МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Інститут післядипломної освіти

Кафедра інформаційних технологій

Кваліфікаційна робота бакалавра

на тему: Біометрична система доступу працівників із автоматичним температурним скринінгом

Виконав студент групи КН-5
спеціальності 122 «Комп'ютерні науки»
Андоній Анатолій Андрійович

Керівник к.геогр.н., доцент
Кузніченко Світлана Дмитрівна

Консультант _____

Рецензент к.техн.н., доцент
Гнатовська Ганна Арнольдівна

Одеса 2023

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД – база даних

ОС – операційна система

ПЗ – програмне забезпечення

ADT – Android Development Tools – Інструменти розробки в ОС Android

API – Application Programming Interface – програмний інтерфейс програми

AR – Augmented Reality – доповнена реальність

IDE – Integrated Development Environment – інтегроване середовище розробки

SDK – Software Development Kit – набір засобів розробки

ВСТУП

Сьогодні інформаційні технології є невід'ємною частиною будь-якої сфери нашого життя, вже нема жодної галузі, де не було б комп'ютеризовано важливі процеси. Людство намагається автоматизувати усі процеси, які можливо, оскільки це економить час, людські ресурси й також надає можливість робити це без втручання людини. Через прискорений розвиток комп'ютерних технологій впроваджуються нові системи, що надають людям більш комфортне перебування у будь-якому місці або спрощує ручну роботу.

Задля створення нових способів організації людського життя створюються все більш складні інформаційні системи, що можуть поєднувати багато різних модулів, наприклад, поєднання вебзастосунків та програмованих мобільних застосунків із сервером, що має власну базу даних, може отримувати запити із апаратних модулів – датчиків відеонагляду, звукових або температурних. Це призводить до вимушеного розвитку нових напрямів у IT, від IoT або обробки великої кількості даних, що генерується складними системами до оптимізації процесів – Big Data.

Оскільки ще недавно була актуальною проблема пандемії коронавірусу, а також сезонні вірусні хвороби є постійними, слід направити набуті знання на поєднання автоматизації та запобіганні розповсюдження респіраторних хвороб, наприклад, автоматизувати ідентифікацію людей на робочому місці та визначення їх температури.

Ця тематика має наступні цілі:

- автоматизувати доступ працівників до місця праці;
- швидка ідентифікація особи за обличчям;
- безконтактне вимірювання температури;
- аналітика параметрів відвідування працівників.

Отже, метою кваліфікаційної роботи є розробка біометричної системи доступу робітників з автоматичним температурним скринінгом.

Для досягнення цілі необхідно вирішити наступні задачі:

− проаналізувати аналогічні системи біометричного доступу, а також виявити функціонал, що слід запозичити або вдосконалити;

− обрати інструменти для проєктування;

− спроєктувати вебчастину, серверну частину, комунікацію апаратної частини із сервером;

− виконати реалізацію фронтенду, бекенду, скриптів для апаратної частини, перенавчити нейромережу для розпізнавання;

− протестувати взаємодію частин системи.

Структура кваліфікаційної роботи складається вступу, трьох розділів, висновків, переліку посилань на 15 найменувань, додатків. Повний обсяг роботи становить 67 сторінок, містить 36 рисунків і 7 таблиць.

1 ОГЛЯД ІСНУЮЧИХ СИСТЕМ БІОМЕТРИЧНОГО ДОСТУПУ ТА ТЕХНОЛОГІЙ ЇХ РОЗРОБЛЕННЯ

При розробці будь-якого продукту після визначення ідеї слід проаналізувати ринок на наявність аналогів. Це є корисним, оскільки в умовах здорової конкуренції виграє той, що має переваги перед іншими. Це можливо завдяки покращення наявного функціоналу, визначення та усунення недоліків, пропозиції щодо маркетингових умов, наприклад, більш приємні ціни або надання певних послуг з супроводі як підтримка.

Аналоги можуть відрізнятися не лише за функціоналом, а складовими частинами, що можуть відрізнятися технологіями розробки.

Також аналіз конкурентних пропозицій надає можливість визначитися із стеком технологій для розробки продукту.

Для біометричної системи доступу робітників слід розглянути конкурентів, що також надають можливість розпізнавати робітників за обличчям, можливо, вони мають якість свої певні цікаві функції, які слід взяти до уваги, або навпаки застережити у своїй роботі.

1.1 Порівняння аналогів систем біометричного доступу

1.1.1 Біометрична система доступу «ANVIZ Face Pass Pro Recognition System»

Як заявляє виробник, їх продукт повністю має новий базовий алгоритм BioNANO (рис. 1.1) та потужну апаратну платформу, що гарантує, що швидкість ідентифікації терміналу становить менше 1 секунди. Інфрачервоне джерело світла дозволяє терміналу забезпечувати освітлення у темряві. Має обмежену кількість для користувачів до 500 та тільки 100000 записів. До недо-

ліків також можна віднести відсутність серверної частини, неможливість виміряти температуру.

Рисунок 1.1 – Система контролю “ANVIZ Face Pass Pro Recognition System”

1.1.2 Біометрична система доступу «ZKTECO LX50»

Термінал AMG-FRX представляє собою також вбудовану камеру із детектором температури (рис. 1.2). Як заявляє розробник, система дозволяє розпізнавати людей у масці, а також вимірювати їх температуру. Із рекламного ролику відомо, що також система відправляє повідомлення, якщо людина із температурою, а також надає змогу передивлятися хто приходив у який час. Система є досить дорогою – 3000$.

1.1.3 Біометрична система доступу «R20-Face»

Не потрібне встановлення додаткового ПЗ і запуск окремого сервера для відеоаналітики.

У разі відсутності зв'язку з сервером зв'язка "Термінал-Контролер" продовжує працювати автономно на підставі раніше занесених даних. Тобто

співробітники можуть як і раніше проходити на об'єкт. Після відновлення зв'язку з сервером всі події, що відбулися за період відсутності зв'язку, заносяться в термінал і контролер.

Рисунок 1.2 – Система контролю “ZKTECO LX50”

Моделі терміналу, обладнані спеціальним датчиком, можуть вимірювати температуру користувачів (рис. 1.3).

Система є вкрай зручною, оскільки також може вимірювати температуру та має ідентифікацію за обличчям, проте система не дає можливість вручну вести прихід людини та робити статистику.

1.1.4 Біометрична система доступу «Swiftlan»

Даний аналог представляє собою також термінал із камерою, що може розпізнавати людей (рис. 1.4). Як зазначено на сайті, це є підсистема великої інформаційної системи, що базується на IoT. Вона надає змогу регулювати доступ людей до приміщень, може повідомляти інших людей про намір зайти у кімнату.

Рисунок 1.3 – Система контролю “ R20-Face”

З опису схоже, що це система створена для власної екосистеми великих будівель, проте також може використовуватися для офісів.

Рисунок 1.4 – Система контролю “Swiftlan”

1.1.5 Результат порівнянь

Після аналізу аналогів можна зробити порівняльну таблицю із функціоналом аналогів та зробити висновки, що має бути реалізовано та враховано у системі, що розробляється (табл. 1.1).

Таблиця 1.1 – Порівняння аналогів

Характеристики	Системи				
	ANVIZ	R20-Face	ZKTECO LX50	Swiftlan	Кваліфікаційна
Ідентифікація за обличчям	+	+	+	+	+
Необмежена кількість користувачів та записів	–	–	–	–	+
Ведення статистики	–	+	+	+	+
Наявність централізованого серверу	+	+	+	+	+
Вимірювання температури	–	+	+	+	+
Можливість самостійно вносити запис щодо прибуття	–	-	+	+	+
Можливість встановлювати графік роботи (для визначення запізнення)	–	–	–	+	+

Продукт кваліфікаційної роботи має ідентифікувати людей за обличчям, враховуючи аналоги. Потрібно не обмежувати користувачів та кількість осіб, що можуть бути розпізнані. Також при авторизації за обличчям потрібно вимірювати температуру особи та записуватися у базу даних. Завдяки чому буде можливість отримувати статистику щодо запізнень та фізичного стану осіб, наприклад, за певний проміжок часу.

Система повинна мати централізований сервер, який буде оброблювати дані, також у майбутньому це дозволить розширювати можливості продукту, у разі якщо будуть виникати нові вимоги до функціональності.

На жаль, невідомо, які технології використані у аналогах, оскільки це комерційна інформація, яка не поширюються, але у розробляємій системі мають бути використані сучасні інструменти, щоб забезпечити усі функціональні та нефункціональні вимоги, а також створити умови для легкої підтримки написаного коду та легкого розширення кодової бази.

Отже, було розглянуто аналоги системи, що розробляється. Можна зауважити, що поки цей напрямок набирає обертів, і такі системи не дуже розповсюджені. Серед існуючих функцій конкурентів можна побачити, що головне – це мати змогу швидко ідентифікувати особу за обличчям. Серед основного недоліку можна виділити велику ціну продуктів.

1.2 Вибір інструментів для проєктування

Майбутній продукт складатиметься з трьох частин: серверу, фронтчастини та апаратної частини, тому треба обґрунтовувати вибір технологій окремо для кожної з частин.

Для серверу треба обрати мови програмування, фреймворки та базу даних.

Для фронтенд частини доцільно обрати фреймворк мови Javascript, оскільки вона є найпопулярнішою для розробки веб-застосунків.

Для апаратної чатини треба обрати мови програмування, камеру, завдяки якій буде відбуватися ідентифікація, та засіб для заміру температури.

1.2.1 Вибір архітектурного шаблону

Для взаємодії окремих частин продукту обрано “клієнт-серверну архітектуру” (рис.1.5).

Є загальні ресурси та сервіси, до яких потрібно забезпечити доступ великої кількості розподілених клієнтів, і при цьому необхідно контролювати доступ або якість обслуговування.

Керуючись набором загальних ресурсів і сервісів, можна забезпечити модифіційність і повторне використання, для чого загальні сервіси винесуться окремо, щоб їх можна було змінювати в одному місці або в невеликій кількості місць. Можна поліпшити масштабованість і доступність до використання за рахунок централізованого управління цими ресурсами і сервісами при одночасному розподілі самих ресурсів між декількома фізичними серверами [1].

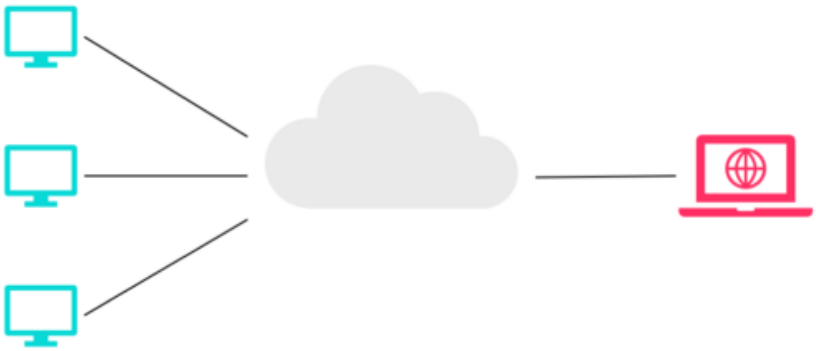

Рисунок 1.5 – Схема взаємодії “Клієнт-серверної архітектури”

У підході «клієнт - сервер» компоненти і сполучні елементи мають певну поведінку:

− компоненти, звані «клієнтами», відправляють запити компоненту, званому «сервер», і чекають відповіді;

− компонент «сервер» отримує запит від клієнта і відправляє йому відповідь.

Підхід «клієнт - сервер» можна застосовувати в моделюванні частини системи, що має багато компонентів, які відправляють запити до іншого компонента який забезпечує роботу сервісів, наприклад, онлайн-додатки (електронна пошта, обмін документами і банківська справа).

1.2.2 Вибір мови програмування

1.2.2.1 Вибір мови серверної частини

Найбільш використовуваними для серверу є мови Java та C#, що майже є сестрами. Java – об'єктно-орієнтована мова програмування, випущена 1995 року компанією «Sun Microsystems» як основний компонент платформи Java. З 2009 року мовою займається компанія «Oracle», яка того року придбала «Sun Microsystems». В офіційній реалізації Java-програми компілюються у байт-код, який при виконанні інтерпретується віртуальною машиною для конкретної платформи.

«Oracle» надає компілятор Java та віртуальну машину Java, які задовольняють специфікації Java Community Process, під ліцензією GNU General Public License.

Мова значно запозичила синтаксис із C і C++. Зокрема, взято за основу об'єктну модель C++, проте її модифіковано. Усунуто можливість появи деяких конфліктних ситуацій, що могли виникнути через помилки програміста та полегшено сам процес розробки об'єктно-орієнтованих програм. Ряд дій, які в C/C++ повинні здійснювати програмісти, доручено віртуальній машині.

Передусім Java розроблялась як платформо-незалежна мова, тому вона має менше низькорівневих можливостей для роботи з апаратним забезпеченням, що в порівнянні, наприклад, з C++ зменшує швидкість роботи програм. За необхідності таких дій Java дозволяє викликати підпрограми, написані іншими мовами програмування.

C# – об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft Research (належить Microsoft).

Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато від своїх попередників – мов C++, Object Pascal, Модула і Smalltalk – C#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад, мова C#, на відміну від C++, не передбачає множинне успадкування класів [2, 3].

Таблиця 1.2 – Порівняння мов програмування для серверу

Параметри	C#	Java
Типізація	Строга	Строга
Спеціалізація	Сервери, ігри	Сервери, мобільні пристрої
Ком'юніті	6.2 мільйони	7.6 мільйони
Фреймворки	ASP.NET, .NET	Java EE, Spring

Отже, найкращим варіантом є мова Java, оскільки розвиненість мови та наявність готових рішень підтверджується великою кількістю розробників, а

також фреймворк Spring є вирішальним, бо він надає можливість створювати програми наявною великою кількістю підключаємих модулів.

1.2.2.2 Вибір мови фронт частини

JavaScript – це мова програмування, яка зазвичай використовується при веб-розробці . Спочатку вона була розроблена Netscape як засіб для додавання динамічних та інтерактивних елементів до веб-сайтів. Хоча на JavaScript впливає Java , синтаксис більше схожий на C і базується на ECMAScript, мові сценаріїв, розробленій Sun Microsystems [4].

JavaScript – це мова сценаріїв на стороні клієнта, що означає, що вихідний код обробляється веб-браузером клієнта, а не веб-сервером . Це означає, що функції JavaScript можуть працювати після завантаження веб-сторінки без зв’язку із сервером. Наприклад, функція JavaScript може перевірити веб-форму перед її поданням, щоб переконатися, що всі обов’язкові поля заповнені. Код JavaScript може видавати повідомлення про помилку до того, як будь-яка інформація фактично передається на сервер.

Оскільки у подальшому буде використовуватися фреймворк Angular, то для фронта використовуватиметься модифікована версія Javascript – TypeScript.

TypeScript має незвичне відношення до JavaScript. TypeScript пропонує всі функції JavaScript, а також додатковий шар: система типів TypeScript.

Наприклад, JavaScript надає примітиви мови такі як string, number і object, але не перевіряє , що вони послідовно проініціалізовані. TypeScript це перевіряє.

Головна перевага TypeScript полягає в тому, що він може виявити несподівану поведінку у коді, знижуючи ймовірність помилок.

