Київський національний університет імені Тараса Шевченка
Факультет радіофізики, електроніки та комп’ютерних систем
Кафедра комп’ютерної інженерії

СТВОРЕННЯ МОДУЛЯ ПОЗИЦІОНУВАННЯ В СИСТЕМІ АВТОМАТИЗАЦІЇ ЗАДАЧ ЛІЗИНГУ

Дипломна робота магістра
студента 2 року навчання
спеціальність: 123 «Комп'ютерна інженерія»
Данила КОСТЮЧЕНКА

Науковий керівник
канд. фіз..-мат. наук Дмитро ІВАНЕНКО,
асистент кафедри комп’ютерної інженерії

Рецензент
проф. каф. аглебри Георгій ШЕВЧЕНКО

До захисту допускаю:

Завідувач кафедрою
Ганна КАРЛАШ /
Анатолій ВЕКЛИЧ /
Ігор АНІСІМОВ /
Сергій РАДЧЕНКО /
Юрій БОЙКО /

Ухвалено на засіданні кафедри “____” _____ 2022 р., протокол № ___

Київ - 2022

РЕФЕРАТ

Обсяг роботи 73 сторінки, 20 ілюстрацій, 2 таблиці, 16 джерел посилань, 4 додатки.

БАЗА ДАНИХ, ВЕБ-ЗАСТОСУВАННЯ, ІНТЕРФЕЙС КОРИСТУВАЧА, 2D ВІЗУАЛІЗАЦІЯ, FABRIC.JS, CLIENT-SERVER ARTHITECTURE, ANGULAR, CONTINIOUS DEPLOYMENT, NODEJS.

Об’єктом даної роботи є візуалізація та взаємодія з 2-d моделями автомобільних терміналів, а предметом – програмний засіб для візуалізації та можливості зміни 2-d моделей терміналів.

Метою даної роботи є створення веб-додатку для побудови та відображення 2-d моделі терміналу з можливістю розширення для забезпечення навігації за допомогою мобільного пристрою, а також дослідити можливості 2-d візуалізації у веб-застосунках та побудови інфраструктури для неперервної розгортки.

Методи розроблення: метод ручного створення програмного продукту за допомогою платформи розробки веб-застосувань. Інструменти розроблення: безкоштовне, вільно поширюване інтегроване середовище розробки Visual Studio Code, мова програмування JavaScript, бібліотека Node.js для реалізації серверної частини, фреймворк Angular для створення клієнтського додатку.

Результати роботи: було проведено аналіз можливостей відображення 2-d об'єктів у веб-застосунках. Розроблено та удосконалено інтегровану систему, до якої входить веб-сервіс для зберігання та обробки даних та односторінковий веб-інтерфейс, який дозволяє користувачам будувати та маніпулювати 2-d моделлю автомобільного терміналу.

ЗМІСТ

С.

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

GUI – Графічний інтерфейс користувача

MVC **–** Model View Controller

HTML – Hypertext Markup Language

CSS – Cascading Style Sheets

JS – JavaScript

HTTP – Hypertext Transporting Protocol

API – Application Programming Interface

CI – Continuous Integration

CD – Continuous Deployment

ВСТУП

Немає ніякого сумніву у тому, що за останнє десятиріччя технології телекомунікаційної навігації, проникли у життя людей та міцно закріпилися у всіх сферах. Адже з безперервно зростаючими обсягами завдань пов’язаними з логістикою, які потрібно вирішувати у різноманітних галузях, використання автоматизованих інструментів значно покращує ефективність бізнес-процесів.

Масштабування будь-якого бізнесу кидає нові виклики для менеджменту, багато з який може вирішити діджиталізація. Діджиталізація бізнесу - рушійна сила, що сприяє його просуванню. Її основні переваги - економія часу і підвищення продуктивності за рахунок автоматизації виробництва та інших внутрішніх процесів компанії; оптимізація та покращення комунікацій - як внутрішніх, так і зовнішніх. Сфера лізингу у автомобільній індустрії не є винятком. Щоденно, процеси, які потребують багато людського ресурсу, можуть слугувати джерелом значних збитків, через власну недосконалість та фактору людської помилки. Влучна автоматизація здатна значно зменшити операційні ресурси.

Комплексне та якісне вирішення науково-прикладної проблеми позиціонування та навігації в середині закритих будівель набуває все більшої актуальності. Слід зазначити, що більша частина відомих інформаційно-технологічних розроблень за означеним профілем спрямована на забезпечення підтримки і допомоги працівникам та відвідувачам в орієнтуванні в офісних будівлях складної, багаторівневої і розгалуженої структури.

Особливо часто з проблемами навігації стикається автомобільна індустрія. Логістика для великих автомобільних терміналів, яка виконується у ручному

режимі часто призводить до значних грошових втрат, через відносно велику кількість помилок за рахунок людського фактору. Для подолання недоліків ручного керування та зменшення витрат було вирішено створити систему автоматизії задач лізингу, основним модулем якої є модуль позиціонування.

Модуль позиціонування передбачає автоматизацію процесів всередині автомобільних терміналів. Для забезпечення належного рівня автоматизації доцільним буде вирішення проблеми навігації всередині закритих приміщень.

Значні ресурси наукові дослідники, так і великі компанію інвестують для розроблення інформаційно-технічних рішень, що визначають поточне місцезнаходження користувача на відкритому просторі, водночас не достатньо часу та уваги приділяють розробкам, для вирішення проблеми позиціонування та навігації всередині закритих будівель[1].

Для побудови якісних програмних продуктів, що використовують інформацію про місце розташування користувача всередині закритих будівель, можливе застосування широкого спектру взаємопов’язаних інформаційно-комунікаційних технологій.

