

```
# Створіть 100-d вектор для #введення
    result = s.run(C, feed_dict={x: input})
# Вартість вибірки, подача x #= вхід
print step, result
```

Таблиця 2.1 – Приклади типів операцій TensorFlow

Категорія	Приклади
Елементарні математичні операції	Add, Sub, Mul, Div, Exp, Log, Greater, Less, Equal, …
Операції масиву	Concat, Slice, Split, Constant, Rank, Shape, Shuffle, …
Матричні операції	MatMul, MatrixInverse, MatrixDeterminant, …
Міжмережеві операції	Variable, Assign, AssignAdd, …
Нейромережеві блоки	SoftMax, Sigmoid, RelU, Convolution2D, MaxPool, …
Операції перевірки	Save, Restore
Операції черги та синхронізації	Enqueue, Dequeue, MutexAcquire, MutexRelease, …
Управління потоком операцій	Merge, Switch, Enter, Leave, NextIteration

Підводячи підсумки, слід відзначити, що на сьогодні існує велика теоретична та практична база по трактуванню поняття штучного інтелекту, машинного навчання та штучних нейронних мереж. Окремо слід виділити місце штучних нейронних мереж в формуванні базису для генерування веб-додатків. Розглянуті основні інструменти для створення програмної системи, зокрема розглянута мова програмування Python, фреймворк Django, програмна бібліотека Pix2code та бібліотека машинного навчання TensorFlow. В результаті сформований вибір інструментів для створення програмної системи.

3 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАВДАННЯ

3.1 Постановка задачі

В таблиці 3 виділено основні задачі, що потрібно вирішити для розробки програми з генерації веб-сайтів в реальному часі. Щоб розв’язати глобальну задачу із створення програмної системи, що задовольняє всі ці потреби або більшість з них, проаналізуємо рішення від сторонніх розробників на наявність інструментів для вирішення кожної з оголошених проблем окремо, а потім згрупуємо отримані результати. В разі успіху в нас з’явиться комплексний набір інструментів для вирішення задачі з генерування нових сайтів і отримання як готового програмного коду, так і рішення на основі Boostrap (див. табл. 3.1).

Таблиця 3.1 – Проблеми та задачі, які потрібно вирішити в процесі розробки додатку

Задача	Дії до вирішення задачі
Алгоритм роботи	Створення алгоритму роботи програми з генерації веб-сайту.
	Можливість перетворення зображення на повноцінний макет веб-сайту
	Додатковий функціонал з обрання ключових слів та напрямків генерації веб-додатку
	Можливість динамічної зміни створених елементів
Варіативність отриманого результату	Отримання результату у вигляді готового рішення на основі Bootstrap
	Можливість отримання програмного коду без використання систем керування вмістом
Інтерфейс користувача	Створення візуального інтерфейсу веб-версії
	Адаптація інтерфейсу для мобільних пристроїв
Інше	Створення можливостей з обрання додаткових параметрів для отримання кращого результату

Отже, виходячи з наведених задач і проблем які потрібно вирішити в процесі розробки веб-додатку, потрібно розпочати з побудови алгоритму штучної нейронної мережі.

3.2 Створення алгоритму роботи штучних нейронних мереж з гіпертекстовими документами

3.2.1 Розробка середовища для роботи нейронної мережі

Для початку роботи з штучною нейронною системою створимо версію гіпертекстового документу з довільним змістом. Наступним кроком буде передача документу до нейронної мережі і розпочата робота з створювання готового рішення.

По-перше, нейронна мережа відображає макет проекту в список значень пікселів. Від 0 до 255 px трьох каналах − червоний, синій і зелений (див. рис .3.1).

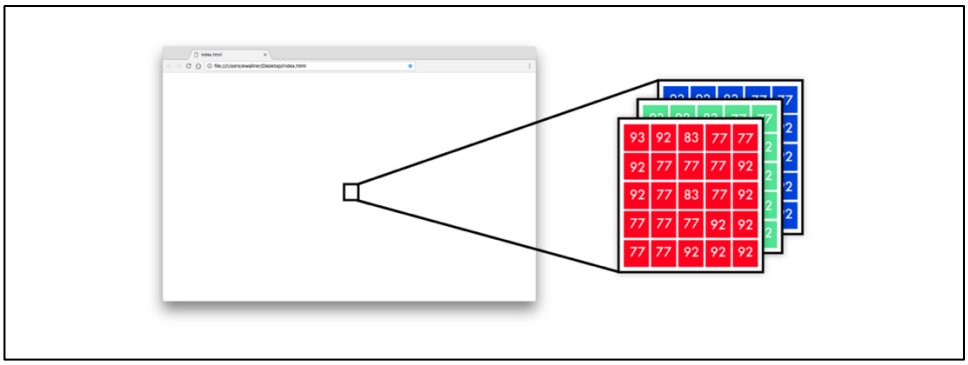

Рисунок 3.1 – Відображення макету проекту

Щоб уявити розмітку в розумінні нейронної мережі, потрібно звернутися до технології палкого кодування (від англ. Hot Encode). Таким чином, ми можемо спостерігати відображення розмітки в пам’яті нейронної мережі (див. рис. 3.2).

СЛОВНИК	ПОЧАТОК	СТВОРЕННЯ	НЕЙРОННОЇ	МЕРЕЖІ	КІНЕЦЬ
ПОЧАТОК	1	0	0	0	0
СТВОРЕННЯ	0	1	0	0	0
НЕЙРОННОЇ	0	0	1	0	0
МЕРЕЖІ	0	0	0	1	0
КІНЕЦЬ	0	0	0	0	1

Рисунок 3.2 – Розмітка в пам’яті нейронної мережі

Фрази дотримуються тієї ж логіки, що і слова. Їм також потрібна однакова довжина введення. Замість того, щоб обмежуватися словником, вони обмежені

максимальною довжиною пропозиції. Якщо фраза коротша максимальної довжини, то її потрібно заповнити нулями (див. рис. 3.3).

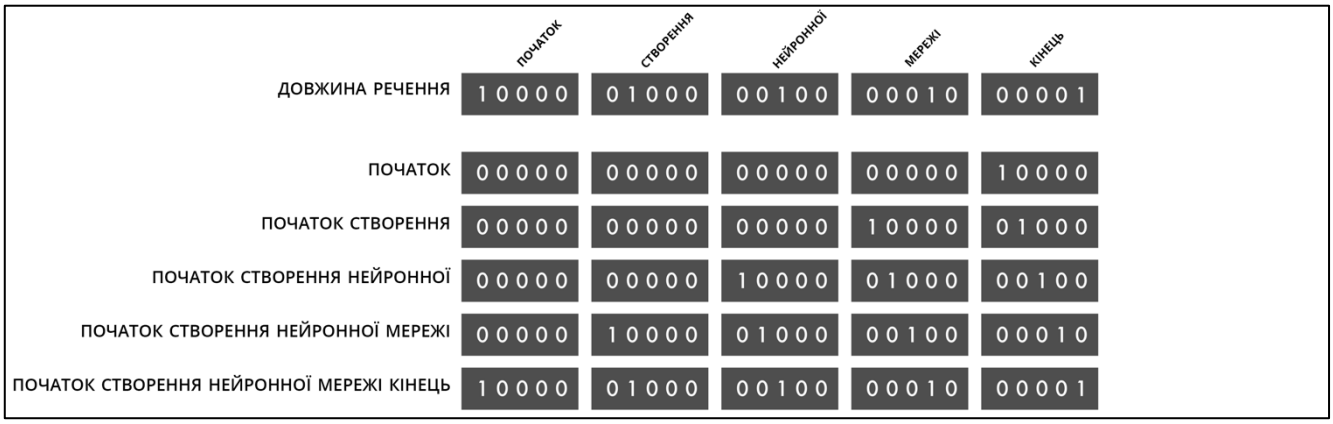

Рисунок 3.3 – Відображення моделі послідовності

Для вхідних даних ми будемо використовувати пропозиції, починаючи з першого слова, а потім додаючи кожне слово одне за іншим. У вихідних даних завжди одне слово.

На рисунку 3.4 представлені чотири варіанти прогнозування рішення. Зліва знаходяться зображення, представлені в трьох колірних каналах: червоний, зелений і синій, а також попередні слова. Поза дужками наведені прогнози одне за іншим, закінчуються червоним квадратом для позначення кінця.

Рисунок 3.4 – Чотири варіанти прогнозування рішення

```
    # Завантаження функції в мережу
    vgg_feature = Input(shape=(1000,))
    vgg_feature_dense = Dense(5)(vgg_feature)
    vgg_feature_repeat =
RepeatVector(max_caption_len)(vgg_feature_dense)
    # Отримання вхідних данних
    language_input = Input(shape=(vocab_size, vocab_size))
    language_model = LSTM(5,
return_sequences=True)(language_input)
    # Об’єднання інформації з вхідного і вихідного сигналу
```

```
    decoder = concatenate([vgg_feature_repeat,
language_model])
    # Отримання об’єднанної інформації
    decoder = LSTM(5, return_sequences=False)(decoder)
    # Прогнозування наступного значення
    decoder_output = Dense(vocab_size,
activation='softmax')(decoder)
    # Компіляція і запуск нейронної мережі
    model = Model(inputs=[vgg_feature, language_input],
outputs=decoder_output)
    model.compile(loss='categorical_crossentropy',
    optimizer='rmsprop')
```

3.2.2 Тестування отриманих результатів

FloydHub – це навчальна платформа для глибокого навчання. Для тестування отриманих результатів потрібно підключитися через термінал до git репозиторія.

```
    # Клонування репозиторія
    git clone https://github.com/emilwallner/Screenshot-
to-code-in-Keras.git
    #Вхід в систему та доступ до командної строки
    cd Screenshot-to-code-in-Keras
    floyd login
    floyd init s2c
```

У процесі тестування програми не було виявлено жодних порушень і програма працювала стабільно після проходження всіх тестів.