1.2.2.3 Вибір мови апаратної частини

C – мова програмування загального призначення, надзвичайно популярна, проста та гнучка у використанні. Це структурована мова програмування, яка не залежить від машини та широко використовується для написання різних програм, операційних систем, таких як Windows, та багатьох інших складних програм, таких як база даних Oracle, Git, інтерпретатор Python тощо. Розглянемо деякі характеристики мови C:

- широко використовується у вбудованих системах;
- використовується для розробки системних застосунків;
- широко використовується для розробки настільних застосунків;
- більшість програм Adobe розроблені з використанням мови програмування C;
- використовується для розробки браузерів та їх розширень, наприклад, Google Chromium;
- використовується для розробки баз даних, наприклад найпопулярнішої БД MySQL;
- використовується при розробці операційної системи, таких як Apple X OS, Microsoft Windows і Symbian;
- використовується для виробництва компіляторів;
- широко використовується в додатках IOT.

Проте є інший конкурент для програмування апаратної частини – Python.

Python – надзвичайно популярна мова програмування, що використовується для різного роду речей, починаючи від розробки вебзастосунків та програм, закінчуючи програмуванням одноплатних комп'ютерів та мікроконтролерів. Python – це те, що більшість людей використовують для програмування своїх Raspberry Pi. До недавнього часу більшість орієнтованих на виробників мікроконтролерів, включаючи Arduino, використовували мови, засновані на C і C ++ [5].

Чим саме Python краще C? C та C ++ – це компільовані мови. Коли завантажується код на МК, комп'ютер компілює код у двійкові інструкції, а потім надсилає ці інструкції на МК. Python – це інтерпретована мова, тому при запуску скриптів не потребує нової перекомпіляції та перепрошивки.

Отже, з вищезгаданих переваг для програмування апаратного модулю буде використано мову Python.

1.2.3 Вибір середовища проєктування

1.2.3.1 Вибір середовища проєктування серверної частини

Для написання коду для серверу на Java в основному використовуються дві IDE: Intelij IDEA та NetBeans.

NetBeans – це повноціне середовище розробки, яке дозволяє створювати всі програмні продукти, з якими доводитться зустрічатися. Сюди входять як додатки для вебу, так і настільні і мобільні програми. Завдяки підтримці широкого кола мов програмування середовище може називати себе однією з найбільш універсальних в світі. Причому, різнобічний функціонал не перевищує пріоритет якості виконання.

Важливим аспектом, який змушує програмістів вибирати NetBeans, є, звичайно ж, безкоштовність середовища розробки. Набагато легше використовувати те програмне забезпечення, яке знаходиться у відкритому доступі. Контрольний постріл для забезпечення лояльності - це універсальність. В рамках діяльності веб-розробника важко придумати хоч найменшу причину, яка змусила б змінити NetBeans на щось інше.

Програмне забезпечення JetBrains IntelliJ IDEA – це провідне середовище швидкої розробки на мові Java. IntelliJ IDEA є високотехнологічним комплексом тісно інтегрованих інструментів програмування, що включає інтелектуальний редактор вихідних текстів з розвиненими засобами автоматизації, потужні інструменти рефакторинга коду, вбудовану підтримку

технологій J2EE, механізми інтеграції з середовищем тестування Ant / JUnit і системами управління версіями, унікальний інструмент оптимізації та перевірки коду Code Inspection, а також інноваційний візуальний конструктор графічних інтерфейсів.

Унікальні можливості JetBrains IntelliJ IDEA позбавляють програміста від вантажу рутинної роботи, допомагають своєчасно усунути помилки і підвищити якість коду, піднімаючи продуктивність розробника на нову висоту.

Оскільки Intelij IDEA є дуже зручним інструментом для роботи із фреймворком Spring, то його буде використано для розробки серверу.

1.2.3.2 Вибір середовища проєктування фронт частини

Тут ми маємо список найкращих Angular IDE, які потрібно використовувати, і які можуть допомогти у розробці мобільних та веб-додатків нового покоління.

Найкращі IDE та інструменти:

- Webstorm;
- Visual Studio Code;
- Brackets;
- Atom.

Webstorm – це потужний редактор коду, створений IntelliJ, розроблений Jetbrains, і є чудовим вибором для кодування програм Angular 2 на основі TypeScript. За допомогою Webstorm можна починати розробку компонентів Angular 2 з першого дня, оскільки він забезпечує вбудований TypeScript. Він також усуває вашу залежність від сторонніх плагінів, оскільки можна легко компілювати код, створений у TypeScript, використовуючи чистий ванільний JavaScript.

Перелік переваг використання Webstorm:

– підтримує JavaScript, Node.js, HTML та CSS;

– пропонує надійну навігацію та рефакторинг;

– забезпечує інтеграцію з VSC та інтелектуальну допомогу в кодуванні;

– надає додатковий плагін для екосистеми з повною конфігурацією, функцією локальної історії тощо;

– підтримує новітні технології з підтримкою налагодження, трасування та тестування.

Код Visual Studio від Microsoft для Windows, OS X та Linux. Це редактор вихідного коду, який підтримує TypeScript, включаючи налагодження, інтелектуальне завершення коду на основі типів змінних, функцій та імпортованих модулів. Він надає різні функції, такі як фрагменти та рефакторинг коду. Visual Studio Code пропонує виділення синтаксису та автозаповнення за допомогою IntelliSense.

Переваги використання коду Visual Studio:

– підтримує різні мови;

– вбудовані команди Git;

– швидке налагодження;

– легко налаштувати.

Sublime Text – це вишуканий вибір для текстового редактора коду, розмітки. Він має гладкий та акуратний користувальницький інтерфейс з надзвичайними функціями та надзвичайною продуктивністю. Сьогодні це широко використовуваний редактор коду. Це також один з найкращих кутових редакторів.

Список переваг використання Sublime Text:

– забезпечує підтримку редагування коду TypeScript, але потрібно встановити плагін Microsoft TypeScript;

– має можливість запустити процес збірки та скомпілювати файл до JavaScript, над яким йде праця, натиснувши функціональну клавішу F7;

– можливість ввімкнути помилку коду в реальному часі;

– Sublime Text доступний для Linux, OS X та Windows. Для використання Sublime Text на кожному комп'ютері незалежно від операційної системи, яку він використовує, потрібна одна ліцензія. Sublime Text використовує спеціальний набір інтерфейсів користувача, оптимізований для швидкості та краси, одночасно використовуючи переваги власних функцій на кожній платформі;

– доступний для OS X, Windows та Linux з однією ліцензією для будь-якої операційної системи;

– пропонує власний набір інструментів UT;

– має можливість виконати крос-платформне редагування;

– підтримує редагування коду TypeScript.

Якщо необхідне поєднання візуальних інструментів у редакторі коду, тоді Brackets – це ваш ідеальний вибір. Brackets – це легкий, але надзвичайно потужний IDE для розробки мобільних та вебзастосунків за допомогою Angular. Він підтримує різноманітні мови та ОС, такі як Windows, Mac та Linux. Він широко використовується для творчих розробок.

Переваги використання Brackets:

– має вбудовані редактори;

– має підтримку препроцесора;

– забезпечує попередній перегляд.

Atom розроблений GitHub і є обраним користувачами завдяки легко налаштовуваному середовищу, простоті установки. Можна встановити пакет Atom TypeScript, щоб покращити досвід використання TypeScript за допомогою програм Angular. Можна використовувати APM CLI як вбудований інсталятор пакетів.

Функціональні можливості подібні до Sublime, оскільки він пропонує автоматичні підказки коду, статичну перевірку типу, інтроспекцію коду і може використовувати автоматичну побудову, щоб зберегти та назвати декілька. Він також містить зручний вбудований генератор tsconfig.json. Це фантастичний текстовий редактор, який є сучасним, доступним, але при цьо-

му налаштовується, щоб дозволити нам продуктивно використовувати його, не торкаючись конфігураційного файлу. Він також підтримується активною спільнотою, яка сприяє появі нових плагінів тощо. Він має довгий список пакетів, вбудований менеджер пакетів, тему синтаксису за замовчуванням та вбудований менеджер пакетів.

Переваги редактора Atom:

- швидке та легке авто заповнення;
- дозволяє крос-платформне редагування;
- забезпечує зручний перегляд з кількома панелями;
- підтримує браузер файлової системи;
- має вбудований менеджер пакетів.

Таблиця 1.3 – Порівняння середовищ програмування для фронт частини

Характеристики	Webstorm	VSC	Brackets	Atom
Навігація та редактор	+	+	+	–
Можливість тестування та трасування	+	–	–	+
Інтеграція з іншими редакторами	+	+	–	+
Плагіни з екосистемами та локальної історії	+	–	+	+

Оскільки Webstorm підтримує навігацію, надає можливість тестування та трасування, інтегрується з іншими редакторами, а також має безліч плагінів, то він є кращим варіантом для розробки фронт частини.

1.2.4 Вибір фреймворків та бібліотек

1.2.4.1 Вибір фреймворків серверної частини

Spring Framework, або просто Spring – один з найпопулярніших фреймворків для створення вебзастосунків на Java.

Перше, що потрібно розуміти при розробці додатків: вони складаються з безлічі шарів і залежностей (рис.1.6). Типовий вебзастосунок буде складатися з шару доступу до бази даних, вебконтролерів, сервісів (місце де йде обробка даних), юніт тестів. І це тільки найпростіші програми.

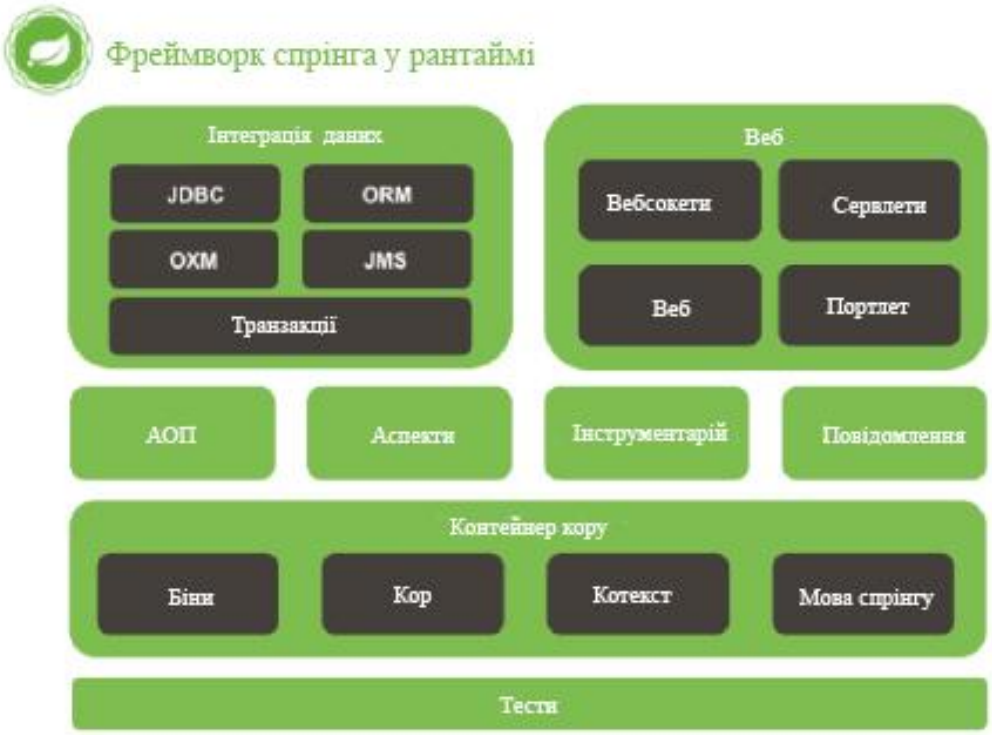

Рисунок 1.4 – Структура Spring

Якщо писати програму використовуючи тільки мову Java, швидше за все буде багато труднощів і необхідності писати так званий «зайвий» код. На прикладі простих вебзастосунків, можна помітити скільки зусиль потрібно щоб підключити базу даних, налаштувати сервлет, або відтворення стилів на вебсторінці.

Використовуючи Spring можна домогтися тих же результатів що і на чистій Java, але набагато меншими зусиллями і меншою кількістю коду.

Мета даного фреймворка – зосередити програміста на вирішенні бізнес завдання замість того, щоб витрачати час на налаштування коду. [6, 7]

Оскільки Spring є гнучким, легковісним, масштабуючим та за допомогою нього можна легко налагоджувати різні модулі, то вибір зупиняється на Java та Spring.

1.2.4.2 Вибір фреймворків фронт частини

React — відкрита JavaScript бібліотека для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту вебсторінки, з якими стикаються в розробці односторінкових застосунків. Розробляється Facebook, Instagram і спільнотою індивідуальних розробників.

React дозволяє розробникам створювати великі вебзастосунки, які використовують дані, котрі змінюються з часом, без перезавантаження сторінки. Його мета полягає в тому, щоб бути швидким, простим, масштабованим. React обробляє тільки користувацький інтерфейс у застосунках. Це відповідає видові у шаблоні модель-вид-контролер (MVC), і може бути використане у поєднанні з іншими JavaScript бібліотеками або в великих фреймворках MVC, таких як AngularJS. Він також може бути використаний з React на основі надбудов, щоб піклуватися про частини без користувацького інтерфейсу побудови вебзастосунків. Як бібліотеку інтерфейсу користувача React найчастіше використовують разом з іншими бібліотеками, такими як Redux.

Vue.js – JavaScript-фреймворк з відкритим вихідним кодом для створення користувацьких інтерфейсів. Легко інтегрується в проєкти з використанням інших JavaScript-бібліотек. Може функціонувати як вебфреймворк для розробки односторінкових додатків в реактивному стилі.

На даний момент підтримується творцем Еваном Ю. і іншими активними членами основної команди з різних компаній, таких як Netlify, Netguru, Baidu, Livestorm.

, який розробляється під керівництвом Angular Team у компанії Google, а також спільнотою приватних розробників та корпорацій. Angular — це AngularJS, який переосмислили та який був повністю переписаний тією ж командою розробників.

Angular має велику кількість переваг у порівнянні з іншими:

- велике ком'юніті;
- декларативний стиль коду;
- використання директив;
- високу швидкість розробки;
- MVC з коробки;
- корисні фічі для SPA;
- модульність;
- наявність готових рішень.

Таблиця 1.4 – Порівняння середовищ програмування для фронт частини

Характеристики	React	Vue.js	Angular
Автозаповнення HTML-компонент	–	+	+
Вбудоване інжектування залежностей	–	–	+
Одностороння прив’язка даних	+	+	+
Архітектура для великах застосунків	–	–	+

Отже, завдяки вищенаведеним перевагам фреймворк Angular є пріоритетнішим для розробки вебчастини.

1.2.5 Вибір бази даних

Інформацію в системі потрібно зберігати та накопичувати, тому треба приділити увагу вибіру базі даних. БД можуть відрізнятися швидкістю обробки, налагоджування, своєю специфікою.

PostgreSQL (вимовляється «Пост-ГРЕС-Кью-Ел») – вільна об'єктно-реляційна система управління базами даних (СКБД).

Існує в реалізаціях для безлічі UNIX-подібних платформ, включаючи AIX, різні BSD-системи, HP-UX, IRIX, Linux, macOS, Solaris / OpenSolaris, Tru64, QNX, а також для Microsoft Windows.

Порівняно з іншими проектами з відкритим кодом, такими як Apache, FreeBSD або MySQL, PostgreSQL не контролюється якоюсь однією компанією, її розробка можлива завдяки співпраці багатьох людей та компаній, які хочуть використовувати цю СУБД та впроваджувати у неї найновіші досягнення.

Сервер PostgreSQL написаний на мові C. Зазвичай розповсюджується у вигляді набору текстових файлів із сирцевим кодом. Для інсталяції необхідно відкомпілювати файли на своєму комп'ютері і скопіювати в деякий каталог. Весь процес детально описаний в документації [8].

