Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“___” _____ 20__ р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи розпізнавання та пошуку інформації за номером автомобіля на базі нейронної мережі”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-22М-2
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Гахов Д.А.
«____» _____ 20 р.

Керівник проекту
доктор технічних наук, професор
_____ Єлизавета МЕЛЕШКО
«____» _____ 20 р.
Рецензент _____
_____

КбПЗ-2023

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

Галузь знань *12 "Інформаційні технології"*

Спеціальність *123 "Комп'ютерна інженерія"*

Освітньо-професійна (освітньо-наукова) програма *"Комп'ютерна інженерія"*

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____Олексій СМІРНОВ
« __ » _____ 20__ року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Гахову Данилу Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи розпізнавання та пошуку інформації за номером автомобіля на базі нейронної мережі*

2. Керівник роботи *Мелешко Єлизавета Владиславівна, доктор техн. наук, професор*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу №__-__ від __.__.20__ року

3. Строк подання роботи до захисту __.__.*2023 р.*

4. Мета та завдання випускної кваліфікаційної роботи: *Дослідження та програмна реалізація системи розпізнавання та пошуку інформації за номером автомобіля на базі нейронної мережі*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи в промислову експлуатацію.
6. Наукова новизна
7. Економічна ефективність розробленої програми.
8. Заходи з охорони праці та техніки безпеки.
9. Висновки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*
Діаграма процесів	*1 аркуш*
Показники економічної ефективності	*1 аркуш*

КОПІЯ 2023

6. Консультанти по роботі, із зазначенням розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В., к.т.н., доцент	25.10.2023	10.11.2023
Охорона праці	Оришака О.В., к.т.н., доцент	03.11.2023	21.11.2023

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2023 р.	
2.	Постановка задачі, оформлення ТЗ	16.10.2023 р.	
3.	Розробка моделі компонента	20.10.2023 р.	
4.	Розробка структур даних	25.10.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2023 р.	
6.	Програмування алгоритмів	10.11.2023 р.	
7.	Розрахунок економічної ефективності	13.11.2023 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2023 р.	
9.	Оформлення ПЗ	17.11.2023 р.	
10.	Попередній захист роботи	10.12.2023 р.	

Дата видачі завдання
«__»_____20 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
«__»_____20 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

КОПЗ – 2023

АНОТАЦІЯ

Гахов Д.А. Дослідження та програмна реалізація системи розпізнавання та пошуку інформації за номером автомобіля на базі нейронної мережі. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній магістерській роботі розроблено програмне забезпечення, яке призначено для реалізації системи розпізнавання та пошуку інформації за номером автомобіля на базі нейронної мережі.

Метою розробки є дослідження та програмна реалізація системи розпізнавання та пошуку інформації за номером автомобіля на базі нейронної мережі.

Об'єктом дослідження є процес ідентифікації автомобіля за зображенням з камери мобільного телефону.

Предметом дослідження є методи побудови нейронних мереж для розпізнавання зображень і текстів та методи розробки мобільних додатків.

Методи дослідження базуються на теорії алгоритмів та структур даних, теорії штучного інтелекту, теорії інженерії програмного забезпечення та теорії об'єктно-орієнтованого програмування.

Результат роботи – програмна реалізація системи розпізнавання та пошуку інформації за номером автомобіля на базі нейронної мережі.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на мобільних телефонах з ОС Android.

Програму розроблено на мові програмування високого рівня Kotlin.

Ключові слова: комп'ютерна інженерія, нейронні мережі, розпізнавання зображень, розпізнавання тексту, ідентифікація автомобіля.

КПЗ-2023

ABSTRACT

Hakhov D. A. Research and software implementation of the system of recognition and retrieval of information by car number on the basis of neural network. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi 2023.

In this master's thesis, software designed for a system for recognizing and searching for information by car number based on a neural network.

The purpose of the development is the research and program implementation of a system for recognizing and searching for information by car number based on a neural network.

The object of the research is the process of car identification based on the image from the mobile phone camera.

The subject of the research is the methods of building neural networks for recognizing images and texts and methods of developing mobile applications.

Research methods are based on the theory of algorithms and data structures, the theory of artificial intelligence, the theory of software engineering, and the theory of object-oriented programming.

The result of the work is the software implementation of a system for recognizing and searching for information by car number based on a neural network.

In the process of working on a software model an analysis of existing hardware and software was performed. All components of the software developed are fully described.

User-friendly user interface is developed. These are instructions for working with software.

The program can be used on mobile phones with Android OS.

The program was developed in the high-level programming language – Kotlin.

Keywords: computer engineering, neural networks, image recognition, text recognition, car identification.

КОТЗ – 2023

ЗМІСТ

КПІ – 2023

КОПЗ – 2023

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

Android – операційна система для мобільних пристроїв, створена компанією Google на базі ядра Linux.

Windows – група сімейств комерційних пропрієтарних операційних систем корпорації Microsoft, орієнтованих на управління за допомогою графічного інтерфейсу.

Нейронна мережа – математична модель, а також її програмне або апаратне втілення, побудована за принципом організації та функціонування біологічних нейронних мереж – мереж нервових клітин живого організму;

ОС – операційна система;

ПЗ – програмне забезпечення;

ПК – персональний комп’ютер.

КбПЗ_2023

ВСТУП

Актуальність теми. У сучасному світі автомобільні поїздки та перевезення є важливою складовою нашого повсякденного життя. Зі зростанням кількості автівок на дорогах, з'являється потреба в збільшенні контролю за безпекою на дорогах. Одним із способів поліпшити безпеку та ефективність дорожнього руху є розробка систем розпізнавання та пошуку інформації за номерами автомобілів. Системи розпізнавання номерів автомобілів використовуються в різних сферах, включаючи дорожнє патрулювання, платні автостоянки, перевірку вживаних автомобілів перед купівлею, забезпечення безпеки різних об'єктів, забезпечення контролю доступу тощо. Однак традиційні методи розпізнавання номерів автомобілів можуть бути обмежені з точки зору точності та швидкості. В цьому контексті нейронні мережі, зокрема глибокі нейронні мережі, виявилися дуже ефективними у завданнях розпізнавання номерів автомобілів. Їх здатність адаптуватися до різних умов освітлення, атмосферних умов та типів номерів робить їх потужним інструментом для автоматизованого збору та обробки інформації. Таким чином дослідження та реалізація систем на базі нейронних мереж для розпізнавання автомобілів є актуальними задачами та дозволять створити надійний та ефективний інструмент, який може використовуватися для підвищення безпеки на дорогах.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи розпізнавання та пошуку інформації за номером автомобіля на базі нейронної мережі.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Дослідження існуючих систем для розпізнавання та пошуку інформації за номером автомобіля.

– Розробка методів та алгоритмів системи для розпізнавання та пошуку

КПІ – 2023

інформації за номером автомобіля на базі нейронної мережі.

– Програмна реалізація системи розпізнавання та пошуку інформації за номером автомобіля на базі нейронної мережі.

Об'єктом дослідження є процес ідентифікації автомобіля за зображенням з камери мобільного телефону.

Предметом дослідження є методи побудови нейронних мереж для розпізнавання зображень і текстів та методи розробки мобільних додатків.

Методи дослідження базуються на теорії алгоритмів та структур даних, теорії штучного інтелекту, теорії інженерії програмного забезпечення та теорії об'єктно-орієнтованого програмування.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

1. Запропоновано метод розпізнавання номера автомобіля на зображенні, отриманому з камери мобільного телефону, з використанням нейронних мереж.

2. Розроблено вітчизняний продукт системи розпізнавання та пошуку інформації за номером автомобіля на базі нейронної мережі, який має більш широкі можливості, на відміну від існуючих аналогів та може бути використаний при ідентифікації автомобіля на фото та відео.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі ідентифікації автомобіля за зображенням з камери мобільного телефону та отримання інформації про нього з відкритих баз даних за його номером.

Достовірність наукових результатів підтверджена теоретичними викладками, результатами тестування розробленого програмного забезпечення, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій та спеціалізованій літературі.

Тож, розробка системи розпізнавання номерів автомобілів на базі нейронних мереж є актуальною задачею, яка відкриває нові можливості забезпеченню безпеки на дорогах та вирішувалася у цій магістерській роботі.

КПЗ-2023

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Призначення системи розпізнавання та пошуку інформації за номером автомобіля на базі нейронної мережі може бути наступним:

– Автоматизовані системи забезпечення безпеки на дорогах. Системи розпізнавання номерів допомагають виявляти автомобілі, які порушують правила дорожнього руху, викликають загрозу безпеці або здійснюють несанкціонований доступ до об’єктів. Це сприяє покращенню дорожньої безпеки та зменшенню ризику аварій.

– Допомога у поліцейському патрулюванні. Поліцейські можуть використовувати системи розпізнавання номерів для автоматичного перевіряння автомобілів на предмет викрадень, розшукуваних осіб або інших кримінальних діянь.

– Забезпечення контролю доступу. Системи розпізнавання номерів можуть бути використані для контролю доступу до охороняємих об’єктів, парковок або в’їзду на територію підприємств. Вони дозволяють автоматично розпізнавати автомобілі та надавати доступ лише авторизованим користувачам.

– Забезпечення моніторингу паркування. Системи розпізнавання номерів можуть допомагати в управлінні паркуванням, визначаючи, які автомобілі паркуються в певних зонах та відстежуючи тривалість паркування.

– Ведення обліку автотранспорту. Державні органи та підприємства можуть використовувати системи розпізнавання номерів для ведення обліку автотранспорту, зокрема для збору статистики, оплати мита, податків та інших адміністративних процедур.

– Збір та аналіз статистичних даних. Системи збору та аналізу даних за номерами автомобілів можуть допомагати у вивченні дорожнього руху, патернів

КПЗ – 2023

руху автомобілів та в інших статистичних дослідженнях.

– Перевірка вживаних автомобілів перед купівлею. Перевірка автомобіля перед покупкою є дуже важливою процедурою, щоб переконатися, що покупець отримає якісне авто та уникне потенційних проблем. На основі номера авто можна перевірити його історію та поточний стан з відкритих джерел і фото з різних площадок для продажу (за наявності оголошень).

Таким чином призначення розроблюваної системи досить широке.

1.2 Область застосування

Система розпізнавання та пошуку інформації за номерами автомобілів на базі нейронних мереж може бути корисною для різних організацій та звичайних користувачів. Розглянемо основні приклади організацій та користувачів, що можуть використовувати таку систему.

Поліція та правоохоронні органи. Поліція може використовувати системи розпізнавання номерів для розшукування викрадених автомобілів та авто, які брали участь у правопорушеннях або належать розшукуваним особам.

Органи безпеки та контролю доступу. Органи безпеки на об’єктах, такі як аеропорти, порти, промислові комплекси, можуть використовувати системи розпізнавання номерів для контролю доступу та безпеки.

Митниця та прикордонний контроль. Митниця та прикордонний контроль можуть використовувати системи розпізнавання номерів для перевірки автомобілів, що в’їжджають чи виїжджають з країни.

Паркування та транспортні компанії. Оператори паркування можуть використовувати системи для контролю доступу та збору плати за паркування. Транспортні компанії можуть використовувати їх для ведення обліку автотранспорту.

Приватні житлові комплекси та бізнеси. Власники приватних житлових комплексів та бізнесів можуть використовувати системи для контролю доступу на

КПІ – 2023

територію та забезпечення безпеки своїх об’єктів.

Автомобільні дилери та страхові компанії. Дилери можуть використовувати системи для автоматичного внесення даних автомобіля в бази даних. Страхові компанії можуть використовувати їх для визначення тарифів та розрахунку страхових внесків.

Приватні особи через мобільні додатки та онлайн-платформи. Мобільні додатки для продажу автомобілів, пошуку паркування, розшукування та обслуговування авто, пошуку інформацію про авто тощо можуть використовувати системи розпізнавання номерів для автоматичного пошуку і збору інформації.

Громадські організації та активісти. Громадські організації можуть використовувати системи розпізнавання номерів для стеження за обсягами дорожнього руху, моніторингу транспортних засобів та публікації даних для громадськості.

Тож, розроблювана система може бути використана різними організаціями та користувачами для різних цілей, зокрема для покращення безпеки, контролю доступу, управління автопарками та багатьох інших сценаріїв.

КБПЗ_2023

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми магістерської роботи

Перш ніж починати розробку програмного забезпечення необхідно провести збір інформації щодо вже існуючих систем розпізнавання та пошуку інформації за номером автомобіля. Потрібно дослідити їх переваги та недоліки та сформувати висновки. Нижче наведено результати такого дослідження.

NumberOk Lite 2 – програма для розпізнавання автомобільних номерних знаків з функцією надсилання результатів детекції в зовнішні додатки.

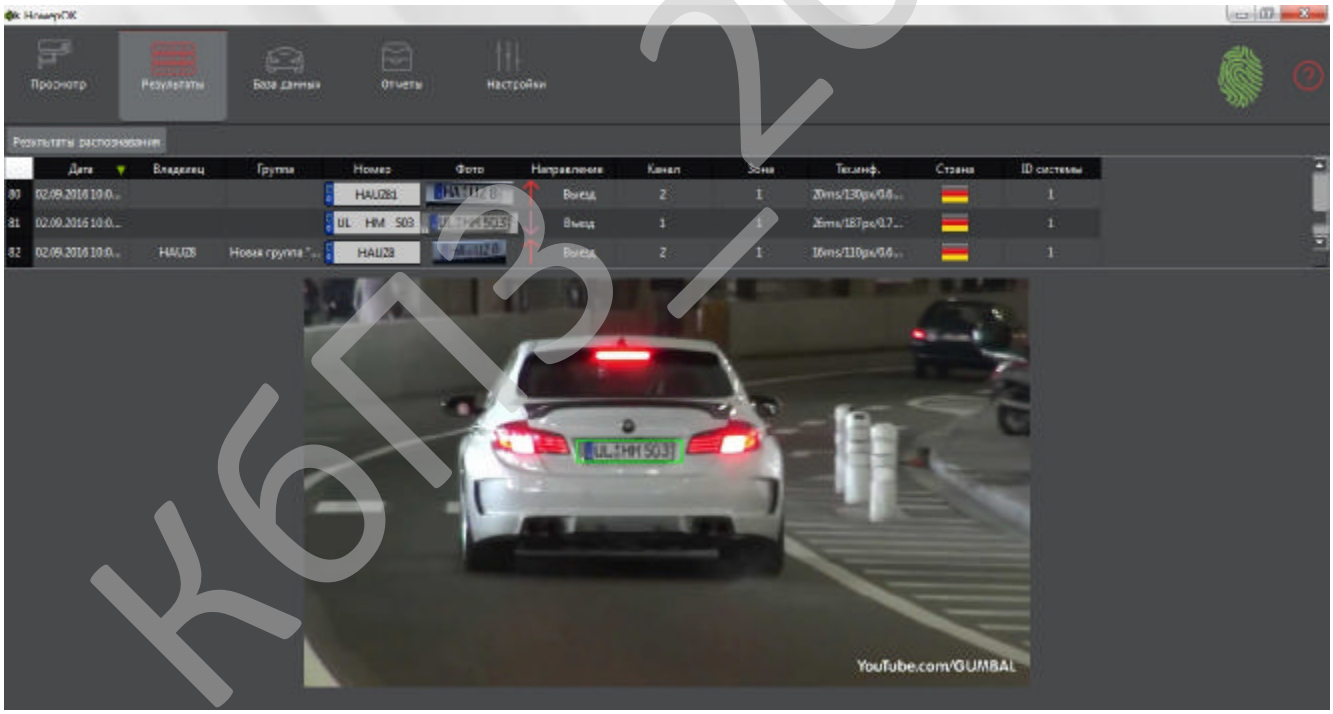


Рисунок 2.1 – Приклад роботи додатку NumberOk Lite 2

Версія NumberOk Lite 2 – це базова версія ПЗі - розрахована на 2 канали та 8 зон розпізнавання (підключається дві відеокамери).

КбПЗ – 2023

Принцип роботи NumberOk Lite

При фіксації автомобільного номера в відеопотоці, програма NumberOk зберігає скріншот транспортного засобу і робить запис в базу даних про подію (дата / час) з результатами розпізнавання.

Версія програмного забезпечення Lite включає модуль розпізнавання автомобільних номерів і передбачає можливість подальшої інтеграції з системами контролю доступу, 1С, ваговими і / або іншими додатками. NumberOk легко інтегрується з іншими додатками, використовуючи базу даних безкоштовного формату (FireBird). Також можливо отримувати дані з ПО NumberOk в режимі реального часу

Країни, номерні знаки яких розпізнаються програмою NumberOk Lite: Україна, Білорусь, Молдова, СНД, Євро Союз, Туреччина, Ізраїль

Формат поставки ПЗ

– інсталяційний файл з необхідними драйверами.

– ліцензійний ключ.

Для активації ПЗ, на робочому сервері, на якому в подальшому буде працювати програма NumberOk, має бути разове підключення до мережі Інтернет при установці.

Мінімальні вимоги до сервера:

– процесор - Intel Core i5 4440 відеокарта (інтегрована) - OpenGL 2.0 оперативна пам’ять - DDR 4Gb жорсткий диск - HDD 500Gb ОС - Windows XP / 7/8/10, server 8 (32/64 bit)

Переваги ПЗ NumberOk:

– сумісно з будь-якою версією ОС Windows.

– висока якість розпізнавання - до 95%.

– робота з будь-якими IP-камерами і аналоговими системами відеоспостереження, що підтримують передачу RTSP-потоку.

– розпізнавання номера при швидкості автомобіля до 250 км / год (в залежності від використовуваної відеокамери).

КПЗ – 2023

– фіксує розпізнані номер, зберігає скріншотів базу.

Події в базі даних: дата і час, розпізнаний номер, скріншот номера і тд.

odb-auto-app – ця програма створена волонтерами, її ціль – розпізнавання номерів автомобілів на основі відкритих даних з реєстру України. Містить останні дані станом на 1 лютого 2022 року – модель, виробник, колір, які за лічені хвилини допоможуть швидко перевірити транспорт на блок-постах.

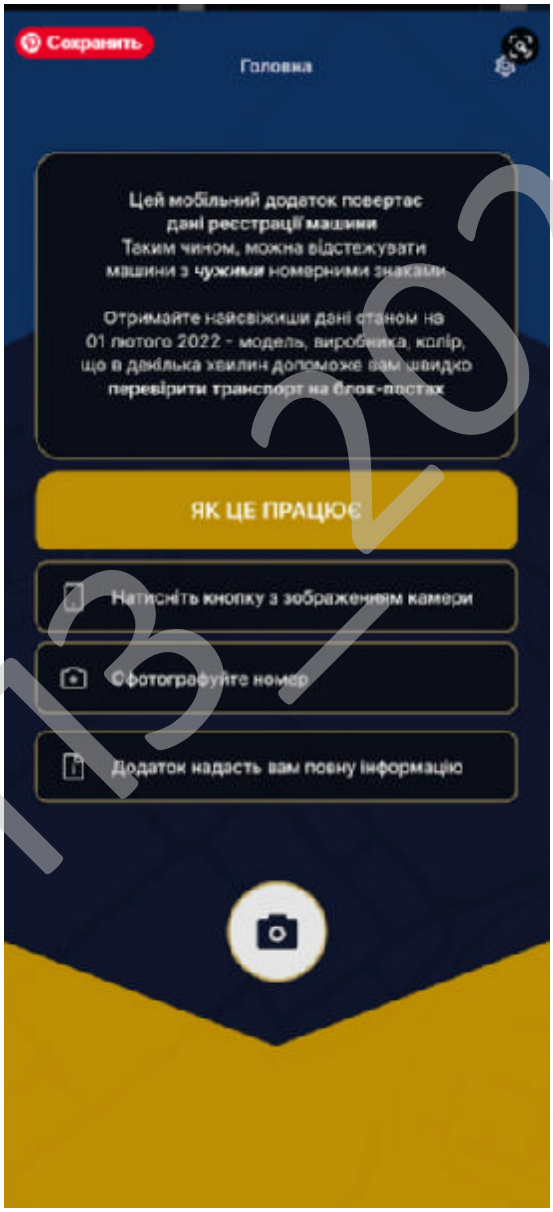


Рисунок 2.2 – Приклад роботи додатку odb-auto-app

КПІ – 2023

В основі додатку лежить машинний алгоритм розпізнавання автомобільних номерів, який працює з даними Opendatabot.

Програма допомагає знизити завантаження співробітників ЗСУ та ТРО на блокпостах та кордонах українських областей. Раніше їм для перевірки номерів доводилося витрачати більше часу через те, що “руками” вносилася інформація про номерний знак машини в комп’ютер, яку іноді доводилося запам’ятовувати на ходу, тому особливо вночі така перевірка займає більше часу.

Розробники також стверджують, що програма майже не витрачає зарядку телефону через продуману хмарну структуру обробки даних, тому підійде для власників економ-смартфонів зі зношеними батареями.

Основний мінус програми – немає поля ручного введення номера, а якщо номер брудний він не розпізнається і відповідно інформацію отримати не можна.

Перевірка автономера – Україна від VIApps – додаток для перевірки автомобіля в Україні. Одна з його переваг – розпізнавання номера автомобіля в реал-тайм режимі, для цього необхідно навести камеру на номер авто, а програма самостійно розпізнає номер і вставить його у форму для введення номера авто.

Рисунок 2.3 – Приклад роботи додатку Перевірка автономера від VIApps

КОПІЯ 2023

Додаток надає наступну інформацію про автомобіль за його номером:

– Дата першої реєстрації авто.
– Кількість реєстрацій автомобіля.
– Назва операції.
– Адреса сервісного центру.
– Марка – модель автомобіля.
– Рік випуску.
– Колір автомобіля.
– Тип транспортного засобу.
– Кузов.
– Паливо.
– Об’єм двигуна.
– Вага без/з навантаженням.
– Адреса реєстрації.
– Власник.
– Наявність у базі угону.
– Середня вартість.
– Фото автомобіля за номером в Україні (якщо доступне).
– Історія попередніх перевірок.
– Історія попередніх перевірок.
– Перевірка авто ввезених в Україну на участь у закордонних аукціонах.
– Область реєстрації транспортного засобу.
– Оцінка номера авто.
– Пошук оголошень про продаж автомобіля.
– VIN декодер, базова інформація про транспортний засіб.

Доступна перевірка автомобілів, зареєстрованих раніше 2013 року.

Джерело даних під час перевірки авто: Єдиний державний веб портал відкритих даних http://data.gov.ua.

КБТЗ – 2023

Перевірка АвтоНомери – Україна від CarPlates LLC – додаток для швидкої перевірки авто на українських номерах по базі МВС.

Має наступний функціонал:

– Пошук авто за номером і VIN.

– Перевірка телефону продавця.

– Пошук по свідоцтву о реєстрації.

– Перевірка на Арешт, викрадення, заставу по базах ДРОРМ.

– Перевірка авто по базах США, Європи, України.