3.3 Створення HTML-версії програмного коду

На рисунку 3.5, зображені основні розділи, які відповідають за побудову HTML документа. Перший з них є кодер. Він відповідає за створення елементів зображення і попередніх функцій розмітки. Функціональні можливості – це будівельні блоки, які мережа створює для з’єднання макетів дизайну з розміткою.

В кінці кодера ми додаємо елементи зображення до кожного слова в попередній розмітці.

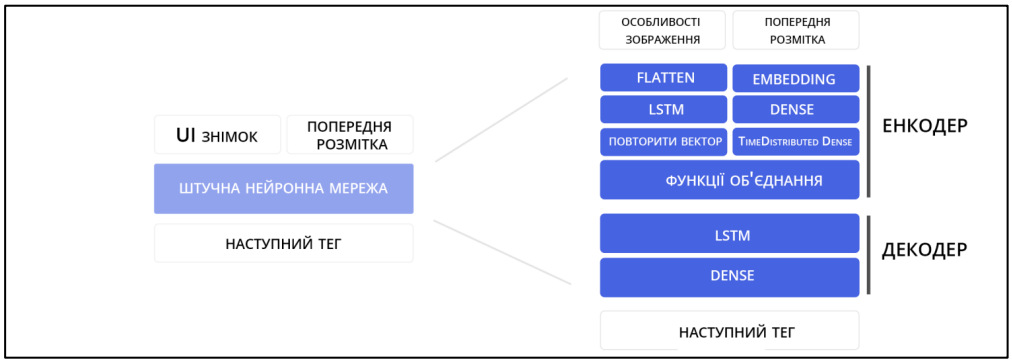

Рисунок 3.5 – Основні розділи, які відповідають за побудову HTML документа

Потім декодер бере об’єднану процедуру оформлення і розмітки і створює функцію тега. Ця функція запускається через повністю підключену нейронну мережу, щоб передбачити наступний тег.

3.3.1 Особливості розмітки

На відміну від попередньої версії генерування програмного коду, в цій ревізії кожне речення залишається незмінним, але спосіб відображення кожного токена змінився. Гаряче кодування обробляє кожне слово, як окрему одиницю. Замість цього ми конвертуємо кожне слово у вхідних даних в списки цифр. Вони представляють відносини між тегами розмітки (див рис. 3.6).

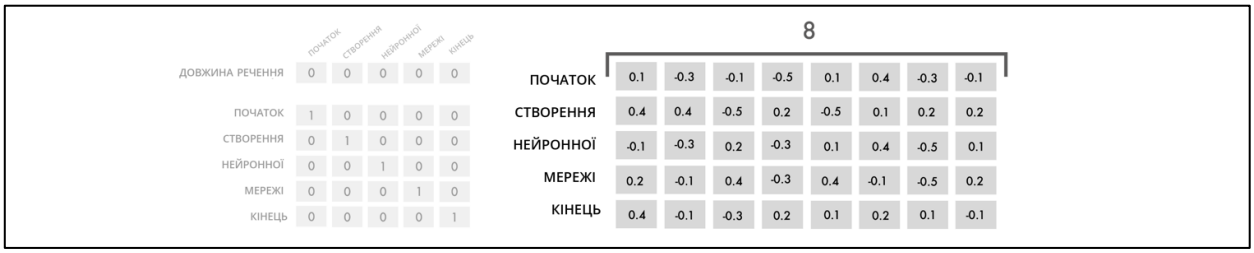

Рисунок 3.6 – Відносини між тегами розмітки

Головною особливістю є те, що нейронна мережа розвивається, щоб зв’язати вхідні дані з вихідними даними.

У більшості мов програмування і мов розмітки елемент оголошується з відкритого токену; якщо дочірні елементи або інструкції містяться всередині блоку, то для інтерпретатора або компілятора звичайно потрібно закрити токен. У такому сценарії, де число дочірніх елементів, що містяться в батьківському елементі, є змінним, важливо змоделювати довгострокові залежності, щоб мати можливість закрити відкритий блок. Традиційні архітектури RNN страждають від зникнення і вибуху градієнтів, що не дозволяють їм моделювати такі відносини між точками даних, розподіленими в тимчасових рядах (тобто в цьому випадку токени поширюються в послідовності). Хохрейтер і Шмідхубер запропонували нейронну архітектуру з короткочасною пам’яттю (LSTM) для вирішення цієї самої проблеми. Різні вихідні дані LSTM-шлюзу можуть бути обчислені в такий спосіб:

$$i_t = \phi(W_{ix^{x_t}} + W_{iy^{h_t}} - 1 + b_i \quad (1)$$

$$f_t = \phi(W_{fx^{x_t}} + W_{fy^{h_t}} - 1 + b_f \quad (2)$$

$$o_t = \phi(W_{ox^{x_t}} + W_{oy^{h_t}} - 1 + b_o \quad (3)$$

$$c_t = f_t \cdot c_t - 1 + i_t \cdot \sigma(W_{cx^{x_t}} + W_{cy^{h_t}} - 1 + b_c) \quad (4)$$

$$h_t = o_t \cdot \sigma(c_t) \quad (5)$$

Таким чином, блок пам'яті LSTM може використовувати i, щоб вирішити, коли записувати інформацію в c, і використовувати o, щоб вирішити, коли читати інформацію з c. Ми використовували варіант LSTM, запропонований Джерсі і Шмідхубер, з ключем забуття f, щоб скинути пам'ять і допомогти моделі мережі безперервних послідовностей.

3.3.2 Енкодер

Для кращого розуміння потрібно вкласти вихідні значення і провести їх через LSTM і повернути послідовність функцій розмітки (див. рис. 3.7).

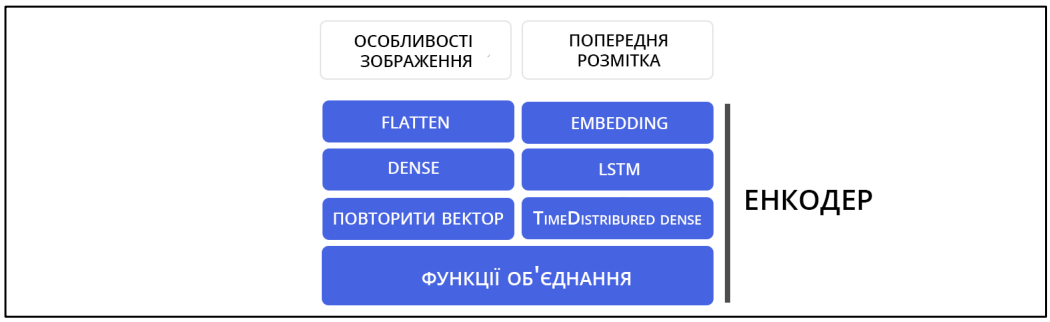

Рисунок 3.7 – Координація через LSTM

Паралельно елементи зображення спочатку згладжуються. Незалежно від того, як цифри структуровані, вони перетворюються в один великий список чисел. На останньому етапі функції зображення об’єднуються з функціями розмітки.

Особливості розмітки через LSTM. На рисунку 3.8 зображений процес доповнення до максимального значення трьох токенів.

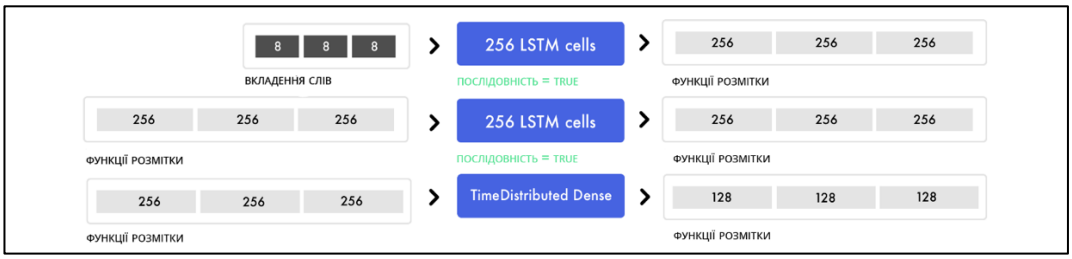

Рисунок 3.8 – Процес доповнення до максимального значення трьох токенів

Водночас для зображень потрібно всі функції перетворити на нумерований список. Інформація при цьому не змінена, внаслідок цього вона реорганізована (див. рис. 3.9).