MySQL – вільна система керування реляційними базами даних.

MySQL був розроблений компанією «TcX» для підвищення швидкодії обробки великих баз даних. Ця система керування базами даних (СКБД) з відкритим кодом була створена як альтернатива комерційним системам. MySQL з самого початку була дуже схожою на mSQL, проте з часом вона все розширювалася і зараз MySQL – одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення

динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування.

Можливості сервера MySQL[9]:

- простота у встановленні та використанні;
- підтримується необмежена кількість користувачів, що одночасно працюють із БД;
- кількість рядків у таблицях може досягати 50 млн;
- висока швидкість виконання команд;
- наявність простої і ефективної системи безпеки.

MongoDB – документо-орієнтована система керування базами даних (СКБД) з відкритим вихідним кодом, яка не потребує опису схеми таблиць. MongoDB займає нішу між швидкими і масштабованими системами, що оперують даними у форматі ключ/значення, і реляційними СКБД, функціональними і зручними у формуванні запитів.

MongoDB підтримує зберігання документів в JSON-подібному форматі, має досить гнучку мову для формування запитів, може створювати індекси для різних збережених атрибутів, ефективно забезпечує зберігання великих бінарних об'єктів, підтримує журналювання операцій зі зміни і додавання даних в БД, може працювати відповідно до парадигми Map/Reduce, підтримує реплікацію і побудову відмовостійких конфігурацій. У MongoDB є вбудовані засоби із забезпечення шардінгу (розподіл набору даних по серверах на основі певного ключа), комбінуючи який з реплікацією даних можна побудувати горизонтально масштабований кластер зберігання, в якому відсутня єдина точка відмови (збій будь-якого вузла не позначається на роботі БД), підтримується автоматичне відновлення після збою і перенесення навантаження з вузла, який вийшов з ладу. Розширення кластера або перетворення одного сервера на кластер проводиться без зупинки роботи БД простим додаванням нових машин. [4]

Оскільки для роботи системи непотрібна надлишкова функціональність Postgres та зберігання у JSON або швидкість запису як у MongoDB, то вибір зупиняється на MySQL.

Таблиця 1.5 – Порівняння баз даних

Параметри	MongoDB	Postgres	MySQL
Тип	Документо-орієнтована	Об'єктно-реляційна	Реляційна
Архітектура	Windows, Linux та MacOS	Клієнт / сервер	Клієнт / сервер
Формат даних	JSON	Таблиці	Таблиці
Сервер	MongoDB Server	HP UX, Linux, Unix, Solaris,OS X, FreeBDS	Linux, Unix, Solaris,OS X, FreeBDS
Використання SQL	–	+	+

Оскільки для збереження даних для проєкту достатньо звичайної реляційної швидконалаштованої бази, то вибір зупиняється на MySQL.

1.2.6 Вибір патернів проєктування

Для розробки системи буде використано три патерни: MVC, dependency injection та singleton.

MVC – це шаблон дизайну програмного забезпечення, який зазвичай використовується для розробки користувальницьких інтерфейсів, що розділяє відповідну логіку програми на три взаємопов’язані елементи. Це робиться для відокремлення внутрішнього подання інформації від способів подання та прийняття інформації від користувача [10].

Впровадження залежності (англ. Dependency injection, DI) – шаблон проєктування програмного забезпечення, що передбачає надання зовнішньої залежності програмному компоненту, використовуючи «інверсію управління» (англ. Inversion of control, IoC) для розв'язання (отримання) залежностей.

Впровадження – це передача залежності (тобто, сервісу) залежному об'єкту (тобто, клієнту). Передавати залежності клієнту замість дозволити клієнту створити сервіс є фундаментальною вимогою до цього шаблону проєктування. Впровадження залежності буде використовуватися в сервісах для використання об’єктів репозиторію.

Singleton покладає контроль над створенням єдиного об'єкта на клас. Використання до цього об'єкту здійснюється через статичне поле класу, яке повертає покажчик або посилання на нього. Цей об'єкт буде створений тільки при першому зверненні до методу, а всі наступні виклики просто повертають його адресу. Для забезпечення унікальності об'єкта, конструктори і оператор присвоювання оголошуються закритими. Цей патерн буде використаний для класів із конфігурацією [11].

1.2.7 Вибір апаратної частини

Для апаратної частини потрібен мікрокомп’ютер, що поєднував би модуль вимірювання температури, а також камеру для ідентифікації.

Raspberry Pi – одноплатний комп'ютер, розроблений британським фондом Raspberry Pi Foundation. Головне призначення – сприяти вивченню базових комп'ютерних навичок школярами.

Raspberry Pi побудований на системі-на-чипі (SoC) Broadcom BCM2835, яка включає в себе процесор ARM із тактовою частотою 700 МГц, графічний процесор VideoCore IV, і 512 чи 256 мегабайтів [12] оперативної пам'яті. Твердий диск відсутній, натомість використовується SD карта. Така

апаратна начинка дозволяє відтворювати відео формату H.264 в роздільній здатності 1080p, і запускати комп'ютерні ігри на зразок Quake III Arena [14, 14].

Оперативна пам'ять сягає від 256 МБ до 4 ГБ.

Початкова ціна є від 25$.

Оскільки програма має стати MVP (minimum viable product) – мінімально розробленим продуктом, що дозволить отримати зворотній відгук від користувачів, щоб визначити, що додати або прибрати з функціоналу , а також модулі камери та температури можуть бути з легкістю замінені, то для ідентифікації підійде звичайна вебкамери компанії Logitech.

Роздільна здатність відео – HD (1280x720)

Частота кадрів на секунду – 30

Інтерфейс підключення – USB 2.0

Щодо температури, то буде взято безконтактний термометр з наступною його модифікацією для підключення до Raspberry Pi.

Тип – інфрачервоний (безконтактний)

Зона вимірювання температури – чоло

Точність вимірювання – 0.2 °C

Діапазон вимірювань температури – 34-42.9 °C

Тривалість вимірювання температури – 2 сек

Тип елемента живлення

Особливості:

- можливість міняти шкалу вимірювання (Цельсій або Фаренгейт);
- вимірювання температури без контакту зі шкірою, щоб уникнути перехресного поширення інфекції.

1.3 Огляд можливості розпізнавання за обличчям

У системі планується задіяти нейронну мережу для того, щоб розпізнавати осіб за обличчям – це дозволить автоматизувати ідентифікацію та при-

швидшити цей процес. Одним з популярних методів для використання нейронних мереж є transfer learning (трансферне навчання).

Трансферне навчання – це метод машинного навчання, коли модель, розроблена для завдання, використовується повторно як відправна точка для моделі іншого завдання.

Це популярний підхід у глибокому навчанні, де попередньо навчені моделі використовуються як відправна точка для завдань комп'ютерного зору та обробки природної мови, також вони враховують необхідні величезні обчислювальні та часові ресурси, необхідні для розробки моделей нейронних мереж щодо вирішення певних завдань.

Треба попередньо розглянути, який алгоритм задіяно для розпізнавання зображень.

Зазвичай, для розпізнавання об'єктів використовують згорткові нейронні мережі.

Згорткова нейронна мережа (CNN) – це алгоритм глибокого навчання, який може приймати вхідне зображення, призначати важливість (зважувані ваги) різним аспектам або об'єктам на зображенні та мати можливість диференціювати один від іншого. Попередня обробка, необхідна в CNN, набагато нижча порівняно з іншими алгоритмами класифікації. Хоча в примітивних методах фільтри розробляються вручну, при достатній підготовці, CNN має можливість вивчати ці фільтри або характеристики [15].

Архітектура CNN аналогічна структурі зв'язку нейронів в мозку людини і прототипом є організація зорової кори. Окремі нейрони реагують на подразники лише в обмеженій області зорового поля, відомому як рецептивне поле. Колекція таких полів перекривається, щоб охопити всю зорову зону.

CNN здатна успішно фіксувати просторові та тимчасові залежності в зображенні за допомогою застосування відповідних фільтрів. Архітектура краще підходить до набору даних зображення завдяки зменшенню кількості задіяних параметрів та повторному використанню ваг. Іншими словами, мережу можна навчити краще розуміти витонченість зображення.

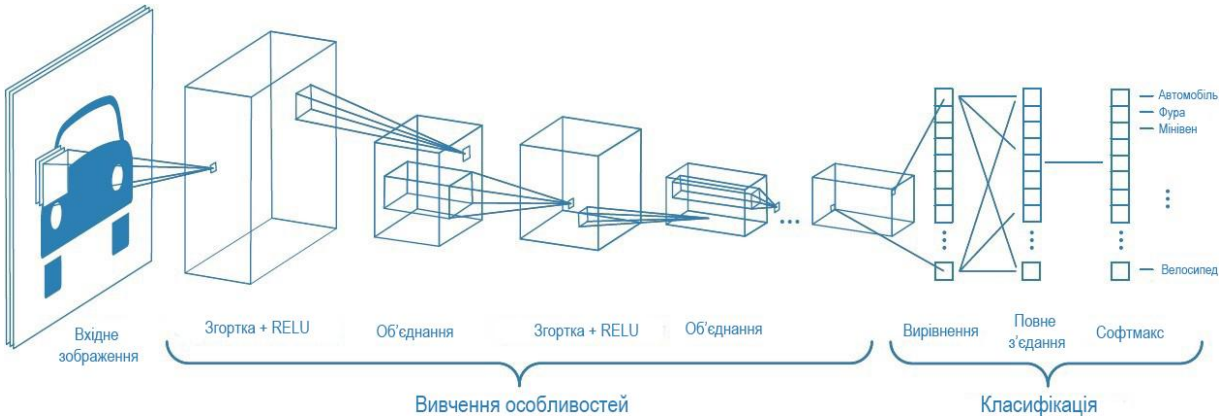

Рисунок 1.6 – Архітектура CNN

Планується знайти підготовлену згорткову мережу для розпізнавання людей.

Додавання нового робітника планується наступним чином: фотографія людини при заповненні формуляра буде зберігатися у базі, нейронна мережа буде після цього перевчатися, щоб це ураховувалося при ідентифікація.

Ідентифікація проходитиме наступним чином: підключена камера буде фіксувати обличчя людини, звертатися до бази із фотографіями, після цього нейронна мережа визначатиме, хто прийшов, а потім мікрокомп'ютер створюватиме запит для фіксації приходу у системі.

Для роботи із мережею буде використовуватися бібліотека Tensorflow — відкрита програмна бібліотека для машинного навчання цілій низці задач, розроблена компанією Google для задоволення її потреб у системах, здатних будувати та тренувати нейронні мережі для виявлення та розшифровування образів та кореляцій, аналогічно до навчання й розуміння, які застосовують люди.

Після аналізу існуючих аналогів було розглянуто три системи, що надають можливість ідентифікувати людину за біометрикою та пришвидшити процес контролю людей на підприємстві. Було визначено спільний функціо-

нал та проаналізовано переваги та недоліки для створення власного бачення ідеальної системи.

Було розглянуто та обрано технології для створення продукту згідно з сучасними тенденціями, а також беручи до уваги важливість найменшої ціни при розробці.

Система буде складатися з трьох частин: вебсервер, фронтенд частина та апаратна частина з камерою за модифікованою версію термометру.

Сервер буде написано на Java за допомогою фреймворку Spring, фронтенд частину буде розроблено за допомогою Angular. Щодо апаратної частини, буде використано мікрокомп’ютер Raspberry Pi для створення запитів та використання нейромережі. Для зберігання даних буде використано СУБД MySQL.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Мета та задачі ІС

Біометрична інформаційна система, що розробляється – це клієнт-серверне застосування, основною функцією якого є надання швидкої ідентифікації користувачів за обличчям із вимірюванням температури та записом у сховище даних.

Мета ІС: інформаційна система повинна надавати змогу створювати аккаунт нового співробітника з подальшою успішною ідентифікацією за допомогою розпізнавання його обличчя, а також зі збереженням даних про час приходу та температуру працівника. Додатково передбачана можливість перегляду статистики по окремих працівниках, наприклад, дати та час приходу на роботу за певний період часу.

Бачення продукту:

For: для підприємств та закладів з великої кількістю людей, що працюють або навчаються у закритому приміщенні, та часто відвідують своє робоче місце, наприклад, для шкіл, вишів, офісів.

WHO: продукт вирішує проблему швидкого запису прибуття людини на підприємство з часом та виміром температури для запобігання можливого контакту людини з ознакою респіраторного захворювання з іншими людьми.

THE: Продукт є спеціалізованим сервісом автоматизації. Аналогами продукта (UNLIKE) є термінал біометричний розпізнавання осіб ANVIZ Face Pass Pro Recognition System, ZKTECO LX50, R20-Face.

OUR PRODUCT: відрізняється від аналогів меншою ціною, можливістю регулювати доступ до приміщення, , можливістю вимірювати температуру та збором та створенням статистики щодо часу прибуття та покидання підприємства.

Опис вхідних та вихідних інформаційних потоків продукту (рис. 2.1).

Вхідні:

- ФІО робітників;
- дані щодо графіку роботи;
- типи робітників;
- час прибуття;
- час покидання;
- поточна температура.

Вихідні:

- повідпомлення про успішну або неуспішну авторизацію;
- запис у базі даних щодо прибуття, покидання підприємства, а також виміряної температури.

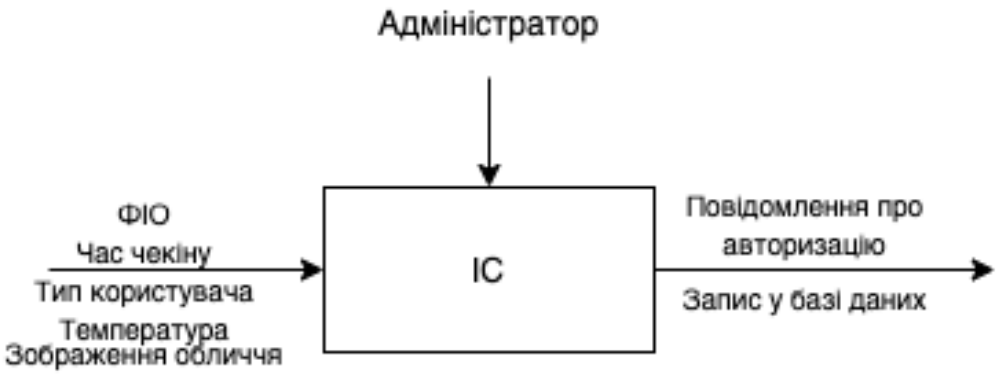

Рисунок 2.1 – Схема інформаційних потоків

Типи користувачів:

- Звичайний робітник – робітник або студент чи школяр, що має авторизуватися, щоб зайти до приміщення, за допомогою розпізнавання за обличчям та провівши замір. Низький рівень відповідальності;
- Адміністратор – людина, що може змінювати статус робітника, його графік відвідування, може змінювати особисту інформацію, отримує статистику щодо відвідування. Високий рівень відповідальності.

- Робітник охорони – може вручну зробити запис прибуття людини. Низький рівень відповідальності.

2.2 Функціональні вимоги

Ідентифікація

- FR 1.1 Система має проводити авторизацію за обличчям;
- FR 1.2 Система має замірювати температуру людини;
- FR 1.3 Система має надавати змогу відхиляти доступ до приміщення;
- FR 1.4 Система має звуком та кольором лампочки сповіщати щодо успішної або неуспішної авторизації;
- FR 1.5 Система має робити запис у базу даних щодо успішної авторизації.

Авторизація

- FR 2.1 Система має давати доступ після успішної аутентифікації для адміністратора або охоронця.