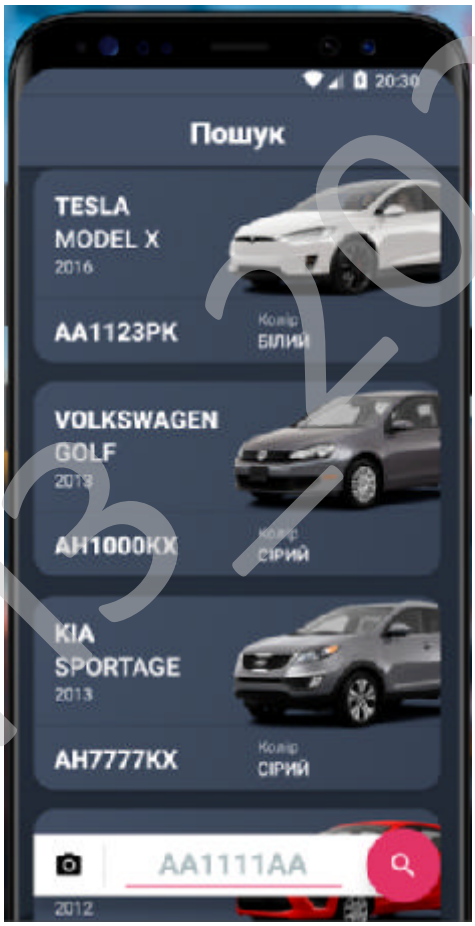


Рисунок 2.4 – Приклад роботи додатку АвтоНомери від CarPlates LLC

Сервіс Автономери надає відразу повну історію про авто перед його покупкою:

– VIN код автомобіля.

КПЗ 2023

– Тип мотора.

– Фото авто.

– Рік випуску автомобіля.

– Колір авто.

– Маса авто.

– Детальна інформація про історій реєстрації транспортного засобу, дата імпорту в Україну.

– Перевірка страховки осага, інформація про страховий поліс.

– Робиться повна перевірка автономеру по базі Викрадення, можна дізнатися чи не перебуває автомобіль в розшуку по базах МВС та Інтерполу.

– Середня вартість автомобіля в Україні.

– Коротка інформація про власника. Категорія власника (Фізичне, Юридична особа).

– Регіон проживання власника авто.

– Дата першої реєстрації авто в Україні.

– Перевірка авто по базах закордонних аукціонів США, ЄВРОПА, КОРЕЯ.

– Пробіг авто, ціна авто, інформація про ДТП. Повний звіт з аукціону і фото авто на аукціоні.

– Пошук авто за оголошеннями про продаж в Україні.

– Пошук і перевірка продавця за номером телефону.

– Пошук по Українському техпаспорту.

– Розшифровка ВІН коду.

– Перевірка авто на будь-які види обмежень і обтяжень по базі ДРОРМ (борги, банки, арешти, кредити, застави).

– Сканування авто номера через камеру.

Перевірка телефону продавця допомагає дізнатися коли і які авто продавалися за номером телефону.

Дані у додатку поширюються відповідно до закону "Про доступ до публічної інформацій" і головним джерелом даних є "Портал відкритих даних"

КПЗ-2023

data.gov.ua. Дані по всіх транспортних засобах, актуальні для всіх авто зареєстрованих в держ.реєстрі України, починаючи з 1 січня 2013 року.

Додаток не здійснює розпізнавання та пошук авто за фотографією.

ZetPro VMS для Windows – програмне забезпечення українського підприємства Kobi software для розпізнавання номерів з RTPS; розпізнавання номерів у камерах ZetPro Smart, UNV, Dahua, Hanwa Techwin; розпізнавання марки та моделі авто; розпізнавання облич та з функцією тривожних оповіщень.

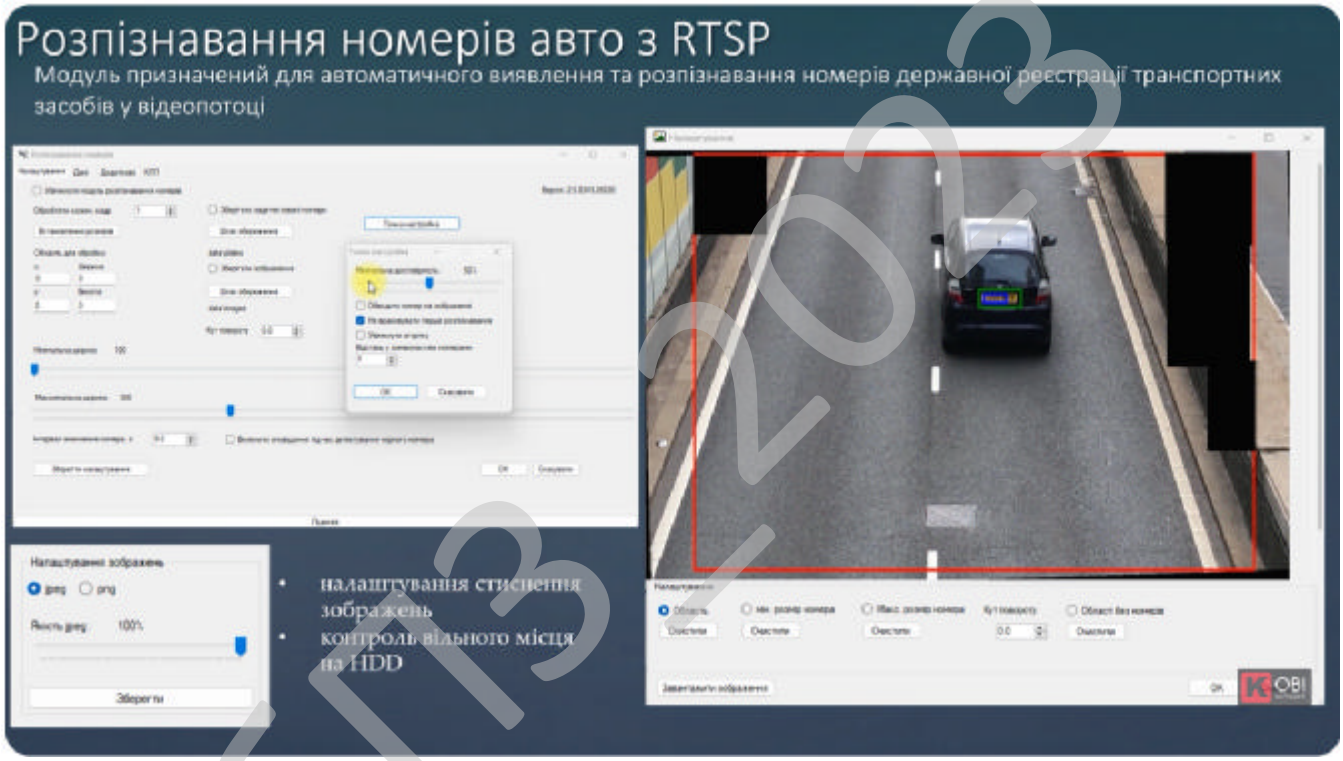


Рисунок 2.5 – Приклад роботи додатку ZetPro VMS для Windows

ZetPro VMS – програмне забезпечення з інтелектуальними модулями для встановлення на ПК, розроблене для роботи з різними виробниками систем відеоспостереження (камери/мережеві реєстратори). Програмний комплекс ZetPro VMS пропонує рішення для різноманітних завдань. Використовуючи передові технології штучного інтелекту, аналітичні модулі цього додатку здійснюють розпізнавання облич, номерів авто, марки та моделі авто, працюють із високою

КПІ 2023

точністю. Окрім цього, алгоритми розпізнавання номерів можуть працювати на периферійних пристроях, що дозволяє значно знизити вимоги до потужності процесора. Для тривожних сповіщень використовується Telegram чат-бот. Виконавши прості налаштування, всі тривожні події «чорних» списків надходитимуть до груп месенджера Telegram. Базова версія програмного забезпечення містить ядро ZetPro VMS, 4 канали розпізнавання осіб, тривожні повідомлення «Telegram».

Інтелектуальні модулі ZetPro VMS:

– Розпізнавання облич.

– Розпізнавання номерів з RTPS та розпізнавання номерів у камерах ZetPro Smart, UNV, Dahua, Hanwa Techwin.

– Розпізнавання марки та моделі авто

– Тривожні оповіщення «Telegram».

Мінімальні системні вимоги до програмного та апаратного забезпечення для базової версії:

– ОС: Microsoft Windows 7, Windows 8.1 або Windows 10.

– Процесор: Intel® Core i3 або краще.

– ОЗУ: 8 ГБ.

– Жорсткий диск: 1 ГБ вільного простору.

– Відеокарта: 1 ГБ пам’яті.

До недоліків даного додатку можна віднести наступні:

– висока ціна та відсутність безкоштовної версії;

– неможливість використовувати на мобільних додатках.

Nomeroff Net – бібліотека автоматичного визначення і розпізнавання автомобільних номерів, яка написана на мові програмування Python v3 з використанням сучасної згорткової мережі Mask RCNN, яка відповідає за визначення позиції номерного знаку, та спеціалізованої OCR, яка відповідає за розпізнавання символів та побудована на різновиді рекурентної нейронної мережі з GRU-шарами. Була створена українськими програмістами Черніем О. та

КПЗ – 2023

Пробачаєм Д. у 2018 році для компанії AUTO.RIA. При завантаженні користувачами фотографій на AUTO.RIA бібліотека Nomeroff Net автоматично визначає номер автомобіля, що дає змогу сервісам сайту робити додаткові перевірки за даними реєстру. Дані про авто беруться з "Єдиного державного вебпорталу відкритих даних", а саме, з його розділу "Відомості про транспортні засоби та їх власників".

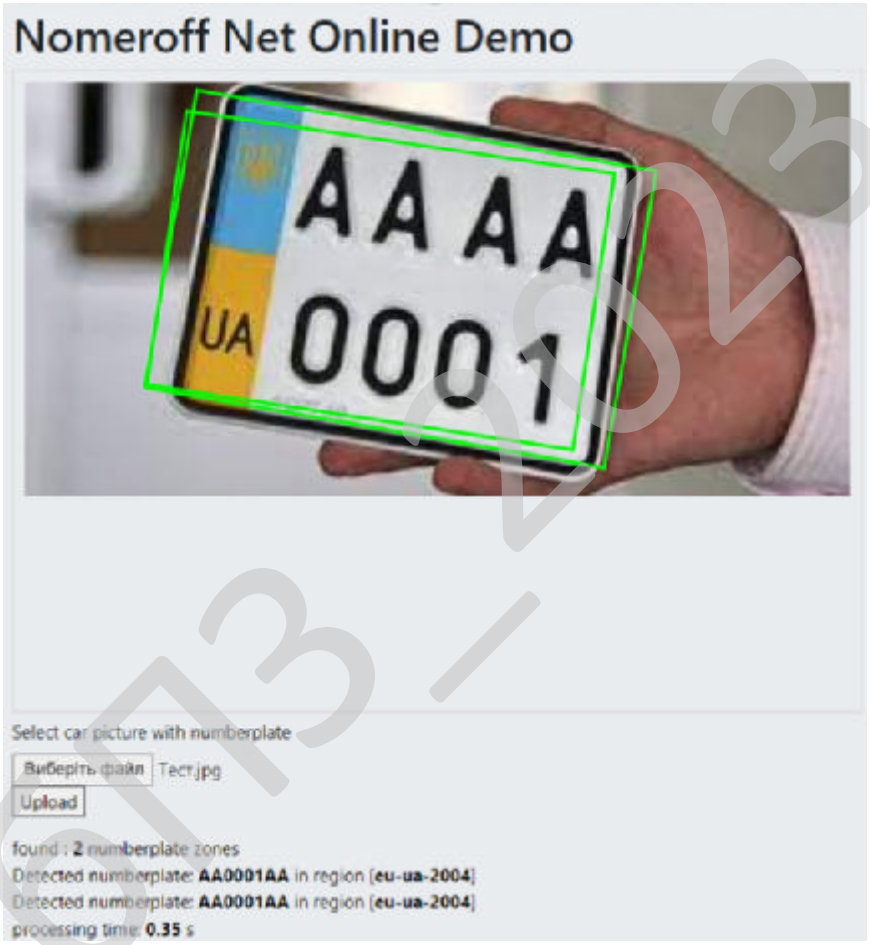


Рисунок 2.6 – Приклад роботи демо-версії роботи бібліотеки Nomeroff Net

Переваги бібліотеки Nomeroff Net:

– висока якість розпізнавання – 97% для українських номерів та 94% для Європейських номерів;

– містить натреновані моделі для розпізнавання автомобільних номерів для різних країн.

КОПЗ_2023

– код бібліотеки опублікований на GitHub під відкритою ліцензією GNU GPL v3;

– проєкт продовжує підтримуватися та знаходиться в стадії активного розвитку.

Серед недоліків слід зазначити, що даний продукт є бібліотекою, а не готовим програмним забезпеченням і буде корисний тільки програмістам, а не звичайним користувачам. Звичайні користувачі можуть користуватися ним лише у вигляді частини функціоналу веб-сайту AUTO.RIA, або у вигляді демо-версії на веб-сайті бібліотеки.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Kotlin представляє сучасну, статично типізовану мову програмування, що швидко розвивається. Вона створена і розвивається компанією JetBrains. Kotlin можна використовувати для створення різних програм. Це і програми для мобільних пристроїв – Android, iOS. Причому Kotlin дозволяє писати кросплатформовий код, який застосовуватиметься на всіх платформах. Це і веб-додатки, причому як серверні програми, які відпрацьовують на стороні сервера – бекенда, так і браузерні клієнтські програми - фронтенд. Kotlin також можна застосовувати для створення десктопних програм, для Data Science і так далі.

Таким чином, коло платформ, для яких можна створювати програми на Kotlin, надзвичайно широке – Windows, Linux, Mac OS, iOS, Android.

Найпопулярнішим напрямком, де застосовується Kotlin, є насамперед розробка під ОС Android. Причому настільки популярним, що компанія Google на конференції Google I/O 2017 проголосила Kotlin однією з офіційних мов для розробки під Android (поряд з Java і C++), а інструменти по роботі з цією мовою були включені в функціонал середовища розробки Android Studio по замовчуванню із версії 3.0.

КПІ – 2023

Kotlin зазнав впливу багатьох мов: Java, Scala, Groovy, C#, JavaScript, Swift і дозволяє писати програми як в об'єктно-орієнтованому, так і функціональному стилі. Він має ясний та зрозумілий синтаксис і досить легкий для навчання.

Також варто зазначити, що Kotin розвивається як opensource, вихідний код проекту можна переглянути в репозиторії на github за адресою https://github.com/JetBrains/kotlin/.

Тож, мова програмування Kotlin стала популярною для розробки додатків для платформи Android.

Вона має декілька переваг, які можуть бути корисні при створенні додатків для розпізнавання автомобільних номерів по фото:

1. **Інтероперабельність з Java**. Kotlin підтримує інтероперабельність з мовою Java, що означає, що ви можете використовувати існуючий Java-код в додатку на Kotlin. Це особливо корисно для проектів, які вже мають Java-код.

2. **Безпека**. Kotlin надає покращену безпеку в порівнянні з Java. Вона має строгу систему типів, яка допомагає запобігти багатьом типовим помилкам програмування і підвищує надійність програм.

3. **Короткий та зрозумілий код**. Kotlin спрощує написання більш короткого та зрозумілого коду, що допомагає підвищити продуктивність розробників.

4. **Розширені можливості при роботі з Android API**. Kotlin має покращену підтримку Android API, що полегшує роботу з Android-специфічними можливостями та ресурсами.

5. **Користувацька розробка DSL**. Kotlin дозволяє розробникам створювати власні користувацькі DSL (мови для доменних областей), що полегшує роботу з певними областями програмування.

6. **Користувацька обробка розподілених операцій та паралельних потоків**. Якщо розпізнавання автомобільних номерів вимагає обробки багатьох операцій паралельно, Kotlin надає зручність у роботі з корутинами та обробці паралельних потоків.

КПЗ – 2023

2.3 Розгорнута постановка завдання

У цій кваліфікаційній роботі необхідно створити систему розпізнавання та пошуку інформації за номером автомобіля на базі нейронної мережі для різних областей застосування. Завдання включає в себе дослідження предметної області та розробку програмного забезпечення, навчання нейронної мережі, інтеграцію з сучасними технологіями та створення зручного інтерфейсу для користувачів. Важливим аспектом є забезпечення точності розпізнавання та обробки великої кількості даних.

Конкретні етапи виконання поставленої задачі:

1. Дослідження існуючих методів, технологій та систем для розпізнавання номерів автомобілів по відео та зображеннях. Дослідження їх переваг та недоліків. Вибір методів та технологій для реалізації власної системи.

2. Збір та підготовка даних. Необхідно зібрати базу даних для навчання та тестування нейронної мережі.

3. Вибір архітектури нейронної мережі. Вибрати архітектуру нейронної мережі, яка буде відповідати вимогам точності та швидкості розпізнавання.

4. Навчання нейронної мережі. Використовувати зібрані та розмічені дані для навчання нейронної мережі.

5. Інтеграція зі зовнішніми системами. Розробити інтерфейси та API для взаємодії системи розпізнавання з іншими системами, такими як системи безпеки, контролю доступу, транспортні бази даних тощо.

6. Розробка мобільного додатку та інтерфейсу користувача. Створити мобільний додаток для користувачів, який дозволяє отримувати з камери телефону фотографії номерів, розпізнавати ці номери нейронною мережею та шукати і отримувати інформацію про автомобілі за їх номерами з відкритих баз даних.

7. Оптимізація та масштабування. Забезпечити оптимальну швидкість та ефективність роботи системи під час обробки великого обсягу запитів.

КПЗ 2023

8. Тестування та валідація. Провести тестування системи на різних наборах даних для підтвердження точності та надійності розпізнавання.

9. Впровадження та підтримка. Впровадити систему в реальних умовах.

10. Забезпечення конфіденційності та безпеки. Захистити дані користувачів та систему від несанкціонованого доступу та злому.

11. Документація. Надати інструкції та рекомендації для користувачів та адміністраторів к працювати з системою, підготувати документацію.

КбПЗ_2023

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Для розробки програмного забезпечення було використано рекомендовану архітектуру для Android-додатків.

Враховуючи загальні архітектурні принципи, кожна програма повинна мати принаймні два рівні:

– *Рівень інтерфейсу користувача*, який відображає дані програми на екрані.

– *Рівень даних*, який містить бізнес-логіку вашої програми та відкриває дані програми.

Можна додати додатковий рівень під назвою *рівень домену*, щоб спростити та повторно використовувати взаємодію між інтерфейсом користувача та рівнями даних.

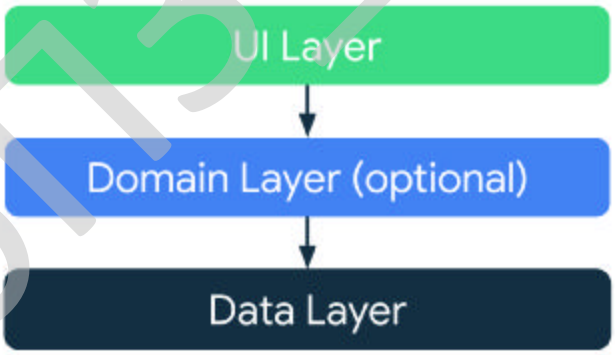


Рисунок 3.1 – Діаграма типової архітектури Android-додатку

Сучасна архітектура програми

Ця *сучасна архітектура програми* заохочує використання таких методів, зокрема:

– Реактивна та багаторівнева архітектура.

КПЗ 2023

– Односпрямований потік даних (UDF) на всіх рівнях програми.

– Рівень інтерфейсу користувача з власниками стану для керування складністю інтерфейсу користувача.

– Співпрограми та потоки.

– Найкращі практики впровадження залежностей.

Рівень інтерфейсу користувача

Роль рівня інтерфейсу користувача (або *рівня презентації*) полягає у відображенні даних програми на екрані. Щоразу, коли дані змінюються через взаємодію користувача (наприклад, натискання кнопки) або зовнішній вхід (наприклад, відповідь мережі), інтерфейс користувача має оновлюватися, щоб відобразити зміни.

Рівень інтерфейсу користувача складається з двох речей:

– Елементи інтерфейсу користувача, які відображають дані на екрані. Ці елементи створюються за допомогою функцій Views або Jetpack Compose .

– Власники стану (наприклад, класи ViewModel), які зберігають дані, надають їх інтерфейсу користувача та обробляють логіку.

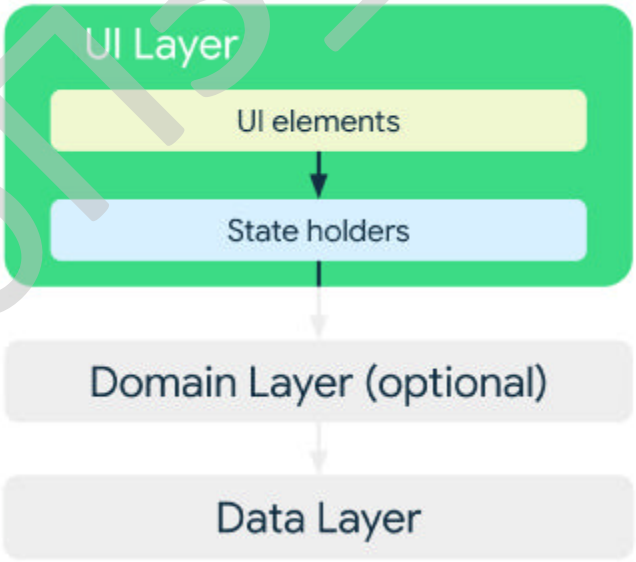


Рисунок 3.2 – Роль рівня інтерфейсу користувача в архітектурі програми

КбПЗ 2023

Рівень даних

Рівень даних програми містить *бізнес-логіку* . Бізнес-логіка – це те, що надає цінність вашій програмі — вона складається з правил, які визначають, як ваша програма створює, зберігає та змінює дані.

Рівень даних складається із *сховищ* , кожне з яких може містити від нуля до багатьох *джерел даних* . Ви повинні створити клас сховища для кожного окремого типу даних, які ви обробляєте у своїй програмі. Наприклад, ви можете створити MoviesRepository клас для даних, пов'язаних із фільмами, або PaymentsRepositoryуклас для даних, пов'язаних із платежами.

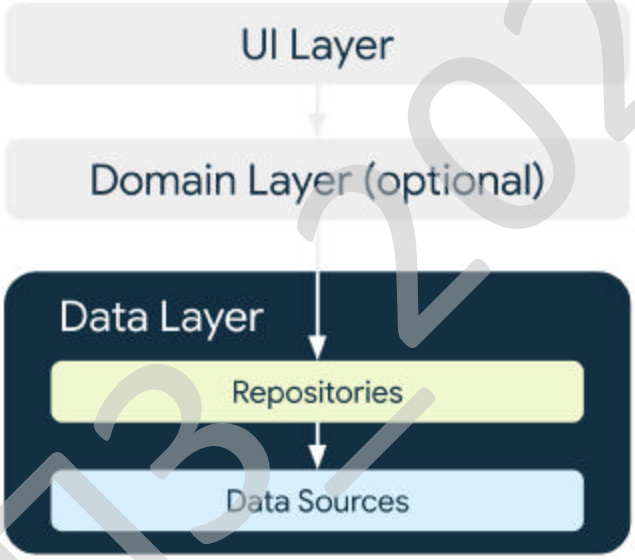


Рисунок 3.3 – Роль рівня даних в архітектурі програми.

Класи репозиторію відповідають за наступні завдання:

- Відкриття даних решті програми.
- Централізація змін даних.
- Вирішення конфліктів між кількома джерелами даних.
- Абстрагування джерел даних від решти програми.
- Містить бізнес-логіку.

Кожен клас джерела даних повинен нести відповідальність за роботу лише

КПІ 2023

з одним джерелом даних, яким може бути файл, мережеве джерело або локальна база даних. Класи джерел даних є мостом між додатком і системою для операцій з даними.

Доменний рівень

Рівень домену – це необов'язковий рівень, який знаходиться між інтерфейсом користувача та рівнями даних.

Рівень домену відповідає за інкапсуляцію складної бізнес-логіки або простої бізнес-логіки, яка повторно використовується кількома ViewModels. Цей рівень є необов'язковим, оскільки не всі програми мають ці вимоги. Використовуйте його лише тоді, коли це необхідно, наприклад, щоб усунути складність або віддати перевагу повторному використанню.