У списку нижче подано телекомунікаційні технології, що використовуються для реалізації навігації в закритих приміщеннях за допомогою мобільного пристрою [1]:

- Навігація за допомогою WI-FI модулів;
- GSM навігація;
- Навігація з використанням сітки BLE(Bluetooth Low Energy) маяків;
- Indoor GPS;
- Технологія ідентифікації за допомогою радіочастот (RFID);

Майже для усіх представлених вище методів, потрібна система для побудови топології закритого приміщення. Для коректної навігації, необхідно знати розміщення усіх наявних об’єктів у певному масштабі, що надасть можливість зберегти координати та використовувати їх для обчислення положення користувача у просторі.

В даній роботі буде розроблена система для побудови топології автомобільного терміналу, яка буде використовуватись при реалізації навігації за допомогою BLE(Bluetooth Low Energy) маяків.

РОЗДІЛ 1. ОГЛЯД ТЕХНОЛОГІЙ АУТЕНТИФІКАЦІЇ

1.1 Застосування аутентифікації

Ми стикаємося з аутентифікацією досить часто: на роботі, при перегляді сайтів, при користуванні своїм смартфоном. Потреба у захисті даних набуває все більш значущих масштабів з кожним днем. Крадіжка даних для аутентифікації несе велику загрозу, як для компаній, які можуть зазнати серйозних наслідків як з точки зору фінансів, так і репутації, а також для користувачів, які можуть бути причетними до незаконних дій, вчинених атакуючою особою. Володіючи, наприклад, доступом до вашого мобільного телефону та маючи ваші паспортні дані, зловмисник легко може зняти з вашого банківського рахунку усі кошти. Маючи доступи до великих інформаційних систем, таких як: поштові провайдери, соціальні мережі, банківські системи, системи документообігу, зловмисних має доступ до так званої «чутливої» інформації, яка відноситься до великого кола людей, та може вільно використовувати її для шантажу, або для перепродажу на чорному ринку.

Аутентифікація — процес підтвердження інформаційною системою, що користувач є тим, ким він себе ідентифікував.

Авторизація — встановлення переліку ресурсів до яких дозволено доступ користувачу.[2]

Однією з найважливіших умов розробки безпечної інформаційної системи є впровадження безпечного і надійного методу автентифікації користувачів, який не дасть змоги зловмисникам отримати доступ до закритого ресурсу.

Вибір прийнятної моделі автентифікації для інформаційної системі має ґрунтуватися, перш за все, міркуваннями безпеки та у наступну чергу зручністю використання. Якщо для доступу до веб-ресурсу, потрібно буде проходити

через надмірні перевірки, це може значно знизити рівень користувацького досвіду.

1.2 Аналіз існуючих моделей автентифікації

1.2.1 Базова автентифікація

Це найпростіший спосіб захисту вашого API. Користувач вводить ім’я та пароль, після чого введені дані відправляються у зашифрованому вигляді на сервер. У стандартній специфікації HTTP є поле заголовка авторизації, яке можна використовувати для цієї мети. Ім’я користувача та пароль закодовані кодуванням Base64. Будь-хто, хто перехопить ці дані на шляху до сервера, побачить довгий ряд випадкових символів. Проте кодування Base64 можна легко декодувати менш ніж за секунду.

Базова автентифікація — це найпростіший тип автентифікації, яким зловмисники можуть зловживати. Використовуючи перехоплювач запитів можна скопіювати закодований логін та пароль і використовуючи декодер, повернути ваше ім’я користувача та пароль.

1.2.2 Аунтефікація за допомогою токенів

У даній модель користувач надає своє ім’я користувача та пароль (облікові дані), а сервер генерує токен на основі цих облікових даних. Після чого сервер надсилає цей токен користувачеві, а також зберігає його копію в базі даних[2].

У результаті користувач обміняв облікові дані на згенерований сервером токен та тепер йому не потрібно надсилати облікові дані для входу з кожним запитом. Замість цього він просто надсилає закодований токен, який при проходженні певного часу потребує оновлення.

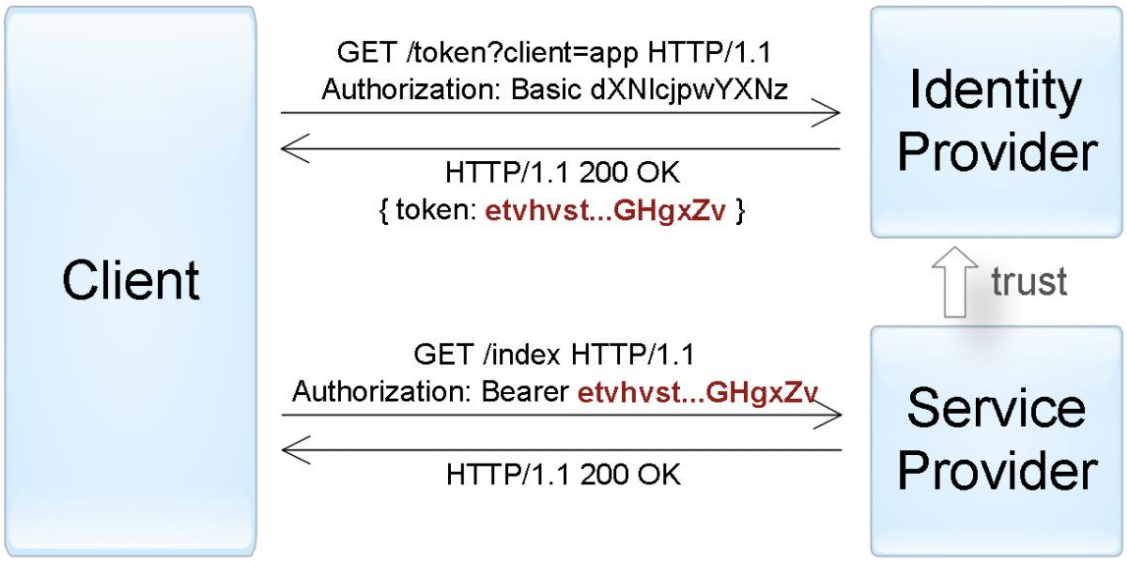

Рисунок 1.1 Схема аутентифікації клієнта за допомогою токена, переданого за допомогою Bearer схеми.