Ручний запис

- FR 3.1 Система має надавати змогу вручну зробити запис про прибуття людини.

Перегляд статистики

- FR 4.2 Система має надавати змогу дивитися відвідування людиною підприємства;
- FR 4.3 Система має надавати змогу дивитися у браузері чек-ін працівників.

Завантаження та редагування робітника

- FR 5.1 Система має давати змогу завантажувати нову людину для авторизації;
- FR 5.2 Система має надавати змогу редагувати інформацію відвідувачів;

– FR 5.3 Система має надавати змогу змінювати графік часу прибуття відвідувачів.

Рисунок 2.2 – Діаграма прецедентів

2.3 Нефункціональні вимоги

Попри функціональні вимоги, до системи висуваються нефункціональні вимоги щодо безпеки, надійності тощо. Це описано у нижченаведеній таблиці (табл. 2.3).

2.4 Ідентифікація архетипу ІС

Інформаційна система буде мати клієнт-серверну архітектуру. Клієнтська частина являє собою вебзастосунок, розроблений за допомогою фреймворку Angular. Серверна частина буде розроблена завдяки технології Spring

Framework. Оскільки система буде складатися з двох частин – бекенду та фронтенду та буде використовуватися ресурси клієнта, то можна зробити висновок, що дана система відноситься до архетипу Rich WEB Application (RWA).

Таблиця 2.3 – Нефункціональні вимоги системи

Унікальний ідентифікатор	Вимога	Опис
01	Надійність	Система має безвідмовно працювати при стабільному живленні та доступу до інтернету
02	Безпека	Система має зберігати дані такими як захищені та недоступними для сторонніх людей
03	Багатоплатформованість	Система має не залежати від ОС
04	Швидкість	Система має працювати швидко, без довгої обробки даних
05	Простота	Система має мати юзер-дружній інтерфейс

Усі вимоги є атомарними, упорядкованими за категоріями, верифікованими, зрозумілими і ясними, повними та несуперечать одна іншій.

2.5 Користувацький інтерфейс (UI View)

Щоб розробити фронтенд частину, небхідно мати попередній вигляд усіх вікон для системи. Мокапи було виконано у напівбезкоштовому застосунку Figma.

Усього 6 вікон (табл. 2.3): авторизація, головне вікно, вікно ручного додавання запису, вікно додавання нового робітника, вікно із статистикою та вікно для редагування.

Таблиця 2.3 – Вікна та ідентифікатори

Ідентифікатор вікна	Назва вікна
1	Вікно авторизації
2	Вікно записів (головне)
3	Вікно додавання нового співробітника
4	Вікно ручного запису прибуття людини
5	Вікно статистики
6	Вікно редагування інформації людини

Початкове вікно для нашого застосунку представлено на рис. 2.3. Тут користувач має ввести свої логін та пароль, щоб авторизуватися та отримати повну змогу користуватися програмою.

Головне вікно за стосунку наведено на рис. 2.4. Тут адміністратор має змогу продивитися час, дату, прізвище та ім’я людини, що прийшла.

Якщо людина має виміряну високу температуру, то система підсвітить цю людину червоним кольором.

На цьому вікні можна перейти на вкладку з додаванням нової людини, до вікна з ручним записом приходу людини, до вікна зі статистикою та до вікна з редагуванням.

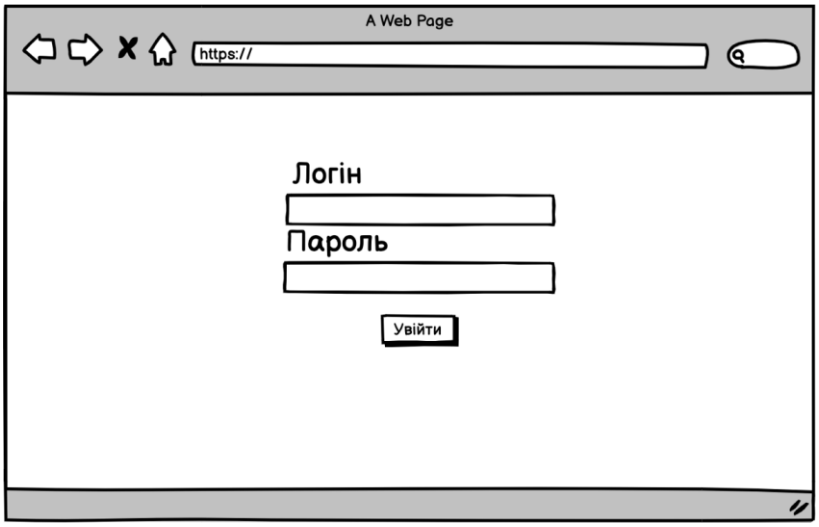

Рисунок 2.3 — Вікно авторизації

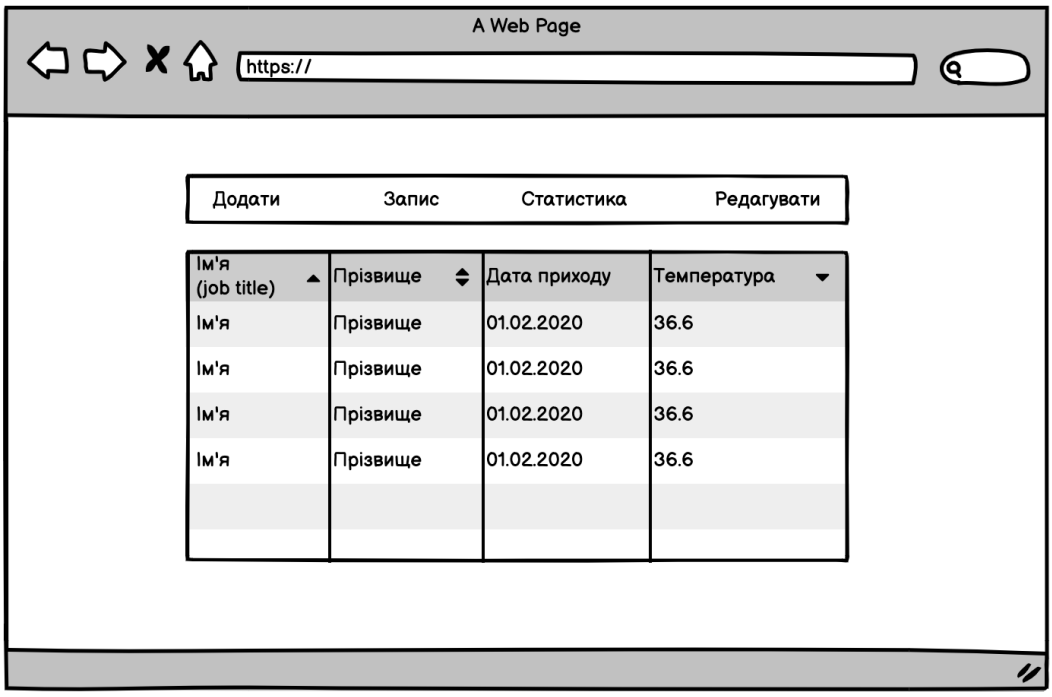

Рисунок 2.4 — Вікно головної сторінки

У вікні, що зображене на рис. 2.5 адміністратор може додати нового співробітника. Для цього вводиться ФІО, його посада та час для перевірка вчасного приходу. Також необхідно додати його фото.

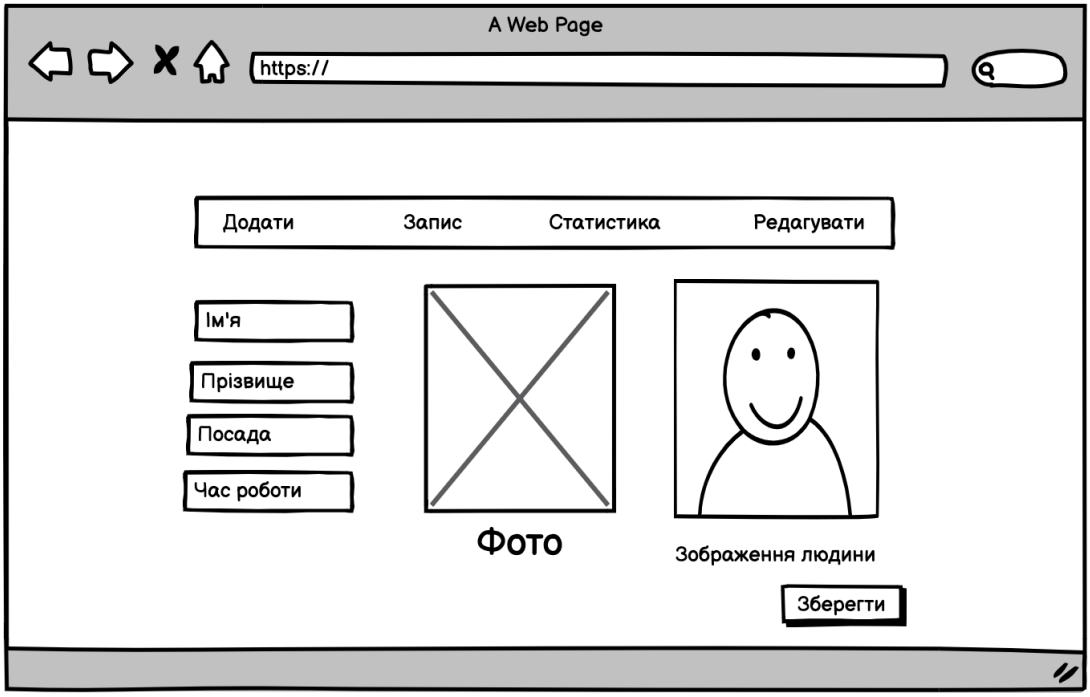

Рисунок 2.5 — Вікно сторінки з додаванням нового співробітника

Вікно на рис.2.6. необхідно для працівника охорони. Якщо система не може розпізнати людину або людина не є співробітником, то це запис людини можна зробити вручну.

На рис. 2.7 наведено вікно адміністратора. В цьому вікні система автоматично робить аналітику щодо прибуття працівників на роботу, а також про їх середню температуру. Це дозволить робити висновки щодо захорюванності та дисципліни в колективі. Також можна продивитися аналітику за певний час.

Вікно на рис.2.8. необхідне для редагування інформації щодо працівника. Наприклад, щодо зміни його персональних даних або зміни посади та графіку роботи, а також там можна оновити фото.

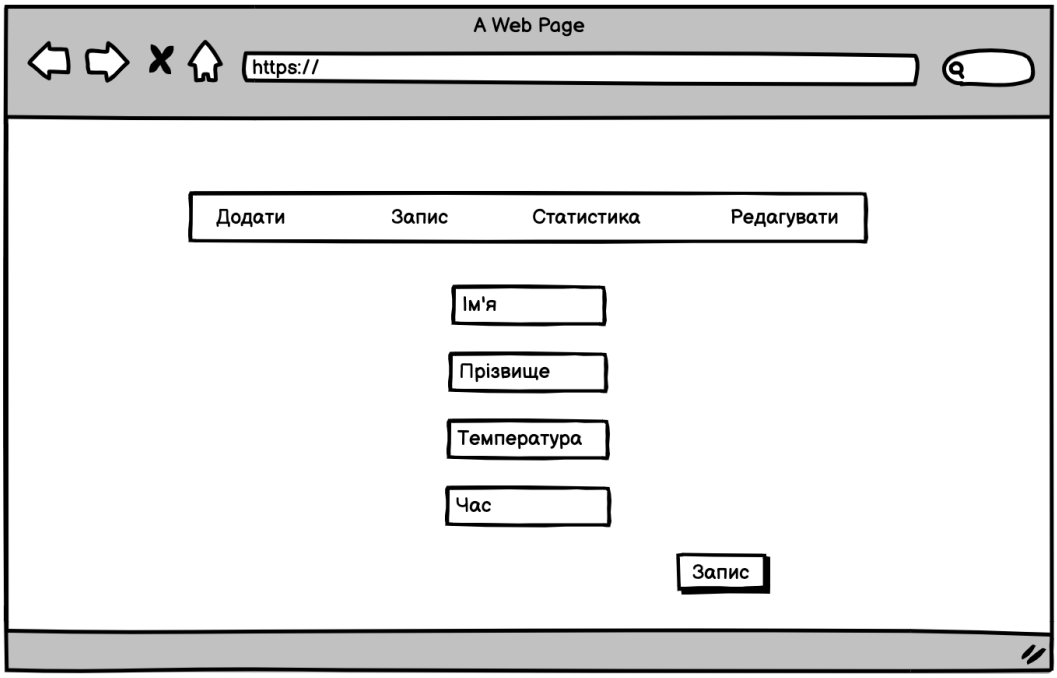

Рисунок 2.6 — Вікно ручного запису прибуття людини

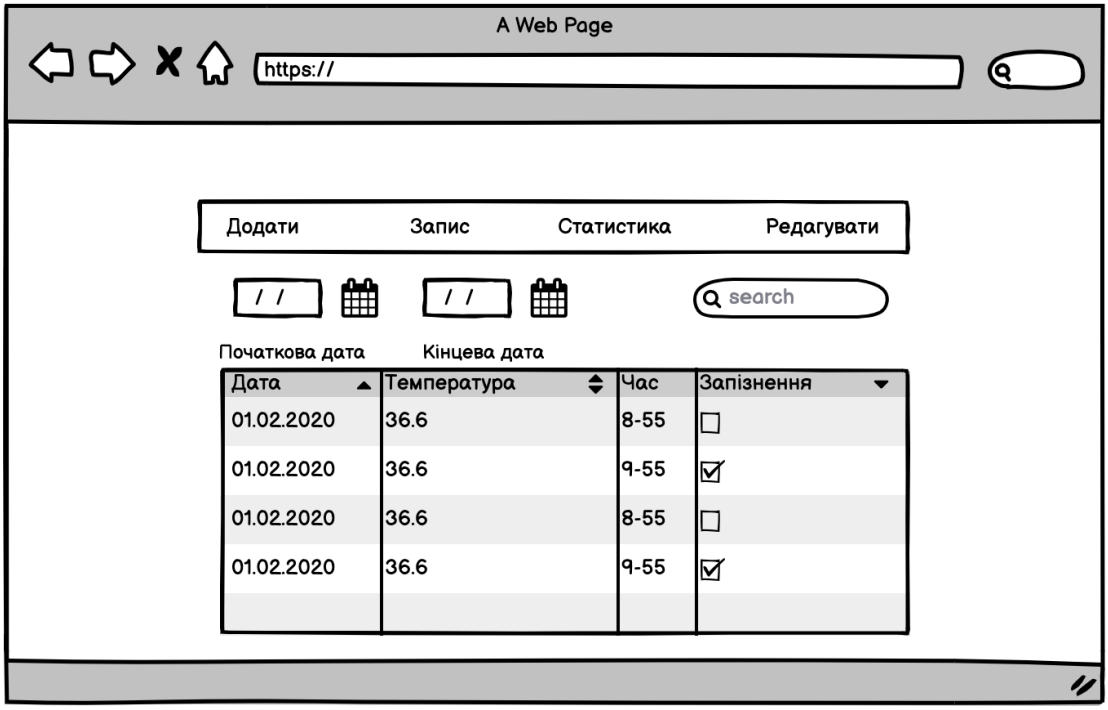

Рисунок 2.7 — Вікно статистики

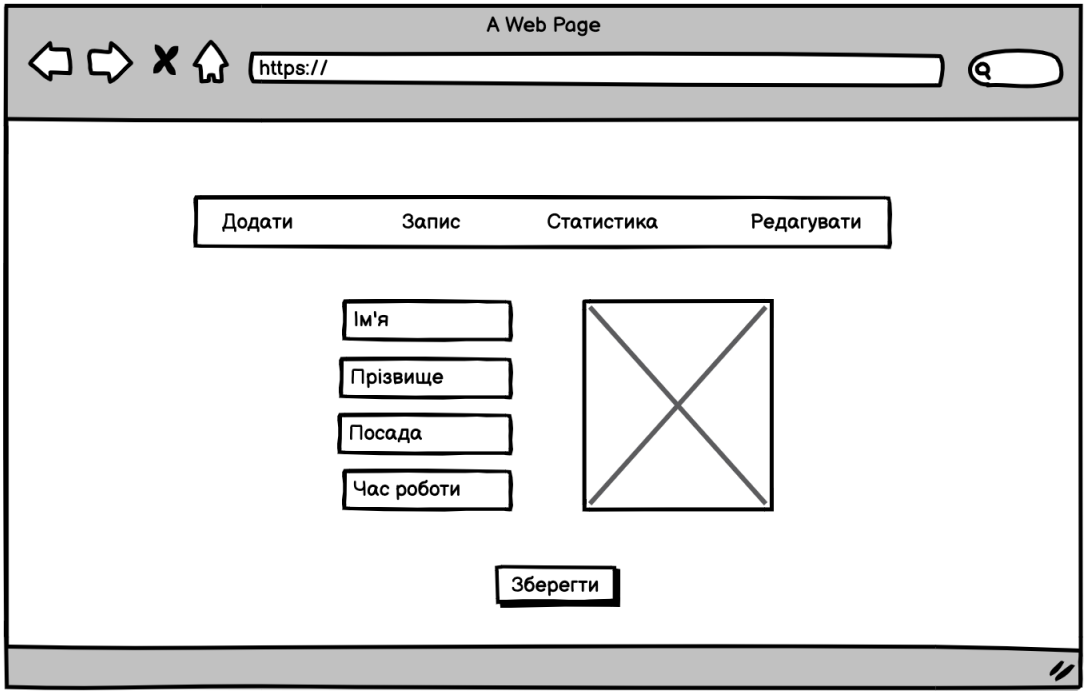

Рисунок 2.8 — Вікно редагування інформації людини

Усі переході між вікнами можна зобразити у діаграмі переходів стану (рис. 2.10).