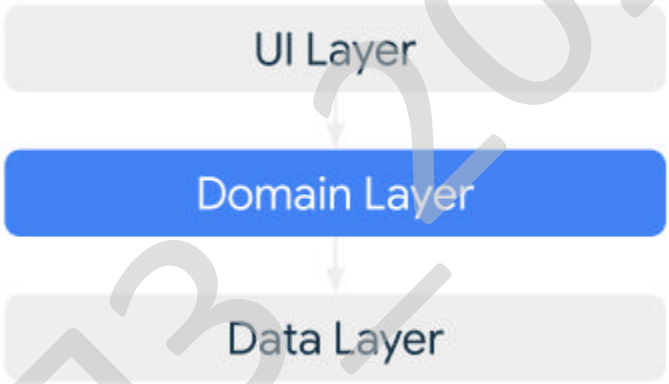


Рисунок 3.4 – Роль доменного рівня в архітектурі програми

Класи на цьому рівні зазвичай називають *варіантами використання* або *інтеракторами*. Кожен варіант використання повинен відповідати за *окрему* функцію. Наприклад, ваша програма може мати GetTimeZoneUseCaseклас, якщо кілька ViewModels покладаються на часові пояси для відображення правильного повідомлення на екрані.

Розпізнавання тексту на зображеннях

Для ідентифікації автомобілів за фото або відео, отриманих з камери телефону, необхідно використати засоби машинного навчання для розпізнавання

КбПЗ-2023

тексту на номерних знаках автомобілів. Для вирішення цієї задачі у роботі була використана бібліотека Firebase ML Kit Google для роботи з нейронними мережами.

Бібліотека Firebase ML Kit Google надає наступні можливості:

– Розпізнавання тексту (Text recognition).

– Розпізнавання обличь та емоцій на них (Face detection).

– Розпізнавання штрих-кодів та QR-кодів (Barcode scanning).

– Виділення об’єктів на зображенні (Image labeling).

– Визначення місцевості (Landmark recognition).

Також за допомогою даного рішення можна використовувати свою власну TensorFlow Lite model. Надає зручний засіб створення та тренування своїх кастомних моделей за допомогою зображень. Є безкоштовна версія, яка дозволяє тримати один проект. Але безкоштовна версія обмежує кількість зображень для навчання – до 3000 шт. Та цього вистачить щоб спробувати і зробити середню за точністю нейронну мережу. Для більш складних завдань потрібна платна версія. Все, що потрібно від розробника для організації процесу навчання – додати зображення з позначками (наприклад, image1 – "sea", image2 – "sun"), навчити і експортувати граф для подальшого використання.

Глибоке навчання на основі TensorFlow розпізнає об’єкти на зображеннях за допомогою трьох або більше рівнів штучних нейронів у нейронній мережі, у яких кожен рівень відповідає за виділення однієї або кількох характеристик зображення (рис. 3.5).

Нейронна мережа – це обчислювальна модель, аналогічна біологічним нейронним мережам у мозку людини. Кожен нейрон приймає вхідні дані, виконує операцію, а потім надсилає вихідні дані одному чи кільком сусіднім нейронам.

TensorFlow, випущений Google у 2015 році, – це бібліотека програмного забезпечення з відкритим кодом для машинного навчання, яка швидко стала дуже популярною бібліотекою для створення нейронних мереж.

КПІ – 2023

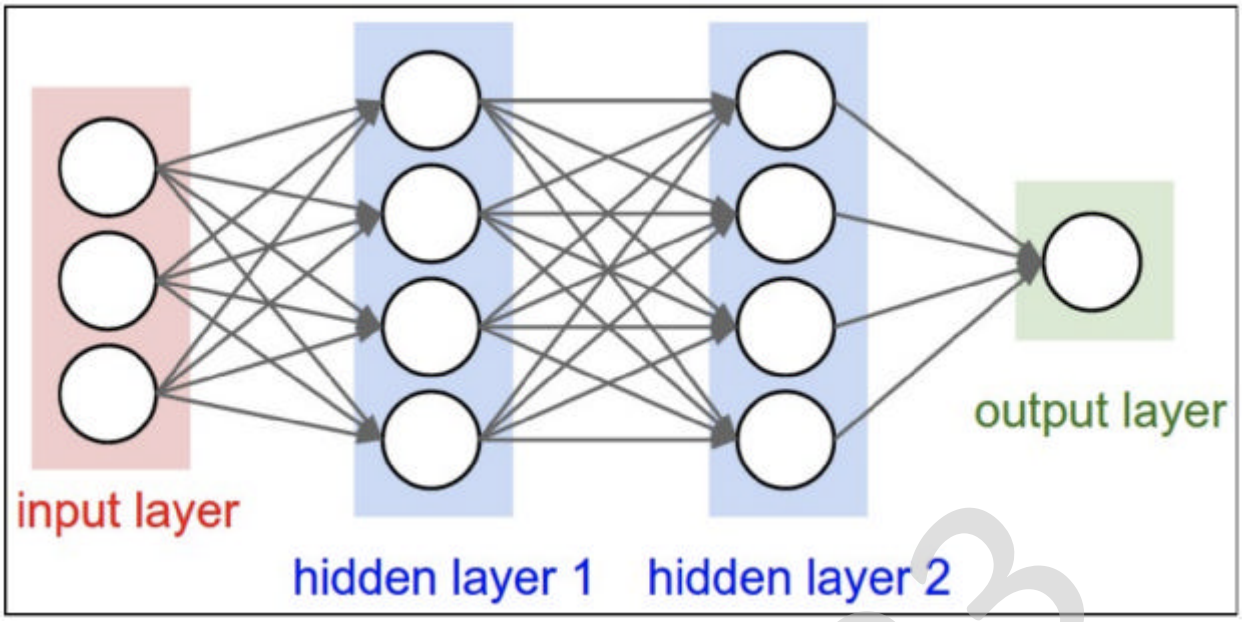


Рисунок 3.5 – Загальна схема нейронної мережі для розпізнавання зображень

Нейронна мережа найчастіше навчається за допомогою процедури зворотного поширення – поширення сигналів помилки від виходів нейронної мережі до її входів, у бік, зворотний прямому поширення сигналів у звичайному режимі роботи. Для створення нейронної мережі, яка може працювати із зашумленими вхідними даними, необхідно навчити мережу, подаючи на вхід дані як з шумом, так і без.

Після навчання нейронної мережі проводиться її тестування та вимірюється похибка роботи. Якщо рівень похибки задовільний, то нейронну мережу можна використовувати. Якщо ж ні, то треба змінювати її архітектуру або розширювати набори вхідних даних та проходити знов етапи навчання та тестування.

Для роботи з TensorFlow через Firebase ML Kit Google необхідні:

– остання версія Android Studio;

– пристрій або емулятор під керуванням Android API рівня 21 або вище;

– обліковий запис у Firebase;

– Google Cloud аккаунт.

КПЗ – 2023

Щоб підключити служби Firebase для розроблюваної програми, треба створити для неї проект Firebase. Для цього треба увійти у консоль Firebase (рис. 3.6), на екрані привітання натиснути кнопку «Додати проєкт» і слідувати інструкціям, зокрема, додати назву проєкту та погодитися з умовами використання. Після створення нового проєкту у розділі Build оберіть Machine Learning.

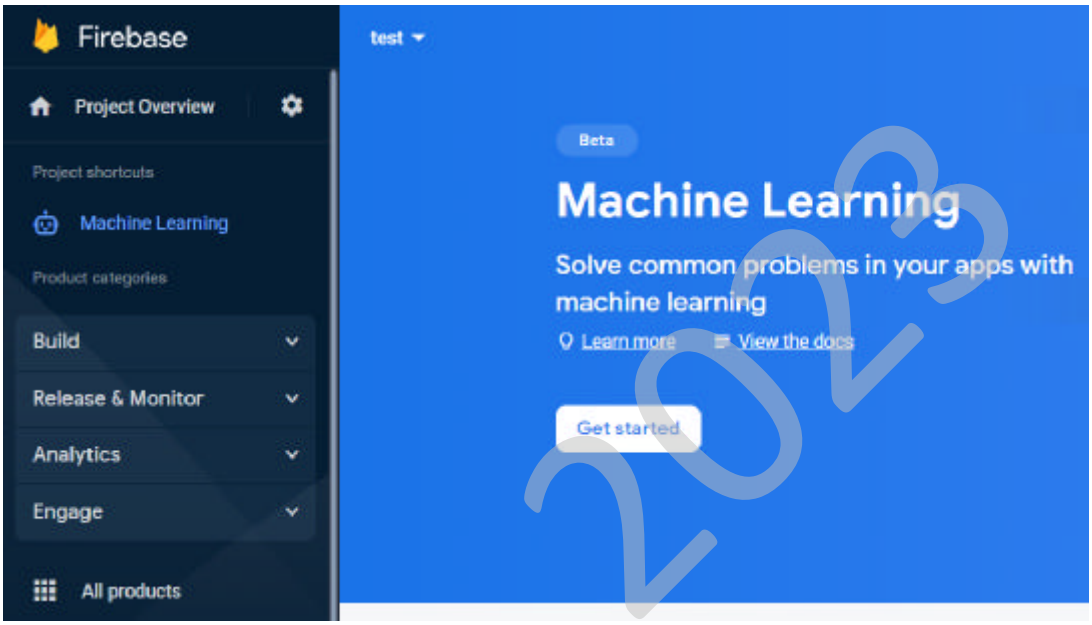


Рисунок 3.6 – Консоль Firebase – створення нового проекту

Перш ніж почати використовувати API Firebase ML Kit, треба встановити з'єднання між своїм проєктом Android Studio і проєктом Firebase, створеним на попередньому кроці, у консолі Firebase.

Після того, як з'єднання буде успішно встановлено, слід натиснути кнопку Додати Google Analytics у розроблювану програму, щоб додати різні базові залежності Firebase до файлу build.gradle свого модуля app.

Потім, щоб додати бібліотеку ML Kit, треба відкрити файл build.gradle і ввести такі implementation:

```
implementation 'com.google.firebase:firebase-ml-vision'
implementation 'com.google.firebase:firebase-ml-vision-image-label-model'
```

За промовчанням локальні моделі Firebase ML Kit автоматично завантажуються на пристрої користувача лише за необхідності. Однак, якщо є

КПЗ – 2023

необхідність, щоб вони були завантажені відразу після інсталяції програми, треба додати наступний код до файлу AndroidManifest.xml:

```
<meta-data
    android:name="com.google.firebase.ml.vision.DEPENDENCIES"
    android:value="text,face,label" />
```

Firebase ML Kit має окремі класи детекторів для всіх операцій розпізнавання зображень, які він пропонує. Щоб розпізнати текст, треба використати клас FirebaseVisionTextDetector, який залежить від локальної моделі, або використати клас FirebaseVisionCloudTextDetector, який залежить від хмарної моделі. Будемо використовувати першу. Вона набагато швидша, але може обробляти текст, написаний тільки на латинському алфавіті.

Детектор ML Kit очікує, що його вхід буде як об'єкт FirebaseVisionImage. Щоб створити такий об'єкт, все, що потрібно зробити, це викликати метод fromBitmap() класу FirebaseVisionImagei передати йому растрове зображення. Наступний код, який має бути доданий в обробник події recognizeText(), показує, як перетворити зображення, яке відображається у віджетах ImageView, на растрове зображення, а потім створити з нього об'єкт FirebaseVisionImage:

```
val textImage = FirebaseVisionImage.fromBitmap(
        (image_holder.drawable as BitmapDrawable).bitmap
)
```

Потім, щоб отримати посилання на об'єкт FirebaseVisionTextDetector, треба використовувати екземпляр FirebaseVision.

```
val detector = FirebaseVision . getInstance (). visionTextDetector
```

Тепер можна розпочати процес розпізнавання тексту, викликавши метод detectInImage() і передавши йому об'єкт FirebaseVisionImage. Оскільки метод виконується асинхронно, він повертає об'єкт Task. Отже, щоб мати можливість обробляти результат, коли він доступний, слід додати до нього екземпляр OnCompleteListener. Ось як:

```
detector.detectInImage(textImage)
        .addOnCompleteListener {
            // …
        }
```

Всередині слухача буде отримано доступ до списку об'єктів Block.

КПЗ – 2023

Загалом кожен блок можна розглядати як окремий абзац, виявлений на зображенні. Переглянувши властивості text усіх об'єктів Block, можна визначити весь текст, який був виявлений. Наступний код показує, як це зробити:

```
var detectedText = ""
it.result.blocks.forEach {
    detectedText += it.text + "\n"
}
```

Те, як буде використовуватися виявлений текст, звичайно залежить від мети додатку. В розроблюваному програмному забезпеченні він буде порівняний з шаблонами, що відповідають формату номерів автомобілів. І якщо знайдений текст відповідає шаблонам, то він буде повернутий в основну програму, а користувач зможе здійснити на основі розпізнаного номера машини пошук інформації про неї у відкритих базах даних.

Розглянемо спосіб виведення отриманого тексту номера автомобіля на екран додатку. Використовуємо функцію alert():

```
runOnUiThread {
    alert(detectedText, "Text").show()
}
```

У наведеному коді метод runOnUiThread() гарантує, що функція alert() запускається в основному потоці програми.

Нарешті, як тільки програміст закінчує використовувати детектор, він повинен викликати його метод close(), щоб звільнити всі ресурси, які той зберігає:

```
detector.close()
```

При розпізнаванні тексту Firebase ML Kit сегментує текст на блоки, рядки, елементи та символи (рис. 3.7):

– *Блок* – це безперервний набір текстових рядків, наприклад абзац або стовпець.

– *Рядок* – це безперервний набір слів на одній осі.

– *Елемент* – це безперервний набір буквено-цифрових символів («слово») на одній осі в більшості латинських мов або слово в інших.

КбПЗ – 2023

– *Символ* – це одиночний буквено-цифровий символ на одній осі в більшості латинських мов або символ в інших.

Для всіх виявлених блоків, ліній, елементів та символів API повертає обмежувальні рамки, кутові точки, інформацію про поворот, показник достовірності, розпізнані мови та розпізнаний текст.

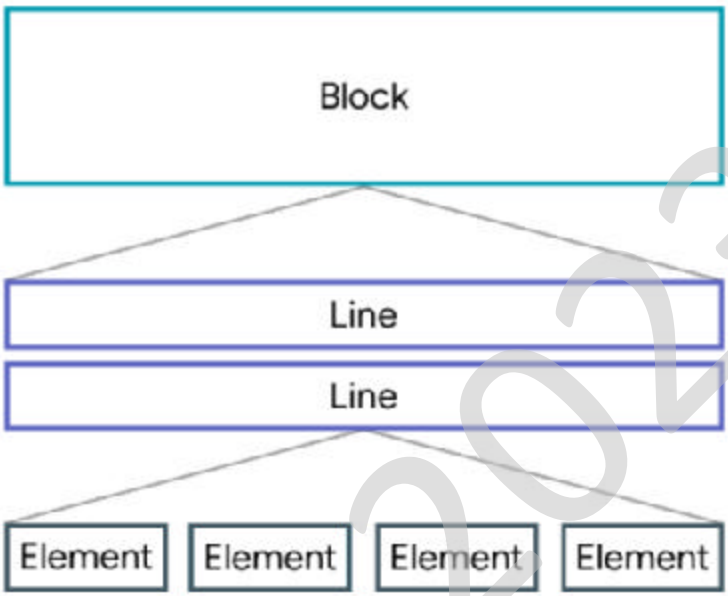


Рисунок 3.7 – Структура розпізнаного тексту в Firebase ML Kit

Щоб Firebase ML точно розпізнавав текст, вхідні зображення повинні містити текст, представлений достатньою кількістю піксельних даних. В ідеалі для латинського тексту розмір кожного символу має бути не менше ніж 16x16 пікселів. Для китайського, японського та корейського тексту кожен символ повинен мати розмір 24x24 пікселів. Так, наприклад, зображення розміром 640×480 може добре підійти для сканування візитної картки, що займає всю ширину зображення. Для сканування документа, надрукованого на папері формату Letter, може знадобитися зображення розміром 720×1280 пікселів. Погане фокусування зображення може знизити точність розпізнавання тексту.

Якщо операція розпізнавання тексту завершиться успішно, об’єкт FirebaseVisionText буде передано обробнику успішного розпізнавання. Об’єкт

КПЗ – 2023

FirebaseVisionText містить повний текст, розпізнаний на зображенні, та нуль або більше об’єктів TextBlock.

Кожен TextBlock є прямокутним блоком тексту, який містить нуль або більше об’єктів Line. Кожен об’єкт Line містить нуль або більше об’єктів Element, які представляють слова та подібні до них об’єкти (дати, числа і т. д.).

Для кожного об’єкта TextBlock, Line і Element можна отримати текст, що розпізнається у фрагменті зображення, і координати, що обмежують цей фрагмент.

Приклад виймання тексту із блоків розпізнаного тексту:

```
val resultText = result.text
for (block in result.textBlocks) {
    val blockText = block.text
    val blockConfidence = block.confidence
    val blockLanguages = block.recognizedLanguages
    val blockCornerPoints = block.cornerPoints
    val blockFrame = block.boundingBox
    for (line in block.lines) {
        val lineText = line.text
        val lineConfidence = line.confidence
        val lineLanguages = line.recognizedLanguages
        val lineCornerPoints = line.cornerPoints
        val lineFrame = line.boundingBox
        for (element in line.elements) {
            val elementText = element.text
            val elementConfidence = element.confidence
            val elementLanguages = element.recognizedLanguages
            val elementCornerPoints = element.cornerPoints
            val elementFrame = element.boundingBox
        }
    }
}
```

3.2 Розробка структурної схеми

На рис. 3.8 показано структурну схему системи розробленого програмного забезпечення.

КПЗ – 2023

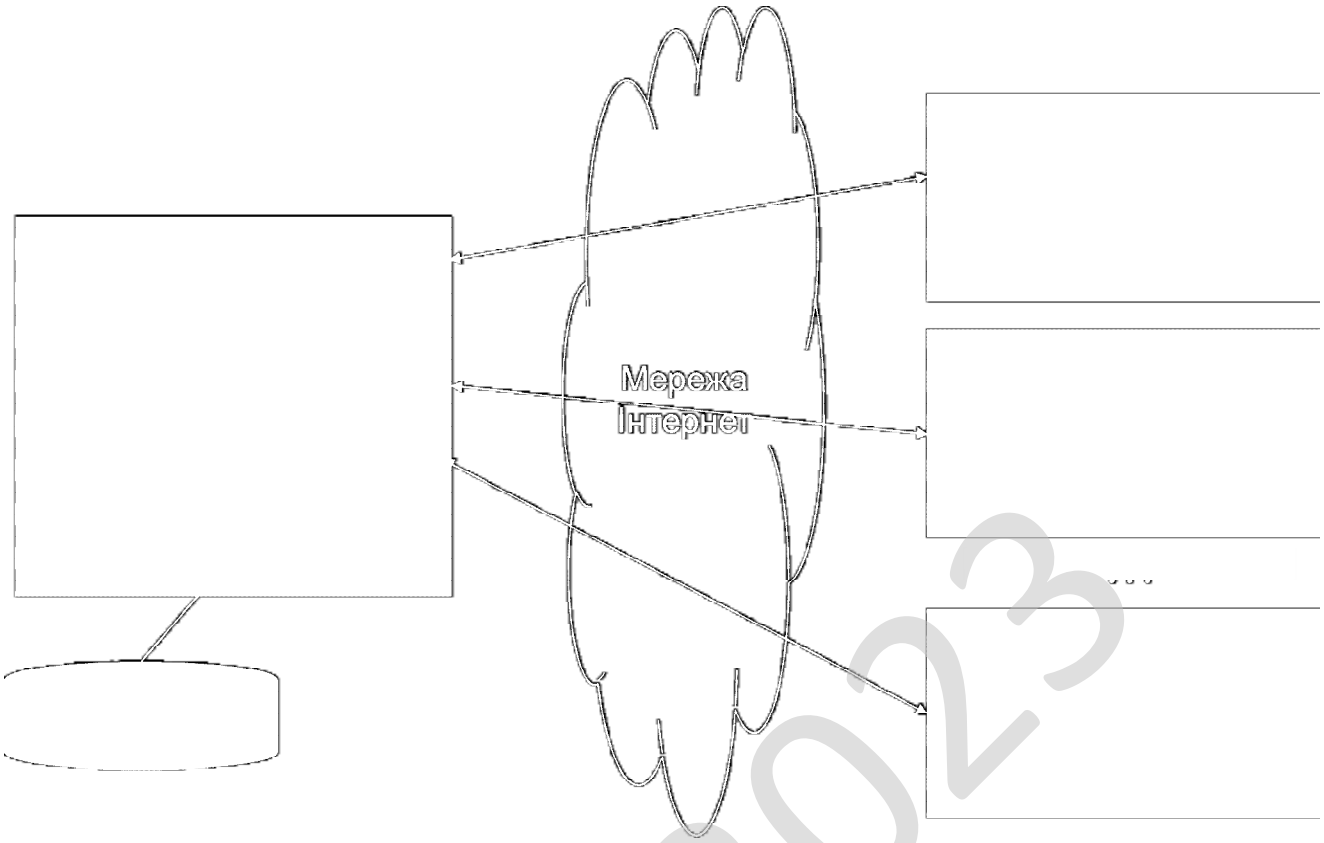


Рисунок 3.8 – Структурна схема системи

Як видно з рисунку, запропонована система складається з клієнтської та серверної частини.

На сервері знаходяться:

– Серверне програмне забезпечення, яке надає API-функції для роботи з нейронною мережею для клієнтських додатків, що дозволяє здійснювати розпізнавання номерів автомобілів на зображеннях.

– Штучну нейронну мережу для розпізнавання текстів на зображеннях, що пройшла процес навчання.

– Базу даних автомобілів, яка наповнена і оновлюється із відкритих державних баз даних.

На мобільних телефонах із встановленим додатком містяться:

– Клієнтське програмне забезпечення.

– Локальна база даних.

КПЗ – 2023

3.3 Розробка функціональної схеми

Основні блоки розробленого програмного забезпечення: app, data, domain. Решта модулів призначені для налаштування збирача, проекту тощо.

app – основний блок, що включає основні налаштування програми, а також реалізує "шар представлення (Presentation Layer)": Вікна, їх верстка, навігація між ними, відображення помилок і так далі.

data – блок, що відповідає в першу чергу за отримання даних із різних джерел та їх кешування. Він реалізується за рахунок патерну Repository.

domain – блок бізнес логіки. Саме до цього модуля звертається модуль подання для виконання запитів та отримання даних.

Блок app складається з наступних модулів:

– arch – набір базових та/або абстрактних класів для всього функціоналу, відповідає за роботу всього додатку вцілому. Так, наприклад, у файлі router модуля arch знаходиться код, необхідний для навігації між екранами додатку зі зберіганням минулого екрану в пам’яті (щоб повернутися на нього за кнопкою "назад") та анімацією переходу. Файл di забезпечує зв’язок всіх модулів та їх компонентів.

– common – набір кастомних класів-екстеншенів, які розширюють стандартний андроїдний ui функціонал.

– feature – безпосередньо основний код ui. Тут фрагменти (екрани) та їх в’ю-моделі. Кожна окрема папка – це окремий екран програми. Наприклад, у main зберігається MainActivity, це цілісне вікно програми, на якому відображаються всі інші.

У блоку data описані моделі та таблиця бази даних (entities), описана сама база даних (Room, заснована на SQLite, local), а також сполучна ланка модулів: repository і mapper. Репозиторій служить мостом для запитів, mapper конвертує дані.