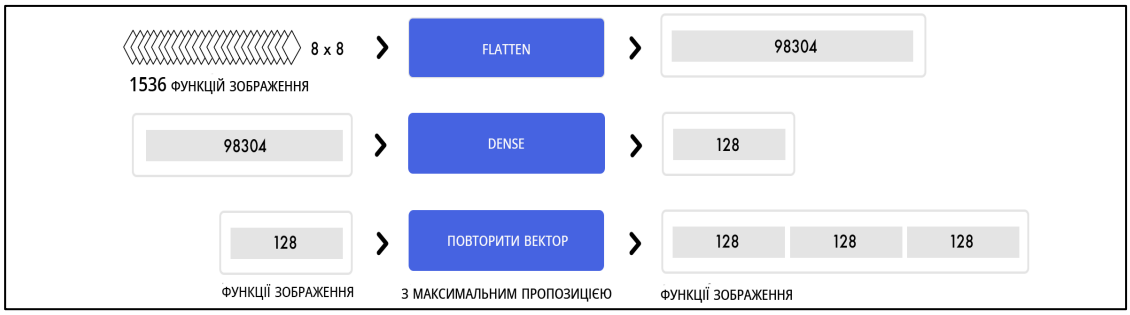

Рисунок 3.9 – Перетворення міні-зображень на нумерований список

Це спеціальні моделі глибокого навчання, які навчаються на наборі даних в якості попередника для вирішення завдання з цими даними. Автоенкодери вчаться

стискати дані в низькорозмірному програмному коді, а потім також вчаться розпаковувати кодування до вихідних даних. Це відмінний спосіб дізнатися важливі особливості даних. Частина кодера часто буде імпортована в інші архітектури з глибоким навчанням для передачі функцій, які вона вивчила для кодування. Це відомо як трансферне навчання. Частина декодера представляє особливий інтерес для нас, тому що вона вчиться інтерпретувати абстрактні значущі кодування і перетворювати їх в зображення для користувача інтерфейсів [26].

Якщо ми навчаємо автокодеру, який може стискати і розпаковувати графічні інтерфейси, то ми отримуємо безкоштовний «декодер зображень» для використання в code2pix (див. рис. 3.10).

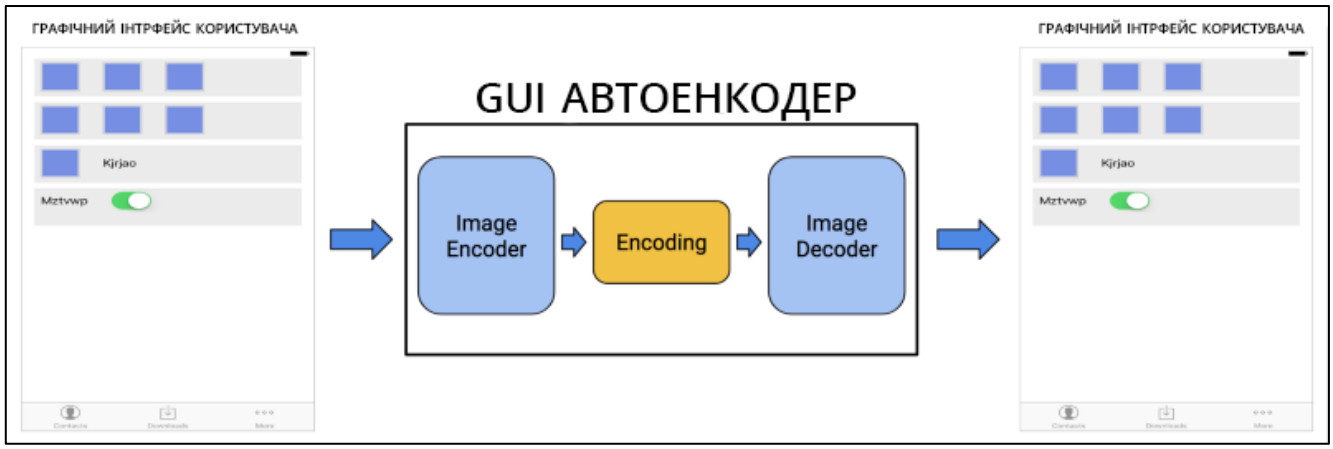

Рисунок 3.10 – Безкоштовний «декодер зображень» для використання в code2pix

Ми випробували стандартну архітектуру автоенкодера, щоб вирішити цю задачу, але вивчені декодери були дуже корисні для вирішення нашої задачі. Таким чином, ми розробили нову архітектуру: multi-headed модель для одночасного вирішення завдання для всіх трьох наборів даних: Android, iOS і Web. У цій моделі, що отримала назву «Hydra», декодер, який інтерпретує кодування зображення, залежить від набору даних, з якого отримано зображення. Наприклад, щоб пропустити зображення Android через модель, ми повинні вручну пропустити його через головку декодера Android, щоб отримати остаточний результат [26].

Конкретний дизайн автоенкодера, який ми використовували, спільно використовував один і той же кодер зображення для всіх трьох декодерів

зображень. В результаті вийшла дуже ефективна модель (див. рис. 3.11), і були створені узагальнені декодери зображень.

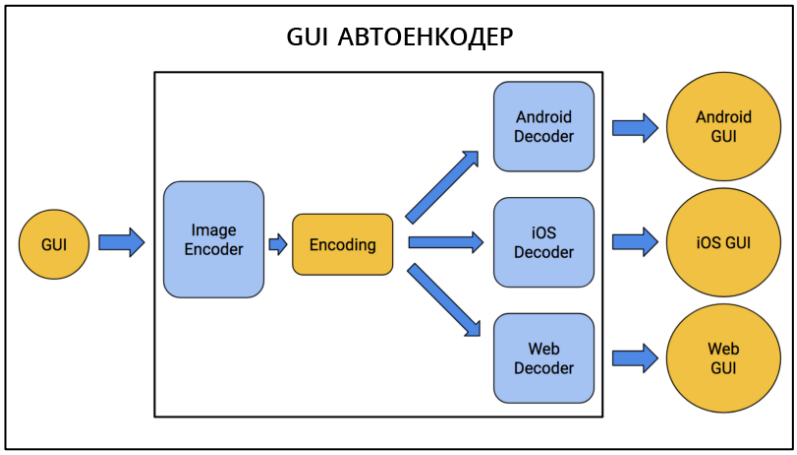

Рисунок 3.11 – Ефективна модель автоенкодера

Ми використовуємо головки декодера в дизайні моделі Code2Pix. За допомогою зазначеної моделі не потрібно адаптувати дизайн для мобільних пристроїв самостійно.

3.3.3 Декодер

На противагу енкодеру, в декодері прийнято використовувати комбіновані функції розмітки зображення, щоб передбачити наступний тег (див. рис. 3.12). У наведеному нижче прикладі ми використовуємо три пари об’єктів з розміткою зображення і виводимо один наступний об’єкт тега

Рисунок 3.12 – Перетворення міні-зображень на нумерований список

Варто звернути увагу, що для рівня LSTM послідовність має значення false. У нашому випадку ця функція для наступного тега буде містити інформацію про остаточний крок (див. рис. 3.13).

Наша модель підготовлена в режимі контрольованого навчання через подачу зображення і контекстної послідовності X з T токенов xt, t ∈ {0. ,, T - 1} в якості вихідних даних; і токен xT в якості цільової мети.

Рисунок 3.13 – Послідовність LSTM

Як показано на рисунку 3.13, модель зразків на основі CNN кодує вхідне зображення, яке ми надіслали на векторне зображення. Вхідна токен xt кодується мовою на основі LSTM в проміжному уявленні, що дозволяє більше орієнтуватися на певні токи і менше на інших. Ця модель першої мови реалізована у вигляді стека з двох шарів LSTM по 128 елементів кожен. Кодований вектор p та кодований вектор qt об'єднуються в один вектор ознак rt, який потім подається в другу засновану на LSTM модель, декодує уявлення, отримані як в моделі бачення, так і в моделі мови. Таким чином, декодер обумовлений модифікацією відносин між об’єктами, наявними у вхідному зображенні GUI, і відповідними токенами, наявними в кодуванні DSL. Наш декодер реалізований у вигляді стек з двох слоїв LSTM з 512 елементами в кожному. Ця архітектура може бути виражена математично наступним чином:

$$p = CNN(I) \quad (1)$$

$$q_t = LSTM(x_t) \quad (2)$$

$$r_t = (q, p_t) \quad (3)$$

$$y_t = (q, p_t) \quad (4)$$

$$y_t = softmax(LSTM'(r_t)) \quad (5)$$

$$x_t + 1 = y_t \quad (6)$$

Ця архітектура дозволяє повністю оптимізувати всю модель pix2code. Дискретний характер виведення (тобто словник фіксованих розмірів токенів в

DSL) дозволяє нам звести задачу до проблеми класифікації. Тобто вихідний шар нашої моделі має таку саму кількість осередків, що і розмір словника; таким чином, генеруючи розподіл ймовірностей токенів-кандидатів на кожному часовому кроці, дозволяючи використовувати рівень softmax для виконання багатокласової класифікації.

Довжина T послідовностей, використовуваних для навчання, важлива для моделювання довгострокових залежностей; наприклад, щоб мати можливість закрити блок коду, який був відкритий. Після проведення емпіричних експериментів вхідні файли DSL, використовувані для навчання, були сегментовані за допомогою протокола Sliding Window розміром 48; іншими словами, ми розгортаємо рекурентну нейронну мережу за 48 кроків. Було встановлено, що це є задовільним компромісом між вивченням довгострокових залежностей і обчислювальними витратами. Тому для кожного токена у вхідному файлі DSL модель забезпечується як вхідним зображенням, так і контекстно. послідовністю токенів T = 48. Хоча контекст (тобто послідовність токенів), який використовується для навчання, оновлюється на кожному часовому кроці (тобто кожному токені) шляхом ковзання вікна, одне й те ж зображення і повторно використовується для вибірок, пов’язаних з одним і тим же GUI. Спеціальні маркери <ST ART> та <END> використовуються для відповідного префікса і суфікса файлів DSL аналогічно методу, що використовується в роботах Karpathy та Fei-Fei. Навчання виконується шляхом обчислення приватних похідних втрат за вагами мережі, розрахованим зі зворотним поширенням, щоб мінімізувати втрати журналу мультікласа:

$$L(I, X) = -\sum_{t=1}^{T} x_t + 1\log\left(y_t\right)$$

З x_t+ 1 очікуваний токен і yt прогнозований токен. Модель оптимізована від початку до кінця, тому втрати L зводяться до мінімуму з урахуванням всіх параметрів, включаючи всі рівні в моделі бачення на основі CNN і всі рівні в обох моделях на основі LSTM. Навчання з використанням алгоритму RMSProp дало найкращі результати при швидкості навчання, встановленої на 1e - 4, і обрізання вихідного градієнта до діапазону [-1,0, 1,0], щоб впоратися з чисельної

нестабільністю. Щоб уникнути переоснащення регуляризація відсіву, встановлена на 25%, застосовується до моделі зору після кожної операції максимального пулу і на 30% після кожного повністю підключеного шару. У моделях на основі LSTM випадання встановлено на 10% і застосовується тільки до односторонніх з'єднаннь. Наша модель була навчена з міні-пакетами з 64 пар послідовності зображень.