Найпопулярнішим форматом токенів є JSON Web Tokens (JWT). Поля токена є парами ключ/значення. Ключі називаються claims. Claims, по суті, повідомляють API, що мають означати значення. Список усіх стандартних claims можна знайти в RFC 7519[3].

Вигляд декодованого JWT токена подано нижче:

```
{
  "iss": "http://www.danylokostiuchenko.com",
  "name": "first last",
  "scope": "read write",
  "email": "danylokostiuchenko@gmail.com",
}
```

1.2.3 Аунтефікація за допомогою ключів доступу (Api Keys)

Використання ключів API – це спосіб автентифікації програми, яка отримує доступ до API, без посилання на реального користувача. Програма додає ключ до кожного запиту API, і API може використовувати ключ для ідентифікації програми та авторизації запиту. Спосіб надсилання ключа залежить від API. Деякі API використовують параметри запиту, деякі

використовують заголовок Authorize, деякі використовують параметри тіла запиту, тощо

Переваги:

- Клієнтам відносно легко використовувати ключі API. Незважаючи на те, що більшість провайдерів використовують різні методи, додати ключ до запиту API досить просто.

Недоліки:

- Ключ API ідентифікує лише програму, а не користувача програми. Часто буває важко зберегти ключ в таємниці.
- API ключі не стандартизовані, тобто кожен API має унікальну реалізацію.

1.2.4 Аунтефікація за допомогою OAuth2

OAuth 2.0, що означає «Відкрита авторизація», — це стандарт, розроблений для того, щоб дозволити веб-сайту або програмі отримати доступ до ресурсів, розміщених іншими веб-програмами від імені користувача. Він замінив OAuth 1.0 у 2012 році і тепер є де-факто галузевим стандартом для онлайн-авторизації. OAuth 2.0 надає згоден доступ і обмежує дії, які клієнтська програма може виконувати з ресурсами від імені користувача, ніколи не повідомляючи облікові дані користувача. [4]

Використовуючи OAuth 2.0, запити на доступ ініціюються клієнтом, наприклад, мобільним додатком, веб-сайтом, настільною програмою тощо. Запит, обмін та відповідь токенів виконуються за таким загальним процесом:

1. Користувач ініціює контакт з Клієнтом.
2. Клієнт робить переадресацію на сторінку авторизації та надає змогу ввести дані для подальшої авторизації.
3. Користувач вводить дані авторизації.
4. Клієнт передає зашифровані дані авторизації разом з OAuth ключами Серверу Авторизації.
5. Сервер Авторизації повертає Токен Доступу.
6. Клієнт використовуючи Токен Доступу створює формує запит до Серверу Ресурсів для отримання ресурсів.
7. Сервер Ресурсів валідує Токен Доступу та надає ресурс Клієнту.
8. Користувач отримує доступ до ресурсу.

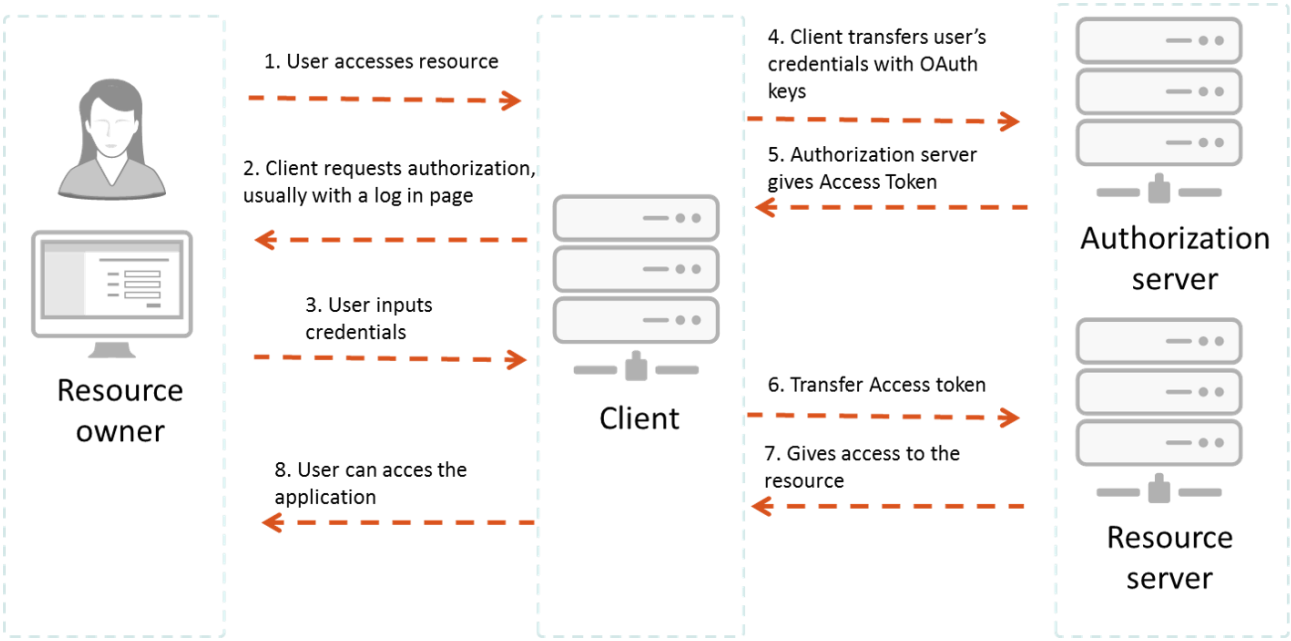

Рисунок 1.2 Схема аутентифікації клієнта за фреймворку OAuth2.

У рамках дипломної роботи поставлені завдання передбачають впровадження аунтентифікації на основі фреймворку OAuth2.

РОЗДІЛ 2. ОГЛЯД ТЕХНОЛОГІЙ СТВОРЕННЯ ВЕБ-ДОДАТКІВ

2.1 Найбільш популярні фреймворки та бібліотеки

Фреймворки надають розробникам базову основу, необхідну для створення програм JavaScript. Кожен фреймворк JavaScript призначений для різних цілей. JavaScript є непохитним вибором для веб-розробки, і багато його фреймворків розроблені саме для цього напряму.