Блок domain складається з модулів:

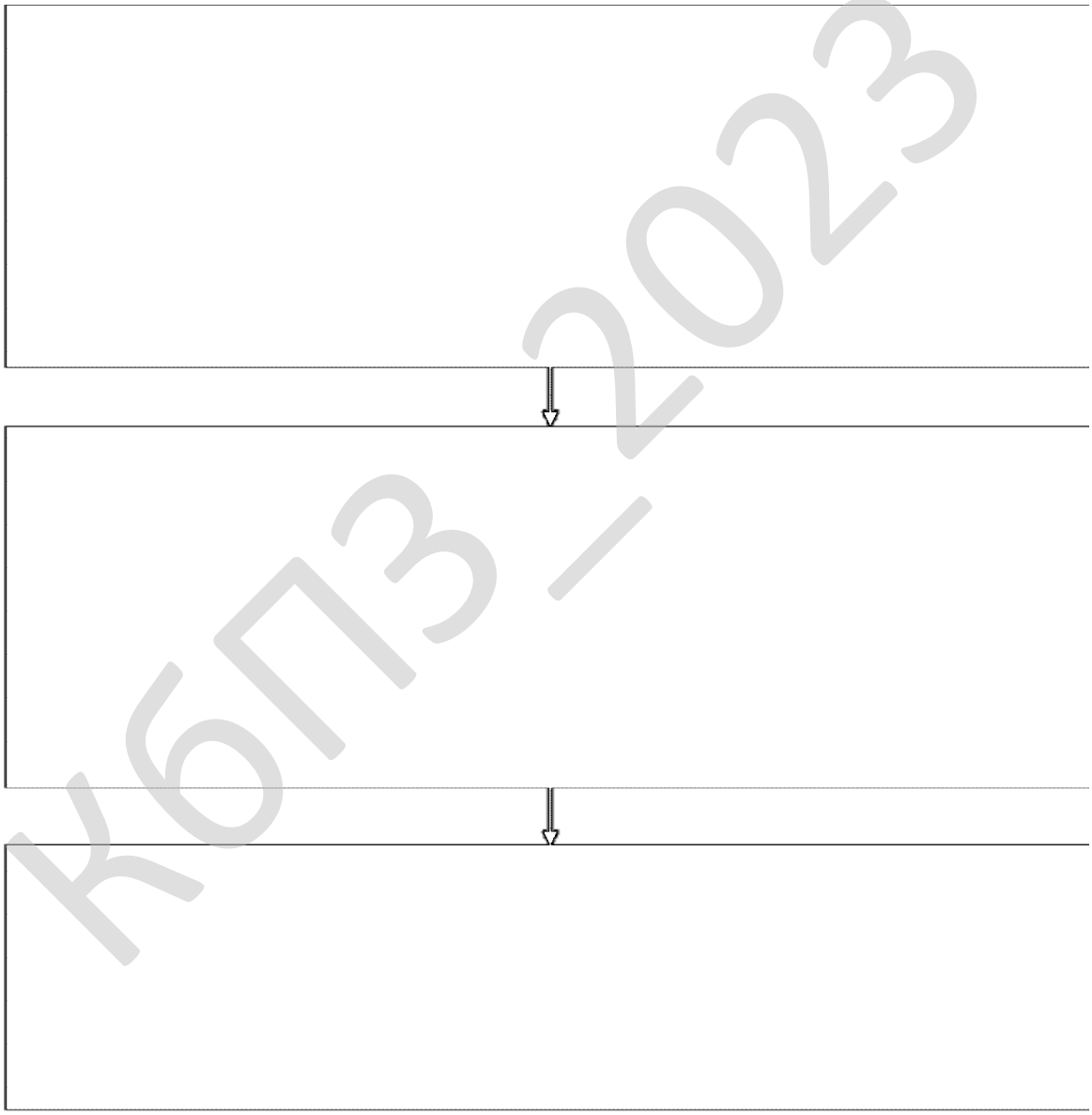
КПЗ – 2023

– entity – тут описані моделі у тому вигляді, в якому приходять із сервера.

– usecase – тут описані класи, які так само є частиною зв’язку модуля подання з бізнес-логікою.

У модулі navigation міститься "візуальне" подання в xml екранів, їх ієрархія та навігація.

Структурна схема розробленого програмного забезпечення зображена на рисунку 3.9.



КбПЗ_2023

Рисунок 3.9 – Структурна схема системи

3.4 Розробка діаграми процесів

На рисунку 3.10 зображена діаграма процесів, що описує взаємодію наявних у системі процесів.

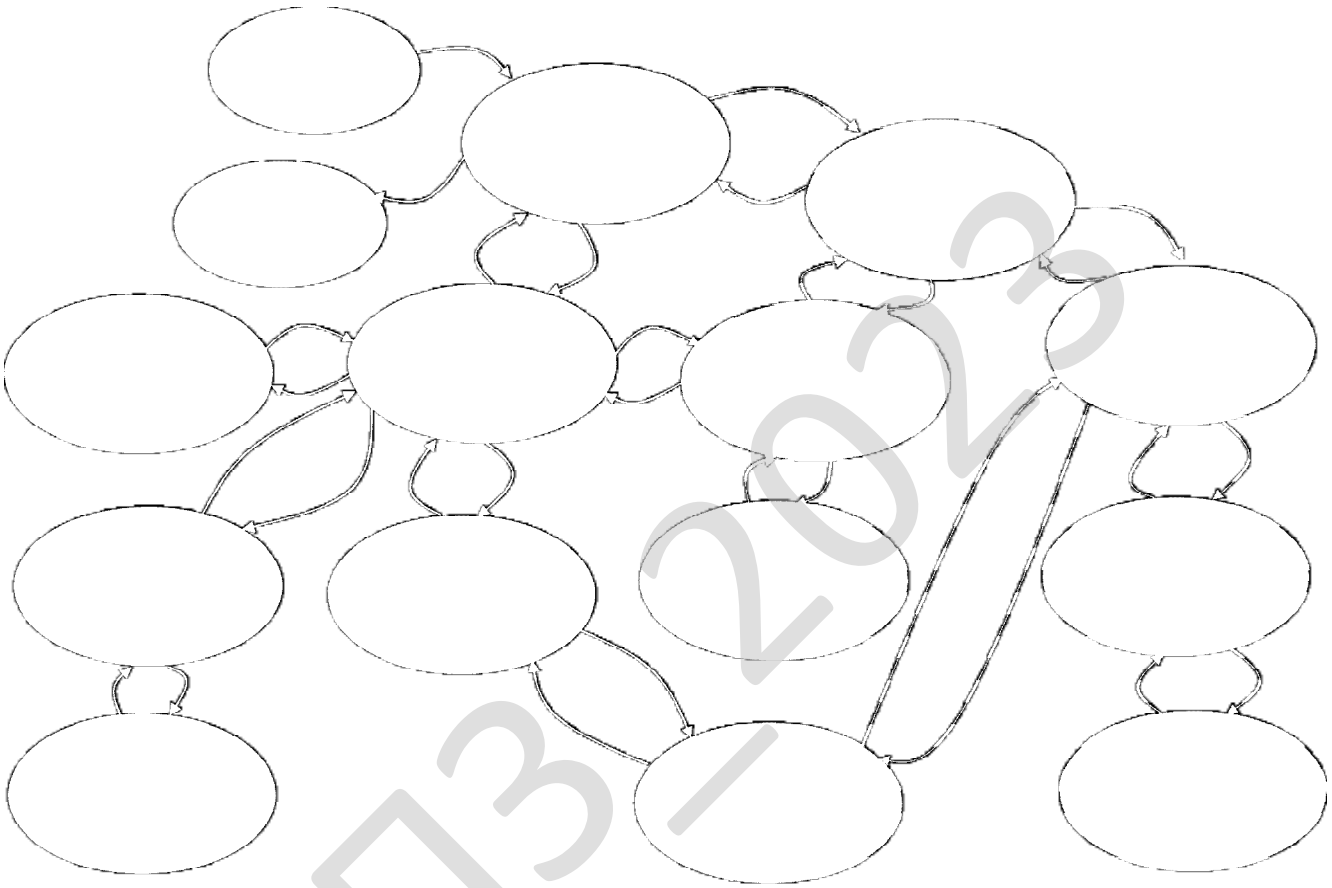


Рисунок 3.10 – Діаграма процесів системи

У розробленому програмному забезпеченні присутні наступні процеси:

- Головне вікно програми.
- Отримання зображення з камери мобільного пристрою.
- Ідентифікація автомобіля.
- Розпізнавання тексту на номерному знаку.
- Нейронна мережа.
- Пошук інформації про автомобіль.

КПЗ 2023

- Виведення інформації про автомобіль.
- Додавання інформації про автомобіль.
- База даних автомобілів.
- Перегляд та пошук.
- Редагування.
- Оновлення.
- Завантаження відкритих баз даних.

Таким чином опис функціонування системи, її структури, складових і процесів повністю виконані та готові до використання. Це дозволяє перейти від проектування до розробки програмного забезпечення.

КбПЗ_2023

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

На рис. 4.1 показана блок-схема основної програми.

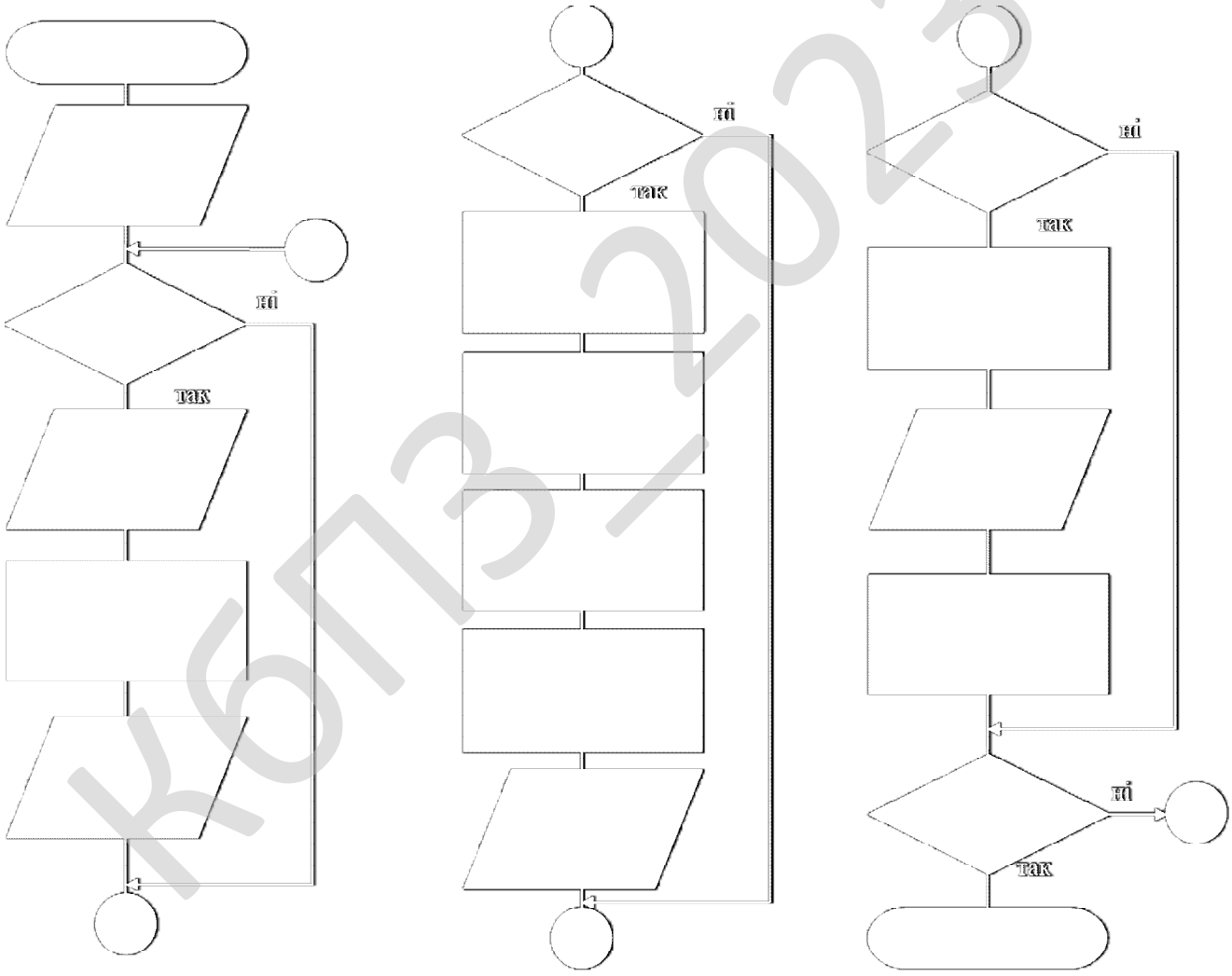


Рисунок 4.1 – Блок-схема основної програми

КПЗ 2023

Графічний інтерфейс користувача для додатку будувався за допомогою ієрархії View та ViewGroup об’єктів. View об’єкти це віджети інтерфейсу користувача, такі як кнопки або текстові поля і ViewGroup це невидимий вид контейнерів, які визначають розташування дочірніх уявлень, як наприклад, в сітці або вертикальному списку.

Android надає XML словник, який відповідає підкласам Viewі ViewGroup, так що можна визначити інтерфейс користувача в XML, використовуючи ієрархію елементів інтерфейсу користувача.

Оголошення макету інтерфейсу користувача в XML, а не під час виконання коду, корисне з кількох причин, але особливо важливо, що таким чином можна створювати різні макети для різних розмірів екрану. Наприклад, можна створити дві версії макету і вказати системі використовувати один на "малих" екранах, а інший на "великих" екранах.

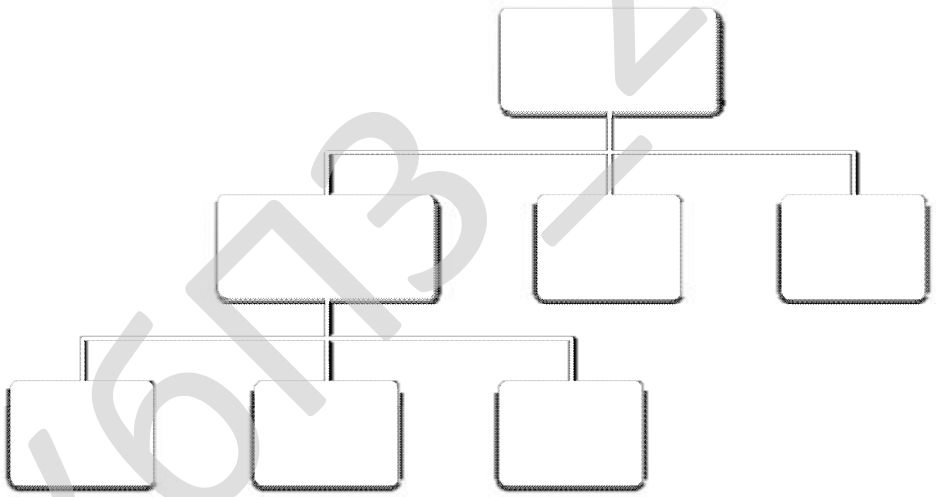


Рисунок 4.3 – Ілюстрація того, як ViewGroup об’єкти утворюють гілки в макеті та містять інші View об’єкти

LinearLayout є групою уявлень (підклас ViewGroup), який розкладає дочірні уявлення у вертикальному або горизонтальному положенні, як зазначено в android:orientation атрибуті. Кожен дочірній елемент LinearLayout з’являється на

КБПЗ-2023

екрані в порядку, в якому він з’являється в XML.

Два інших атрибути, android:layout_width і android:layout_height, потрібні для всіх уявлень, щоб вказати їх розмір.

LinearLayout є коренем у макеті, він повинен заповнити всю область екрана, яка доступна для програми, встановивши ширину та висоту в "match_parent". Це значення вказує, що уявлення має розширити свою ширину чи висоту до відповідності ширині чи висоті батьківського уявлення.

Щоб створити текстове поле, яке можна редагувати, необхідно додати <EditText> елемент усередині <LinearLayout>. Як і будь-якому View об’єкту, необхідно вказати певні XML атрибути для вказівки EditText властивостей об’єкта.

Об’єкт ресурсу це унікальне ціле ім’я, яке асоціюється з ресурсом програми, таким як растрове зображення, файл макета або рядки. Кожен ресурс має відповідний об’єкт ресурсу, визначений у проекті.

Коли потрібно додати текст до інтерфейсу користувача, треба вказувати кожен рядок як ресурс. Рядкові ресурси дозволяють керувати всім текстом інтерфейсу користувача в одному місці, що дозволяє його легше знайти і оновити. Використання зовнішніх ресурсів для рядків також дозволяє локалізувати програму під різні мови, надаючи альтернативні визначення для кожного рядкового ресурсу.

Кнопка – один із найпоширеніших елементів управління у програмуванні. Наслідує TextView і є базовим класом для класу CompoundButton. Від класу CompoundButton у свою чергу успадковуються такі елементи як CheckBox, ToggleButton та RadioButton. В Android для кнопки використовується клас android.widget.Button. На кнопці знаходиться текст і на кнопку потрібно натиснути, щоб отримати результат. Альтернативою їй може бути компонент ImageButton (android.widget.ImageButton), у якого замість тексту використовується зображення.

КОПЗ-2023

У Android студії кнопка представлена компонентом Button у розділі Widgets. Керувати розміром шрифту, кольором тексту та іншими властивостями можна через атрибут textAppearance, який використовує системні стилі. Випадаючий список цього типу містить великий перелік варіантів. Також можна вручну встановити конкретні індивідуальні налаштування через окремі властивості.

Якщо розтягувати кнопку по всій ширині екрана (android:layout_width="match_parent"), то додатково бажано використовувати атрибут android:layout_margin (або споріднені з ним layout_marginRight і layout_marginLeft) для створення відступів від країв екрана (веб-майстри) знайомі.

Так як кнопка є спадкоємцем TextView, то використовує багато таких же атрибутів: textColor , textSize та ін.

Існує три способи обробки подій натискань на кнопку.

– метод setOnClickListener().

– метод setOnClickListener().

– інтерфейс OnClickListener.

Більш традиційний спосіб управління кнопкою – через метод setOnClickListener(), який прослуховує натискання на кнопку.

Для роботи з камерою мобільного телефону є інтерфейс камери, що використовується для керування потоком даних для випадків використання, керування камерою за допомогою CameraControl, а також публікації стану камери за допомогою CameraInfo.

fun getCameraControl(): CameraControl – повертає CameraControl для Camera. Забезпечує CameraControl різні асинхронні операції, такі як масштабування, фокусування та вимірювання. CameraControl готовий розпочати роботу одразу після прив’язки варіантів використання до Camera. Коли всі UseCases розблоковані, або коли камера закривається, або закривається через життєвий цикл onStop, буде CameraControlвідхилено всі операції.

КОПЗ – 2023

Кожен метод CameraControlповертає, ListenableFutureякі програми можуть використовувати для перевірки асинхронного результату. Якщо операція не дозволена в поточному стані, повернута помилка ListenableFuture негайно завершиться з CameraControl.OperationCanceledException.

fun getCameraInfo(): CameraInfo – Повертає інформацію про цю камеру. Повернену інформацію можна використовувати для запиту статичних характеристик камери або спостереження за станом роботи камери.

Блок-схема роботи підпрограми розпізнавання номеру автомобіля на зображенні представлена на рис. 4.2.

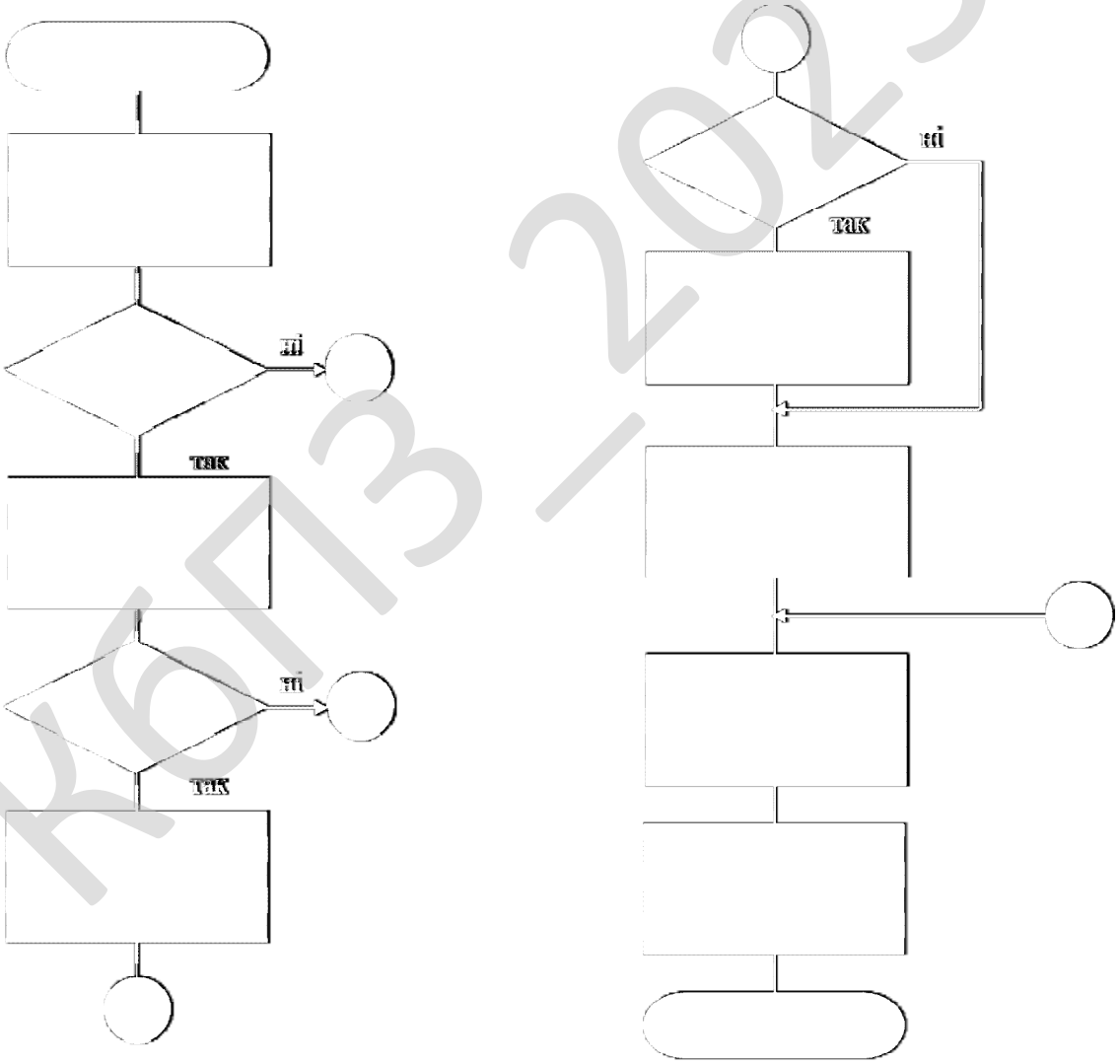


Рисунок 4.2 – Блок-схема роботи підпрограми розпізнавання номеру автомобіля на зображенні