3.4 Створення Bootstrap-версії програмного коду

У остаточній версії буде використовуватися набір даних згенерованих сайтів початкового завантаження з документації pix2code. Використовуючи завантажувач Twitter Bootstrap, ми можемо комбінувати HTML і CSS і зменшувати розмір словника.

У цій моделі потрібно внести зміни до функції зображення нейронної мережі. Потрібно використовувати максимальний пул для збільшення щільності інформації. Це зберігає положення і колір елементів інтерфейсу (див рис. 3.14).

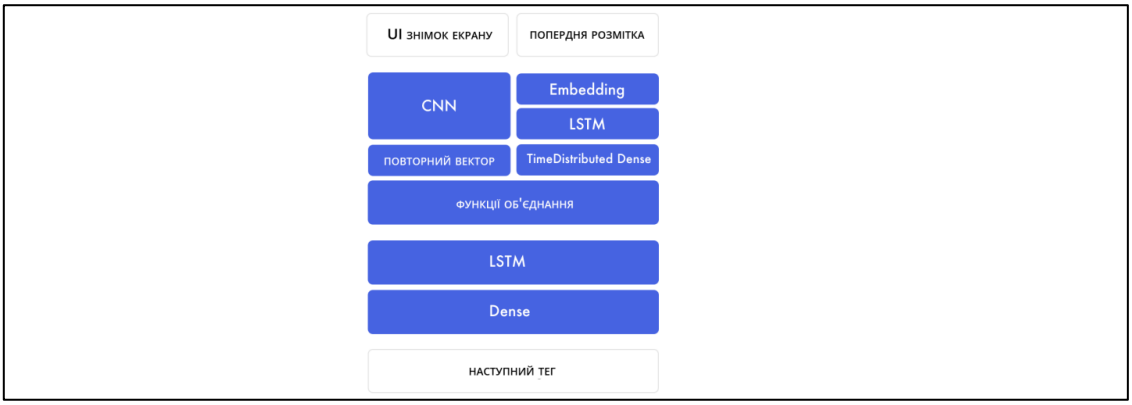

Рисунок 3.14 – Основні моделі нейронних мереж застосовуваних при створенні розмітки для Bootstap

LSTM нейронна мережа, налаштована на отримання інформації в порядку черги. На рисунку 3.15 зображена спрощена версія процесу для кожного тимчасового кроку в LSTM.

На противагу минулим версіям програмного коду, потрібно використовувати 17 спрощених токенів, які потім зміняться на файли з розміткою HTML і CSS. Набір

даних включає в себе 1500 тестових знімків екрану і 250 перевірочних зображень. На кожен скріншот в середньому припадає 65 токенів, що надає 96925 прикладів ймовірних результатів.

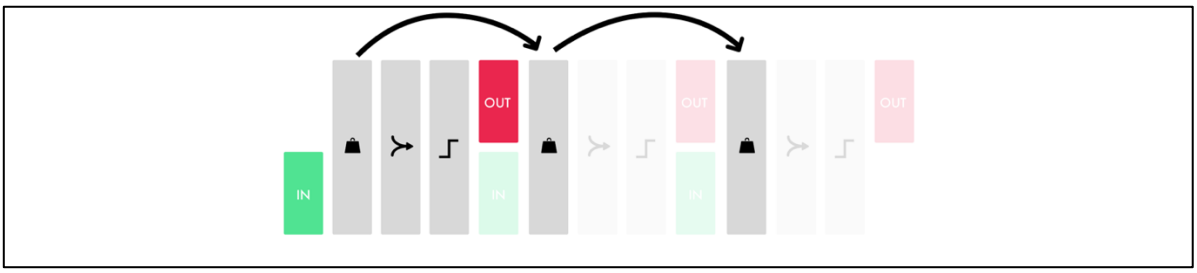

Рисунок 3.15 – Спрощена версія процесу для кожного тимчасового кроку в LSTM

```
    #Створення енкодеру
    image_model = Sequential()
    image_model.add(Conv2D(16, (3, 3), padding='valid',
activation='relu', input_shape=(256, 256, 3,)))
    image_model.add(Conv2D(16, (3,3), activation='relu',
padding='same', strides=2))
    image_model.add(Conv2D(32, (3,3), activation='relu',
padding='same'))
    image_model.add(Conv2D(32, (3,3), activation='relu',
padding='same', strides=2))
    image_model.add(Conv2D(64, (3,3), activation='relu',
padding='same'))
    image_model.add(Conv2D(64, (3,3), activation='relu',
padding='same', strides=2))
    image_model.add(Conv2D(128, (3,3), activation='relu',
padding='same'))
    image_model.add(Flatten())
    image_model.add(Dense(1024, activation='relu'))
    image_model.add(Dropout(0.3))
    image_model.add(Dense(1024, activation='relu'))
    image_model.add(Dropout(0.3))
    image_model.add(RepeatVector(max_length))
    visual_input = Input(shape=(256, 256, 3,))
    encoded_image = image_model(visual_input)
    language_input = Input(shape=(max_length,))
    language_model = Embedding(vocab_size, 50,
input_length=max_length, mask_zero=True)(language_input)
    language_model = LSTM(128,
return_sequences=True)(language_model)
```

```
    language_model = LSTM(128,
return_sequences=True)(language_model)
    #Створення декодеру
    decoder = concatenate([encoded_image, language_model])
    decoder = LSTM(512, return_sequences=True)(decoder)
    decoder = LSTM(512, return_sequences=False)(decoder)
    decoder = Dense(vocab_size,
activation='softmax')(decoder)
```

Щоб отримати остаточний результат, потрібно вирішити наведену формулу (4/5)*0,25+(2/4)*0,25+(1/3)*0,25+(0/2)*0,25=0,2+0,125+0,083+0=0,408. Оскільки довжина вірна, вона стає остаточною оцінкою.

Результатом аналізу та будування штучної нейронної мережі за допомогою Python, TensorFlow та Pix2Code став робочий синтез конкретних методів, із допомогою яких можна досягти поставленої цілі. Проаналізований і складений алгоритм побудови програмного коду з вхідних зображень та заданого тексту. Виділені способи трансформування отриманого результату в html та фреймворка bootstrap.

4 ПРОГРАМНА СИСТЕМА, ЩО ГЕНЕРУЄ ВЕБ-САЙТИ ЗА ДОПОПОМОГОЮ ШТУЧНОГО ІНТЕЛЕКТУ

4.1 Постановка задачі

Для вирішення спектру проблем та потреб, що виникли у зв’язку автоматизація розробки веб-додатків необхідно створити два web-додатки, для яких було обрано наступні назви: Automation ANN та Packaging AI. Перший – для генерації з існуючих шаблонів та графічних зображень програмний код, другий – для зібрання програмного коду, графічних зображень та інших матеріалів в завершене рішення. Обидва додатки мають бути реалізовані в рамках одного проекту, написаного на мовах програмування Python, або Java та створені як окремі APK-файли, придатні для публікації у мережі Інтернет.

Для створення додатків потрібно використати алгоритм, покладений в основу pix2code, який був досліджений у попередній частині роботи.

Список вимог до нових компонентів веб-додатку наведено нижче:

– перший додаток повинен обирати графічні зображення та готові макети з бази даних, об’єднувати їх та генерувати за допомогою ШНМ в програмний код;

– другий додаток покликаний отримувати інформацію від першого і збирати їх в файловий архів, придатний для користування;

– компонент спільного інтерфейсу програм повинні бути реалізовані в мінімалістичному дизайні з мінімальною кількістю додаткових функцій;

– компонент інтерфейсу не повинен мати жодних засобів для доступу до адміністративної панелі та бази даних;

– компонент інтерфейсу має бути виконаний у вигляді статичного інтерфейсу, без можливості вносити зміни до функціонала додатка.

Додаток, що створює службу, повинен мати 5 сторінок з інтерфейсом користувача:

– головна сторінка із панеллю навігації, формою реєстрації, доступом до особового кабінету та областю для основного контенту. Також на старінці потрібно розмістити допоможіні відомості.

– сторінка з доступом до особового кабінету, на якій розміщена форма реєстрації та панель доступу.