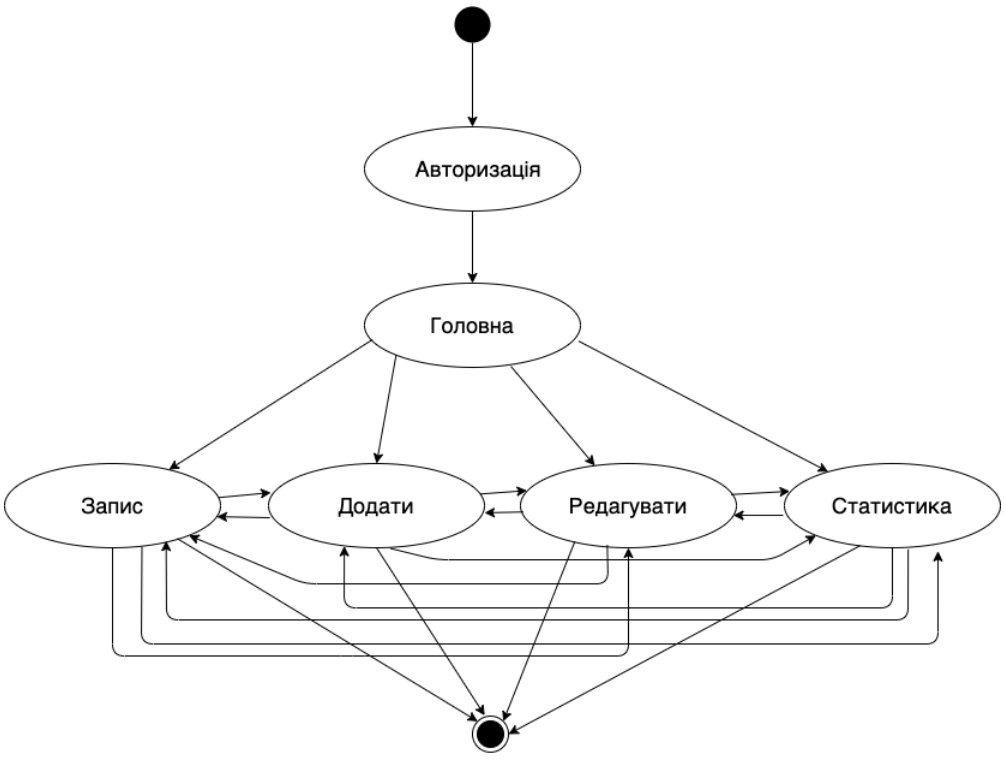

Рисунок 2.10 — Діаграма переходів станів

2.6 Логічне уявлення про ІС (Logical View)

Для того, щоб уявити майбутню систему, необхідно побудувати логічну діаграму (рис.2.11), що буде показувати зв’язок компоненті. Система буде складатися з адміністратора, що матиме змогу додавати нових співробітників, редагувати інформацію, а також дивитися статистику щодо відвідуваності людей.

Модель складається з Angular-застосунку, що надсилає запити серверу, а він взаємодіє із базою даних.

Модулю із відеокамерою та модулю вимірювання температури, що під’єднані до Raspberry. Він розпізнає людину, а потім із виміряною температурою відправляє запит на сервер для запису приходу людини.

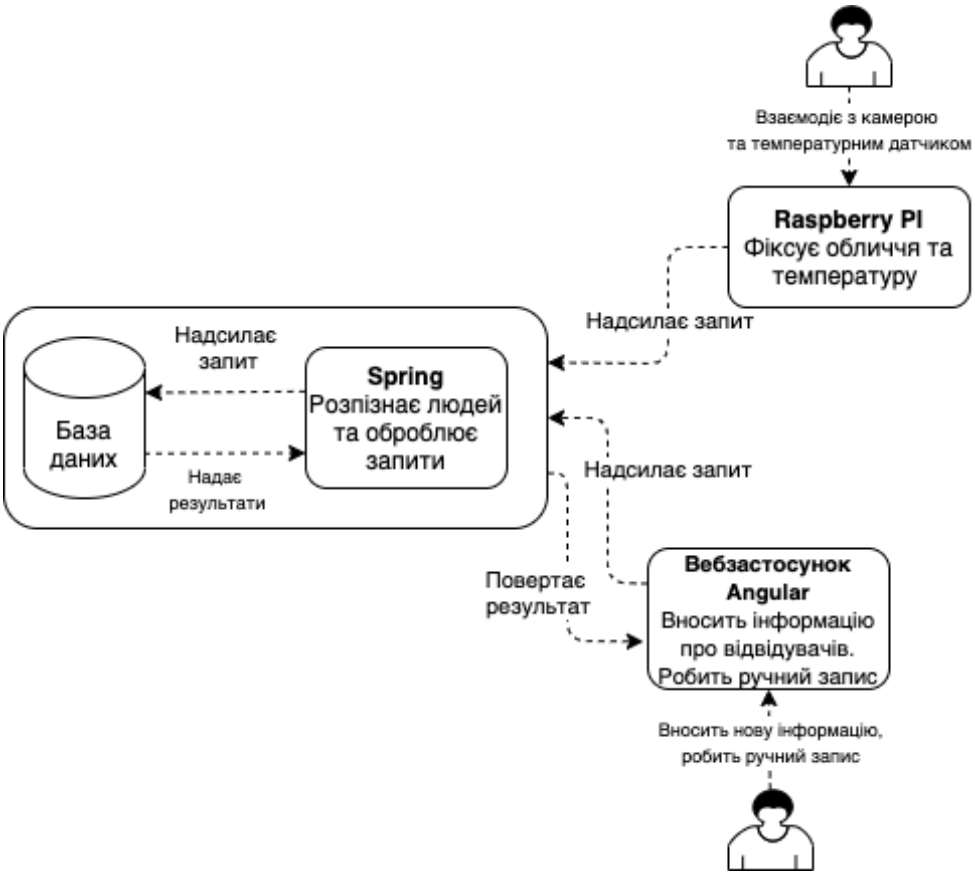

Рисунок 2.11 — Діаграма логічного уявлення ІС

2.7 Уявлення розгортання ІС (Deployment View)

Клієнтський застосунок, а також серверна частина існують завдяки передачі запитів по Інтернету. Ця частина буде хостуватися на Azure.

Сервер буде обробляти усі запити, що будуть надходити до нього. За архітектурний стиль узято REST. Усі виклики методів будуть прописані маршутами (рис. 2.12).

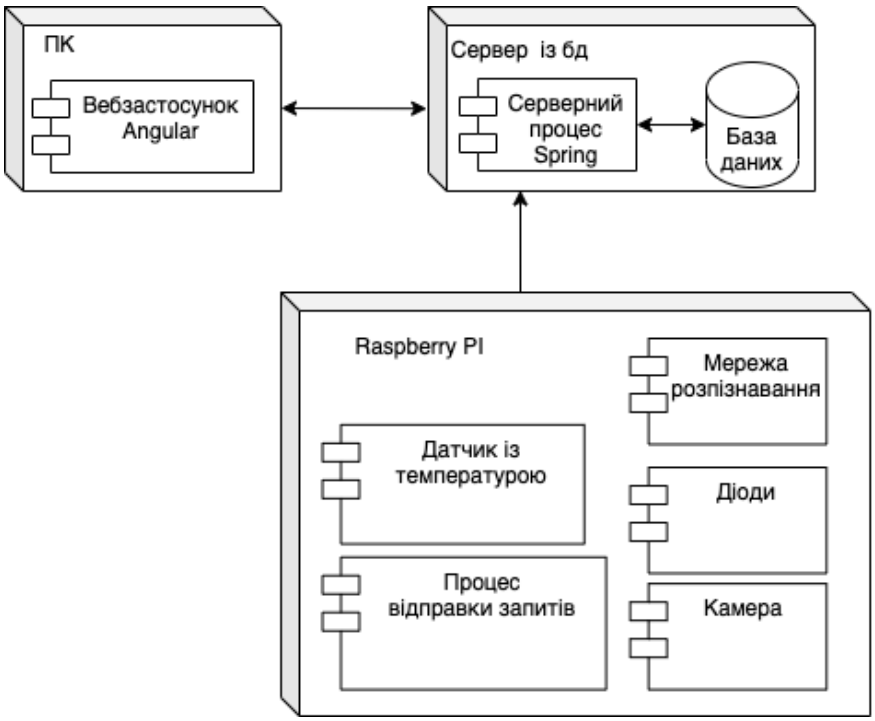

Рисунок 2.12 — Діаграма розгортання ІС

Сервер буде формувати відповіді та у форматі JSON відправляти відповідь на вебчастину.

2.8 Уявлення слоїв ІС (Design View)

Серверна частин матиме наступні слої:

- Domain layer – цей слой потрібен для розпарсування даних у класах та наступною роботою з ними як з об’єктами. Також цей слой надає змогу працювати із даними як комунікацією з базою даних.
- Application layer – цей слой передбачає собою увесь функціонал застосунку, як-от авторизацію, додавання та редагування робітників, отримання статистики. Цей слой напрямує залежить від
- Communication layer – на цьому слої відбувається обробка запитів, що виконуються REST Api рівнем.

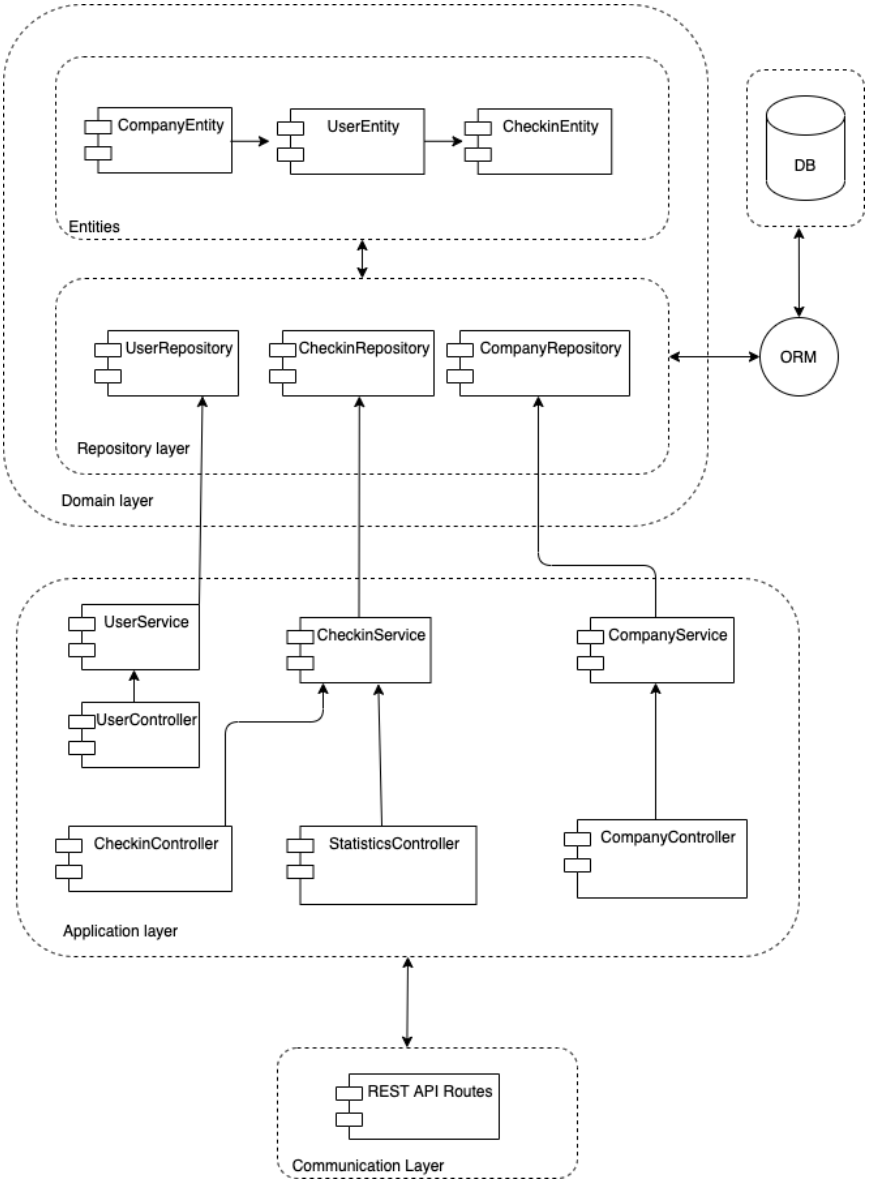

Рисунок 2.13 — Діаграма слоїв ІС серверу

Вебчастина матиме наступні слої:

- View – цей слой має складатися із html для представлення та вводу даних.
- CommandAngular – цей слой має роутинг для переключення необхідних компонентів ангуляра та виклику запитів до серверу.
- Services – на цьому слої відбувається обробка даних або певні функції для створення даних.
- Model – слой потрібен для парсування даних у об’єкти.