КбПЗ – 2023

Розпізнавання номера авто на фото реалізовано наступним чином:

```
package com.gahov.plates_recognition.feature.search


import android.annotation.SuppressLint
import android.content.Context
import android.net.Uri
import android.os.Environment
import com.gahov.domain.entity.cars.CarEntity
import com.gahov.domain.entity.common.Either
import com.gahov.domain.entity.failure.Failure
import com.gahov.domain.usecase.carplate.GetRemoteCarInfoUseCase
import com.gahov.domain.usecase.carplate.params.GetCarParams
import com.gahov.plates_recognition.arch.controller.BaseViewModel
import com.gahov.plates_recognition.feature.search.command.CarSearchCommand
import com.google.firebase.ml.vision.FirebaseVision
import com.google.firebase.ml.vision.common.FirebaseVisionImage
import com.google.firebase.ml.vision.text.FirebaseVisionText
import java.io.File
import java.io.IOException
import java.text.SimpleDateFormat
import java.util.Date
import javax.inject.Inject

@Suppress("DEPRECATION")
class CarSearchViewModel @Inject constructor(
    private val loadCarInfoUseCase: GetRemoteCarInfoUseCase
) : BaseViewModel() {

    private fun onResultFailure(failureResult: Failure) {
        handleCommand(CarSearchCommand.OnNetworkError(failureResult))
        handleFailure(failureResult)
    }

    fun processImage(context: Context, photoUri: Uri) {
        val image = FirebaseVisionImage.fromFilePath(context, photoUri)
        val detector = FirebaseVision.getInstance().onDeviceTextRecognizer

        detector.processImage(image)
            .addOnSuccessListener { firebaseVisionText ->
                processResultText(firebaseVisionText)
```

КПЗ – 2023

```
            }
            .addOnFailureListener {
                handleFailure(Failure.Common(Throwable(message =
PICTURE_HANDLING_ERROR)))
            }
    }

    private fun processResultText(resultText: FirebaseVisionText) {
        if (resultText.textBlocks.size == EMPTY_TEXT) {
            handleFailure(Failure.Common(Throwable(message =
TEXT_HANDLING_ERROR)))
            return
        }
        for (block in resultText.textBlocks) {
            val blockLine = block.lines
            for (i in blockLine) {
                var carPlate = i.text.replace("\\s".toRegex(), "")
                if (carPlate.length == MAX_POSSIBLE_CAR_DIGITS_LENGTH) {
                    carPlate.submitRecognitionAndSearch()
                } else {
                    carPlate = removeExtraSymbolsInDigits(carPlate)
                    if (carPlate.length == MAX_POSSIBLE_CAR_DIGITS_LENGTH)
{
                        carPlate.submitRecognitionAndSearch()
                    } else {
                        handleFailure(Failure.Common(Throwable(message =
WRONG_PLATED_INPUT_ERROR)))
                    }
                }
            }
        }
    }

    private fun String.submitRecognitionAndSearch() {
        CarSearchCommand.OnRecognitionResult(digits = this)
        searchCarInputDigits(GetCarParams(this))
    }

    private fun removeExtraSymbolsInDigits(fullCarDigits: String): String {
        var carPlate = fullCarDigits
```

КПЗ_2023

```
        if (fullCarDigits.length == FULL_CAR_DIGITS_LENGTH &&
fullCarDigits.contains(UA_SYMBOL)) {
            carPlate = fullCarDigits.replace(UA_SYMBOL, "")
        }
        return carPlate
    }

    fun onNewCarDigitsInput(carDigits: String) {
        if ((carDigits.chars()).count().toInt() ==
MAX_POSSIBLE_CAR_DIGITS_LENGTH) {
            searchCarInputDigits(GetCarParams(carDigits))
        }
    }

    private fun searchCarInputDigits(param: GetCarParams) {
        launch {
            when (val result = loadCarInfoUseCase.execute(param = param)) {
                is Either.Right -> onResultSuccess(result = result.success)
                is Either.Left -> onResultFailure(result.failure)
            }
        }
    }

    private fun onResultSuccess(result: CarEntity) {
        navigateDirection(

CarSearchBottomDialogFragmentDirections.actionCarSearchToCarDetails(
                carData = result,
                carDigits = result.digits
            )
        )
    }

    @SuppressLint("SimpleDateFormat")
    @Throws(IOException::class)
    fun createImageFile(context: Context): File {
        val timeStamp: String =
SimpleDateFormat(IMAGE_DATE_FORMAT).format(Date())
        val storageDir: File? =
context.getExternalFilesDir(Environment.DIRECTORY_PICTURES)
        return File.createTempFile(
            JPEG_PREFIX + "${timeStamp}_", JPEG_SUFFIX, storageDir
```

КОПІЯ 2023

```
            ).apply {
                absolutePath
            }
        }

        companion object {
            const val REQUEST_IMAGE = 100
            const val STORAGE = 1
            const val FLOAT = "%.2f"
            const val LOAD_DELAY = 1500L
            const val AUTHORITY = "com.gahov.plates_recognition.provider"
            private const val JPEG_SUFFIX = ".jpg"
            private const val JPEG_PREFIX = "JPEG_"
            private const val MAX_POSSIBLE_CAR_DIGITS_LENGTH = 8
            private const val FULL_CAR_DIGITS_LENGTH = 10
            private const val EMPTY_TEXT = 0
            private const val PICTURE_HANDLING_ERROR = "Помилка обробки
зображення!"
            private const val TEXT_HANDLING_ERROR = "Помилка обробки номеру!"
            private const val WRONG_PLATED_INPUT_ERROR = "Невірний номер!"
            private const val IMAGE_DATE_FORMAT = "yyyyyMMdd_HHmmss"
            private const val UA_SYMBOL = "UA"
```

Безпосередньо взаємодія з нейронною мережею відбувається у:

com.gahov.plates_recognition.feature.search.CarSearchViewModel

У функції processImage(context: Context, photoUri: Uri) із зображення дістається текст.

У функції processResultText(resultText: FirebaseVisionText) з отриманого тексту визначаються критерії номера автомобіля.

Функція requestCameraIntent() запускає флоу з камерою. Взаємодію з камерою реалізовано наступним чином:

```
package com.gahov.plates_recognition.feature.search
import android.annotation.SuppressLint
import android.content.Intent
import android.net.Uri
import android.os.Bundle
import android.os.Handler
import android.provider.MediaStore
```

КПЗ-2023

```
import android.view.View
import androidx.core.content.FileProvider
import androidx.core.view.isVisible
import androidx.fragment.app.DialogFragment
import com.gahov.domain.entity.failure.Failure
import com.gahov.plates_recognition.R
import com.gahov.plates_recognition.arch.router.command.Command
import com.gahov.plates_recognition.arch.ui.dialog.BaseBottomSheetFragment
import com.gahov.plates_recognition.databinding.FragmentCarSearchBinding
import com.gahov.plates_recognition.feature.main.MainActivity
import
com.gahov.plates_recognition.feature.search.CarSearchViewModel.Companion.AU
THORITY
import
com.gahov.plates_recognition.feature.search.CarSearchViewModel.Companion.LO
AD_DELAY
import
com.gahov.plates_recognition.feature.search.CarSearchViewModel.Companion.RE
QUEST_IMAGE
import com.gahov.plates_recognition.feature.search.command.CarSearchCommand
import com.permissionx.guolindev.PermissionX
import dagger.hilt.android.AndroidEntryPoint
import java.io.File
import java.io.IOException

@Suppress("DEPRECATION")
@AndroidEntryPoint
class CarSearchBottomDialogFragment :
    BaseBottomSheetFragment<CarSearchViewModel, FragmentCarSearchBinding>(
        layoutId = R.layout.fragment_car_search,
        viewModelClass = CarSearchViewModel::class.java
    ) {

    private var takePictureDestinationFile: File? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setStyle(DialogFragment.STYLE_NO_FRAME,
R.style.ThemeOverlay_Material3_BottomSheetDialog)
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
```

КПЗ_2023

```
        super.onViewCreated(view, savedInstanceState)
        setClickListeners()
    }

    private fun setClickListeners() {
        binding.searchCameraButton.setOnClickListener {
            dispatchTakePictureIntent()
        }
        binding.searchButton.setOnClickListener {
            onLoading(true)

viewModel.onNewCarDigitsInput(binding.numberInput.text.toString())
        }
    }

    private fun onLoading(isLoading: Boolean) {
        binding.searchProgressBar.isVisible = isLoading
    }

    override fun handleFeatureCommand(command: Command.FeatureCommand) {
        with(command) {
            if (this is CarSearchCommand) {
                when (this) {
                    is CarSearchCommand.OnNetworkError ->
displayError(failure)
                    is CarSearchCommand.OnRecognitionResult ->
displayRecognizedPlates(digits)
                }
            } else {
                super.handleFeatureCommand(command)
            }
        }
    }

    private fun displayRecognizedPlates(digits: String) {
        onLoading(false)
        binding.numberInput.setText(digits)
    }

    private fun dispatchTakePictureIntent() {
        PermissionX.init(activity as MainActivity)
            .permissions(listOf(android.Manifest.permission.CAMERA))
```

КПЗ_2023

```
                .request { allGranted, _, _ ->
                    if (allGranted) {
                        requestCameraIntent()
                    } else {
                        viewModel.handleFailure(Failure.Common())
                    }
                }
        }

        @SuppressLint("QueryPermissionsNeeded")
        private fun requestCameraIntent() {
            Intent(MediaStore.ACTION_IMAGE_CAPTURE).also { takePictureIntent ->

takePictureIntent.resolveActivity(requireContext().packageManager)?.also {
                    // Create the File where the photo should go
                    takePictureDestinationFile = try {
                        onLoading(true)
                        viewModel.createImageFile(requireContext())
                    } catch (ex: IOException) {
                        viewModel.handleFailure(Failure.Common())
                        null
                    }
                    // Continue only if the File was successfully created
                    takePictureDestinationFile?.also {
                        val photoURI: Uri = FileProvider.getUriForFile(
                            requireContext(), AUTHORITY, it
                        )
                        takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT,
photoURI)
                        startActivityForResult(takePictureIntent,
REQUEST_IMAGE)
                    }
                }
            }
        }

        @Deprecated("Deprecated in Java")
        override fun onActivityResult(
            requestCode: Int,
            resultCode: Int,
            data: Intent?
        ) {
```

КБПЗ-2023

```
            super.onActivityResult(requestCode, resultCode, data)
            binding.numberInput.setText("")

            Handler().postDelayed({
                val photoURI: Uri = takePictureDestinationFile?.let {
                    FileProvider.getUriForFile(
                        requireContext(), AUTHORITY, it
                    )
                }!!
                photoURI.let { viewModel.processImage(requireContext(), it) }
            }, LOAD_DELAY)
        }
    }
```

4.2 Захист розробленого програмного забезпечення

Для інформаційного захисту розробленого програмного забезпечення використано засоби безпеки Android, а саме модуль безпеки Android Keystore.

Android Keystore – це компонент безпеки на платформі Android, який призначений для зберігання і керування ключами та іншою чутливою інформацією в безпечному середовищі. Цей інструмент важливий для забезпечення безпеки додатків та даних на Android-пристроях. Android Keystore надає доступ до апаратних модулів безпеки, які доступні на деяких Android-пристроях.

Основні функції Android Keystore включають:

– Зберігання ключів. Android Keystore забезпечує можливість зберігати криптографічні ключі в безпечному середовищі. Це можуть бути ключі шифрування, ключі ідентифікації або інші чутливі дані.

– Операції з ключами. Модна використовувати Android Keystore для виконання операцій з ключами, таких як генерація ключів, підпис даних, шифрування та розшифрування.

– Автоматичний апаратний захист. Якщо пристрій має апаратний модуль безпеки, Android Keystore автоматично використовує його для зберігання та

КБПЗ-2023

операцій з ключами, що забезпечує вищий рівень захисту.

– Підтримка відновлення даних. Можна налаштувати Android Keystore для відновлення ключів після скидання пристрою до заводських налаштувань або інших подібних подій.

Можливі дії з Android Keystore:

1. Створення ключів. Для створення ключів необхідно використовувати KeyGenerator або KeyPairGenerator для симетричних або асиметричних ключів відповідно.

2. Збереження ключів. Після створення ключів вони зберігаються автоматично в Android Keystore.

3. Отримання ключів. Щоб отримати ключі з Android Keystore, слід використовувати KeyStore та відповідні API для отримання доступу до ключів.

4. Виконання операцій з ключами. Модна виконувати операції з ключами, такі як підпис, шифрування тощо, за допомогою ключів із Android Keystore.

Android Keystore автоматично захищає доступ до ключів і використовує систему автентифікації пристрою, таку як PIN-код, пароль або біометричні дані.

Також Android Keystore надає можливість оновлення ключів та відновлення їх у випадку втрати.

Загалом, Android Keystore є потужним інструментом для забезпечення безпеки ключів та інших чутливих даних в додатках для Android. Він дозволяє забезпечити високий рівень захисту програмного забезпечення.

КОПЗ – 2023

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Для роботи розробленого програмного забезпечення необхідно встановити його на мобільний телефон з операційною системою Android.

На рис. 5.1-5.3 показані приклади вигляду вікон розробленого програмного забезпечення.

Рисунок 5.1 – База даних авто у додатку

КПІ 2023

Для розпізнавання номера авто необхідно здійснити наступні дії:

- Натиснути кнопку "Відкрити камеру".

- Дати дозвіл на використання камери додатком.

- Після відкриття камери зробити фото номера машини та натиснути кнопку "ОК".

Рисунок 5.2 – Пошук авто за номером

КПЗ – 2023

Після цих дій створюється директорія під новий файл для фотографії. Фотографія зберігається в цей файл, програма отримує до неї шлях. Нейронна мережа розпізнає текст на зображенні. Результат парситься і розбивається на блоки. Якщо щось із результатів розпізнавання відповідає критеріям пошуку номеру авто серед тексту, це визначається як номер, повертається у вигляді "усвідомленої" моделі та відправляється на сервер, щоб отримати інформацію за цим номером.

Рисунок 5.3 – Перегляд інформації про знайдене авто

КПЗ-2023

6 НАУКОВА НОВИЗНА

У магістерській роботі розроблено програмне забезпечення, яке призначено для системи розпізнавання та пошуку інформації за номером автомобіля на базі нейронної мережі.

Метою роботи є дослідження та програмна реалізація системи розпізнавання та пошуку інформації за номером автомобіля на базі нейронної мережі.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Дослідження існуючих систем для розпізнавання та пошуку інформації за номером автомобіля.

– Розробка методів та алгоритмів системи для розпізнавання та пошуку інформації за номером автомобіля на базі нейронної мережі.

– Програмна реалізація системи розпізнавання та пошуку інформації за номером автомобіля на базі нейронної мережі.

Об'єктом дослідження є процес ідентифікації автомобіля за зображенням з камери мобільного телефону.

Предметом дослідження є методи побудови нейронних мереж для розпізнавання зображень і текстів та методи розробки мобільних додатків.

Методи дослідження базуються на теорії алгоритмів та структур даних, теорії штучного інтелекту, теорії інженерії програмного забезпечення та теорії об'єктно-орієнтованого програмування.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

1. Запропоновано метод розпізнавання номера автомобіля на зображенні, отриманому з камери мобільного телефону, з використанням нейронних мереж.

КПЗ 2023

2. Розроблено вітчизняний продукт системи розпізнавання та пошуку інформації за номером автомобіля на базі нейронної мережі, який має більш широкі можливості, на відміну від існуючих аналогів та може бути використаний при ідентифікації автомобіля на фото та відео.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі ідентифікації автомобіля за зображенням з камери мобільного телефону та отримання інформації про нього з відкритих баз даних за його номером.

КбПЗ_2023

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

7.1 Техніко-економічне обґрунтування теми дипломного проекту

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 24 днів (один місяць).

В магістерській роботі було проведено дослідження та виконана програмна реалізація системи розпізнавання та пошуку інформації за номером автомобіля на базі нейронної мережі.

Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликий розмір;

б) невеликі системні потреби;

в) незалежність від встановлених на комп'ютері баз даних;

г) зручність у користуванні та надійність.

Таблиця 7.1 – Початкові дані

Показники		Позначення	Характеристика або величина
1		2	3
1.	Кількість розроблених програм період, шт.	N	1
2.	Кількість екземплярів програм, шт.	Ne	30
3.	Запланований термін розробки, днів	Fpq	24 (1 місяць)
4.	Група задачі підсистеми управління (1-6)	–	1
5.	Ступінь новизни задачі (А, Б, В, Г)	–	Г
6.	Складність алгоритму (1, 2, 3)	–	2

КОПІЯ – 2023

Продовження таблиці 7.1

1	2	3
7. Кількість макетів вхідної інформації	–	8
8. Кількість форм вихідної інформації.	–	6
9. Мова програмування (1-6)	–	6
10. Попередній досвід (1-6)	–	3
11. Гнучкість проекту ПП (1-6)	–	3
12. Детальність проекту ПП (1-6)	–	1
13. Рівень спрацьованості колективу (1-6)	–	2
14. Ступінь вимірності процесів (1-6)	–	3
15. Необхідна надійність програмного забезпечення (1-6)	–	3
16. Розмір бази даних (порівняно з розміром програми) (1-6)	–	4
17. Складність кінцевого програмного продукту (1-6)	–	5
18. Необхідний рівень забезпечення повторного використання (1-6)	–	2
19. Документованість відповідно до планованого життєвого циклу (1-6)	–	3
20. Вимоги до швидкодії ПП (1-6)	–	3
21. Обмеження на розміри основного сховища даних (1-6)	–	2
22. Різноманітність використовуваних обчислювальних платформ (1-6)	–	4
23. Професійний рівень аналітиків (1-6)	–	3
24. Професійний рівень програмістів (1-6)	–	4
25. Постійність складу команди розробників (1-6)	–	2
26. Досвід розробки додатків (1-6)	–	1
27. Досвід роботи з обчислювальною платформою (1-6)	–	2

КОПЗ – 2023

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	3
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	30000
33. Норматив додаткової зарплати, % :	$Н_Д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	40
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

Значення трудомісткості розробки програмного забезпечення для стадій ТЗ, ЕК, ТП та ВП визначаємо по типовим нормам часу приведеним в додатках МВ. Стадія РП є найбільш тривалою і трудомісткою, що робить значний вплив на інші стадії проекту.

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

$$T_{ном} = A \text{ Size}^B, \quad (7.1)$$

де: A – коефіцієнт Боема, A = 2,45;

КПЗ – 2023

Size – загальний об'єм відлагодженого програмного коду, тис. рядків;

B – показник ступеня, що визначається співвідношенням:

$$B = 1{,}01 + 0{,}001\sum W_i, \quad (7.2)$$

де: *Wi* – сумарне значення п'яти показників (МВ, додаток 2), що відображають особливості розробки проекту програмного продукту (ПП) і колективу розробників.

$$B = 1{,}01+0{,}001(2{,}43+3{,}64+4{,}22+3{,}95+2{,}73) = 1{,}027.$$

$$T_{ном} = 2{,}45 \cdot 2{,}2^{1{,}027} = 5{,}5 \text{ люд-міс.}$$

Визначаємо уточнені (з урахуванням приведених в МВ додатку 3 сімнадцяти додаткових коефіцієнтів) трудовитрати, люд-міс.:

$$T_{уточн} = T_{ном}\, ПВ_j, \quad (7.3)$$

де: $ПВ_j$ – добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн} = 5{,}5 \cdot (1 \cdot 1{,}09 \cdot 1{,}30 \cdot 0{,}91 \cdot 1 \cdot 1 \cdot 1 \cdot 1{,}15 \cdot 1 \cdot 0{,}87 \cdot 1{,}10 \cdot$$
$$\cdot 1{,}22 \cdot 1{,}12 \cdot 1{,}10 \cdot 1 \cdot 1 \cdot 1{,}10) = 12{,}9 \text{ люд-міс.}$$

Ці коефіцієнти дозволяють диференційовано оцінювати результати роботи програмістів, беручи до уваги швидкодію програми, використання різноманітних обчислювальних платформ і інструментів розробки, взаємодію декількох серверів, вимоги до об'ємів баз даних і ін.

Визначаємо підсумкові трудовитрати по стадії робочий проект, люд-дні:

$$T_{РП} = 0{,}3 C T_{уточн}^{0{,}33+0{,}2(B-1{,}01)} S, \quad (7.4)$$

де: *C* – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4);

S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПЗ згідно встановленим вимогам. Вибираємо в межах (25…350)%.

$$T_{РП} = 0{,}3 \cdot 2{,}75 \cdot 12{,}9^{0{,}33+0{,}2(1{,}027-1{,}01)} \cdot 58 = 112 \text{ люд/день.}$$

КПІ-2023

Для зручності визначення загальної трудомісткості на розробку програмного забезпечення результати розрахунків по стадіям зводимо до таблиці 7.2.

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	**Трудомісткість за типовими нормами та розрахунками**	
	Величина, люд/дні	**Підстава**
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	15	Д7
Робочий проект	112	Ф 7.1-7.4
Впровадження	15	Д13
Всього	161	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

Чисельність ставок інженерів-програмістів для розробки програмного забезпечення визначається за формулою:

$$Ч = \frac{T_{пз} N}{F_{рq} - H_{ев}}, \quad (7.5)$$

де: $F_{рq}$ – плановий фонд робочого часу одного спеціаліста, днів;

$T_{пз}$ – трудомісткість розробки програмного забезпечення люд-дні.

$$Ч = \frac{161 \cdot 1}{24\text{-}3} = 7,7 \text{ ставки.}$$

Чисельність інженерів-електронщиків для проведення технічного обслуговування та ремонту комп'ютерних мереж визначається в залежності від наявності технічних засобів і норм витрат часу на виконання профілактичних робіт на протязі року.

КПІ – 2023

Визначаємо затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за період розробки. Результати розрахунку зводимо до таблиці 7.3.

Таблиця 7.3 – Затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за розрахунковий період

Найменування обладнання	**Профілактичне обслуговування**			
	Кількість хв. на один. обл.	**Кількість облад-нання**	**Затрати часу в хв.**	**Затрати часу в год.**
Системний блок ПК	90	10	900	15
Монітор	60	10	600	10
Клавіатура	30	10	300	5
Маніпулятор «мишка»	30	10	300	5
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	1	120	2
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор–маршрутизатор	30	2	60	1
Кабельні господарства ЛВС на 1 м. п.	2,5	150	375	6,25
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	47,91

Час на профілактику обладнання в загальному балансі робочого часу інженерів-електронщиків неповинен складати більше 10%.

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$\Phi_{др}^{с} = \frac{З_ч \cdot n_{міс}}{1{,}2}, \qquad (7.6)$$

КбПЗ–2023

$$Ф_{др}^{с} = \frac{48 \cdot 1}{1,2} = 40 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = \frac{Ф_{др}^{с}}{F_{др} \cdot T_{зм}}, \quad (7.7)$$

$$Ч_{ел} = 40/(24 \cdot 8) = 0,2 \text{ ставки.}$$

Для забезпечення нормального технічного обслуговування засобів ТО та мереж, необхідно прийняти найбільше ціле значення розрахункової чисельності інженерів-електронщиків. Чисельність інженерів-системотехніків, адміністраторів мережі, дизайнерів WEB вузлів, системних програмістів (аналітиків), бухгалтерів-економістів визначається за потребою в залежності від функціональних обов'язків. Після визначення чисельності персоналу складається штатний розклад.