Створення веб-програм і веб-сайтів від початку до кінця може зайняти чимало роботи. Веб-фреймворки — або насправді, фреймворки JavaScript — використовують той факт, що кожен веб-сайт і веб-програма мають спільні функції.

Фреймворки JavaScript компілюють попередньо написаний код JavaScript, який включає у собі вирішення рутиних задач у веб-програмуванні, що в кінцевому підсумку полегшує розробку.

Відповідно до трендів NPM, React, Angular і Vue.js є найбільш завантажуваними фреймворками та бібліотеками JavaScript.

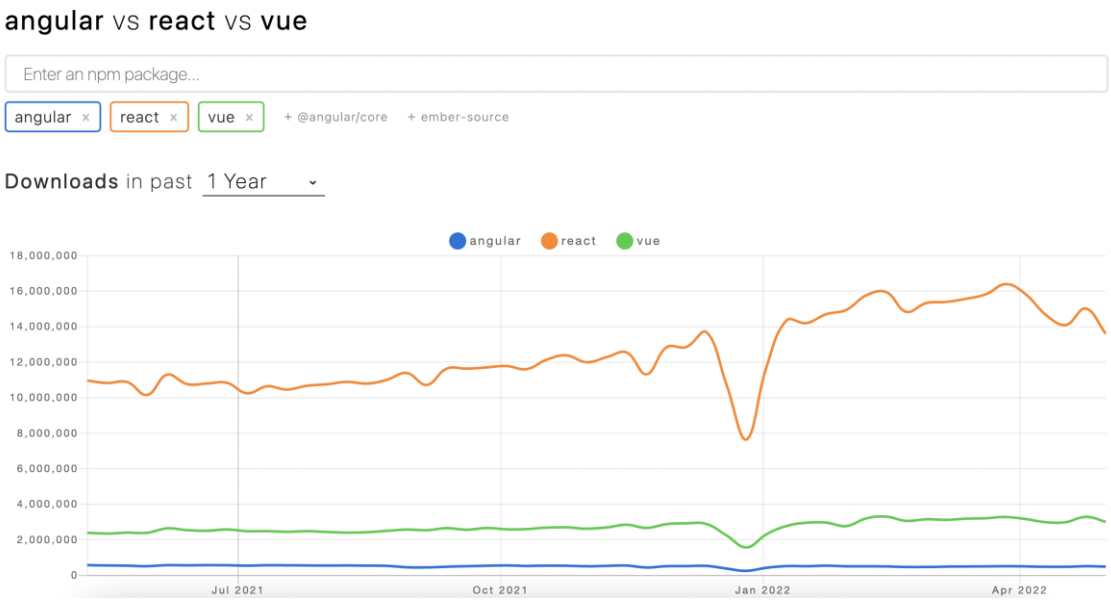

Рисунок 2.1 Порівняльна діагграма кількості завантажень React, Angular і Vue.js.

2.2 Фреймворк Angular

Angular був розроблений Google у 2009 році як фреймворк динамічного веб-додатка з відкритим вихідним кодом Міско Хевері та Адамом Абронсом. Безсумнівно, Angular є найбільшим серед усіх інших. Насправді, іноді його називають платформою, а не фреймворком. Після цього Angular пропонує готову підтримку багатьох речей, завдяки чому розробники мають повний контроль над інтерфейсом користувача, реакцією на введення користувача, перевіркою введених даних у формах, маршрутизацією та керуванням станом, надсилаючи запити AJAX HTTP тощо [5].

Завдяки широкому вибору вбудованих функцій Angular дозволяє створювати, керувати та тестувати свою програму набагато ефективніше. Крім того, це широко популярний фреймворк для front-end розробки, тому такі гігантські організації, як Google, Forbes, Whatsapp та інші 500 компаній, виявили інтерес до цього фреймворку. Крім того, ось плюси та мінуси використання Angular для проекту розробки додатків

2.3 Фреймворк Vue.js

Vue.js – це найбільш обговорювана та швидко зростаюча платформа на основі JavaScript, ініційована колишнім співробітником Google Еваном Ю. Коли Еван Ю працював там, у нього виникла необхідність швидко побудувати прототип складного інтерфейсу і був потрібен інструмент, щоб уникнути написання повторюваного HTML. AngularJS та Backbone були занадто громіздкі для прототипування, а React лише починав розроблятись. Тому Еван створив свій фреймворк. З того часу Vue еволюціонував і дає змогу писати не тільки

прототипи, а й складні вебзастосунки. Це фреймворк, який знаходиться десь між React і Angular. Vue не такий великий, як Angular, він містить такі функції, які роблять його конкурентоспроможним з React і Angular. Поточну стабільну версію 3.2.26 випустили у квітні 2022 року.

Vue.js — це прогресивна платформа, яка дозволяє створювати прогресивні односторінкові програми. Як і Angular і React, Vue також розробляє інтерфейси користувача шляхом поєднання компонентів, які можна використовувати повторно. Але крім цього, Vue дає вам більше React і менше Angular, тому він перевершує Angular [6].

2.4 Бібліотека React

React – це бібліотека JavaScript, зосереджена на створенні декларативних інтерфейсів користувача (UI) з використанням концепції на основі компонентів. Він використовується для обробки шару перегляду та може використовуватися для веб- та мобільних додатків. Основна мета React — бути розширеним, швидким, декларативним, гнучким і простим.

React - це не фреймворк, це конкретна бібліотека. Пояснення цьому полягає в тому, що React займається лише відтворенням інтерфейсів інтерфейсу користувача і зберігає багато речей на розсуд окремих проектів. Стандартний набір інструментів для створення програми за допомогою ReactJS часто називають стеком.