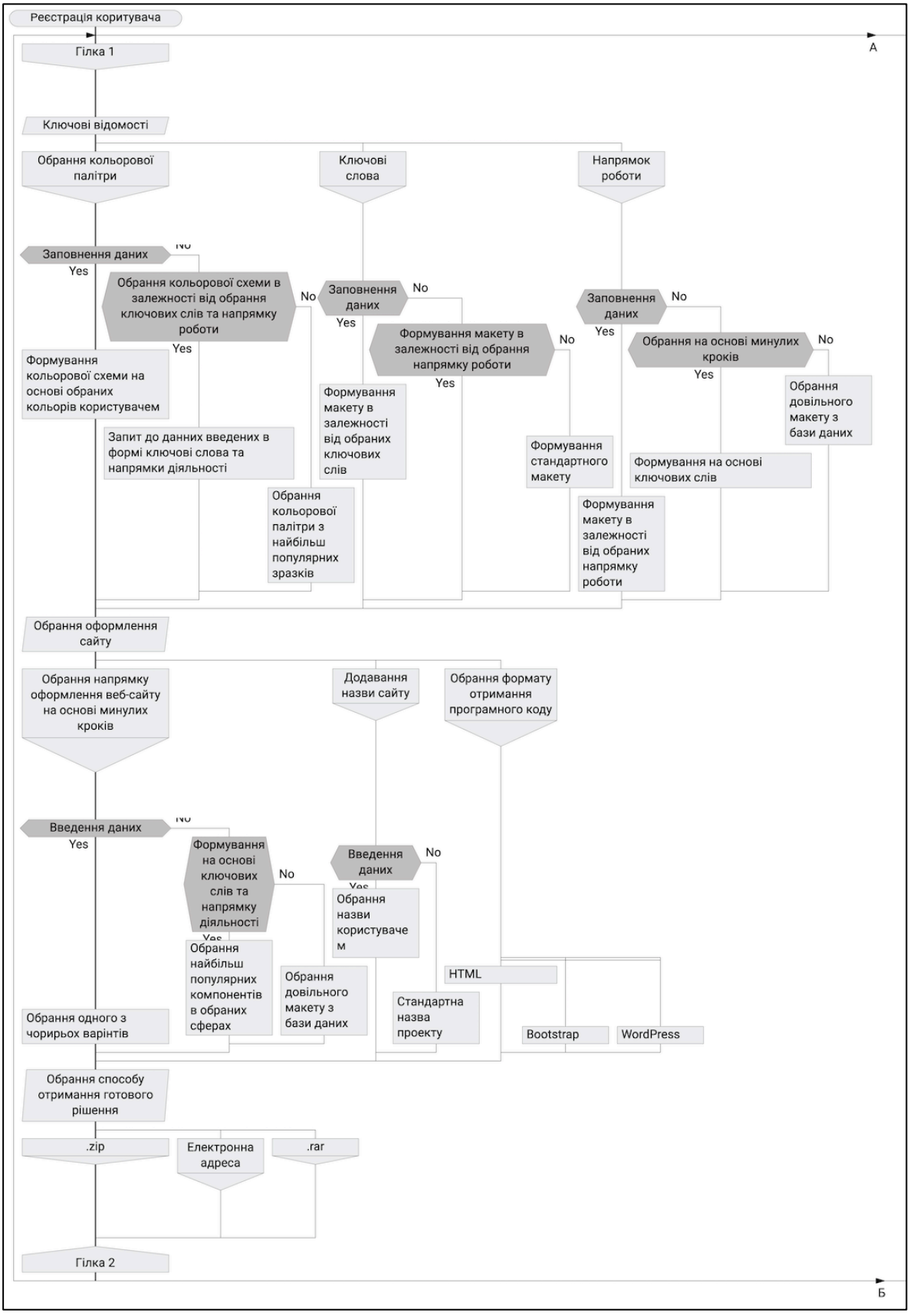

Рисунок 4.1 – Блок-схема роботи програмної системи з генерації веб-сайтів

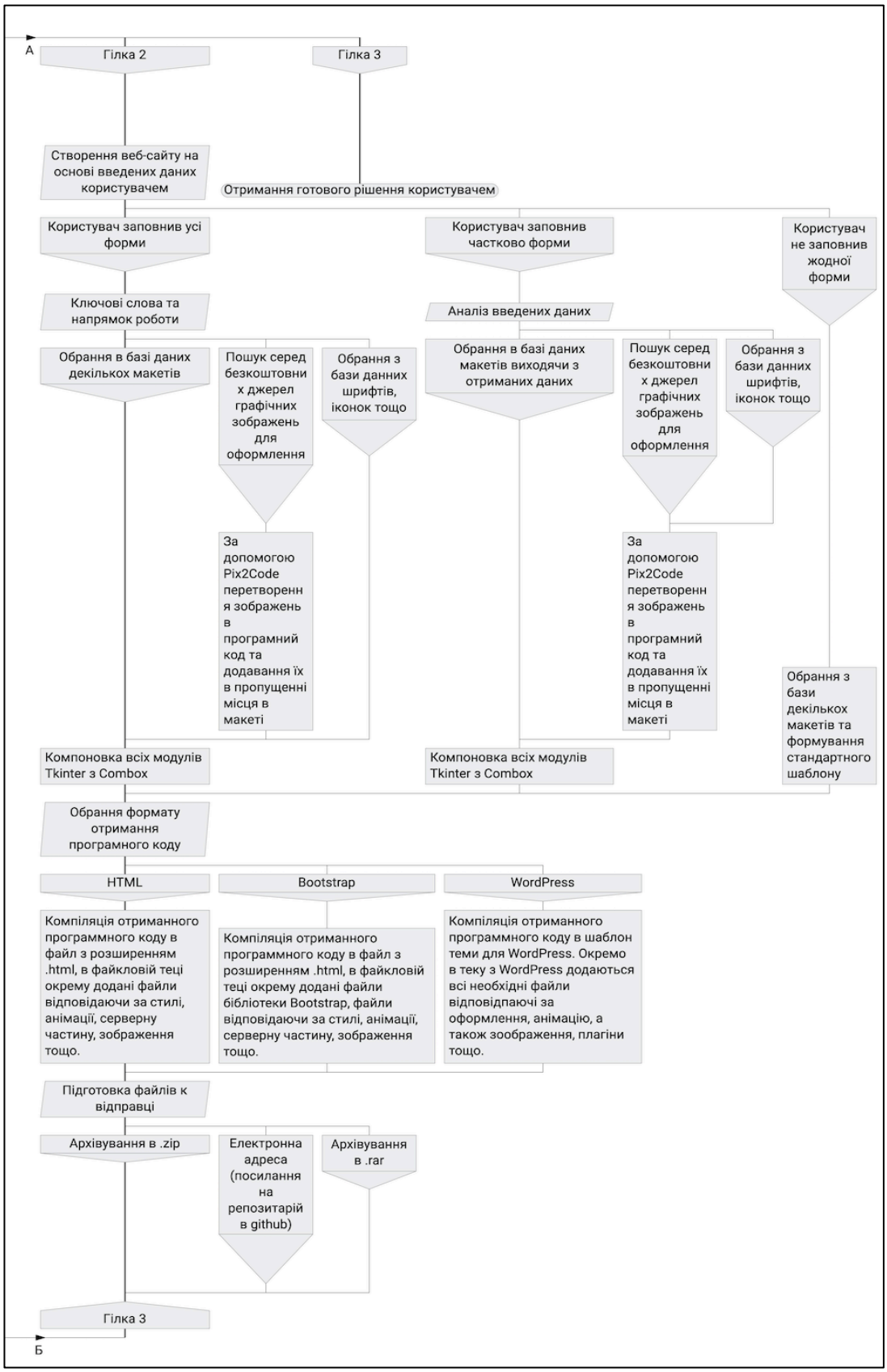

Рисунок 4.1, аркуш 56

На рисунку 4.1 зображена блок-схема роботи програмної системи з генерації веб-сайтів.

4.2 Перелік вимог до програмної системи

Додатки повинні мати інтуїтивний та логічний інтерфейс, забезпечувати плавний відгук на дії користувача за допомогою анімацій. Додатковою вимогою є мінімізація текстових елементів, що полегшить локалізацію та розповсюдження. Программа повинна забезпечувати якісний UX, тобто мінімізувати кількість дій, необхідних для виконання функціоналу додатку.

У разі активації функціоналу додатків, вони мають працювати стабільно і не припиняти свою роботу за будь яких умов, окрім бажання користувача. У разі вимкнення або аварійного завершення роботи браузера, додатки повинні відновлювати свою роботу на останньому збереженому місці. Додатки мають відповідати сучасним архітектурним стандартам, а саме використовувати патерн MVC та дотримуватись принципів проектування SOLID, що спростить їх майбутню підтримку та налагодження автоматичного тестування.

Програмна система має бути вільною від помилок, та забезпечувати стабільну роботу принаймні в сучасних браузерах.

4.3 Архітектура програмної системи

4.3.1. Патерн Model-View-Presenter

Модель–вигляд–контролер (або Модель–представлення–контролер, англ. Model-view-controller, MVC) − архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення.