Таблиця 7.4 – Розрахунок чисельності штатного персоналу сектору системного та адміністративного обслуговування засобів ОТ та комп'ютерних мереж

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server 2019, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN, PPPoE, Frame Relay, Wi-Fi	1	0,5
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	1	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	1	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	1	
Всього		4	

КПІ – 2023

Продовження таблиці 7.4

Посада	Вид роботи	Час	К-ть штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	0,25	0,25
	Підтримка постійних клієнтів	0,5	
	Оформлення договорів, ведення тендерів	0,5	
	Контроль взаєморозрахунків з постачальниками	0,75	
Всього		2	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	1	0,5
	Створення графічних і стилістичних елементів сайту	1	
	Оформлення банерів і промо-сторінок	1	
	Розміщення графіки і контенту на Інтернет сторінках	1	
Всього		4	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	1	0,5
	Верстка друкованих видань	1	
	Додрукова підготовка макетів	1	
	Розміщення графіки і контенту на Інтернет сторінках	1	
Всього		4	

КПЗ – 2023

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (IT-менеджер)	1	8385	8385
Продакт-менеджер	0,25	8500	2125
Інженер-програміст	7,7	8780	67606
Інженер-електронщик	0,2	7000	1400
Інженер-системотехнік	0,5	7000	3500
Адміністратор мережі	0,5	8000	4000
Дизайнер WEB	0,5	10000	5000
Всього за період розробки	$R_{сп}$ = 10,65	-	$Ф_{роб}$ = 92016

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рд}}, \quad (7.8)$$

де: $Ф_{роб}$ – загальна сума зарплати за плановий період, грн.

$$З_{сд} = \frac{92016}{10,65 \cdot 24} = 360 \ грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

Балансова вартість будівель визначається з урахуванням кількості робочих місць виконавців, питомої площі на одне робоче місце, та вартості одного квадратного метра виробничої площі:

$$Б_{уд} = R^1_{сп} S_у Ц_{пл}, \quad (7.9)$$

де: $R^1_{сп}$ – кількість робочих місць виконавців, шт. Приймаємо 13 робочих місць;

$S_у$ – питома площа на одне робоче місце, $м^2$;

КПЗ 2023

$Ц_{пл}$ – вартість одного квадратного метра площі, грн.

Згідно даних інтернет ресурсу DOM.RIA (https://dom.ria.com) ціна одного квадратного метра площі, вік якої не перевищує 30 років, по місту складає 500…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 38 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На кожне робоче місце у середньому потрібно 8 $м^2$. З урахуванням цього:

$$Б_{уд} = 13 \cdot 8 \cdot 20000 = 2080000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 208000 грн. Балансова вартість інвентарю розраховується за нормою 3500 грн. на одне робоче місце. Тобто:

$$I_{нв} = R^1_{сп} \cdot Ц_м, \qquad (7.10)$$

де: $Ц_м$ – ціна меблів для одного робочого місця, грн.

$$I_{нв} = 13 \cdot 3500 = 45500 \text{ грн.}$$

Балансова вартість обчислювальної техніки визначається по оптовим цінам постачальника з врахуванням витрат на транспортування.

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу Інтернет магазину Компбест за 14.10.23 – джерело https://compbest.com.ua.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп'ютер		11457
Системний блок		7509
Процесор	Intel Core i7-4790 (4(8) ядра по 3.6 - 4.0 GHz); Cache Memory 8 MB	–

КПІ – 2023

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Системна плата	1st Player ATX NEW	–
Відеокарта	PCIeX: ATI HD5670 SAPPHIRE 1024MB/128bit/DDR3/TV/DualDVI	–
Жорсткий диск	120Gb SSD+ HDD: 500 Gb 7200 Serial ATA WD 32MB	–
Оперативна пам’ять	Kingston DDR3 4GB (KVR1333D3N9/4G) Intel/AMD – 2 шт	–
DVD-привод	-	-
Корпус	ATX Middle Tower GIGABYTE GZ-X4 Silver 500W (GZ-X4 Silver)	–
Кулер	–	–
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-B, int. 3.5", 1*USB2.0+AUDIO+1394, multi: All Type Cards, black	–
інше	Клавіатура, мишка	–
Монітор	22" TFT, ASUS VW223D (5ms, 300/3000: 1, 170/160, D-SUB, Wide)	2600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USB cable	5500
Сканер	Epson Perfection V37 Photo	2970
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965
Пристрій безперебійного живлення	UPS APC BACK-UPS ES 525VA 230V RUSSIA (BE525-RS)	1348

КБПЗ-2023

Витрати на транспорт, монтаж та випробування можуть бути прийняті в межах до 10% від оптової ціни.

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп'ютери	13	11457	14894,1	163835,1
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Сканери	1	2970	297	3267
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	185653,6

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відраху-вання, грн.
1	2	3	4
Група 3			
1. Будівлі	2080000	-	-
2. Передавальні пристрої	208000	-	-
Всього по групі	2288000	5	114400

КОПЗ – 2023

Продовження таблиці 7.8

1	2	3	4
Група 4			
3. Обчислювальна техніка	185654	-	-
Всього по групі	185654	50	92827
Група 5, 6			
4. Вимірювальні пристрої	3999	25	-
5. Транспортні засоби	0	20	-
6. Господарський інвентар	45500	25	-
Всього по групі	49499	-	12374,75
7. Нематеріальні активи	30000	10	3000
Разом	$К_р$ = 2553153		$А_р$ = 222602

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

$$З_о = \frac{З_{сд} \cdot Т_{пз}}{N_e}, \quad (7.11)$$

де: N_e – кількість екземплярів програм, шт.

$$З_о = 360 \cdot 161/30 = 1932 \text{ грн.}$$

Визначимо додаткову зарплату (оплата відпусток, виконання державних та суспільних обов'язків) на рівні 10%:

$$З_д = З_о \cdot Н_q \cdot 0{,}01, \quad (7.12)$$

де: $Н_q$ – норматив додаткової зарплати, %.

$$З_д = 1932 \cdot 10 \cdot 0{,}01 = 193 \text{ грн.}$$

КОПЗ_2023

Відрахування на соціальні потреби за нормативом $H_c = 22\%$ від суми основної та додаткової зарплати:

$$C_{оц} = 0{,}01 \cdot H_c (З_о + З_д), \quad (7.13)$$

де: H_c – відрахування на соціальні потреби, %.

$$C_{оц} = 0{,}01 \cdot 22(1932+193) = 468 \text{ грн.}$$

Визначимо загальногосподарські витрати (електроенергію, ремонт і утримання приміщень і т.д) за нормативом $H_г = 15\%$ від основної зарплати:

$$Г_{осп} = З_о \cdot H_г \cdot 0{,}01, \quad (7.14)$$

де: $H_г$ – загальногосподарські витрати, %.

$$Г_{осп} = 1932 \cdot 15 \cdot 0{,}01 = 290 \text{ грн.}$$

Визначимо витрати на матеріали для розробки програмної продукції за нормами споживання та діючими цінами за одиницю виміру:

$$З_М = (З_{М1} + З_{М2} + З_{М3})/N_е, \quad (7.15)$$

де: $З_{М1}$ – вартість паперу, грн.;

$З_{М2}$ – вартість запам'ятовуючих пристроїв, грн.;

$З_{М3}$ – вартість фарби, картриджів, тонеру, грн.;

$N_е$ – кількість екземплярів програм, шт.

Згідно прийнятих норм на підприємстві $n_{вит}$ приймаємо 0,2 пачки паперу на період розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=200 грн., визначаємо вартість паперу за період розробки:

$$З_{М1} = Ц_п \cdot N_м \cdot n. \quad (7.16)$$

$$З_{М1} = 200 \cdot 1 \cdot 0{,}2 = 40 \text{ грн.}$$

Згідно прийнятих норм по комплектації до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків. Їх кількість дорівнює кількості коробочних версій запропонованого продукту (приймаємо 3):

$$З_{М2} = \sum Ц_д, \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR box – 23 грн./шт., DVD-R box – 35 грн./шт.

$$З_{М2} = 35 \cdot 2 + 23 = 93 \text{ грн.}$$

КПЗ – 2023

Згідно норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{МЗ} = \sum Ц_{з}., \quad (7.18)$$

де: $Ц_з$ – вартість розхідних матеріалів друкуючих пристроїв: відновлення та заправка картриджу для Canon i-SENSYS LBP6030W – 574 грн.; картридж для Epson Stylus Photo P50 – 558 грн.; відновлення картриджу для MF217W – 570 грн.

$$З_{МЗ} = 574+558+570 = 1702 \text{ грн.}$$

$$З_{М} = (40 + 93 + 1702)/30 = 61 \text{ грн.}$$

Визначимо витрати на освоєння нових мов програмування або операційних систем за нормативом ($Н_п$ = 15%) від основної зарплати виконавців:

$$О_п = З_о \cdot Н_п \cdot 0{,}01, \quad (7.19)$$

де: $Н_п$ – норматив витрат на освоєння нових мов програмування, %.

$$О_п = 1932 \cdot 15 \cdot 0{,}01 = 290 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 30 прим.):

$$А_м = \frac{А_р \cdot N_{міс}}{N_е \cdot 12}, \quad (7.20)$$

де: $А_р$ – загальна річна сума амортизаційних відрахувань, грн.

$$А_м = 222602 \cdot 1/(30 \cdot 12) = 618 \text{ грн.}$$

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції:

$$С_п = З_о + З_д + С_{оц} + Г_{осп} + З_м + О_п + А_м. \quad (7.21)$$

$$С_п = 1932+193+468+290+61+290+618 = 3852 \text{ грн.}$$

Величини ціна підприємства, податок на додану вартість, відпускна ціна програмної продукції визначаються за формулами, приведеними в таблиці 7.9

КПЗ-2023

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн.
1	2	3
1. Основна зарплата виконавців	$З_о$	1932
2. Додаткова зарплата виконавців	$З_д$	193
3. Відрахування на соціальні потреби	$С_{оц}$	468
4. Загальногосподарські витрати	$Г_{осп}$	290
5. Витрати на матеріали	$З_М$	61
6. Освоєння нових операційних систем, мов програмування	$О_п$	290
7. Амортизація основних фондів	$А_м$	618
8. Повна собівартість програмного забезпечення	$С_п$	3852
9. Плановий прибуток	$П_р$	1541
10.Ціна підприємства $Ц_п = С_п + П_р$	$Ц_п$	5393
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	1077
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	6470

Визначимо плановий прибуток за рівнем рентабельності ($Р_п$) програмної продукції, яка залежить від складності програми та ступеня новизни задачі.

Для даного програмного забезпечення рівень рентабельності складає 40%.

$$П_р = 0{,}01 \cdot Р_п \cdot С_п, \quad (7.22)$$

де: $Р_п$ – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 40 \cdot 3852 = 1541 \text{ грн.}$$

КОПІЯ – 2023

7.6 Визначення об’єму капітальних вкладень у споживача програмної продукції

Об’єм капітальних вкладень у споживача програмної продукції визначаємо на основі балансової вартості основних фондів, яка враховує ціну, транспортно-заготівельні витрати, вартість будівель, монтажних та пусконалагоджувальних робіт, а також витрати на випробування у виробничих умовах. Результати розрахунків зводимо у таблицю 7.9.

Таблиця 7.10 – Розрахунок об’єму капітальних вкладень у споживача програмної продукції

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	6470
Всього капітальних витрат	–	6470

7.7 Визначення експлуатаційних витрат

Експлуатаційні витрати у споживача програмної продукції визначаємо при умові роботи підсистеми на протязі року. Результати зводимо до таблиці 7.11.

Витрати на профілактичні роботи:

$$З_р = Т_р \cdot З_г \cdot (1 + 0{,}01 \cdot Н_q) \cdot (1 + 0{,}01 \cdot Н_с), \qquad (7.23)$$

де: $Т_р$ – кількість годин обслуговування кожного комп’ютера за рік, *год.*;

$З_г$ – заробітна плата обслуговуючого персоналу, *грн / год*.

КБПЗ-2023

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на технічне обслуговування	$З_р$	40260	21472
2. Витрати на електроенергію	$З_{ел}$	22680	22050
3. Витрати на амортизацію	$З_{ам}$	0	3235
Всього витрат за рік	$І$	62940	46757

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилася з 300 годин на рік до 160 годин на рік, тому витрати на технічне обслуговування зменшилися з:

$$З_{р\ баз} = 300 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 40260\ грн,$$

до:

$$З_{р\ нов} = 160 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 21472\ грн.$$

Витрати на електроенергію визначаються з урахуванням спожитої потужності ($П_{ел}$) в кіловатах, часу експлуатації технічних засобів ($Т_р$) в годинах та ціни однієї кіловат-години ($Ц_{ел}$):

$$З_{ел} = П_{ел} \cdot Т_р \cdot Ц_{ел}. \quad (7.24)$$

$$З_{ел\ баз} = 10 \cdot 0{,}15 \cdot 7200 \cdot 2{,}1 = 22680 \text{ грн.}$$

$$З_{ел\ нов} = 10 \cdot 0{,}15 \cdot 7000 \cdot 2{,}1 = 22050 \text{ грн.}$$

Витрати по амортизації визначаються на основі норм амортизаційних відрахувань, вартості програмної продукції і основних фондів. Для розрахунку складаємо таблицю 7.12.

КБПЗ-2023

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн., за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	50	–	6470	–	3235
Всього відрахувань	-	–	6470	–	3235

7.8 Визначення економічної ефективності програмної продукції

Економічна ефективність програмного забезпечення визначається для виготовлювача і споживача за такими показниками.

Величина економічного ефекту при виготовленні програмної продукції, розраховуємо за формулою:

$$E_в = (Ц_п - C_п) \cdot N_е - \sum_{i=1}^{m} E_{p_m} \cdot K_{p_m}, \quad (7.25)$$

де: $K_р$ – балансова вартість основних фондів розробника, грн.; $E_р$ – розрахунковий коефіцієнт капіталовкладень.

$$Ев = (5393 - 3852) \cdot 30 - (0,05 \cdot 2288000 + 0,5 \cdot 185654 + +0,25 \cdot 49499 + 0,1 \cdot 30000) \cdot 1/12 = 27680 \ грн.$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_p^*}{(Ц_п - C_п) \cdot N_е}, \quad (7.26)$$

де: K_p^* – балансова вартість основних фондів розробника без врахування вартості ОФ третьої групи, так як їх строк служби на порядок більший ніж період розробки ПЗ.

$$T_в = \frac{265153}{(5393 - 3852) \cdot 30 \cdot 12 / 1} = 0,5 \ року.$$

КПЗ – 2023

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

$$E_{сп} = (I_б - I_н) - E_н(K_н - K_б), \quad (7.27)$$

де: $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно;

$K_б$, $K_н$ – об'єм капітальних вкладень за варіантами, що порівнюються.

$$E_{сп} = (62940\text{-}46757)\text{-}0{,}5 \cdot 6470 = 12948 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у споживача програмної продукції за рахунок зниження експлуатаційних витрат:

$$T_{сп} = \frac{K_н - K_б}{I_б - I_н}, \quad (7.28)$$

$$T_{сп} = \frac{6470}{62940 - 46757} = 0{,}4 \text{ року}.$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	**Одиниця виміру**	**Величина**
1	2	3
1. Кількість екземплярів програми	Прим.	30
2. Повна собівартість розробленої програми	Грн.	3852
3. Ціна розробленої програми	Грн.	5393
4. Плановий прибуток від реалізації розробленої програми	Грн.	1541
5. Рентабельність програмної продукції	%	40
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	2553153
7. Загальний прибуток від реалізації програмної продукції	Грн.	46230

КОПІЯ 2023

Продовження таблиці 7.13

1	2	3
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	27680
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	0,5
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	6470
11.Величина економічного ефекту у користувача програмної продукції	Грн.	12948
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,4

7.9 Висновки

Розроблена програма економічно вигідна. За рахунок впровадження програмного забезпечення досягається скорочення часу обробки інформації, підвищується культура праці, підвищення якості приймаючих управлінських рішень.

КБПЗ – 2023

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Законом України “Про охорону праці” регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207, який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями», яким затверджено нормативно-правовий акт з охорони праці НПАОП 0.00-7.15-18, «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98.

Програмісти у процесі роботи мають негативний вплив на органи зору, а також мають значну розумову напругою і нервово-емоційне навантаження. Руки (суглоби польців та м’язи рук) при роботі з клавіатурою мають теж істотне навантаженням. До шкідливих факторів, які впливають на робитників галузі інформаційних технологій (ІТ) спеціалісти відносять високочастотні електромагнітні коливання (випромінювання) роботи апаратної частини ЕОМ та виділення шкідливих газів.

Ці шкідливі фактори можуть привести до професійних захворювань.

Розглянемо шкідливі чинники роботи програмістів керуючись наступними нормативно-правовими актами: «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98, та «Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» НПАОП 0.00-7.15-18.

КОПІЯ - 2023

Умови праці програміста включають наступні фактори:

- параметри повітряного середовища в приміщенні;
- вентиляція приміщення;
- освітлення приміщення;
- параметри повітряного середовища в приміщенні, тощо.

Щоб запропонувати заходи щодо зменшення негативного впливу комп'ютера на організм людини визначимо фактори, які можуть викликати професійне захворювання і впливають на працездатність програміста.

8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером

Електронно-обчислювальні машини (ЕОМ) та інше обладнання є джерелами небезпеки ураження електричним струмом. Так як робота програміста характеризується істотним зоровим навантаженням, то вимагає належного освітлення. У приміщенні, в якому працюють люди (у т.ч. програмісти) необхідно створити належний мікроклімат, праметри якого регламентуються, Державними санітарними правилами і нормами, зокрема ДСанПіН 3.3.2.007-98.

При роботі з використанням ЕОМ відзначають наступні небезпечні та шкідливі фактори:

- ризик виникнення надзвичайних ситуацій природного або штучного характеру на об'єкті або території.
- ризик виникнення пожежі;
- негативний вплив на органи зору людини;
- ризики ураження електричним струмом;
- недостатня, або надмірна освітленість робочого місця;
- електромагнітні (у т.ч. високочастотні) електромагнітні випромінювання (коливання);
- несприятливі мікрокліматичні умови;
- нервово-емоційна напруженість праці;

КПЗ 2023

- інтелектуальні навантаження;

- монотонність праці;

- невідповідність ергономічних показників робочого місця діючим вимогам;

- шум;

- статичні навантаження на кістково-м’язовий апарат;

8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

Розглянемо умови праці у приміщенні, в якому працюють програмісти. Геометричні розміри приміщення наведено у таблиці 8.1.

Таблиця 8.1 – Розміри приміщення

Найменування	**Значення, м**
Ширина	5,38
Довжина	5,95
Висота	2,8

Таблиця 8.2 – Площа та обсяг приміщення, на одного працюючого*

Геометрична характеристика	**Одиниця виміру**	**Нормативне значення***	**Фактичне значення**
Площа, S	$м^2$	не менше 6.0	8
Об’єм, V	$м^3$	не менше 20.0	22,4

* Згідно ДСанПіН 3.3.2.007-98 (Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин).

У зазначеному приміщенні працюють четверо людей. За даними, які наведено у табл. 8.1, та табл. 8.2, можна зробити висновок, що площа та об’єм приміщення у розрахунку на одне робоче місце програміста не відповідають

КПЗ – 2023

нормативним вимогам ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин», але відповідають нормативним вимогам Наказу Міністерства соціальної політики України № 207, від 14.02.2018 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» та НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин»). Таким чином можна зробити висновок, що санітарно-гігієнічні умови праці на робочому місці програміста відповідають вимогам.

Температура повітря в приміщенні визначається впливом температури зовнішнього повітря і тепловою енергією, яка виділяється всередині приміщення. Джерелами виділення теплоти в даному приміщенні є електроустаткування, освітлювальні прилади, а також люди. У світлий час доби джерелом надлишкового тепла є сонячна радіація. Згідно Постанови № 42 від 01.12.1999 Головного державного санітарного лікаря України, робота, виконувана в даному приміщенні, відноситься до категорії Іа. В цьому випадку людина витрачає енергії до 120 ккал у годину. Вологість повітря в приміщенні визначається впливом багатьох факторів, серед яких: вологість атмосферного повітря, виділення вологи людьми (при диханні та випарами з поверхні шкіри).

Мікроклімат повітряного середовища в приміщенні характеризується запиленістю та загазованістю повітря. Мікроклімат приміщення визначається діючим на організм людини поєднанням, вологості, температури, швидкості руху повітря та інтенсивності теплового випромінювання. Аналіз мікроклімату складається з визначення зазначених вище факторів і порівняння результатів із встановленими нормами.

У таблиці 8.3 наведено оптимальні та фактичні значення параметрів мікроклімату як для категорії ваги робіт Іа, так і розглянутого приміщення. У приміщеннях, де встановлено ЕОМ, рекомендується застосування тільки оптимальних значень показників мікроклімату.

КОПІЯ 2023

Таблиця 8.3 – Оптимальні і фактичні значення параметрів мікроклімату

Пора року	Оптимальні для Іа			Фактичні		
	Темпе-ратура, °С	Воло-гість, %	Швид-кість повітря, м/с	Темпе-ратура, °С	Воло-гість%	Швид-кість повітря, м/с
Холодна	22-24	40-60	0,1	22-23	40-55	0,1
Тепла	23-25	50-70	0,1	24-25	50-65	0,11

Проведений аналіз показує, що показники мікроклімату в приміщенні відповідають установленим нормам. Штучне опалення застосовується у холодний період року.

В літню пору застосовується кондиціонер.

Для боротьби з пилом робляться регулярні провітрювання та вологі прибирання приміщенні.

У приміщенні знаходяться наступні джерела шуму: принтер HP 1100, електродвигуни вентиляторів ЕОМ.

Одним з найважливіших факторів, які впливають на ефективність трудової діяльності людини, та попереджають травматизм і професійні захворювання програмістів є освітлення на робочому місці.

З 2019 року діють Державні будівельні норми україни “Природне і штучне освітлення” – ДБН В.2.5-28:2018, у яких прописані вимоги до використання всіх освітлювальних приладів, у т.ч. світлодіодних.

Працю робітника, який постійно працює за комп’ютером, згідно ДБН В.2.5-28:2018, можна віднести до роботи з малою точністю (найменший розмір об’єкта розрізнення від 1 до 5 мм) V-го розряду зорової роботи, з великою контрастністю об’єкта розрізнення (символів на екрані дисплея), з темним тлом (під розряд зорової роботи В). Приміщення можна віднести до 1-ої групи приміщень, у яких проводиться розрізнення об’єктів зорової роботи при фіксованому напрямку лінії зору того, що працює на робочу поверхню. Для

КОПЗ-2023

такого типу приміщень і розряду зорової роботи нормоване значення коефіцієнта природної освітленості (КПО) робочої поверхні (при поєднаному, спільному освітленні), повинен становити не більше 1,5%, освітленість при штучному висвітленні повинна становити 300 Лк., Крім того все поле зору повинне бути освітлено достатньо рівномірно – ця основна гігієнічна вимога. Так як яскраве світло на ділянці периферійного зору значно збільшує напруженість очей і, як наслідок, призводить до їх швидкої стомлюваності, ступінь освітлення приміщення і яскравість екрану комп'ютера повинні бути приблизно однаковими.

8.4 Розробка заходів з умов поліпшення охорони праці

Згідно аналізу умов праці в розглянутому приміщенні, ми одержали наступні результати:

- розмірі приміщення, у розрахунку на одному працюючого, відповідають нормативам;
- мікроклімат відповідає нормативному значенню;
- акустичні умови роботи не перевищують нормативних значень;

Таким чином можна припустити, що основною причиною можливого зниження працездатності програміста є психофізіологічний фактор, тому основна пропозиція буде така: дотримання позитивної психологічної атмосфери в колективі та регламентованого режиму праці та відпочинку, організація робочого місця з урахуванням ергономічних вимог.

Рекомендовані заходи: регулярні періодичні наочні огляди персоналом шляхів для евакуації людей із приміщення, відповідно до плану евакуації (який повинен розташовуватись на видному місці у приміщенні), включення до колективного договору мінімально можливого вмісту аптечок з обов'язковою наявністю масок-клапанів, або іншого спорядження для штучного дихання. Регулярна періодична перевірка параметрів заземлення та занулення (вимірювання опору ланцюга).