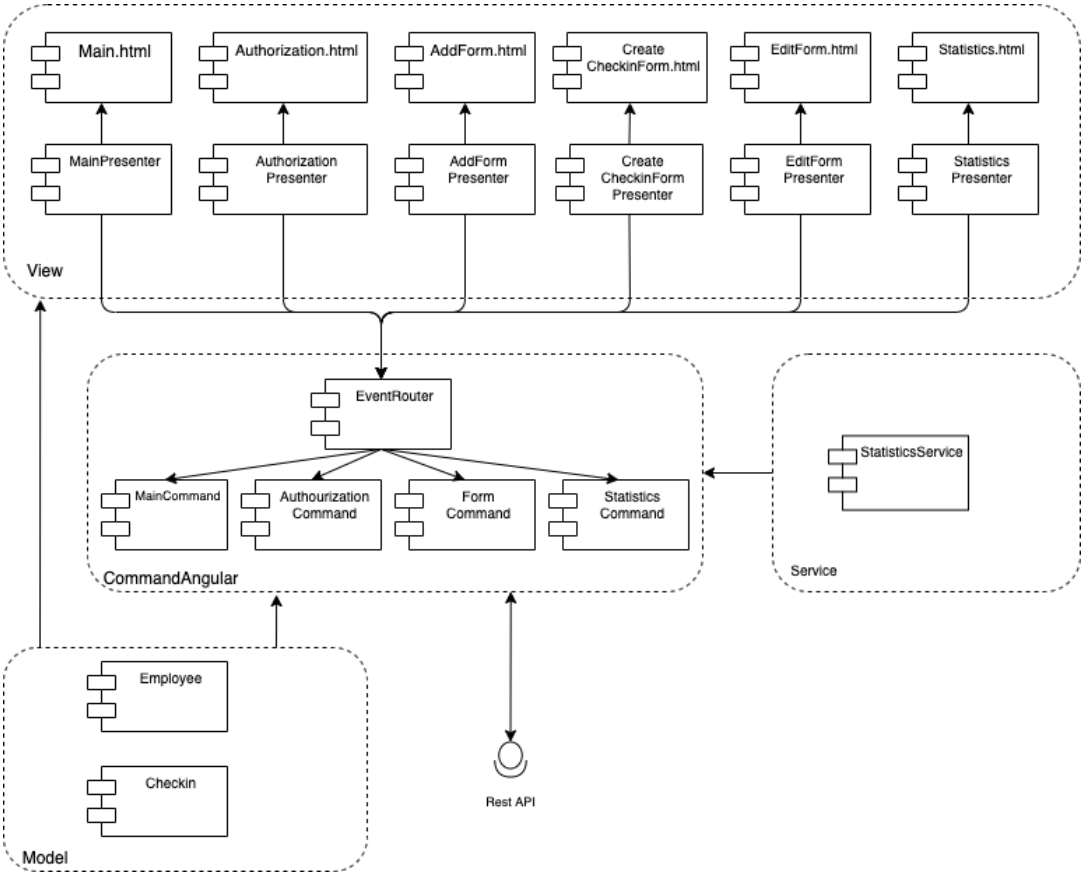

Рисунок 2.14 — Діаграма слоїв ІС вебчастини

Программне забезпечення апаратної частини матиме наступні слої:

- Network – цей слой матиме нейронну мережу, що буде фіксувати та розпізнавати обличчя.

- Manager – цей слой буде керувати викликами функцій, управляти підключеними модулям.
- Model – слой потрібен для парсування даних у об’єкти.

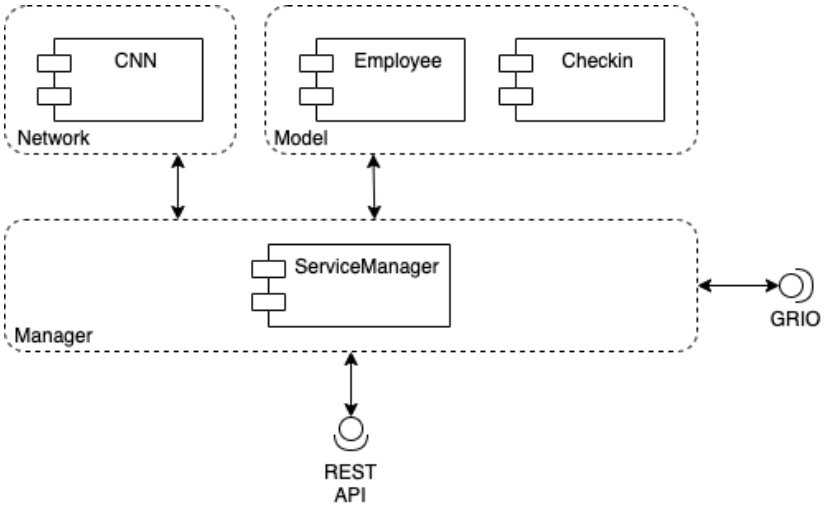

Рисунок 2.15 — Діаграма слоїв ІС апаратної частини

2.9 Уявлення процесів ІС (Process View)

Оскільки основною ціллю системи є ідентифікація робітників, слід виділити два процеси: ідентифікація за запис приходу людини, а також отримання інформації по відвідуваності.

Сценарій 1. Проведення авторизації за обличчям людини.

Робітник приходить на роботу та підходить до апаратної частини. Камера має знайти зображення людини, зафіксувати обличчя, зробити замір температури. Після має буде розпізнано, чи є така людина у бази.

Якщо людина є у базі, то сервер записує інформацію про прихід, а апаратна частину повідомляє про успішну ідентифікацію.

Якщо людину не знайдено, то система повідомляє про неуспішну ідентифікацію.

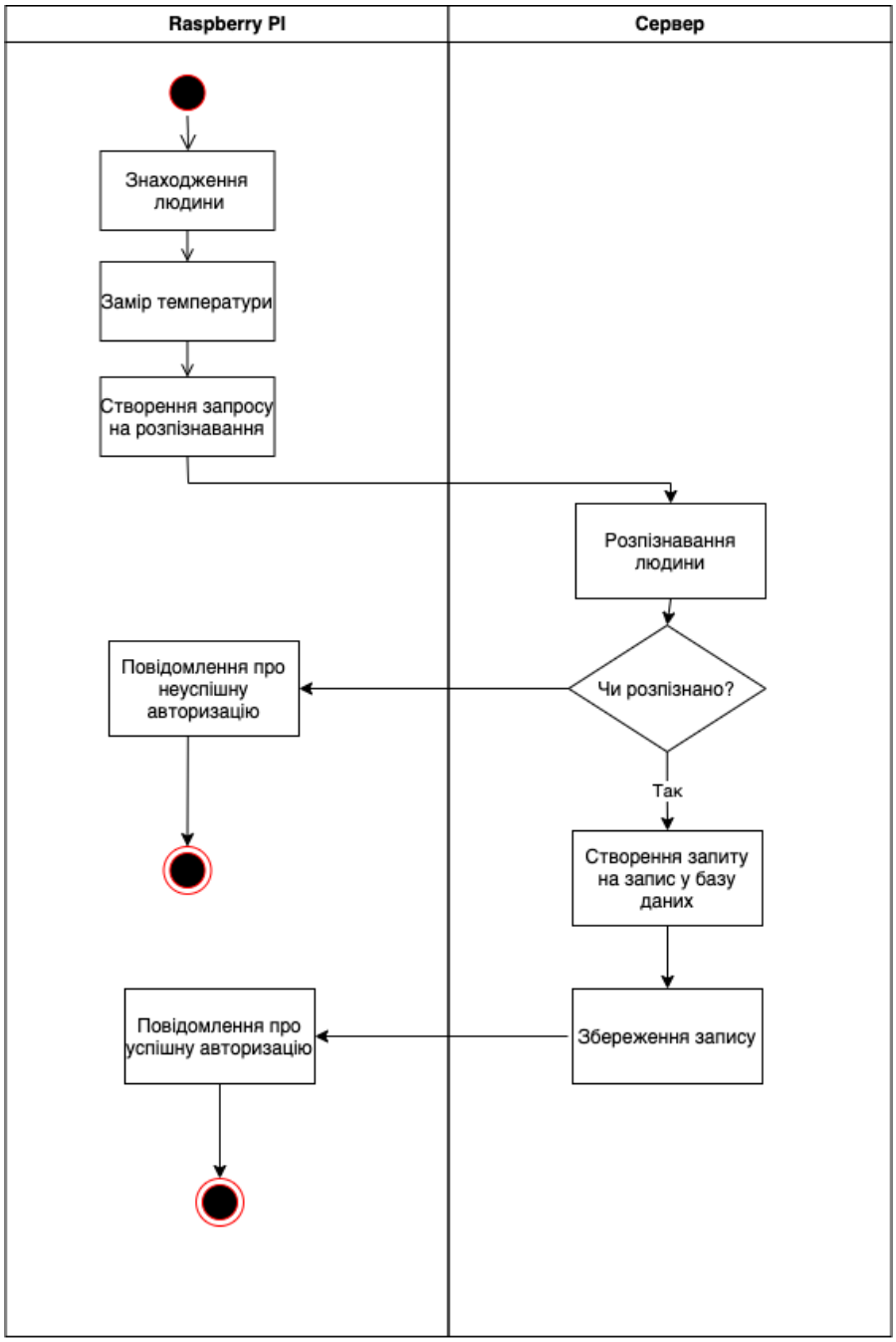

Рисунок 2.14 — Діаграма активності для випадку ідентифікації людини

Сценарій 2. Отримання даних про відвідуваність

Адміністратор має змогу подивитися статистику по людині.

Для цього він має ввести інтервал по часу, а також людину для перевірки. Після цього система відображає шукану інформацію.

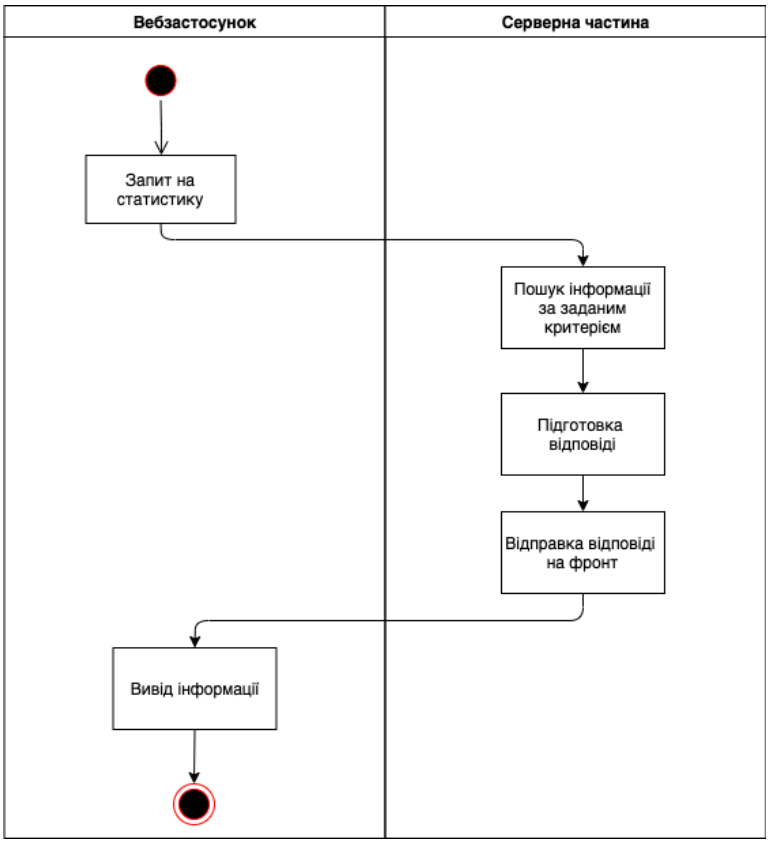

Рисунок 2.15 — Діаграма активності для випадку отримання статистики по відвідуваності