2.5 Порівняльна характеристика

2.5.1 Міграція

Коли справа доходить до створення продукту, одне з основних завдань полягає в тому, щоб обрати фреймворк, який не потребує багато змін бази коду для переходу на нього та переходу між версіями фреймворків. З огляду на це,

Angular отримує оновлення кожні шість місяців. React і Vue забезпечують більш рідкі оновлення.

Angular. Безсумнівно, Angular випускає оновлення частіше, ніж React і Vue, приблизно кожні 6 місяців. Однак, для великих оновлень цей термін складає 1 рік.

React. React надає необхідні скрипти переходу - React codemod, допомагають вам переміщатися зручніше, ніж Angular і Vue.

Vue. З Vue міграція додатків стає набагато простішою, оскільки при переході гарантовано зребігається зворотня сумісність у 90% усієї бази коду, якщо перехід відбувається з однієї основної версії на іншу, наприклад, з 1.x на 2. Крім того, Vue надає допоміжний інструмент міграції.

2.5.2 Масштабованість

Що стосується frond-end розробки, то масштабованість здебільшого відноситься до здатності постійно підтримувати розширювану функціональність. Передбачається, що додатки збільшуються в розмірах і складності, і платформа розробки повинна підтримувати таке зростання.

Спільнота розробників майже одностайно стверджує, що як Angular, так і React вирішують завдання, коли мова йде про створення масштабованих додатків. Angular забезпечує масштабованість завдяки своїй модульній структурі розробки, тоді як React використовує компонентний підхід з однаково чудовими результатами.

У питаннях масштабованості Vue.js приголомшливо програє, оскільки використовує синтаксис на основі шаблонів. У великому додатку повторно використовувати шаблони стає все важче, ніж компоненти JavaScript.

2.5.3 Спільнота розробників

Опитування розробників Stack Overflow 2017 року назвало React найулюбленішим фреймворком серед розробників. Будучи творінням команди Facebook, він користується неперевершеною популярністю та підтримкою. Зараз спільнота розробників React налічує понад тридцять тисяч учасників, усі вони вносять свої знання та досвід у її розвиток [7].

Angular традиційно цінується за свої переваги і може похвалитися досить значною підтримкою спільноти. Хоча статистика Bestof.js показує, що початкова версія Angular – AngularJS, або Angular 1 – все ще має набагато більше послідовників, ніж її пізніші версії.

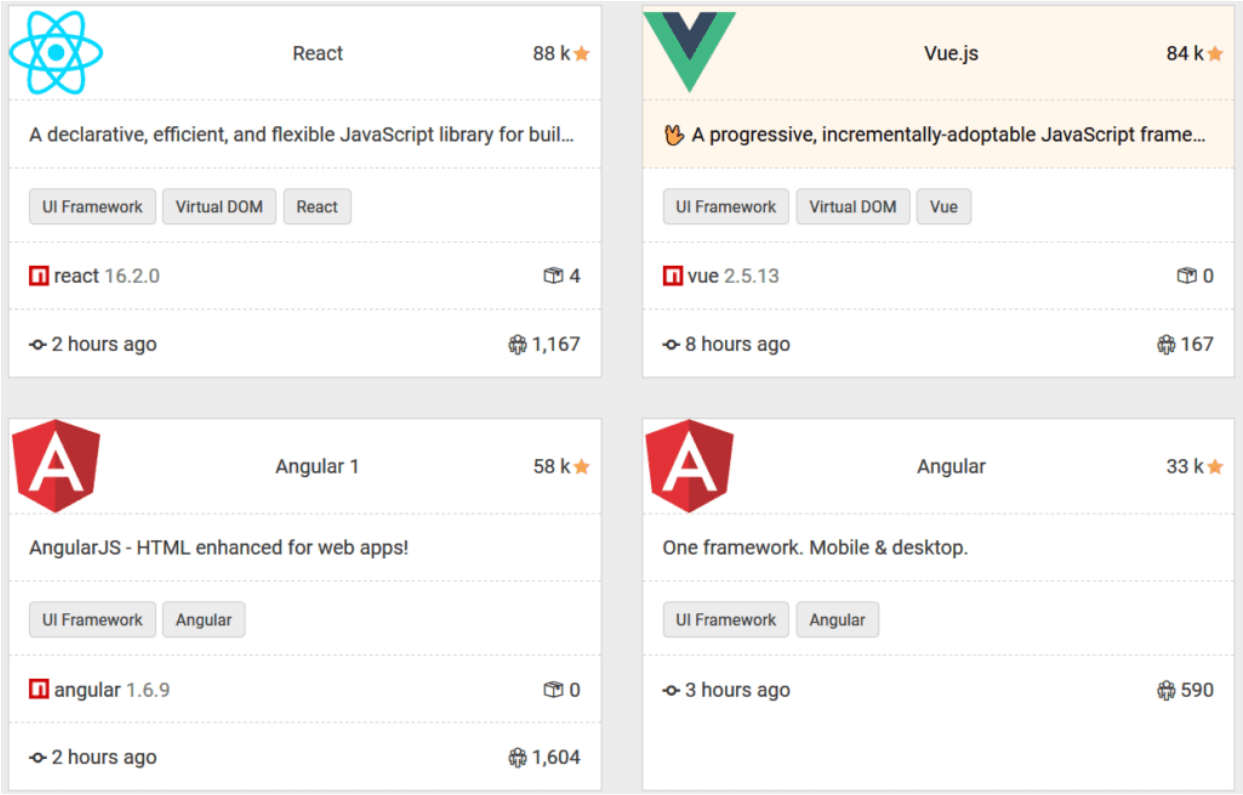

Рисунок 2.2 Порівняння спільнот розробників Angular, Vue та React.

Та ж статистика доводить, що, незважаючи на свої переваги, Vue.js ще далеко не дуже популярний серед розробників.

Чим більше людей співпрацює над функціональними можливостями фреймворка, тим краще він стає і, що ще важливіше, тим більшою стає його бібліотечна колекція. З React ви обов’язково знайдете багаторазові компоненти всіх видів, надані однолітками-розробниками.