КОПІЯ 2023

Регулярна наочне знайомство персоналу із шляхами для евакуації людей із приміщення відповідно до плану евакуації, забезпечення розподільних щитів спеціальними розетками з заземлюючими контактами; організація заземлення всіх приладів і пристроїв, які працюють при напрузі вище 36 В.

Так як при ураженні електричним струмом у людини може статися фібриляція шлуночків серця, в організації бажано мати дефібрилятор і підготовлений персонал для роботи з ним.

8.5 Розрахункова частина

Проведемо розрахунок штучного освітлення за методом коефіцієнта використання світлового потоку для приміщення ширина якого складає 3 м, довжина – 4,6 м, висота – 3 м.

У зазначеному приміщенні працює 1 особа.

Для того, щоб визначити потрібну кількість світильників, які повинні забезпечити нормований рівень освітленості, визначимо світловий потік, що падає на робочу поверхню за формулою:

$$F=(E\cdot S\cdot K\cdot Z)/n,$$

де: F – світловий потік, що розраховується, Лм; E – нормована мінімальна освітленість, Лк; E = 300 Лк; S – площа освітлюваного приміщення (у нашому випадку S=3 х 4,6 = 13,8 м^2); K – коефіцієнт запасу, що враховує зменшення світлового потоку лампи в результаті забруднення світильників в процесі експлуатації (його значення залежить від типу приміщення і характеру робіт, що проводяться в ньому, в нашому випадку K = 1,5); Z – відношення середньої освітленості до мінімальної (зазвичай приймається рівним 1.1... 1.2, в нашому випадку Z = 1,1); n – коефіцієнт використання світлового потоку, (відношення світлового потоку, що падає на розрахункову поверхню, до сумарного потоку всіх ламп і обчислюється в долях одиниці; залежить від характеристик світильника, розмірів приміщення, забарвлення стін і стелі, що характеризуються

КПЗ-2023

коефіцієнтами відбиття від стін ($\rho_{стін.}$) і стелі ($\rho_{стелі}$), значення коефіцієнтів дорівнюють $\rho_{стін}$ = 50% і $\rho_{стелі}$ = 50%.

Обчислимо індекс приміщення за формулою:

$$i=S/(h(A+B)),$$

де: S – площа приміщення, S = 13,8 м^2; h – розрахункова висота підвісу, h = 3 м (співпадає з висотою стелі, т.я. лампи освітлення закріплюються на стелі); A – ширина приміщення, A = 3 м; B – довжина приміщення, B = 4,6 м.

Підставимо всі значення у формулу та визначимо індексу приміщення: i=0,43. Знаючи індекс приміщення, за знаходимо n = 0,23 (з табличних даних коефіцієнтів використання світлового потоку (n) світильників з відповідним типом лампам). Підставимо всі значення у формулу, визначимо світловий потік: *F=29700 Лм.* Для розрахунку будемо використовувати світлодіодні панелі *LED панель 42Вт 6000К SUNLED 000000127*, світловий потік яких $F_л$ = *3990 Лм.*

Число ламп визначається по формулі:

$$N=F/F_л$$

де: F – світловий потік, $F_л$ – світловий потік однієї лампи.

Підставимо всі значення у формулу та визначимо індекса приміщення: *N= 29700/ 3990=7,4 шт.*

Приймаємо необхідну кількість світлодіодних світильників 8 шт.

8.6 Висновки

Дотримання всіх необхідних умов праці не лише сприяє збереженню здоров’я працівників, а також підвищує ефективність виробництва вцілому.

З цих міркувань було здійснено аналіз умов праці, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи. Виконано розрахунок штучного освітлення, як одного з ключових факторів впливу на працездатність та здоров’я програміста. Розроблено заходи з охорони праці.

КПЗ – 2023

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання магістерської роботи, призначено для системи розпізнавання та пошуку інформації за номером автомобіля на базі нейронної мережі.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

У магістерській роботі наведені теоретичне узагальнення й рішення наукового завдання дослідження методів розпізнавання та пошуку інформації за номером автомобіля на базі нейронної мережі.

Рішення даного завдання полягало у вирішенні наступних задач:

– Дослідження існуючих систем для розпізнавання та пошуку інформації за номером автомобіля.

– Розробка методів та алгоритмів системи для розпізнавання та пошуку інформації за номером автомобіля на базі нейронної мережі.

– Програмна реалізація системи розпізнавання та пошуку інформації за номером автомобіля на базі нейронної мережі.

Розроблені під час виконання магістерської роботи алгоритми дозволяють успішно вирішувати завдання ідентифікації автомобіля за зображенням з камери мобільного телефону.

Проведено аналіз предметної галузі в ході якого були виявлені об’єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудований алгоритм і вибране середовище розробки.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

КОПІЯ 2023

При створені програмного забезпечення було використано об’єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові програмування високого рівня Kotlin у середовищі розробки Android Studio. Дана мова програмування дозволяє найбільш ефективно вирішити поставлені задачі. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку.

Програма призначена для виконання на мобільних пристроях під управлінням багатозадачної операційної системи Android Studio.

У роботі даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для інформаційного захисту розробленого програмного забезпечення був обраний компонент безпеки платформи Android – Android Keystore.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 12948 грн. З урахуванням вартості розробки програми та обладнання, строк окупності становить 0,4 роки.

КОПЗ-2023

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Aggarwal C.C. Neural Networks and Deep Learning: A Textbook 1st ed. 2018 Edition. – Springer, 2018. – 520 p.

2. Aho A.V., Hopcroft J.E., Ullman J.D. Data Structures and Algorithms. – Pearson, 2001. – 620 p.13. Кормен Т., Лейзерсон Ч., Риверст Р., Штайн К. Алгоритмы: построение и анализ, 3-е издание – М.: Диалектика, 2019. – 1328 p.

3. Angel Miguel, Torres Castiblanco Learning Concurrency in Kotlin. 2018 ISBN: 9781788627160

4. Bernstein James Android Smartphones Made Easy: The Beginners Guide Made For Beginners (Computers Made Easy) Paperback. Independently published. 2019. 243 pages.

5. Chakraborty Rivu Reactive Programming in Kotlin: Design and build non-blocking, asynchronous Kotlin applications with RXKotlin, Reactor-Kotlin, Android, and Spring. 2017 ISBN: 9781788473026

6. Chung, J., Gulcehre, C., Cho, K. & Bengio, Y. “Empirical evaluation of gated recurrent neural networks on sequence modeling”. – Available from: https://arxiv.org/abs/1412.3555. – [Accessed February 2021].

7. Collier Marsha Android Smartphones For Seniors For Dummies (For Dummies (Computer/Tech)) 1st Edition. For Dummies. 2021. 336 pages

8. Developers Guide to app architecture. – URL: https://developer.android.com/topic/architecture

9. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. Design Patterns. Elements of reusable object-oriented software. Addison Wesley. 1994. 395 c.

10. Greenhalgh David Kotlin Programming: The Big Nerd Ranch Guide, Josh Skeen. 2018 ISBN: 9780135161630

11. Hands-on Design Patterns with Kotlin, Alexey Soshin. 2018 ISBN: 9781788998017

КПІ – 2023

12. Hermans, M. & Schrauwen, B. “Training and analysing deep recurrent neural networks”. in Advances in neural information processing systems. 2013; Vol 2: p. 190–198.

13. Horton John Android Programming for Beginners: Build in-depth, full-featured Android apps starting from zero programming experience, 3rd Edition. Packt Publishing. 2021. 742 pages

14. James Mike Programmer’s Guide To Kotlin. 2017 ISBN: 9781871962536

15. Karanpuria Rashi Kotlin Programming Cookbook, Aanand Shekhar Roy. 2018 ISBN: 9781788472142

16. Knuth, Donald E. The Art of Computer Programming: Fundamental Algorithms. 3rd Ed. Addison-Wesley, 1997.

17. Knuth, Donald E. The Art of Computer Programming: Seminumerical Algorithms. 3rd Ed. Addison-Wesley, 1998.

18. Knuth, Donald E. The Art of Computer Programming: Sorting and Searching. 2nd Ed. Addison-Wesley, 1998.

19. Kotlin docs. Latest stable version: 1.9.10. – URL: https://kotlinlang.org/docs/home.html

20. Michael I. Jordan, Cristopher M. Bishop. Neural Networks. Massachusets Institute of Technology. AI Memo No. 1562. Anonymous ftp://publications.ai.mit.edu.

21. Saumont Pierre-Yves The Joy of Kotlin. 2018 ISBN: 9781617295362

22. Smyth Neil Android Studio Giraffe Essentials - Java Edition: Developing Android Apps Using Android Studio 2022.3.1. Payload Media. 2023. 1201 pages.

23. Subramaniam Venkat Programming Kotlin. 2019 ISBN: 9781680506358

24. The Joy of Kotlin, Pierre-Yves Saumont. 2019 ISBN: 9781617295362

25. Valentin F. Turchin. The phenomenon of the science, a cybernatic approach to human evolution. Addison Wesley.

26. Wood, Derick. Data Structures, Algorithms, and Performance. Addison-Wesley, 1993.

27. Алгоритми і структура даних: Навчальний посібник / В.М.Ткачук. -

КОПЗ - 2023

ІваноФранківськ : Видавництво Прикарпатського національного університету імені Василя Стефаника, 2016. 286 с.

28. Алгоритми та структури даних. Навчальний посібник / Т. О. Коротєєва. Львів : Видавництво Львівської політехніки, 2014. - 280 с.

29. Аналіз методів і технологій розробки мобільних додатків для платформи Android : навч. посіб. / О. В. Шматко, А. О. Поляков, В. М. Федорченко. – Харків : НТУ «ХПІ», 2018. – 284 с.

30. Гуржій А.М., Львов М.С., Співаковський О.В. Основи програмування. 256 с.

31. Державні будівельні норми України: ДБН В.2.5-28:2018. - Режим доступу до ресурсу: https://goo.su/9AkQ

32. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин: ДСанПІН 3.3.2-007-98. - Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/v0007282-98

33. Етапи створення мобільного додатку. URL: https://webcase.com.ua/uk/blog/etapi-stvorennja-mobilnogo-dodatku/

34. Закон України «Про охорону праці» від 14.10.1992 р. № 2694-XII. - Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/2694-12

35. Зеркалов Д. В. Охорона праці в Галузі: Загальні вимоги: навч. посіб. Київ: Основа. 2011. 551 с.

36. Ічанська Н.В., Улько С.І. Основні аспекти створення мобільних додатків та вибір інструментів їх розробки // Системи управління, навігації та зв'язку. Збірник наукових праць. – Полтава: ПНТУ, 2020. – Т. 1 (59). – С. 74-78. – doi:https://doi.org/10.26906/SUNZ.2020.1.074.

37. Кавун С.В., Смірнов О.А., Сорбат І.В., Мелешко Є.В., Коваленко О.В. Системи штучного інтелекту // Навчальний посібник для студентів вищих навчальних закладів напрямів підготовки 6.050102 «Комп'ютерна інженерія». Гриф "Навчальний посібник" надано у відповідності з листом Міністерства освіти і науки, молоді та спорту України від 12.02.13року №1/11-2237 –

КОПІЯ 2023

Кіровоград: КНТУ 2013. – 335 с.

38. Ковалюк Т.В. Алгоритмізація та програмування: Підручник. – Львів: «Магнолія 2006», 2013. – 400 с.

39. Кормен Томас Х., Лейзерсон Чарльз І., Ривест Рональд Л., Штайн Кліффорд, Алгоритми. Побудова і аналіз

40. Мелешко Є. В., Якименко М.С., Поліщук Л.І. Алгоритми та структури даних // Навчальний посібник для студенів технічних спеціальностей денної та заочної форми навчання. – Кропивницький: Видавець – Лисенко В.Ф., 2019. – 156 с. – URL: http://dspace.kntu.kr.ua/jspui/handle/123456789/8944

41. Мелешко Є.В., Якименко М.С. Машинне навчання // Методичні рекомендації до виконання лабораторних робіт студентами денної та заочної форми навчання спеціальності 123 "Комп'ютерна інженерія" – Кропивницький: ЦНТУ, 2021. – 52 с.

42. Методичні рекомендації до виконання розділу "Заходи з охорони праці та техніки безпеки" випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти для здобувачів вищої освіти спеціальностей 123 "Комп'ютерна інженерія" та 122 "Комп'ютерні науки" / М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т, каф. кібербезпеки та програм. забезпечення; [укл. О.В. Оришака, К.М. Марченко]. - Кропивницький: ЦНТУ, 2022. — 19 с. [Електронний ресурс]. – Режим доступу : http://dspace.kntu.kr.ua/jspui/handle/123456789/12240

43. Мови програмування для мобільної розробки [Електронний ресурс]. – Режим доступу: https://code.tutsplus.com/uk/articles/mobile-development-languages-cms-29138

44. Моделі життєвого циклу, принципи і методології розробки програмного забезпечення (ПЗ) [Електронний ресурс]. – https://evergreens.com.ua/articles/ software-development-metodologies.html

45. Моделі і методи проектування інформаційних систем [Електронний ресурс]. – https://elearning.sumdu.edu.ua/free_content/lectured:de1c9452f2a161439

КОПЗ-2023

391120eef364dd8ce4d8e5e/20151203140326/165292/index.html

46. Наказ Міністерства соціальної політики України 14.02.2018 № 207 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями». - Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/z0508

47. Николайчук Я. М. Проектування спеціалізованих комп'ютерних систем : навч. посібник / Я. М. Николайчук, Н. Я. Возна, І. Р. Пітух. - Тернопіль : ТзОВ "Терно-граф", 2010. - 392 с.

48. Оришака О. В. Основи охорони праці: навч. посіб. / О. В. Оришака, Г. П. Горбачова, К. М. Марченко; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. - Кропивницький : ЦНТУ, 2022. - 175 с. – Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/12161 (дата звернення 19.09.22).

49. Охорона праці. Ч. 1. Захисне заземлення: метод. вказ. до викон. розрахунків з викор. персон. ЕОМ ІВМ сумісного типу / Кіровоград. ін-т с.-г. машинобуд.; [укл. О. В. Оришака, Є. К. Солових, В. О. Оришака]. - Кіровоград: КІСМ, 1997. - 20 с. Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/4358

50. Постанова № 42 від 01.12.1999 Головного державного санітарного лікаря України «Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99. - Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/va042282-99 (дата звернення 19.09.22).

51. Смірнов О.А., Євсеєв С.П., Жукарев В.Ю., Король О.Г., Сорокін В.Є., Мелешко Є.В. Технології і стандарти комп'ютерних мереж // Навчальний посібник для студентів вищих навчальних закладів напрямів підготовки 8.050102 «Комп'ютерна інженерія» та 8.050201 «Системна інженерія». За ред. О.А. Смірнова Гриф "Навчальний посібник" надано у відповідності з листом Міністерства освіти і науки, молоді та спорту України від 1.12.2011 року № 1/11-11258. – Кіровоград: КНТУ 2012. – 454 с.

КПІ – 2023

52. Смірнов О.А., Коваленко О.В., Мелешко Є.В., Константинова Л.В., Кожанова А.С. Інженерія програмного забезпечення // Навчальний посібник для студентів вищих навчальних закладів напрямів підготовки 6.050102 «Комп'ютерна інженерія». Гриф "Навчальний посібник" надано у відповідності з листом Міністерства освіти і науки, молоді та спорту України від 18.03.13 року №1/11-5584 – Кіровоград: КНТУ 2013. – 335 с.

53. Смірнов О.А., Мелешко Є.В., Семенов С.Г. Методи та засоби обробки сигналів і даних в інформаційних системах // Навчальний посібник для студентів вищих навчальних закладів напрямів підготовки 8.050102 «Комп'ютерна інженерія». Гриф "Навчальний посібник" надано у відповідності з листом Міністерства освіти і науки, молоді та спорту України від 17.04.2012 року № 1/11-5249. – Кіровоград: КНТУ 2012. – 250 с.

54. Харченко К.В. Методи та засоби розробки програмних додатків для операційної системи Андроїд // Збірник наукових праць. Серія: Нові рішення в сучасних технологіях. – Х.: НТУ «ХПІ». – 2014. - № 17. – С. 68-72.

55. Центр післядипломної освіти та підвищення кваліфікації. - Режим доступу до ресурсу: https://cpo.stu.cn.ua

56. Що це таке тестування програмного забезпечення та яке його призначення? [Електронний ресурс]. – Режим доступу: https://www.quality-assurance-group.com/shho-take-testuvannya-programnogo-zabezpechennya-ta-yake-jogo-znachennya/

КОПІЮВАННЯ

Додаток А
(обов'язковий)

Технічне завдання

Зміст

КПЗ-2023

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи розпізнавання та пошуку інформації за номером автомобіля на базі нейронної мережі.

2 Підстава для розробки

Підставою для розробки служить завдання на магістерську роботу, видане на кафедрі кібербезпеки та програмного забезпечення (нак. №__-__ від __.__.20__ року).

3 Мета та призначення розробки

Метою магістерської роботи є дослідження та програмна реалізація системи розпізнавання та пошуку інформації за номером автомобіля на базі нейронної мережі.

4 Джерела розробки

Джерелом цієї магістерської роботи є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії

КПЗ – 2023

системи з ОС та з користувачем;

– техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;

– аналіз умов праці розробників програмного забезпечення;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– розпізнавання та пошук інформації за номером автомобіля на базі нейронної мережі;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Застосунок, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною

КПЗ – 2023

документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Android і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Android.

5.8.1 Обладнання

Комп'ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14" 1Mb або сумісні з ним.

5.8.2 Мова програмування

Мова програмування високого рівня Kotlin.

КПЗ – 2023

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД.

7 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 1 вересня 2023 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи повинні бути розглянуті умови праці програмістів під час розробки програмного забезпечення.

КбПЗ – 2023

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуші.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 94 аркушів.

10 Етапи розробки

10.1 Збір і обробка інформації по темі магістерської роботи. Постановка задачі на виконання магістерської роботи (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень магістерської роботи.

10.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання магістерської роботи на попередній захист __.__.2023 р.

11.2 Подання магістерської роботи на захист __.__.12.2023 р.

КПЗ – 2023

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ
Керівник випускної кваліфікаційної роботи
за другим (магістерським) рівнем вищої освіти
_____ Є.В. Мелешко

Дослідження та програмна реалізація системи розпізнавання та пошуку інформації за номером автомобіля на базі нейронної мережі

Лістинг програми

Код документу 12

Носій: DVD-диск

Загальна кількість аркушів: 22

Літера: РП

Кропивницький – 2023 року

КбПЗ – 2023

Файл BaseFragment.kt – модуль основної програми для роботи з інтерфейсом користувача (базовий фрагмент інтерфейсу)

```
package com.gahov.plates_recognition.arch.ui.fragment

import android.content.Context
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Toast
import androidx.annotation.LayoutRes
import androidx.databinding.DataBindingUtil
import androidx.databinding.ViewDataBinding
import androidx.fragment.app.Fragment
import androidx.lifecycle.ViewModel
import androidx.lifecycle.ViewModelProvider
import androidx.navigation.fragment.findNavController
import com.gahov.domain.entity.component.logger.Level
import com.gahov.domain.entity.component.logger.Logger
import com.gahov.domain.entity.failure.Failure
import com.gahov.plates_recognition.arch.component.error.ErrorHandler
import com.gahov.plates_recognition.arch.controller.BaseViewModel
import com.gahov.plates_recognition.arch.ktx.getString
import com.gahov.plates_recognition.arch.provider.RouterProvider
import com.gahov.plates_recognition.arch.router.NavComponentRouter
import com.gahov.plates_recognition.arch.router.Router
import com.gahov.plates_recognition.arch.router.command.Command
import com.gahov.plates_recognition.arch.router.command.NavDirection
import com.gahov.plates_recognition.arch.ui.view.BaseView
import com.gahov.plates_recognition.arch.ui.view.model.TextProvider
import javax.inject.Inject

/**
 * An abstract base class for creating Fragments with integrated ViewModel,
DataBinding, and navigation functionality.
 *
 * @param B The type of ViewDataBinding associated with the fragment.
 * @param T The type of ViewModel associated with the fragment.
 * @property contentLayoutID The resource ID of the layout to be inflated.
 * @property viewModelClass The class of the associated ViewModel.
 */

abstract class BaseFragment<B : ViewDataBinding, T : ViewModel>(
```

КПЗ-2023

```
    @LayoutRes private val contentLayoutID: Int,
    private val viewModelClass: Class<T>,
) : Fragment(), BaseView, RouterProvider {

    protected lateinit var binding: B
        private set

    protected lateinit var viewModel: T

    @Inject
    protected open lateinit var logger: Logger

    @Inject
    protected open lateinit var failureHandler: ErrorHandler

    @Inject
    lateinit var viewModelFactory: ViewModelProvider.Factory

    /**
     * The Router instance responsible for navigation within the fragment.
     */
    override val router: Router by lazy {
        NavComponentRouter(
            navController = findNavController(),
            logger = logger
        )
    }

    override fun onAttach(context: Context) {
        super.onAttach(context)
        viewModel = ViewModelProvider(this, viewModelFactory)[viewModelClass]
    }

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        binding = DataBindingUtil.inflate(inflater, contentLayoutID, container,
false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        binding.lifecycleOwner = viewLifecycleOwner
```

КОПЗ – 2023

```
        setBaseObservers()
        setObservers()
    }

    open fun logMessage(message: TextProvider.Text) {
        logger.log(
            message = message.text
        )
    }

    protected open fun navigate(command: Command) {
        when (command) {
            is NavDirection -> router.navigate(command)
            Command.Back -> router.popBackStack()
            Command.Root -> router.popToRoot()
            Command.Close -> requireActivity().finish()
            is Command.FeatureCommand -> navigateByFeature(command)
            is Command.Route -> commandError(command)
        }
    }

    protected open fun setBaseObservers() {
        getCurrentViewModel()?.let {
            it.errorEvent.observe(viewLifecycleOwner, ::displayError)
            it.command.observe(viewLifecycleOwner, ::handleCommandData)
            it.message.observe(viewLifecycleOwner, ::showMessage)
        }
    }

    protected open fun handleCommandData(command: Command) {
        when (command) {
            is NavDirection -> router.navigate(command)
            Command.Back -> router.popBackStack()
            Command.Root -> router.popToRoot()
            Command.Close -> requireActivity().finish()
            is Command.FeatureCommand -> handleFeatureCommand(command)
            is Command.Route -> commandError(command)
        }
    }

    protected open fun handleFeatureCommand(command: Command.FeatureCommand) {
        logger.log(
            level = Level.Warning, "method navigateByFeature isn't implement for
$command"
        )
    }
```

КОПЗ 2023

```
    protected open fun navigateByFeature(command: Command.FeatureCommand) {
        logger.log(
            level = Level.Warning,
            message = "method navigateByFeature isn't implement for $command"
        )
    }

    private fun commandError(command: Command) {
        logger.log(
            level = Level.Warning,
            message = "navigation isn't implement for $command"
        )
    }

    protected open fun setObservers() {}

    protected open fun getCurrentViewModel(): BaseViewModel? {
        return viewModel as? BaseViewModel
    }

    override fun displayError(failure: Failure) {
        failureHandler.parseFailure(failure)
    }

    override fun showMessage(textProvider: TextProvider) {
        context?.let { context ->
            Toast.makeText(
                context.applicationContext,
                textProvider.getString(context),
                Toast.LENGTH_SHORT
            ).show()
        }
    }
}
```