Модель складається з чистої логіки програми, яка взаємодіє з базою даних. Вона включає в себе всю інформацію для подання даних кінцевому користувачеві [30].

```
# model_view_controller.py
import basic_backend
import mvc_exceptions as mvc_exc
```

```
    class ModelBasic(object):
        def __init__(self, application_items):
            self._item_type = 'product'
            self.create_items(application_items)
        @property
        def item_type(self):
            return self._item_type
        @item_type.setter
        def item_type(self, new_item_type):
            self._item_type = new_item_type
        def create_item(self, name, price, quantity):
            basic_backend.create_item(name,          price,
quantity)
        def create_items(self, items):
            basic_backend.create_items(items)
        def read_item(self, name):
            return basic_backend.read_item(name)
        def read_items(self):
            return basic_backend.read_items()
        def update_item(self, name, price, quantity):
            basic_backend.update_item(name,          price,
quantity)
        def delete_item(self, name):
            basic_backend.delete_item(name)
```

Представлення відповідає за відображення інформації на зовнішньому інтерфейсі. Інтерфейс – це те, що бачить користувач, і він може бути реалізований за допомогою веб-сторінки (в разі веб-додатків), або набору інструментів з графічним інтерфейсом, такого як Tkinter (в разі настільних додатків) [31].

```
    # model_view_controller.py
    class View(object):
        @staticmethod
        def show_bullet_point_list(item_type, items):
            print('---          {}          LIST          ---
'.format(item_type.upper()))
            for item in items:
                print('* {}'.format(item))
        @staticmethod
        def show_number_point_list(item_type, items):
            print('---          {}          LIST          ---
'.format(item_type.upper()))
            for i, item in enumerate(items):
```

```
                print('{}. {}'.format(i+1, item))
    …
        @staticmethod
        def display_change_item_type(older, newer):
            print('-------------------------------------')
            print('Change item type from "{}" to
"{}"'.format(older, newer))
            print('-------------------------------------')
        @staticmethod
        def display_item_updated(item, o_price, o_quantity,
n_price, n_quantity):
            print('-------------------------------------')
            print('Change {} price: {} --> {}'
                  .format(item, o_price, n_price))
            print('Change {} quantity: {} --> {}'
                  .format(item, o_quantity, n_quantity))
            print('-------------------------------------')
        @staticmethod
        def display_item_deletion(name):
            print('-------------------------------------')
            print('We have just removed {} from our
list'.format(name))
            print('-------------------------------------')
```

Контролер приймає вхідні дані користувача і делегує уявлення даних в уявлення і обробку даних в модель [32].

```
    # model_view_controller.py
    class Controller(object):

        def __init__(self, model, view):
            self.model = model
            self.view = view

        def show_items(self, bullet_points=False):
            items = self.model.read_items()
            item_type = self.model.item_type
            if bullet_points:

self.view.show_bullet_point_list(item_type, items)
            else:

self.view.show_number_point_list(item_type, items)
```

```
        def show_item(self, item_name):
            try:
                item = self.model.read_item(item_name)
                item_type = self.model.item_type
                self.view.show_item(item_type,   item_name,
item)
            except mvc_exc.ItemNotStored as e:

self.view.display_missing_item_error(item_name, e)
    …

        def update_item_type(self, new_item_type):
            old_item_type = self.model.item_type
            self.model.item_type = new_item_type

self.view.display_change_item_type(old_item_type,
new_item_type)

        def delete_item(self, name):
            item_type = self.model.item_type
            try:
                self.model.delete_item(name)
                self.view.display_item_deletion(name)
            except mvc_exc.ItemNotStored as e:

self.view.display_item_not_yet_stored_error(name,
item_type, e)
```

Оскільки модель, перегляд і контролер відокремлені, кожен з трьох може бути розширений, змінений і замінений без необхідності переписувати два інших (див. рис. 4.2).

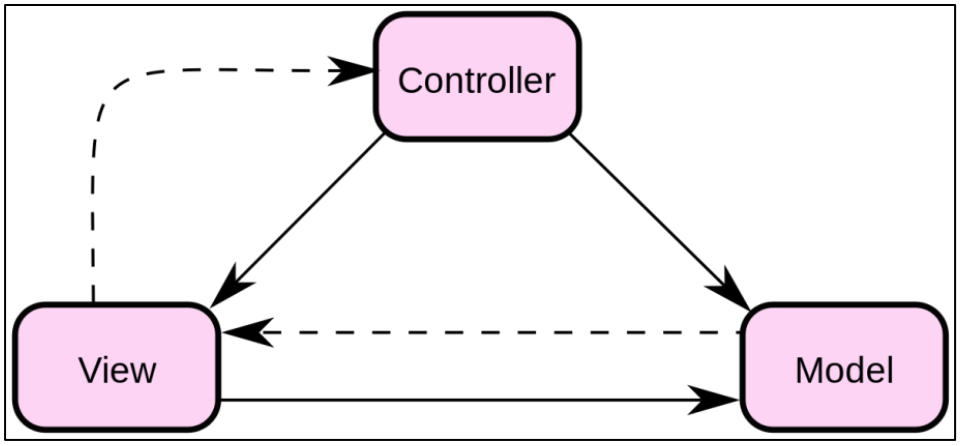

Рисунок 4.2 – Схема патерну MVC при використанні в Python-додатках

4.3.2. Мови програмування, що використано про розробці системи

Основні компоненти програмної системи написано на мові програмування Python. Подальшу розробку та розвиток було вирішено виконувати за допомогою бібліотеки pix2code та фреймворка Django.

Python – одна з провідних мов розробки систем на основі штучного інтелекту. Це об’єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована мова програмування. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій, так і у вихідній формі на всіх основних платформах [33].

Pix2code – додаток, який працює на python2(3), включая в себе TensorFlow і популярні бібліотеки opencv2 та numpy. Серед готових інструментів в комплекті знаходиться база даних з тренуваннями нейронних мереж [34].

Django − високорівневий відкритий Python-фреймворк для розробки веб-систем. Django підтримує парадигму ООП. Надає у розпорядження розробникові розвинутий прикладний програмний інтерфейс для високорівневого доступу до даних [35].

4.4 Структура проекту

Весь програмний код проекту розташовано в архіві automation.zip. В основному архіві розміщені пакети для автоматичної генерації програмного коду за допомогою штучних нейронних мереж, веб-додаток складений за допомогою фреймворка Django, система має три додаткові підпакети: compiler, datasets та model.

Створена програмна система дозволяє генерувати програмний код, виходячи з введених даних користувачем. На рисунку 4.3, зображена початкова сторінка реєстрації користувача з трьома обов’язковими формами. На сторінці також передбачені додаткові відомості для користувача.

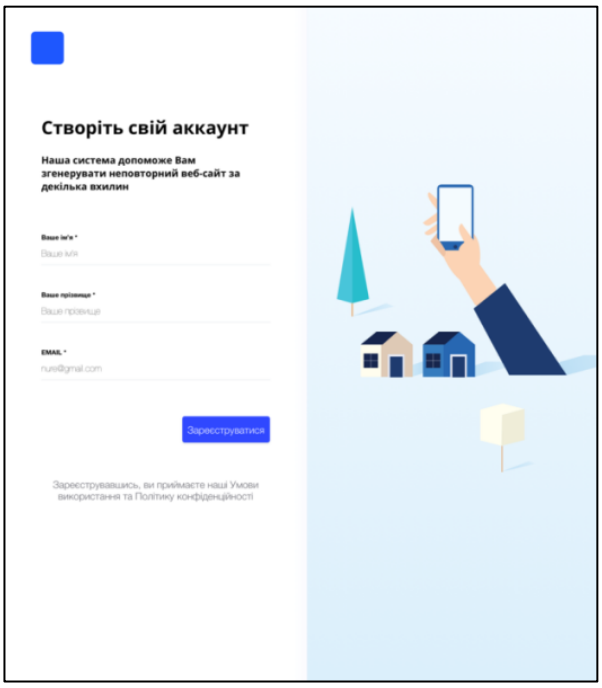

Рисунок 4.3 – Сторінка реєстрації користувача

На наступній сторінці зображений початковий етап генерування веб-сайту. Відображено примірний час заповнення даних та кількість кроків для отримання результату, інформація про надані послуги, а також кнопка переходу до початкової сторінки генерування веб-сайту (див. рис. 4.4).

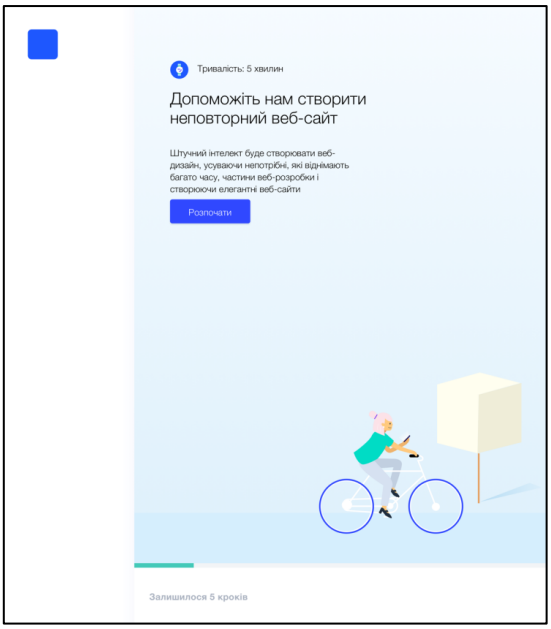

Рисунок 4.4 – Сторінка з початковим етапом заповнення даних

Структура основного пакету поділена відповідно контексту використання. Вона містить 4 вкладених пакетів:

compiler – містить в собі iOS, Android та веб-компілятор;

datasets – база даних;

model – крос-сумісність Python 2 і 3 версії;

listensite – пакет, на якому розміщенні компоненти web-додатку;

ui – реалізацію інтерфейсу користувача додатку;

utils – класи та функції, що вирішують маленькі специфічні задачі (конвертація, розмірів, тощо).

Сторінка налаштувань наділена ключовими інструментами для кращого отримання результату. Зокрема, наведені три форми, з яких у користувача запитують обрати 3 кольори для складання кольорової палітри, ключові слова, а також напрямок роботи майбутнього сайту, керувати параметрами кутів. Слід відзначити, що за допомогою кольорової палітри система на основі штучного інтелекту підставить відповідні значення в заготовлений макет, вибраний на основі напрямку діяльності та ключових слів (див. рис. 4.5).