2.5.4 Загальне порівняння

Безсумнівно, всі наведені фреймворки та біболіотеки для розробки веб-додатків по-своєму є основоположними. Однак єдиного рішення для всіх випадків не існує. Перш за все, навіть після того, як ви знаєте плюси і мінуси кожної платформи, вибір між цими технологіями — важка робота. Через це було побудовано порівняльну таблицю:

Технологія	**React**	**Angular**	**Vue**
Тип	Бібліотека	Фреймворк	Бібліотека
Випуск першої версії	Березень 2013	Вересень 2016	Лютий 2014
Ідеально підходить для	Сучасних веб-додатків та нативно скомпільовані додатки для iOS та Andriod	Побудови великих веб-додатків з великим об’ємом функціоналу та побудови додатки рівня корпорацій	Односторінкових веб-додатків у яких планується перевикористання написаних HTML-шаблонів
Модель	Побудований на основі Virtual DOM	Побудований на основі DOM API	Побудований на основі Virtual DOM

Мова програмування	Javascript	Typescript	Javascript
Зручність для розробника	Забезпечує гнучке середовище розробки	Фреймворк у якого наперед задана архітектура додатку, що дозволяє зосередитись на більш прикладних задачах	Дозволяє сфокусуватись на написанні функціоналу перевикористовуючи HTML-шаблони
Спільнота розробників	Середня	Найбільша	Найменша

Висновки:

Все одно слід відповісти на велике питання – коли вибрати кожну з трьох технологій? Як максимально використати їх переваги?

Виходячи з основних переваг і недоліків кожного з трьох фреймворків, можна виділити розробили наступні критерії, які допоможуть обрати найбільш оптимальну платформу для проекту:

Angular:

- Є намір розробити дуже великий і складний проект (наприклад, створення корпоративних додатків за допомогою Angular);
- Потрібна легка та надійна масштабованість;
- Є намір фокусуватися на більш прикладних задачах, використовуючи передбачену одно варіативну архітектуру додатку Angular;

React:

- Проект використовує багаторазові компоненти;
- Інтерфейс проекту не надто складний;
- Продуктивність і масштабованість мають вирішальне значення;

Vue:

- Обсяг проекту невеликий.
- Потрібна висока продуктивність.

У рамках дипломної роботи поставлені завдання передбачають впровадження функціоналу на основі фреймворку Angular.

РОЗДІЛ 3. ОГЛЯД МОЖЛИВОСТЕЙ 2-D ВІЗУАЛІЗАЦІЇ У ВЕБ-ДОДАТКАХ

Загальновідомою і найбільш розвиненою технологією для 2-d візуалізації у веб-додатках є HTML елемент Canvas. Canvas — це елемент HTML, який можна використовувати для малювання графіки за допомогою сценаріїв (зазвичай JavaScript). За допомогою нього можна малювати графіки, об’єднувати фотографії або створювати анімації [8].

Елемент Canvas створює поверхню малювання фіксованого розміру, яка надає один або кілька контекстів візуалізації, які використовуються для створення та керування показаним вмістом. Canvas API надає два контексти малювання: 2D і 3D.

Canvas — це API растрової графіки — де здійснюється маніпулювання елементами на рівні пікселів. Це означає, що базове програмне забезпечення не знає модель, яка використовується для відображення контексту — чи це коло, чи прямокутник. Виходячи з цього, для того щоб намалювати складну фігуру, потрібно буде описати описати кожну точку, лінію, грань і так далі. Виходячи з цього є сенс дослідити можливості сторонніх бібліотек для 2-d візуалізації на основі Canvas API.

3.1 D3.js

D3.js — це бібліотека JavaScript, яка використовується для створення інтерактивних візуалізацій у браузері. Бібліотека D3.js дозволяє маніпулювати елементами веб-сторінки в контексті набору даних. Ці елементи можуть бути елементами HTML, SVG або Canvas, і їх можна вводити, видаляти або редагувати відповідно до вмісту набору даних [9].

D3.js є одним із найкращих фреймворків візуалізації даних, і його можна використовувати для створення простих і складних візуалізацій разом із взаємодією з користувачем та ефектами переходу. Нижче наведено основні особливості:

- Надзвичайно гнучкий;
- Простий у використанні та швидкий;
- Підтримує великі набори даних;
- Можливість повторного використання коду;
- Має широкий спектр функцій генерування кривих;

3.2 EaselJS

Бібліотека EaselJS надає інструменти для роботи з HTML5 Canvas, включаючи повний ієрархічний список відображення та допоміжні класи, що значно полегшують роботу з 2D-графікою[10]. EaselJS надає прямі рішення для роботи з насиченою графікою та інтерактивністю. EaselJS значно спрощує розробку додатків, використовуючи синтаксис та архітектуру, дуже схожу на мову ActionScript 3.0.

EaselJS надає можливість керувати багатьма типами графічних елементів (векторні форми, растрові зображення, спрайт-таблиці, тексти та елементи HTML), а також він підтримує сенсорні події, анімацію та багато інших функцій.

3.3 PixiJS

PixiJS — це бібліотека візуалізації, яка дозволяє створювати насичену інтерактивну графіку, багатопратформенні програми та ігри без необхідності занурюватися в API WebGL або мати справу з сумісністю браузерів і пристроїв. PixiJS має повну підтримку WebGL та Canvas HTML5 [11].

Основні особливості:

- Підтримка багатьох платформ;
- Рендерер WebGL (з автоматичним інтелектуальним пакетуванням);
- Повний граф об’єктної моделі наявних елементів;
- Мультисенсорна взаємодія;
- Багаторядковий текст;

3.4 Fabric.js

Fabric.js — легка бібліотека JavaScript, яка пропонує інтерактивну модель об’єкта поверх елементу Canvas. Вона забезпечує інтерактивне середовище для роботи з Canvas HTML5. Fabric.js надає можливість створювати об’єкти/форми на Canvas, як з простих геометричних фігур так і достатньо складних [12].