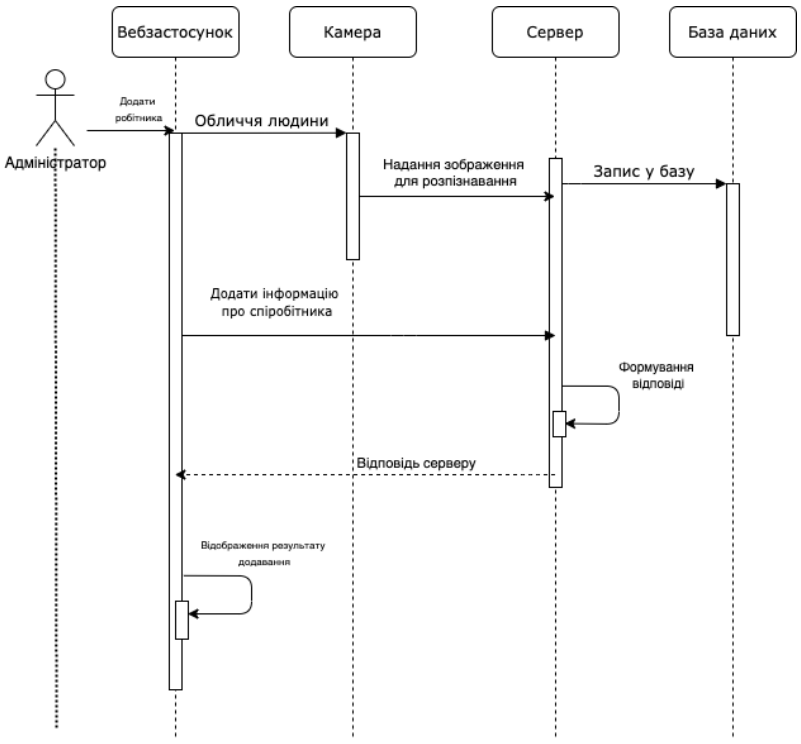

Рисунок 2.16 — Діаграма процесів для додавання нового співробітника

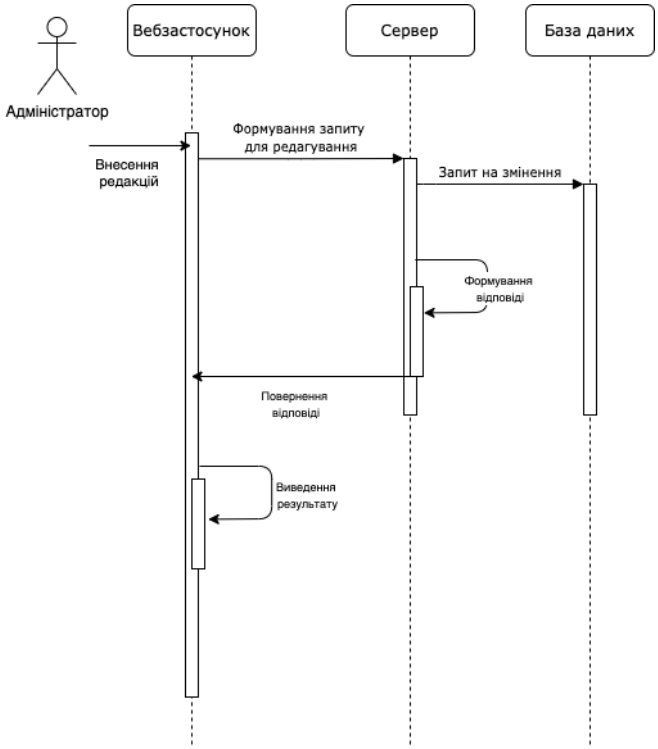

Рисунок 2.17 — Діаграма процесів для редагування інформації про співробітника

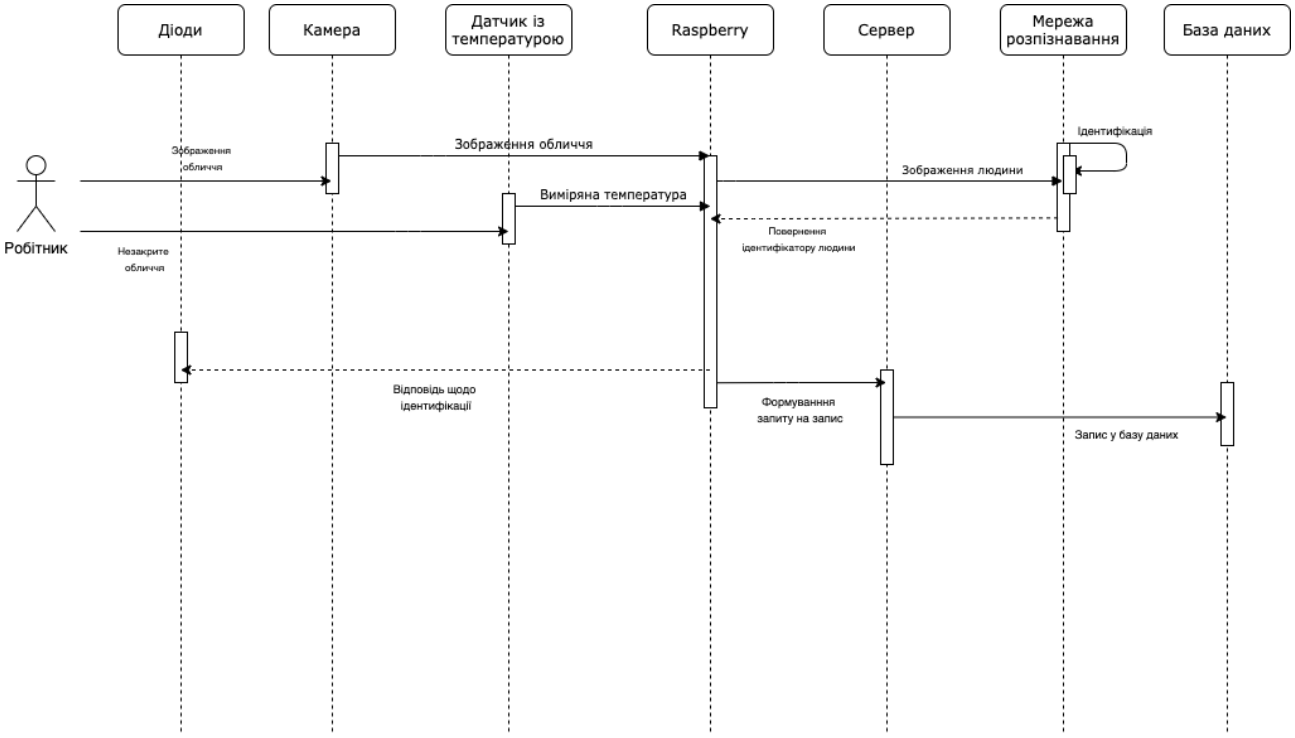

Рисунок 2.18 — Діаграма процесів для ідентифікації людини

2.10 Уявлення даних IC (Data View)

Дані мають зберігатися у базі даних. Для цього у базі будуть наступні наступні таблиці (рис.2.19):

- Company – компанія, що користується продуктом;
- User – співробітник, що ідентифікується та записується у системі. Для уникнення колізій із іменами має унікальний ідентифікатор;
- Checkin – прихід людини, що зберігає час та температуру людини.

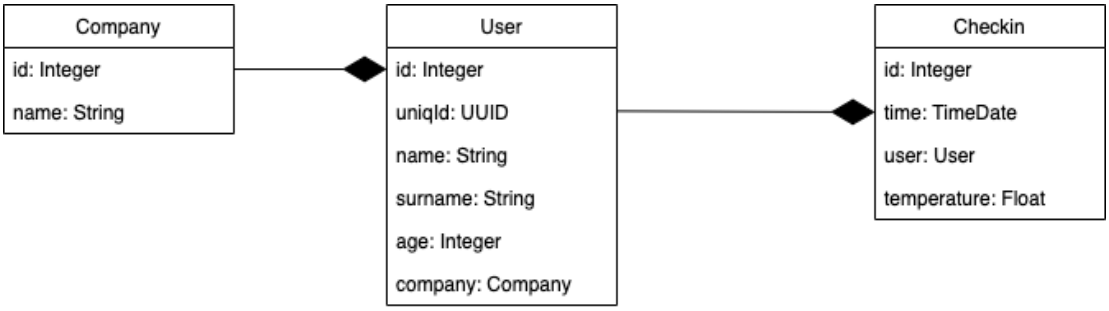

Рисунок 2.19 — Схема реляційної бази даних

Таким чином, другому розділі було сформовано функціональні та нефункціональні вимогу до інформаційної системи, що розробляється. Було змодельовано діаграму прецедентів, визначено, які будуть типи користувачів, їх функції, Визначено архетип продукту. Також було розроблено мокапи майбутнього застосунку з приблизним функціоналом, що буде розроблено. Усю роботу продукту було описано у діаграмах – логічному уявленні, уявленні розгортання, слоїв, процесів, даних. Роблено реляційну модель бази даних.

Отже, було пройдено етап підготовки до реалізації самої системи.

3. РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Навчання та тестування нейронної мережі

Для виконання проекту було знайдено переднавчену нейромережу ResNet, яка має 29 згорткових шарів. Як заявлено в описі, то модель має точність 99,38% за стандартом тесту Labeled Faces in the Wild. Ця модель є мережею. Автор пише, що мережа навчалася з нуля на наборі даних, що складає близько 3 мільйони облич. Цей набір даних походить із декількох датасів та власних знайдених фото автором.

Почнемо розбиратися із мережею. Оскільки вона вже переднавчена, то з кожним нашим новим зображенням, буде проводитися навчання та змінюватися ваги. Для проведення експерименту буде використано Deepnote – аналог Colab.

Треба завантажити наступне фото, щоб мережа перевчилась розпізнавати мене.

Рисунок 3.1 – Зображення для навчання

Зображення переводиться у масив чисел (рис. 3.2).

На жаль, не можна продивитися процес навчання по епохам та визначити точність, з якою мережа розпізнає. Проте, мережа дійсно непогано на-

вчена, адже точно розпізнає обличчя навіть у масці або з викривленими емоціями.

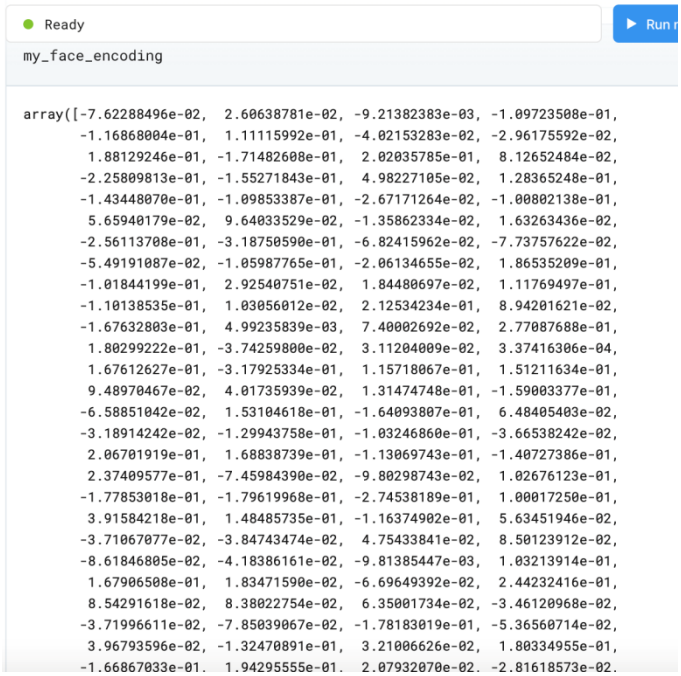

Рисунок 3.2 – Закодоване зображення

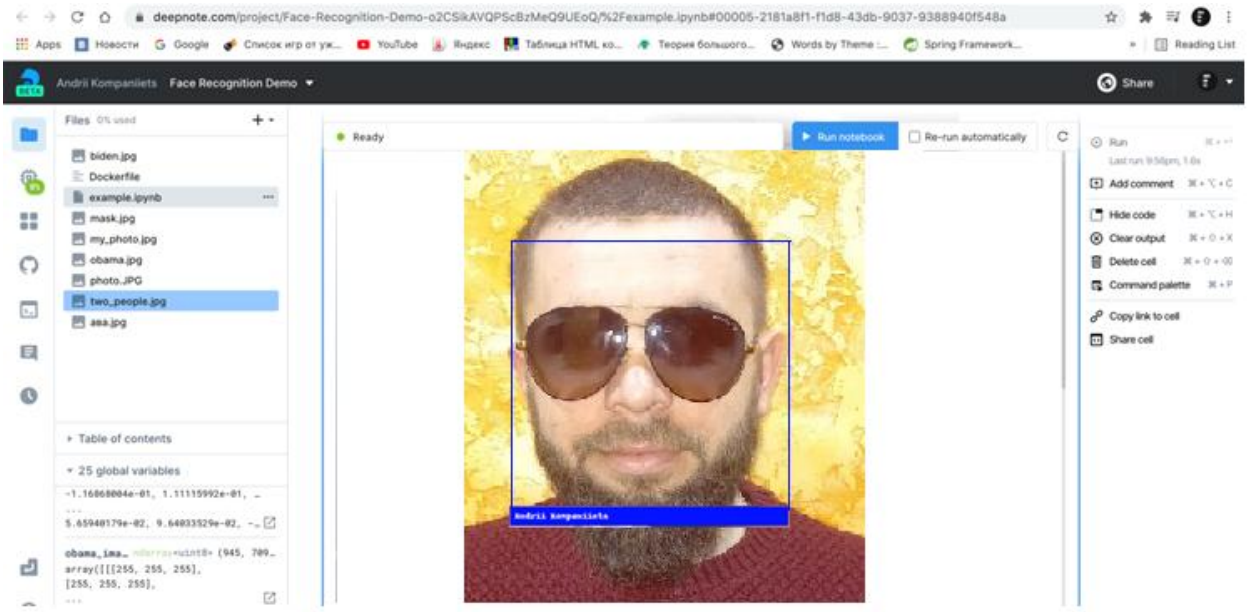

Рисунок 3.3 – Розпізнавання обличчя в окулярах по іншій фотографії

3.2 Уявлення про структуру проекту інформаційної системи

Структура проєкту представлено двома діаграмами для серверу та фронтчастини.

Структуру серверної часини представлено на рис. 3.4.

Рисунок 3.4 – Структура проєкту серверу

Структуру системи можна показати за допомогою представлення взаємодії пакетів бекенд частини (рис. 3.5). Усі взаємодіють між собою та мають у собі класи, які наведено вище.

Бекенд частину розподілено по пакетам, які уособлюють собою класи по певним задачам.

– Model – пакет, що має у собі класи для парсування об’єктів;

– Repository – пакет, що має у собі інтерфейси для роботи з базою даних, оскільки Spring Data JPA дозволяє працювати із реалізованим функціоналом у дефолтних інтерфейсах;
– Service – пакет, що має у собі класи для обробки даних або для реалізації певного функціоналу;
– Controller – пакет, що має класи для обробки;
– Configs – пакет, що містить класи-біни для конфігурації або налаштувань.

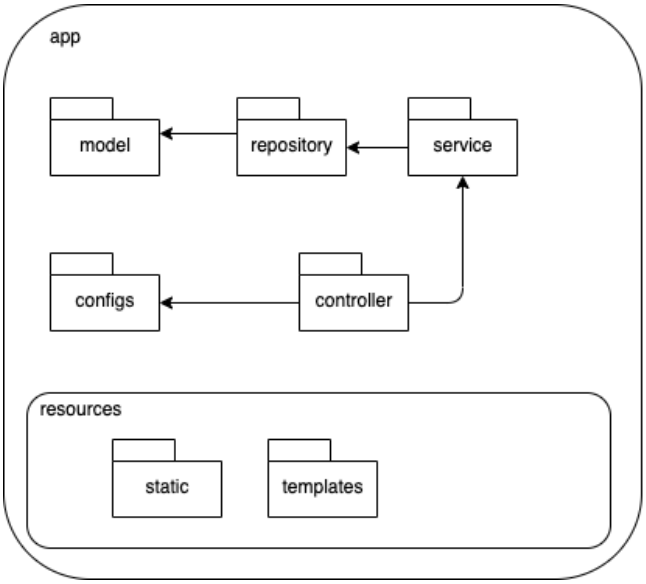

Рисунок 3.5 – Представлення взаємодії пакетів проєкту

Потрібно привести також структуру проєкту вебчастини (рис. 3.6).

Завдяки можливостям фреймворку Angular можна розбити програму на частини (компоненти) та почастково використовувати їх у фронт частини. Усього буде 1 модуль – admin, він матиме 5 компонентів із своїми представленнями і кодом-функціоналом. Також є пакет shared – він зберігатиме спільні класи для використання між компонентами, як-от: обробка помилок, інтер-

септор для роботи із авторизованими запитами. Нижче наведено пакети фронт частини (рис.3.7).

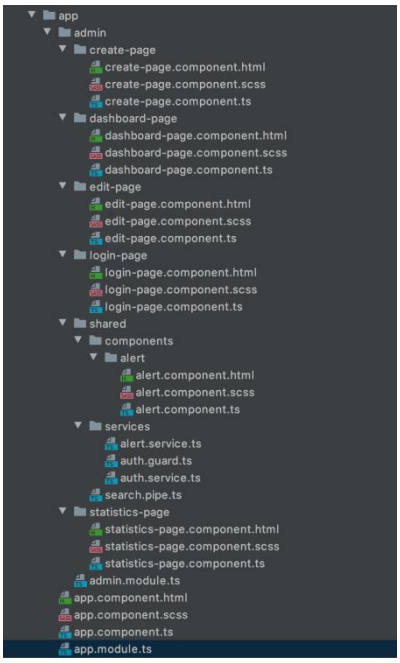

Рисунок 3.6 – Структура проєкту вебчастини

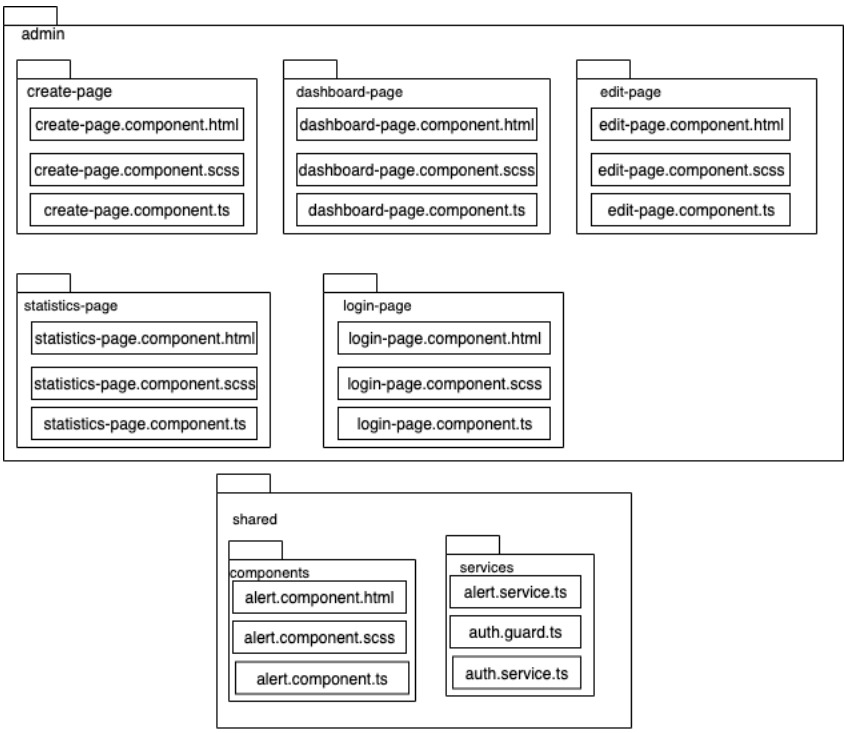

Рисунок 3.7 – Структура вебмодулю вебчастини

3.3 Уявлення про класи ІС

Оскільки система має бекенд- та фронтенд части, то слід навести діаграму класів роботи та їх взаємодію. Наведено наступні патерни:

– Singleton – клас-бін для конфігурації роботи із Azure та налаштуваннями;
– Dependency injection – вбудування об’єкту в інший, у програмі виконується як ін’єкція об’єкту репозиторія в об’єкт сервісу та об’єкт сервісу в контролер;
– MVC – реалізовано у фронт частині.