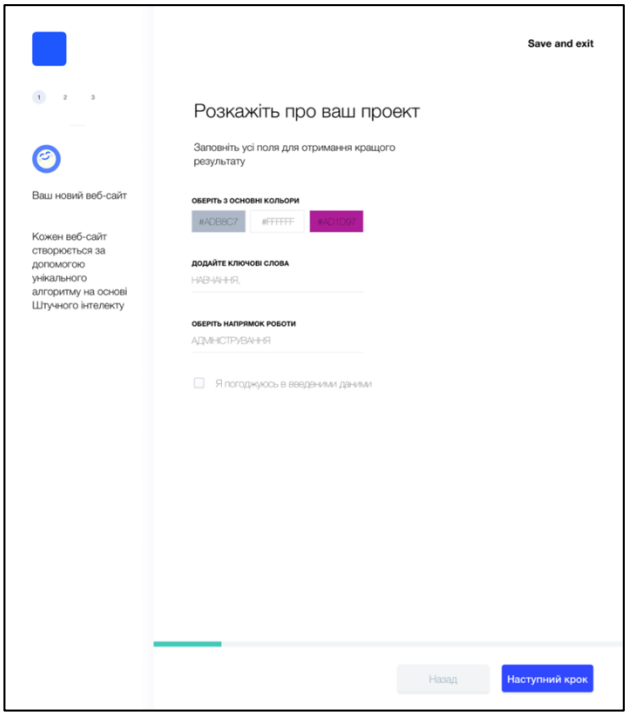

Рисунок 4.5 – Сторінка налаштувань

Сторінка вибору оформлення дизайну веб-сайта дозволяє обрати з 3-6 основних категорій, які обираються після обрання напрямку та ключових слів.

Також на сторінці відображена форма для заповнення з назвою проекту, а також три формати отримання результату: html програмний код (див. рис. 4.6).

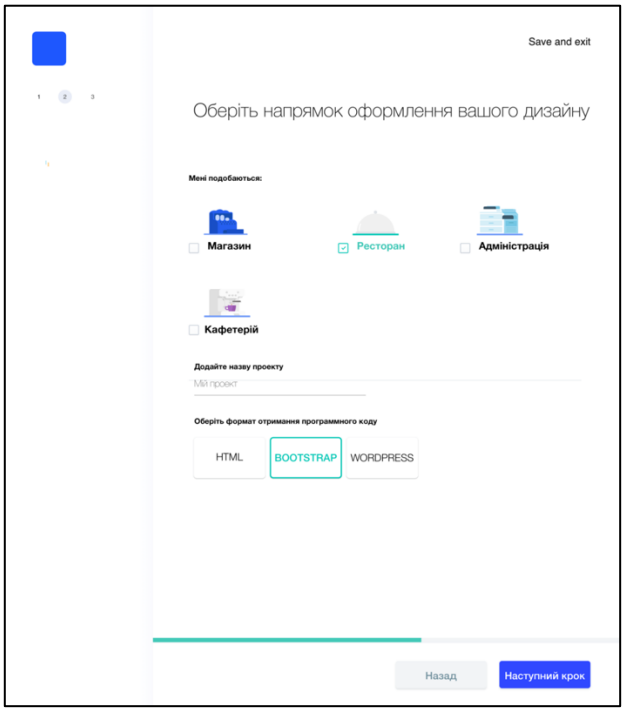

Рисунок 4.6 – Сторінка обрання налаштувань дизайну

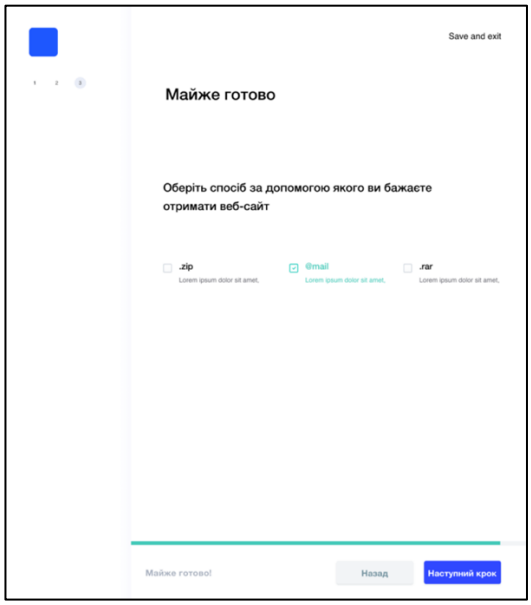

Рисунок 4.7 – Сторінка способу отримання результату

Остання сторінка знаменується обранням форми отримання матеріалів. Зокрема відображені 3 найпоширеніші способи, зокрема формати стиснення даних .zip та .rar, а також отримання по електронній скриньці. В подальшому потрібно

надати можливість обрання з кількох популярних систем спільної розробки програмного забезпечення, такими як GitHub, Bitbucket та ін. (див. рис. 4.7).

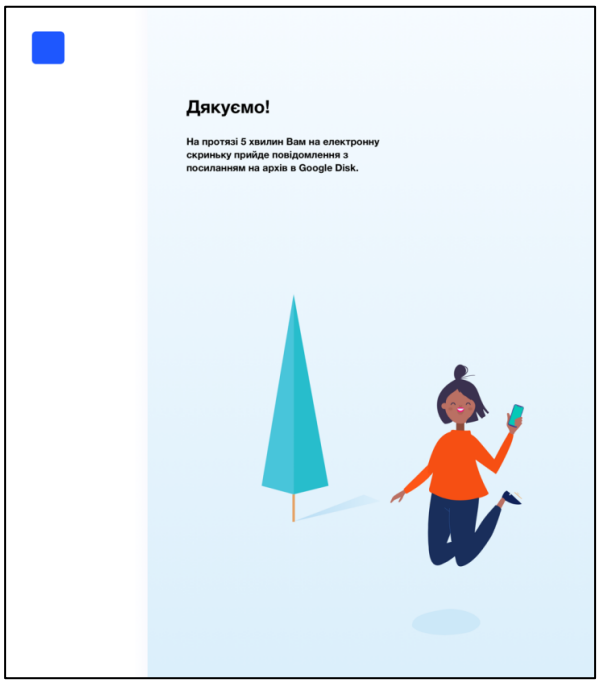

Рисунок 4.8 – Сторінка про успішний результат генерування

На останній сторінці відображений текст про успішний процес генерування веб-додатку. Після успішного заповнення форм користувачем, система починає генерувати веб-сайт. Наступним кроком є розміщення отриманого результату на тимчасовій основі в Google Disk (див. рис. 4.8) та повідомлення користувача через поштовий SMTP-клієнт про успішний результат та доступ до хмарного сховища.

Отже, на основі отриманих результатів було створено програмну систему, що складається із 2 додатків. Перший – для генерації з існуючих шаблонів та графічних зображень програмний код, другий – для зібрання програмного коду, графічних зображень та інших матеріалів в завершене рішення. Також було проаналізовано ключові абстракції програмного коду та розглянуто процес генерації веб-сайтів користувачем.

ВИСНОВКИ

В рамках дослідження методів автоматичної генерації веб-сайтів було проведено глибокий аналіз стандартів web. Зазначено, що сайти розроблені за допомогою ШІ, надзвичайно корисні і прості в створенні. Кількість підприємств, які бажають створити свої веб-сайти, зростає. Згідно з дослідженням, проведеним Clutch, в 2016 році 46 відсотків малих підприємств не мали веб-сайтів, але це число скоротилося до 29 відсотків всього за один рік. Такий величезний попит змусить людей, які керують цим бізнесом, шукати дешеві і швидкі рішення. Слід підкреслити, що веб-сайти, розроблені на основі ШІ, обтяжені великою кількістю проблем, таких як неструктурований програмний код, який буде доволі складно в подальшому змінювати. Утім, для більшості користувачів такий варіант буде прийнятним.

Це також означає, що штучний інтелект в основному здатний справлятися із заданими завданнями так само, як початкові веб-розробники, і що робота останніх може виявитися під загрозою в недалекому майбутньому. Але великі відомі бренди, ймовірно, не будуть задоволені дешевим посереднім зовнішнім виглядом і обмеженим інтерфейсом. Таким чином, у майбутньому масове виробництво веб-додатків буде здійснюватися за допомогою ШІ. Водночас, першокласні, зроблені за індивідуальним замовленням сайти будуть замовляти компанії виключно з високим рейтингом.

На першому етапі аналізу теорії штучного інтелекту, машинного навчання, штучних нейронних мереж було проведено глибокий аналіз існуючих стандартів і виділено велику теоретичну та практичну базу для формування базиса генерування веб-додатків. Розглянуті основні інструменти для створення програмної системи, зокрема розглянуто мову програмування Python, фреймворк Django, програмну бібліотеку Pix2code та бібліотеку машинного навчання TensorFlow. У результаті сформований вибір інструментів для створення програмної системи.

Результатом аналізу та будування штучної нейронної мережі за допомогою Python, TensorFlow та Pix2Code став робочий синтез конкретних методів, із

допомогою яких можна досягти поставленої цілі. Проаналізований і складений алгоритм побудови програмного коду з вхідних зображень та заданого тексту. Виділені способи трансформування отриманого результату в html та фреймворка bootstrap.