Основні особливості:

- Підтримка анімації;
- Можливість змінювати фотографії та геометричні фігури (змінювати положення у просторі, зміна розміру, поворот навколо своєї осі і т.д);
- Функціонал групування який дозволяє формувати групу з окремих елементів та маніпулювати ними, як одним цілим.
- Підтримка подій DOM API.
- Функціонал для серіалізації/десеріалізації наявної конфігурації canvas у формат JSON/SVG, що дозволяє зберігати малюнок у базу даних з подальшим відтворенням.

3.5 Висновок

Завдяки об'єктно-орієнтованому підходу для створення та маніпулюванням елементами Canvas бібліотеки Fabric.js, можна побудувати

зручну систему класів та описати поведінку усіх складових автомобільного терміналу, побудова моделі якого є першочерговою задачею цією дипломної роботу. Ще одним фактором вибору Fabric.js є функціонал серіалізації/десеріалізації «з коробки» для подальшого збереження схеми до бази даних та відтворення її на Canvas.

Отже, для реалізації 2-d конструктору найоптимальнішим вибором буде бібліотека Fabric.js.

РОЗДІЛ 4. ОГЛЯД ХМАРНИХ ПЛАТФОРМ ДЛЯ ЗАБЕЗПЕЧЕННЯ НЕПЕРЕРВНОГО РОЗГОРТАННЯ

4.1 Вступ

Amazon, Microsoft та Google займають лідируючі позиції на ринку хмарних провайдерів, що надають безпечні, гнучкі та надійні хмарні послуги. Їх відповідні хмарні платформи, AWS, Azure та GCP, надають користувачам доступ до обчислюваних ресурсів та аналітичних інструментів, а також до зберігання даних, серверів, ПО тощо.

Amazon Web Services (AWS), Google Cloud Platform (GCP) і Azure пропонують інфраструктуру як послугу (IaaS). Інфраструктура, яку надають ці хмарні постачальники, призначена для розміщення інших послуг, які можуть бути комерційними або відкритими.

4.2 Наявні хмарні рішення

4.2.1 Amazon Web Services

AWS (Amazon Web Services) — це комплексна платформа хмарних обчислень, що розвивається, надана Amazon, яка включає поєднання інфраструктури як послуги (IaaS), платформи як послуги (PaaS) і пакетного програмного забезпечення як послуги (SaaS). Служби AWS можуть запропонувати такі інструменти, як обчислювальні потужності, бази даних і послуги неперервного розгортування.

AWS було запущено в 2006 році з внутрішньої інфраструктури, створеної Amazon.com для роботи з онлайн-роздрібними операціями. AWS була однією з перших компаній, яка представила модель хмарних обчислень з оплатою по мірі використання, яка масштабується, щоб забезпечити користувачам обчислення, сховище або пропускну здатність за потреби.

4.2.2 Microsoft Azure

Azure — це платформа хмарних обчислень, яка може забезпечити все, що потрібно бізнесу для віртуального виконання всіх або частини його обчислювальних операцій, включаючи сервери, сховище, бази даних, мережу, аналітику тощо.

Microsoft Azure підтримує розгортання приватної, загальнодоступної та гібридної хмари. Надійні служби інформаційної безпеки Azure (InfoSec) забезпечують загальну безпеку, сховище, базу даних і мережу, керування ідентифікацією та доступом, резервне копіювання та аварійне відновлення (DR).

Microsoft Azure також підтримує будь-який інструмент, мову чи фреймворк: Node.js, Java, .NET тощо.

4.2.3 Google Cloud Platform

Google Cloud Platform (GCP), набір хмарних ресурсів та сервісів які були розроблені Google. Платформа надає різноманітні послуги, такі як хмарні обчислення, зберігання даних, налаштування віртуальних мереж, робота з великими даними та багато іншого.

Google Cloud Platform відомий як один із провідних постачальників хмарних послуг у сфері IT. Розробники програмного забезпечення та користувачі з невеликими технічними знаннями можуть легко отримати доступ до послуг і функцій. Google займає провідне місце серед усіх провайдерів хмарних послуг, пропонуючи високомасштабовану та найнадійнішу платформу для створення, тестування та розгортання додатків у середовищі реального часу.

4.3 Порівняльна характеристика

4.3.1 Доступність в залежності від регіону

Хмарна інфраструктура AWS побудована навколо регіонів і зон доступності AWS. AWS Cloud працює у понад 77 зонах доступності в більш ніж 24 географічних регіонах по всьому світу, а також оголошені плани щодо додаткових зон та регіонів доступності.

Глобальна інфраструктура Azure складається з двох ключових компонентів — фізичної інфраструктури та компонентів з'єднувальної мережі. Фізичний компонент складається з 160+ фізичних центрів обробки даних, розбитих на регіони та пов'язаних однією з найбільших взаємопов'язаних мереж на планеті.

Служби Google Cloud доступні в країнах Північної Америки, Південної Америки, Європи, Азії та Австралії. Ці локації поділяються на регіони та зони.

Детальне порівняння доступності відрбражено у таблиці нижче:

	AWS	Azure	GCP
Регіони	25	60	24
Зони доступності	80	Щонайменше по 3 на регіон	73
Країни	245	200	200

4.3.2 Порівняння сервісів

Завдяки тому що AWS з'явилась на 5 років раніше, обчислювальні послуги AWS, безумовно, є найбільш розвиненими та функціонально багатими.

AWS пропонує близько 200+ сервісів, тоді як Azure пропонує до 100+ сервісів. Google Cloud, з іншого боку, наздоганяє Azure і AWS, які пропонують

близько 60+ сервісів. Сьогодні спостерігається висока конкуренція між трьома постачальниками хмарних послуг. Відповідно до останніх тенденцій та запитів клієнтів усі три постачальники почали пропонувати різноманітні сервіси та послуги, які мають задовольнити вимоги користувача. Але все ж таки, наведені платформи мають свої ключові сервіси, які розвинуті у них найкраще[13].