КОП 3 – 2023

Файл BaseBottomSheetFragment.kt – модуль основної програми для роботи з інтерфейсом користувача (нижній фрагмент інтерфейсу)

```
package com.gahov.plates_recognition.arch.ui.dialog

import android.content.Context
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Toast
import androidx.annotation.LayoutRes
import androidx.databinding.DataBindingUtil
import androidx.databinding.ViewDataBinding
import androidx.lifecycle.ViewModel
import androidx.lifecycle.ViewModelProvider
import androidx.navigation.fragment.findNavController
import com.gahov.domain.entity.component.logger.Level
import com.gahov.domain.entity.component.logger.Logger
import com.gahov.domain.entity.failure.Failure
import com.gahov.plates_recognition.arch.component.error.ErrorHandler
import com.gahov.plates_recognition.arch.controller.BaseViewModel
import com.gahov.plates_recognition.arch.ktx.getString
import com.gahov.plates_recognition.arch.provider.RouterProvider
import com.gahov.plates_recognition.arch.router.NavComponentRouter
import com.gahov.plates_recognition.arch.router.Router
import com.gahov.plates_recognition.arch.router.command.Command
import com.gahov.plates_recognition.arch.router.command.NavDirection
import com.gahov.plates_recognition.arch.ui.view.BaseView
import com.gahov.plates_recognition.arch.ui.view.model.TextProvider
import com.google.android.material.bottomsheet.BottomSheetDialogFragment
import javax.inject.Inject

/**
 * An abstract base class for creating BottomSheetDialogFragments that are
integrated with ViewModel,
 * DataBinding, and navigation functionality.
 *
 * @param T The type of ViewModel associated with the fragment.
 * @param B The type of ViewDataBinding associated with the fragment.
 * @property layoutId The resource ID of the layout to be inflated.
 * @property viewModelClass The class of the associated ViewModel.
 */

abstract class BaseBottomSheetFragment<T : ViewModel, B : ViewDataBinding>(
    @LayoutRes private val layoutId: Int,
    private val viewModelClass: Class<T>
) : BottomSheetDialogFragment(), BaseView, RouterProvider {

    protected lateinit var binding: B
        private set

    protected lateinit var viewModel: T

    @Inject
    protected open lateinit var logger: Logger

    @Inject
    protected open lateinit var failureHandler: ErrorHandler

    @Inject
    lateinit var viewModelFactory: ViewModelProvider.Factory

    /**
     * The Router instance responsible for navigation within the fragment.
     */
```

ІПЗ-2023

```
    override val router: Router by lazy {
        NavComponentRouter(
            navController = findNavController(),
            logger = logger
        )
    }

    override fun onAttach(context: Context) {
        super.onAttach(context)
        viewModel = ViewModelProvider(this, viewModelFactory)[viewModelClass]
    }

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        binding = DataBindingUtil.inflate(inflater, layoutId, container, false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        binding.lifecycleOwner = viewLifecycleOwner
        setBaseObservers()
        setObservers()
    }

    open fun logMessage(message: TextProvider.Text) {
        logger.log(
            message = message.text
        )
    }

    protected open fun navigate(command: Command) {
        when (command) {
            is NavDirection -> router.navigate(command)
            Command.Back -> router.popBackStack()
            Command.Root -> router.popToRoot()
            Command.Close -> requireActivity().finish()
            is Command.FeatureCommand -> navigateByFeature(command)
            is Command.Route -> commandError(command)
        }
    }

    protected open fun setBaseObservers() {
        getCurrentViewModel()?.let {
            it.errorEvent.observe(viewLifecycleOwner, ::displayError)
            it.command.observe(viewLifecycleOwner, ::handleCommandData)
            it.message.observe(viewLifecycleOwner, ::showMessage)
        }
    }

    protected open fun handleCommandData(command: Command) {
        when (command) {
            is NavDirection -> router.navigate(command)
            Command.Back -> router.popBackStack()
            Command.Root -> router.popToRoot()
            Command.Close -> requireActivity().finish()
            is Command.FeatureCommand -> handleFeatureCommand(command)
            is Command.Route -> commandError(command)
        }
    }

    protected open fun handleFeatureCommand(command: Command.FeatureCommand) {
        logger.log(
            level = Level.Warning, "method navigateByFeature isn't implement for
$command"
        )
```

KSTB 2023

```
    }

    protected open fun navigateByFeature(command: Command.FeatureCommand) {
        logger.log(
            level = Level.Warning,
            message = "method navigateByFeature isn't implement for $command"
        )
    }

    private fun commandError(command: Command) {
        logger.log(
            level = Level.Warning,
            message = "navigation isn't implement for $command"
        )
    }

    protected open fun setObservers() {}

    protected open fun getCurrentViewModel(): BaseViewModel? {
        return viewModel as? BaseViewModel
    }

    override fun displayError(failure: Failure) {
        failureHandler.parseFailure(failure)
    }

    override fun showMessage(textProvider: TextProvider) {
        context?.let { context ->
            Toast.makeText(
                context.applicationContext,
                textProvider.getString(context),
                Toast.LENGTH_SHORT
            ).show()
        }
    }
}
```

КбПЗ_2023

Файл BaseRecyclerListAdapter.kt – модуль основної програми для роботи з інтерфейсом користувача (адаптер списків)

```
package com.gahov.plates_recognition.arch.ui.recycler

import android.view.LayoutInflater
import android.view.ViewGroup
import androidx.annotation.LayoutRes
import androidx.databinding.DataBindingUtil
import androidx.databinding.ViewDataBinding
import androidx.recyclerview.widget.AsyncDifferConfig
import androidx.recyclerview.widget.DiffUtil
import androidx.recyclerview.widget.ListAdapter

/**
 * An abstract base class for creating RecyclerView ListAdapters with simplified
item management
 * and view binding.
 *
 * @param T The type of items to be displayed in the adapter.
 * @property items The list of items to be displayed in the adapter.
 */
abstract class BaseRecyclerListAdapter<T : Any> :
    ListAdapter<T, BaseViewHolder<T, out ViewDataBinding>> {

    var items: List<T> = arrayListOf()
        set(value) {
            field = value
            submitList(value)
        }

    constructor(diffCallback: DiffUtil.ItemCallback<T>) : super(diffCallback)

    constructor(config: AsyncDifferConfig<T>) : super(config)

    protected fun inflate(parent: ViewGroup, @LayoutRes contentLayoutID: Int):
ViewDataBinding {
        return DataBindingUtil.inflate(
            LayoutInflater.from(parent.context), contentLayoutID, parent, false
        )
    }

    override fun onViewAttachedToWindow(holder: BaseViewHolder<T, out
ViewDataBinding>) {
        holder.onAttachToWindow()
```

КОПЗ_2023

```
        super.onViewAttachedToWindow(holder)
    }

    override fun onViewDetachedFromWindow(holder: BaseViewHolder<T, out
ViewDataBinding>) {
        holder.onDetachFromWindow()
        super.onViewDetachedFromWindow(holder)
    }

    override fun onBindViewHolder(holder: BaseViewHolder<T, out
ViewDataBinding>, position: Int) {
        holder.bindView(position, getItem(position))
    }

    /**
     * Removes the item at the specified [index] from the list of items.
     *
     * @param index The index of the item to be removed.
     */
    fun removeAt(index: Int) {
        if (index in items.indices) {
            val newItems = items.toMutableList()
            newItems.removeAt(index)
            items = newItems
            notifyItemRangeChanged(index, items.size - index)
        }
    }
}
```

КОПЗ – 2023

Файл BaseViewHolder.kt – модуль основної програми для роботи з інтерфейсом користувача (пошук та збереження посилань на елементи інтерфейсу користувача)

```
package com.gahov.plates_recognition.arch.ui.recycler

import androidx.databinding.ViewDataBinding
import androidx.recyclerview.widget.RecyclerView

/**
 * An abstract base class for creating RecyclerView ViewHolders with simplified
binding and item handling.
 *
 * @param T The type of item associated with the ViewHolder.
 * @param B The type of ViewDataBinding associated with the ViewHolder.
 * @property binding The ViewDataBinding instance associated with the
ViewHolder's root view.
 */

abstract class BaseViewHolder<T : Any, B : ViewDataBinding>(binding:
ViewDataBinding) :
    RecyclerView.ViewHolder(binding.root) {

    lateinit var item: T
        private set

    @Suppress("UNCHECKED_CAST")
    val binding: B = binding as B

    fun bindView(position: Int, item: T) {
        this.item = item
        bindView(position)
    }

    abstract fun bindView(position: Int)

    open fun onAttachToWindow() {}

    open fun onDetachFromWindow() {}
}
```

КОПЗ – 2023

Файл NavComponentRouter.kt – модуль основної програми для навігації між екранами додатку

```
package com.gahov.plates_recognition.arch.router

import android.os.Bundle
import androidx.annotation.IdRes
import androidx.navigation.*
import com.gahov.plates_recognition.R
import com.gahov.domain.entity.component.logger.Logger
import com.gahov.plates_recognition.arch.router.command.Command
import com.gahov.plates_recognition.arch.router.command.NavDirection

open class NavComponentRouter(
    private val navController: NavController,
    private val logger: Logger
) : Router {

    override fun navigate(command: Command.Route) {
        if (command is NavDirection) {
            navigate(command)
        } else {
            throw Throwable(message = Router.UNKNOWN_COMMAND)
        }
    }

    override fun popBackStack() = navController.popBackStack()

    override fun popToRoot() {
        while (true) {
            if (!navController.popBackStack()) {
                break
            }
        }
        navigateResId(navController.graph.startDestinationId)
    }

    private fun navigate(command: NavDirection) {
        when (command) {
            is NavDirection.Direction -> navigateDirection(command.directions,
command.options)
            is NavDirection.Replace -> navigateWithReplace(command.directions)
            is NavDirection.ResID -> navigateResId(command.resID, command.args)
            is NavDirection.BackTo -> popBackStack(command.destinationId,
command.inclusive)
```

КПЗ – 2023

```
        }
    }

    open fun navigateDirection(directions: NavDirections, options: NavOptions? =
null) {
        try {
            navController.navigate(directions, options)
        } catch (e: Exception) {
            catchException(e)
        }
    }

    open fun navigateWithReplace(directions: NavDirections) {
        navigateDirection(directions, replaceOption())
    }

    open fun navigateResId(@IdRes resId: Int, args: Bundle? = null) {
        try {
            navController.navigate(resId, args, navOptions {
                getAnimBuilder()
            })
        } catch (e: Exception) {
            catchException(e)
        }
    }

    open fun popBackStack(@IdRes destinationId: Int, inclusive: Boolean) =
        navController.popBackStack(destinationId, inclusive)

    private fun replaceOption(): NavOptions? {
        val currentID = navController.currentDestination?.id ?: return null
        return navOptions {
            popUpTo(id = currentID, popUpToBuilder = { inclusive = true })
        }
    }

    private fun catchException(e: Exception) {
        logger.log(
            message = "${logger.getConfiguration().className}:
${Router.UNKNOWN_ROUTE}",
            configuration = logger.getConfiguration().copy(this),
            throwable = e
        )
    }

    private fun getAnimBuilder(): AnimBuilder {
```

КОПІЯ 2023

```
        val builder = AnimBuilder()
        with(builder) {
            popEnter = R.anim.nav_default_pop_enter_anim
            popExit = R.anim.nav_default_pop_exit_anim
            enter = R.anim.nav_default_enter_anim
            exit = R.anim.nav_default_exit_anim
        }
        return builder
    }
}
```

КбПЗ_2023

Файл AndroidLogger.kt – модуль компоненту для журналювання подій

```
package com.gahov.plates_recognition.arch.component.logger

import android.util.Log
import com.gahov.domain.entity.component.logger.Level
import com.gahov.domain.entity.component.logger.Logger
import
com.gahov.domain.entity.component.logger.configuration.LoggerConfiguration
import
com.gahov.plates_recognition.arch.component.logger.configuration.DefaultLoggerCo
nfiguration

class AndroidLogger(
    private var configuration: LoggerConfiguration =
DefaultLoggerConfiguration()
) : Logger {

    override fun getConfiguration(): LoggerConfiguration = configuration

    override fun setConfiguration(configuration: LoggerConfiguration) {
        this.configuration = configuration
    }

    override fun log(
        level: Level,
        message: String?,
        throwable: Throwable?,
        configuration: LoggerConfiguration
    ) {
        if (configuration.isEnabled) {
            val logMessage = message ?: EMPTY_MESSAGE
            when (level) {
                Level.Debug -> d(
                    message = logMessage,
                    exception = throwable,
                    configuration = configuration
                )

                Level.Info -> i(
                    message = logMessage,
                    exception = throwable,
                    configuration = configuration
                )

                Level.Warning -> w(
```

КБПЗ_2023

```
                    message = logMessage,
                    exception = throwable,
                    configuration = configuration
                )

                Level.Error -> e(
                    message = logMessage,
                    exception = throwable,
                    configuration = configuration
                )
            }
        }
    }

    private fun d(message: String, exception: Throwable?, configuration:
LoggerConfiguration) {
        Log.d(configuration.className, message, exception)
    }

    private fun i(message: String, exception: Throwable?, configuration:
LoggerConfiguration) {
        Log.i(configuration.className, message, exception)
    }

    private fun w(message: String, exception: Throwable?, configuration:
LoggerConfiguration) {
        Log.w(configuration.className, message, exception)
    }

    private fun e(message: String, exception: Throwable?, configuration:
LoggerConfiguration) {
        Log.e(configuration.className, message, exception)
    }

    companion object {
        private const val EMPTY_MESSAGE = ""
    }
}
```

КОПЗ – 2023

Файл CarSearchBottomDialogFragment.kt – модуль для зчитування зображення з камери мобільного телефону та передачі його на обробку нейронній мережі

```
package com.gahov.plates_recognition.feature.search

import android.annotation.SuppressLint
import android.content.Intent
import android.net.Uri
import android.os.Bundle
import android.os.Handler
import android.provider.MediaStore
import android.view.View
import androidx.core.content.FileProvider
import androidx.core.view.isVisible
import androidx.fragment.app.DialogFragment

import com.gahov.domain.entity.failure.Failure
import com.gahov.plates_recognition.R
import com.gahov.plates_recognition.arch.router.command.Command
import com.gahov.plates_recognition.arch.ui.dialog.BaseBottomSheetFragment
import com.gahov.plates_recognition.databinding.FragmentCarSearchBinding
import com.gahov.plates_recognition.feature.main.MainActivity
import
com.gahov.plates_recognition.feature.search.CarSearchViewModel.Companion.AUTHORI
TY
import
com.gahov.plates_recognition.feature.search.CarSearchViewModel.Companion.LOAD_DE
LAY
import
com.gahov.plates_recognition.feature.search.CarSearchViewModel.Companion.REQUEST
_IMAGE
import com.gahov.plates_recognition.feature.search.command.CarSearchCommand

import com.permissionx.guolindev.PermissionX
import dagger.hilt.android.AndroidEntryPoint
import java.io.File
import java.io.IOException

@Suppress("DEPRECATION")
@AndroidEntryPoint
class CarSearchBottomDialogFragment :
    BaseBottomSheetFragment<CarSearchViewModel, FragmentCarSearchBinding>(
        layoutId = R.layout.fragment_car_search,
        viewModelClass = CarSearchViewModel::class.java
    ) {

    private var takePictureDestinationFile: File? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setStyle(DialogFragment.STYLE_NO_FRAME,
R.style.ThemeOverlay_Material3_BottomSheetDialog)
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        setClickListeners()
    }

    private fun setClickListeners() {
        binding.searchCameraButton.setOnClickListener {
            dispatchTakePictureIntent()
        }
        binding.searchButton.setOnClickListener {
            onLoading(true)
```

КОПІЯ 2023

```
            viewModel.onNewCarDigitsInput(binding.numberInput.text.toString())
        }
    }

    private fun onLoading(isLoading: Boolean) {
        binding.searchProgressBar.isVisible = isLoading
    }

    override fun handleFeatureCommand(command: Command.FeatureCommand) {
        with(command) {
            if (this is CarSearchCommand) {
                when (this) {
                    is CarSearchCommand.OnNetworkError -> displayError(failure)
                    is CarSearchCommand.OnRecognitionResult ->
displayRecognizedPlates(digits)
                }
            } else {
                super.handleFeatureCommand(command)
            }
        }
    }


    private fun displayRecognizedPlates(digits: String) {
        onLoading(false)
        binding.numberInput.setText(digits)
    }

    private fun dispatchTakePictureIntent() {
        PermissionX.init(activity as MainActivity)
            .permissions(listOf(android.Manifest.permission.CAMERA))
            .request { allGranted, _, _ ->
                if (allGranted) {
                    requestCameraIntent()
                } else {
                    viewModel.handleFailure(Failure.Common())
                }
            }
    }


    @SuppressLint("QueryPermissionsNeeded")
    private fun requestCameraIntent() {
        Intent(MediaStore.ACTION_IMAGE_CAPTURE).also { takePictureIntent ->

takePictureIntent.resolveActivity(requireContext().packageManager)?.also {
                // Create the File where the photo should go
                takePictureDestinationFile = try {
                    onLoading(true)
                    viewModel.createImageFile(requireContext())
                } catch (ex: IOException) {
                    viewModel.handleFailure(Failure.Common())
                    null
                }
                // Continue only if the File was successfully created
                takePictureDestinationFile?.also {
                    val photoURI: Uri = FileProvider.getUriForFile(
                        requireContext(), AUTHORITY, it
                    )
                    takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT,
photoURI)
                    startActivityForResult(takePictureIntent, REQUEST_IMAGE)
                }
            }
        }
    }
```

КОПИЯ 2023

```
    @Deprecated("Deprecated in Java")
    override fun onActivityResult(
        requestCode: Int,
        resultCode: Int,
        data: Intent?
    ) {
        super.onActivityResult(requestCode, resultCode, data)
        binding.numberInput.setText("")

        Handler().postDelayed({
            val photoURI: Uri = takePictureDestinationFile?.let {
                FileProvider.getUriForFile(
                    requireContext(), AUTHORITY, it
                )
            }!!
            photoURI.let { viewModel.processImage(requireContext(), it) }
        }, LOAD_DELAY)
    }
}
```

КбПЗ_2023

Файл CarSearchViewModel.kt – модуль для розпізнавання зображення та номера автомобіля на ньому нейронною мережею

```
package com.gahov.plates_recognition.feature.search


import android.annotation.SuppressLint
import android.content.Context
import android.net.Uri
import android.os.Environment
import com.gahov.domain.entity.cars.CarEntity
import com.gahov.domain.entity.common.Either
import com.gahov.domain.entity.failure.Failure
import com.gahov.domain.usecase.carplate.GetRemoteCarInfoUseCase
import com.gahov.domain.usecase.carplate.params.GetCarParams
import com.gahov.plates_recognition.arch.controller.BaseViewModel
import com.gahov.plates_recognition.feature.search.command.CarSearchCommand
import com.google.firebase.ml.vision.FirebaseVision
import com.google.firebase.ml.vision.common.FirebaseVisionImage
import com.google.firebase.ml.vision.text.FirebaseVisionText
import java.io.File
import java.io.IOException
import java.text.SimpleDateFormat
import java.util.Date
import javax.inject.Inject

@Suppress("DEPRECATION")
class CarSearchViewModel @Inject constructor(
    private val loadCarInfoUseCase: GetRemoteCarInfoUseCase
) : BaseViewModel() {

    private fun onResultFailure(failureResult: Failure) {
        handleCommand(CarSearchCommand.OnNetworkError(failureResult))
        handleFailure(failureResult)
    }

    fun processImage(context: Context, photoUri: Uri) {
        val image = FirebaseVisionImage.fromFilePath(context, photoUri)
        val detector = FirebaseVision.getInstance().onDeviceTextRecognizer

        detector.processImage(image)
            .addOnSuccessListener { firebaseVisionText ->
                processResultText(firebaseVisionText)
            }
            .addOnFailureListener {
                handleFailure(Failure.Common(Throwable(message =
PICTURE_HANDLING_ERROR)))
            }
    }

    private fun processResultText(resultText: FirebaseVisionText) {
        if (resultText.textBlocks.size == EMPTY_TEXT) {
            handleFailure(Failure.Common(Throwable(message =
TEXT_HANDLING_ERROR)))
            return
        }
        for (block in resultText.textBlocks) {
            val blockLine = block.lines
            for (i in blockLine) {
                var carPlate = i.text.replace("\\s".toRegex(), "")
                if (carPlate.length == MAX_POSSIBLE_CAR_DIGITS_LENGTH) {
                    carPlate.submitRecognitionAndSearch()
                } else {
                    carPlate = removeExtraSymbolsInDigits(carPlate)
                    if (carPlate.length == MAX_POSSIBLE_CAR_DIGITS_LENGTH) {
                        carPlate.submitRecognitionAndSearch()
```

КПІ 2023

```
                    } else {
                        handleFailure(Failure.Common(Throwable(message =
WRONG_PLATED_INPUT_ERROR)))
                    }
                }
            }
        }
    }

    private fun String.submitRecognitionAndSearch() {
        CarSearchCommand.OnRecognitionResult(digits = this)
        searchCarInputDigits(GetCarParams(this))
    }

    private fun removeExtraSymbolsInDigits(fullCarDigits: String): String {
        var carPlate = fullCarDigits

        if (fullCarDigits.length == FULL_CAR_DIGITS_LENGTH &&
fullCarDigits.contains(UA_SYMBOL)) {
            carPlate = fullCarDigits.replace(UA_SYMBOL, "")
        }
        return carPlate
    }

    fun onNewCarDigitsInput(carDigits: String) {
        if ((carDigits.chars()).count().toInt() ==
MAX_POSSIBLE_CAR_DIGITS_LENGTH) {
            searchCarInputDigits(GetCarParams(carDigits))
        }
    }

    private fun searchCarInputDigits(param: GetCarParams) {
        launch {
            when (val result = loadCarInfoUseCase.execute(param = param)) {
                is Either.Right -> onResultSuccess(result = result.success)
                is Either.Left -> onResultFailure(result.failure)
            }
        }
    }

    private fun onResultSuccess(result: CarEntity) {
        navigateDirection(
            CarSearchBottomDialogFragmentDirections.actionCarSearchToCarDetails(
                carData = result,
                carDigits = result.digits
            )
        )
    }

    @SuppressLint("SimpleDateFormat")
    @Throws(IOException::class)
    fun createImageFile(context: Context): File {
        val timeStamp: String =
SimpleDateFormat(IMAGE_DATE_FORMAT).format(Date())
        val storageDir: File? =
context.getExternalFilesDir(Environment.DIRECTORY_PICTURES)
        return File.createTempFile(
            JPEG_PREFIX + "${timeStamp}_", JPEG_SUFFIX, storageDir
        ).apply {
            absolutePath
        }
    }

    companion object {
        const val REQUEST_IMAGE = 100
        const val STORAGE = 1
        const val FLOAT = "%.2f"
        const val LOAD_DELAY = 1500L
        const val AUTHORITY = "com.gahov.plates_recognition.provider"
```

КОПІЯ 2023

```
        private const val JPEG_SUFFIX = ".jpg"
        private const val JPEG_PREFIX = "JPEG_"
        private const val MAX_POSSIBLE_CAR_DIGITS_LENGTH = 8
        private const val FULL_CAR_DIGITS_LENGTH = 10
        private const val EMPTY_TEXT = 0
        private const val PICTURE_HANDLING_ERROR = "Помилка обробки зображення!"
        private const val TEXT_HANDLING_ERROR = "Помилка обробки номеру!"
        private const val WRONG_PLATED_INPUT_ERROR = "Невірний номер!"
        private const val IMAGE_DATE_FORMAT = "yyyyyMMdd_HHmmss"
        private const val UA_SYMBOL = "UA"
    }
}
```

КбПЗ_2023