На основі отриманих результатів було створено програмну систему, що складається із 2 додатків. Перший – для генерації з існуючих шаблонів та графічних зображень програмний код, другий – для зібрання програмного коду, графічних зображень та інших матеріалів в завершене рішення. Також було проаналізовано ключові абстракції програмного коду та розглянуто процес генерації веб-сайтів користувачем.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Дрожжинов В.И., Райков А.Н. Веб-технологии, искусственный интеллект и когнитивное правительство [Электронный ресурс]: Современные информационные технологии и ИТ- образование. 2017. Том. 13. №2. С. 153 – 169. URL: https://cyberleninka.ru/article/v/veb-tehnologii-iskusstvennyy-intellekt-i-kognitivnoe-pravitelstvo

2. Radha Guha Toward the Intelligent Web Systems // First International Conference on Computational Intelligence, Communication Systems and Networks. – 2009. c. 459 – 463.

3. Keshab Nath What Comes after Web 3.0? Web 4.0 and the Future [Electronic resource]: International Conference on Computing and Communication Systems (I3CS'15). 2015. URL: https://www.researchgate.net/profile/Keshab_Nath2/publication/281455061_What_Comes_after_Web_30_Web_40_and_the_Future/links/55e8b3eb08ae3e1218425040.pdf

4. Sareh Aghaei, Mohammad Ali Nematbakhsh, Hadi Khosravi Farsani Evolution of the world wide web: from web 1.0 to web 4.0 [Electronic resource]: International Journal of Web & Semantic Technology. 2012. Vol. 3. №1. c. 110. URL: https://pdfs.semanticscholar.org/8cb3/93c3229e8f288febfa4dac12a0f6298efb93.pdf

5. Соловяненко Д.В. Ключевые принципы ВЕБ 2.0 // Культура народов Причерноморья. №100. т. 2. с. 147 – 151.

6. Nupur Choudhury World Wide Web and Its Journey from Web 1.0 to Web 4.0 [Electronic resource]: International Journal of Computer Science and Information Technologies. 2014. Vol. 5(6). c. 8096 – 8100. URL: http://ijcsit.com/docs/Volume%205/vol5issue06/ijcsit20140506265.pdf

7. Nath K., Dhar Mr. S., Basishtha S. Web 1.0 to Web 3.0 – Evolution of the Web and its Various Challenges [Electronic resource]: International Conference on Optimization, Reliabilty, and Information Technology (ICROIT). 2014. URL: https://www.researchgate.net/publication/269310255_Web_10_to_Web_30_-_Evolution_of_the_Web_and_its_various_challenges

8. Fernando Almeida Concept and Dimensions of Web 4.0 [Electronic resource]: Concept and Dimensions Internetional journal of Computers and Technology. 2014. Vol. 16. №7. c. 7040 – 7046. URL: https://pdfs.semanticscholar.org/d38c/0b90e2501a34c8d8da9f985b079dea778d6b.pdf

9. Нургалеева Лариса Владимировна Семантика сетей в оптые развития концепций искусственного интеллекта // Гуманитарная информатика. 2012. №6. URL: https://cyberleninka.ru/article/n/semantika-setey-v-optye-razvitiya-kontseptsiy-iskusstvennogo-intellekta

10. Kosmayer D. How to build a highly converting website with Artificial Intelligence [Electronic resource]: Crank wheel. 2017. URL: https://crankwheel.com/how-to-build-a-highly-converting-website-with-artificial-intelligence/

11. Constandse C. How AI-driven website builders will change the digital landscape [Electronic resource]: Medium. 2018. URL: https://uxdesign.cc/how-ai-driven-website-builders-will-change-the-digital-landscape-a5535c17bbe

12. Ismail K. The Grid AI-Powered Website Builder Doesn’t Check Enough Boxes [Electronic resource]: CMS WiRE. 2017 URL: https://www.cmswire.com/digital-experience/the-grid-ai-powered-website-builder-doesnt-check-enough-boxes/

13. Reed L. Will AI Website Builders Ever Replace Human Web Designers? [Electronic resource]: Dzone. 2018. URL: https://dzone.com/articles/will-ai-website-builders-ever-replace-human-web-de

14. Vukcevic P. A Look at AI Website Builders: The Future or Just a Gimmick [Electronic resource]: COMPARE THE CLOUD LTD. 2017. URL: https://www.comparethecloud.net/articles/ai/look-ai-website-builders-future-just-gimmick/

15. Agarwal D. An Introduction to AI at LinkedIn [Electronic resource]: Linkedin Engineering. 2018. URL: https://engineering.linkedin.com/blog/2018/10/an-introduction-to-ai-at-linkedin

16. Brandon J. This A.I.-Powered Website Creator Stood Up to Rigorous Testing [Electronic resource]: INC. 2018. URL: https://www.inc.com/john-brandon/reviewed-this-ai-powered-website-creator-stood-up-to-rigorous-testing.html

17. Зубик Л. В., Зубик Я. Я. Розвиток систем штучного інтелекту // Вісник Чернігівського національного педагогічного університету. Серія: Педагогічні науки. 2018. Вип. 153. с. 49 – 52. URL: http://nbuv.gov.ua/UJRN/VchdpuP_2018_153_13

18. Tony Beltramelli Pix2code: Generating Code from a Graphical User Interface Screenshot [Electronic resource]: arXiv:1705.07962v2 [cs.LG]. 19 Sep, 2017. URL: https://arxiv.org/pdf/1705.07962.pdf

19. Abdel-Badeeh M. Salem Web Intelligence: A New Paradigm for Virtual Communities and Web Science // Telecommunications in Modern Satellite, Cable and Broadcasting Services. 2007. TELSIKS 2007. 8th International Conference on. 2007.

20. Susloparov D. Artificial Intelligence in Web Development // Vardot. 2017. URL: https://www.vardot.com/en/blog/artificial-intelligence-web-development

21. Jones S. How AI is reshaping web development. Innovation Enterprise // Innovation enterprise. 2019. URL: https://channels.theinnovationenterprise.com/articles/how-ai-is-reshaping-web-development

22. Івченко К. В. Машинне навчання та когнітивні обчислення [Електронний ресурс]: Матеріали міжнародної науково- практичної інтернет-конференції "Використання інноваційних технологій в процесі підготовки фахівців", Вінниця, 28-29 березня 2017 р. 2017. URL: https://conferences.vntu.edu.ua/index.php/itpf/2017/paper/view/3285

23. Bala Venkatesha An Introduction to Machine Learning Theory [Електронний ресурс]: Medium, 27 April, 2018. URL: https://medium.com/@venkateshpnk22/an-introduction-to-machine-learning-theory-2b09a4748187

24. Тихонов Алексей Анатольевич Большие данные и глубокое машинное обучение в искусственных нейронных сетях // Наука и образование сегодня. 2018. №6 (29). URL: https://cyberleninka.ru/article/n/bolshie-dannye-i-glubokoe-mashinnoe-obuchenie-v-iskusstvennyh-neyronnyh-setyah

25. Мошенченко М. С. Штучні нейронні мережі [Електронний ресурс]: Proceedings of the International Scientific Conference «Topical problems of modern

science» (June 16, 2017, Warsaw, Poland). 2017. URL: http://archive.ws-conference.com/wp-content/uploads/pw0060.pdf

26. Новотарський М.А., Нестеренко Б.Б. Штучні нейронні мережі: обчислення // Праці Інституту математики НАН України. Т 50. Київ: Ін-т математики НАН України, 2004. 408 с.

27. Зуенко А.А., Алмаматов А.А. Программирование в ограничениях на языке Python с применением структур и алгоритмов алгебры кортежей // Труды Кольского научного центра РАН. 2014. №5 (24). URL: https://cyberleninka.ru/article/n/programmirovanie-v-ogranicheniyah-na-yazyke-python-s-primeneniem-struktur-i-algoritmov-algebry-kortezhey

28. Васильев П. А. Web-программирование на языке python. Фреймворки django, Flask // Наука, техника и образование. 2016. №8 (26). URL: https://cyberleninka.ru/article/n/web-programmirovanie-na-yazyke-python-freymvorki-django-flask

29. Noah Gundotra Code2Pix — Deep Learning Compiler for Graphical User Interfaces [Electronic resource]: Towards Data Science. 2018. URL: https://towardsdatascience.com/code2pix-deep-learning-compiler-for-graphical-user-interfaces-1256c346950b

30. Gupta A. How Artificial Intelligence Is Reshaping Web Designing And Web Development? [Electronic resource]: Express Computer. 2019. URL: https://www.expresscomputer.in/artificial-intelligence-ai/how-artificial-intelligence-is-reshaping-web-designing-and-web-development/32201/

31. Model View Controller Pattern. Retrieved from [Electronic resource]: Tutorialspoint. URL: https://www.tutorialspoint.com/python_design_patterns/python_design_patterns_model_view_controller.htm

32. Model View Controller (MVC) [Electronic resource]: Djangospin. 2019. URL: https://www.djangospin.com/design-patterns-python/model-view-controller-mvc/

33. Giacomo D. MVC pattern in Python: Introduction and BasicModel [Electronic resource]: Giacomodebidda. 2017. URL: https://www.giacomodebidda.com/mvc-pattern-in-python-introduction-and-basicmodel/

34. Мицель А.А., Погуда А.А., Poguda A. Нейросетевой подход к задаче тестирования // Прикладная информатика. 2011. №5 (35). URL: https://cyberleninka.ru/article/n/neyrosetevoy-podhod-k-zadache-testirovaniya

35. Abadi M. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems [Electronic resource]: arXiv:1603.04467v2 [cs.DC]. 2016. URL: https://arxiv.org/pdf/1603.04467.pdf