Розглянемо діаграму класів для бекенду (3.8).

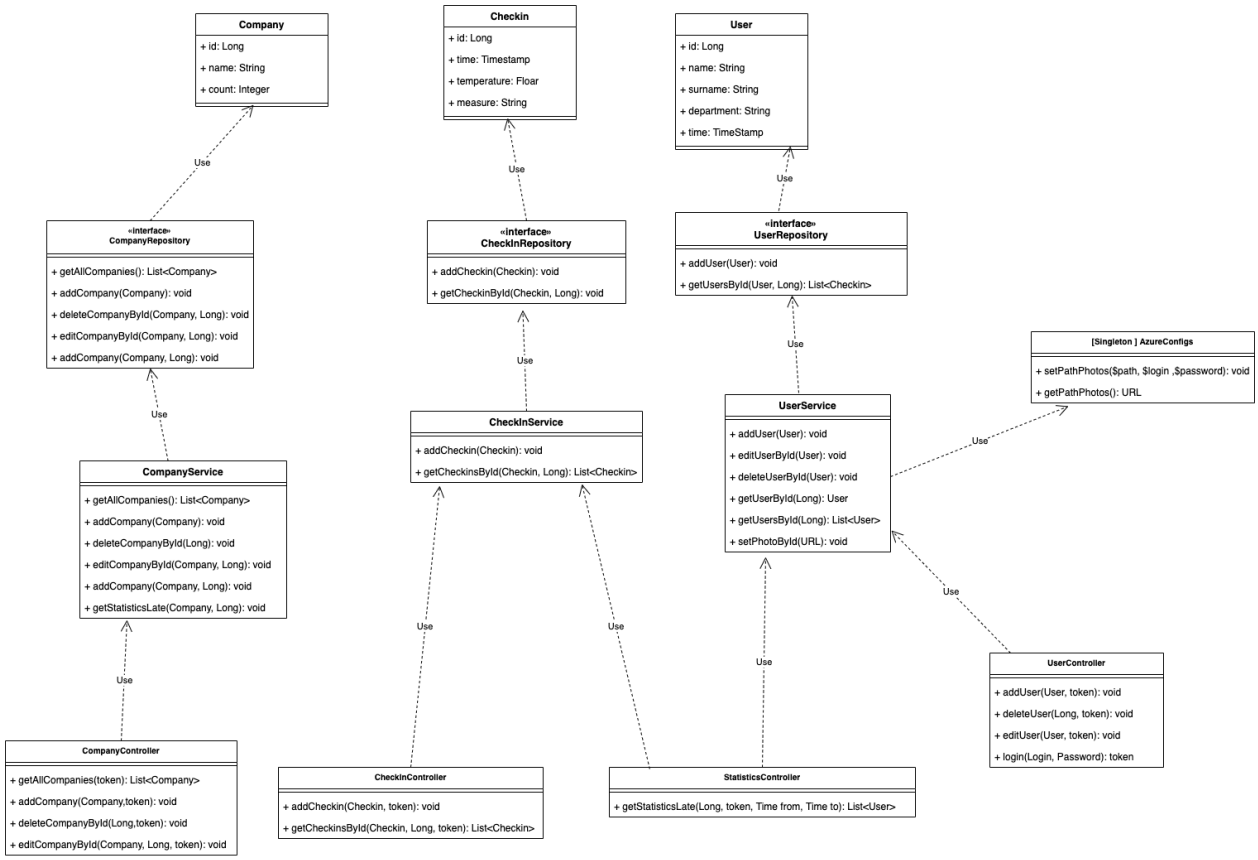

Рисунок 3.8 – Діаграма класів бекенду

Класи Company, Checkin, User – це моделі, для роботи з ними як із даними.

Інтерфейс CompanyRepository – інтерфейс-репозиторій, що потрібен для отримання або зберігання даних із бази даних. Його специфіка, як і усіх інших репозиторіях у Spring Data JPA у тому, що вони самі реалізують функціонал для роботи із даними:

- getAllCompanies(): List<Company> – метод, що дістає із бази даних усі компанії;
- addCompany(Company): void – метод, що додає нову компанію;
- deleteCompanyById(Company, Long): void – метод, що дозволяє видалити компанію за ідентифікатором;
- editCompanyById(Company, Long): void – метод, що обновлює дані компанії за ідентифікатором;
- addCompanyUser(Company, Long): void – метод, що додає нового робітника до компанії.
- Інтерфейс CheckinRepository – інтерфейс-репозиторій, потрібен для роботи із :
- addCheckin(Checkin): void – метод, що додає запис про прибуття людини;
- getCheckinsById(Long): void – метод, отримує усі записи по ідентифікатору людини.

Інтерфейс UserRepository – інтерфейс-репозиторій, потрібен для роботи із працівниками:

- addUser(User): void – метод, що додає нового співробітника;
- getUsersById(Long): List<Checkin> – метод, завдяки якому можна отр

Клас CompanyService – клас, який реалізує логіку для роботи із об’єктами компаній:

- getAllCompanies(): List<Company> – метод, надає усі наявні компанії;
- addCompany(Company): void – метод, що дозволяє додати компанію;

- deleteCompanyById(Long): void – метод, що видаляє компанію;
- editCompanyById(Company, Long): void – метод, що редагує існуючу компанію за ідентифікатором;
- getStatisticsLate(Company, Long): void – метод, дозволяє знайти статистику по запізненням у певній компанії по робітникам.
- Клас CheckInService – клас-сервіс, для роботи із записами про прибуття:
- addCheckin(Checkin): void – метод, що робить запис про прибуття людини;
- getCheckinsById(Checkin, Long): List<Checkin> – метод, що надає усі записи про прибуття по ідентифікатору робітника.
- Клас UserService – клас-сервіс для роботи із об’єктами робітників:
- addUser(User): void – метод, що зберігає робітника;
- editUserById(User): void – метод, що редагує дані робітника за ідентифікатором;
- deleteUserById(User): void – метод, що видаляє робітника за ідентифікатором;
- getUserById(Long): User – метод, що находить робітника за ідентифікатором;
- getUsersById(Long): List<User> – метод, що надає список робітників за ідентифікатором компанії;
- setPhotoById(URL, Long): void – метод, що змінює зображення людини за посиланням.

Клас CompanyController – контролер, що працює з REST-запитами, усі протребують попередньої авторизації, щоб отримати токен для аутентифікації:

- getAllCompanies(token): List<Company> – метод, що дає усі компанії списком;
- addCompany(Company,token): void – метод, що додає компанію;

– deleteCompanyById(Long,token): void – метод, що видаляє компанію за ідентифікатором;
– editCompanyById(Company, Long, token): void – метод, що редагує компанію за ідентифікатором;

Клас CheckInController – контролер для роботи із REST-запитами для записів прибуття:

– addCheckin(Checkin, token): void – метод, додає новий запис;
– getCheckinsById(Long, token): List<Checkin> – метод, отримає усіх записів про прибуття для людини.

Клас StatisticsController – контролер для роботи із REST-запитами із статистикою :

– getStatisticsLate(Long, token, Time from, Time to): List<User> – метод, що витягує дані по прибуттям людини за певний проміжок.

Клас UserController контролер для роботи із REST-запитами для робітників:

– addUser(User, token): void – метод, що додає нового робітника;
– deleteUser(Long, token): void – метод, що видаляє робітника за ідентифікатором;
– editUser(User, token): void – метод, що редагує робітника;
– login(Login, Password): token – метод, що потрібен для авторизації по логіну та паролю, та повертає токен.

Клас [Singleton] AzureConfigs – бін-сінглтон, зберігає лише налаштування для зберігання фото:

– setPathPhotos($path, $login ,$password): void – метод, що встановлює з’єднання для збергання фото;
– getPathPhotos(): URL – метод, що повертає шлях, де зберігаються фото.

Розглянемо каркаси класів для фронт частини для випадку виводу записів про прибуття людей (рис. 3.9).

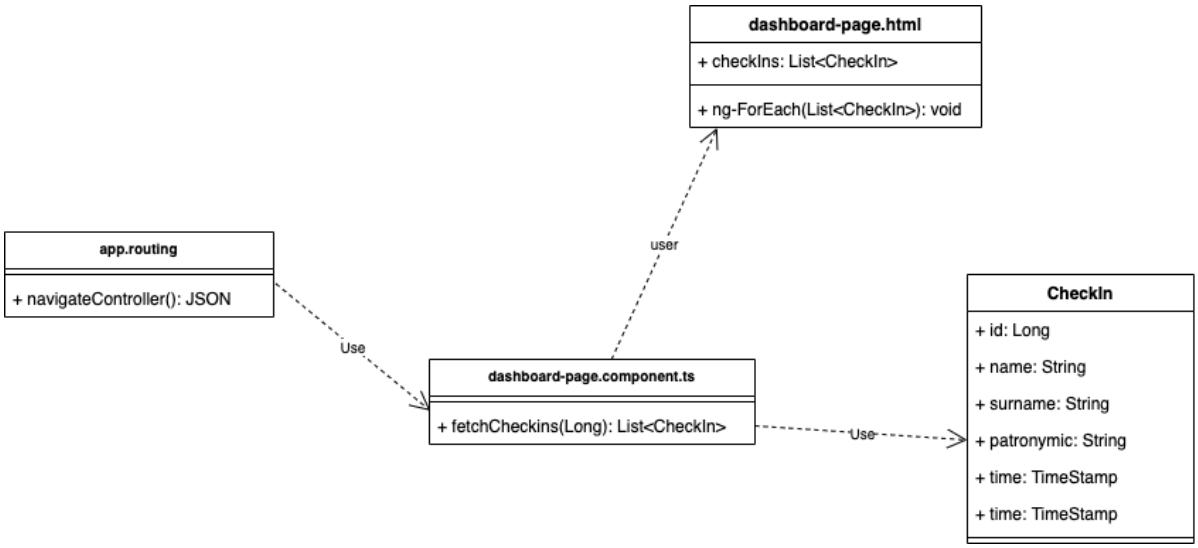

Рисунок 3.9 – Діаграма класів бекенду

Для реалізації цього патерну використовується компонент з Ангуляру, який складається з трьох частин: моделі checkIn, що має ПІБ людини, що прийшла, її час, температуру; контролеру dashboard-page.component.ts, який відповідає за логіку виводу даних та view – dashboard-page.html, що виводить список.

- app.routing – класс, що відповідальний за навігацію по контролерам та компонентам у фронт частині:
- navigateController(): JSON – викликає необхідний контролер
- dashboard-page.component.ts – клас-контролер, що взаємодіє між моделлю та view:
- fetchCheckins(Long): List<Checking> – метод, що отримує нові дані по записам прибуття та вносить їх до view.

ВИСНОВКИ

В процесі виконання бакалаврської роботи були закріплені як теоретичні, так і практичні навички проектування та реалізації біометричної системи доступу працівників із автоматичним температурним скринінгом.

В роботі був проведений аналіз існуючих аналогів, що надають можливість ідентифікувати людину за біометрикою та контролювати графік роботи працівників на підприємстві, на підставі якого визначені їх переваги та недоліки для створення власного бачення оптимальної системи.

Було розглянуто та обрано технології для створення продукту згідно з сучасними тенденціями, а також беручи до уваги важливість найменшої ціни при розробці. Система складається з трьох частин: вебсервер, фронтенд частина та апаратна частина з камерою за модифікованою версію термометру.

Сервер написано на Java за допомогою фреймворку Spring, фронтенд частину розроблено за допомогою Angular. Щодо апаратної частини, то був використаний мікрокомп’ютер Raspberry Pi для створення запитів та використання нейромережі. Для зберігання даних використано СУБД MySQL.

В роботі було сформовано функціональні та нефункціональні вимоги до інформаційної систем. Побудовано діаграму прецедентів, визначені типи користувачів та їх функції. Також було розроблено мокапи майбутнього застосунку з потрібним функціоналом.

Усю роботу ПП було описано у діаграмах – логічному уявленні, уявленні розгортання, слоїв, процесів, даних. Роблено реляційну модель бази даних. Прототип системи був реалізований та протестований на власних зображеннях.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Дудзяний І.М. Об’єктно-орієнтоване моделювання програмних систем: Навчальний посібник. – Львів: Видавничий центр ЛНУ імені Івана Франка, 2007.– 108 с.
2. Java – Вікіпедія. (загол. з екрана). URL: https://uk.wikipedia.org/ wiki/Java (дата звернення 19.05.2023)
3. Коноваленко І.В., Марущак П.О. Платформа .NET та мова програмування C# 8.0: навчальний посібник, Тернопіль: ФОП Паляниця В. А., 2020. 320 с.
4. Цеслів О.В. Основи програмування та веб-дизайн: Навч. посіб. – К.,2020. 149с. URL: https://ela.kpi.ua/bitstream/123456789/40499/1/ OP_veb-dyzain.pdf (дата звернення 19.05.2023)
5. Python – Вікіпедія. (загол. з екрана). URL: https://uk.wikipedia.org/ wiki/Python (дата звернення 19.02.2021)
6. Spring Framework – Вікіпедія. (загол. з екрана). URL:https://uk.wikipedia.org/ wiki/Spring_Framework (дата звернення 14.04.2021)
7. Керівництво з Spring. URL: https://proselyte.net/tutorials/spring-tutorial-full-version/introduction/ (дата звернення 14.04.2023)
8. PostgreSQL – Вікіпедія. (загол. з екрана). URL: https://uk.wikipedia.org/wiki/PostgreSQL (дата звернення 14.04.2021)
9. MySQL – Вікіпедія. (загол. з екрана). URL: https://uk.wikipedia.org/ wiki/MySQL (дата звернення 14.04.2021)
10. Табунщик Г. В. Проектування та моделювання програмного забезпечення сучасних інформаційних систем / Г. В. Табунщик, Т.І. Каплієнко, О.А. Петрова − Запоріжжя : Дике Поле, 2016. – 250 с.

11. Авраменко В.С., Авраменко А.С. Проектування інформаційних систем: навчальний посібник. Черкаси: Черкаський національний університет ім. Б. Хмельницького, 2017. – 434 с.: іл.
12. Raspberry Pi 3 Model B. URL: https://uk.rs-online.com/web/generalDisplay.html?id=raspberrypi (дата звернення: 14.05.2023)
13. Blum, Jeremy (2019). Exploring Arduino: Tools and Techniques for Engineering Wizardry (2nd ed.). Wiley. ISBN 978-1119405375
14. Monk, Simon (2022). Programming Arduino: Getting Started with Sketches (3rd ed.). McGraw-Hill Education. ISBN 978-1264676989
15. Субботін С. О. Нейронні мережі : теорія та практика: навч. посіб. Житомир : Вид. О. О. Євенок, 2020. 184 с