Ключові сервіси AWS:

1. *Штучний інтелект і машинне навчання.* Зі свого списку різноманітних послуг, орієнтованих на штучний інтелект, AWS пропонує DeepLens, камеру на основі штучного інтелекту для розробки та розгортання алгоритмів машинного навчання, щоб використовувати їх для оптичного розпізнавання символів та зображень або об’єктів. AWS також анонсувала бібліотеку глибокого навчання з відкритим кодом під назвою Gluon, яку можуть використовувати як розробники, так і нерозробники для швидкого створення нейронних мереж без будь-яких знань про ШІ.
2. *SageMaker до Severless.* AWS має довгий список послуг у сферах машинного навчання та ШІ. Список послуг AWS також включає SageMaker, який використовується для навчання та розгортання моделей машинного навчання. Він також має розмовний інтерфейс Lex, який підтримує служби Alexa, службу обміну повідомленнями Greengrass IoT і безсерверну службу Lambda.

Ключові сервіси Azure:

1. *Cognitive Services.* Корпорація Майкрософт, вкладаючи значні кошти в сферу машинного навчання та штучного інтелекту, пропонує машинне навчання та службу ботів на Azure. Крім цього,

вона також надає Cognitive Services, які включають Bing Web Search API, Text Analytics API, Face API, Computer Vision API та Custom Vision Service. Крім того, для IoT Microsoft має кілька служб управління та аналітики.

2. *Підтримка програмного забезпечення MSFT.* Azure має кілька інструментів, які допомагають налаштувати інтеграцію з програмним забезпеченням, яке входить до пакету MSFT. Резервне копіювання Azure — це служба, яка пов'язує резервне копіювання Windows Server у Windows Server 2012 R2 і Windows Server 2016. Служби команд Visual Studio розміщують проекти Visual Studio в Azure і т.д.

Ключові сервіси GCP:

1. *IoT to Serverless.* Серед усіх передових технологій, Google Cloud пропонує API для природної мови, мовлення, перекладу тощо. На додаток до цього він пропонує IoT та безсерверні послуги, які наразі знаходиться у бета-тестуванні.
2. *Штучний інтелект.* Наразі Google Cloud є лідером у розробці AI. Заслугою цього є TensorFlow, бібліотека програмного забезпечення з відкритим вихідним кодом для створення програм машинного навчання.

4.3.3 Порівняння можливостей неперервного розгортання

Azure. Azure надає узгоджений набір інструментів, які виконують конкретні завдання:

- Azure Repositories для зберігання та перегляду коду;
- Azure Pipelines для імплементації CI/CD;
- Azure Boards для управління та планування завдань;
- Azure Artifact, для того щоб зберігати результати виконання конвеєра;

Azure Pipelines – сервіс який надає функціонал для реалізації неперервного неперервного розгортання. Будь який сценарій розгортання складається з етапів, які складаються із завдань, які в свою чергу складаються з кроків. Як і для всіх інших хмарних рішеннях, користувач керуєте деталями у файлі YAML. Існує також великий список вбудованих завдань, які надає Azure. За допомогою сценаріїв можна розгортати різноманітні ресурси, такі як: віртуальні машини, реєстри контейнерів, сегменти зберігання тощо [14].

Google Cloud. Cloud Build — це безсерверна платформа CI/CD від Google, у центрі якої, як зазвичай, є декларативний опис кроків, які ви хочете виконати на цій платформі. Служба постачається з обмеженим набором приблизно з 20 офіційних образів докерів для створення хмар. Ці офіційні образи підтримують більшість поширених ланцюжків інструментів і включають bazel, docker, dotnet, go і npm, серед інших.

AWS. Хоча етапи збірки та розгортання — це два кроки в одному конвеєрі для Azure і GCP, AWS використовує інший підхід і представляє два різних рішення: CodeBuild і CodeDeploy.

CodeBuild — це покрокова повністю керована служба збірки. Вона потребує лише дві речі: сховище вихідного коду (може бути CodeCommit або репозиторій GitHub) і деякі інструкції щодо того, як і що створити. Ці інструкції надходять у вигляді файлу buildspec.yaml у сховищі. Отриманий артефакт може

бути збережений у відро S3 або переданий до реєстру контейнерів. Приємним доповненням є локальна опція збірки, яка дозволяє протестувати конвеєр локально перед його фіксацією.

Після етапу збірки AWS CodeDeploy дозволяє розгорнути додаток, пропонуючи список обчислювальних платформ, на яких можна розгорнути, наприклад EC2, Lambda або ECS. Він надає різні стратегії розгортання з коробки, напр. g. canaray, лінійне або синьо-зелене розгортання.

4.4 Висновок

У рамках дипломної для реалізації поставленних завдань найкраще підходить використання платформи Azure. Дане рішення обгрунтовано наявністю великої кількості вбудованих завдань для імплементації неперервного розгортання та збірки проекту. А також наявністю простої інтеграції з сервісами контролю версій.

РОЗДІЛ 5. СТРУКТУРА ТА РОЗРОБКА СПЕЦІАЛІЗОВАНОГО ВЕБ-ДОДАТКУ

5.1. Структура веб-додатку конструктора

На рисунку 5.1 зображена архітектура розробленого веб-додатку, де за допомогою стрілок показані шляхи циркуляції даних.

Переважна більшість веб-додатків складаються з користувацького інтерфейсу (UI) та серверної частини (back-end), що представляє собой прикладний програмний інтерфейс, з яким клієнт (користувацький інтерфейс) взаємодіє.

Представлена архітектура є трирівнева. Трирівнева архітектура поділяє додатки на три логічні рівні: рівень презентації або інтерфейс користувача; рівень програми, на якому обробляються дані; і рівень даних, де зберігаються й керуються дані, пов’язані з програмою [15].

Головна перевага трирівневої архітектури полягає в тому, що, будь-який рівень працює на власній інфраструктурі, кожен рівень може одночасно розроблятися окремою командою розробників і за потреби оновлюватися або масштабуватися, не впливаючи на інші рівні.

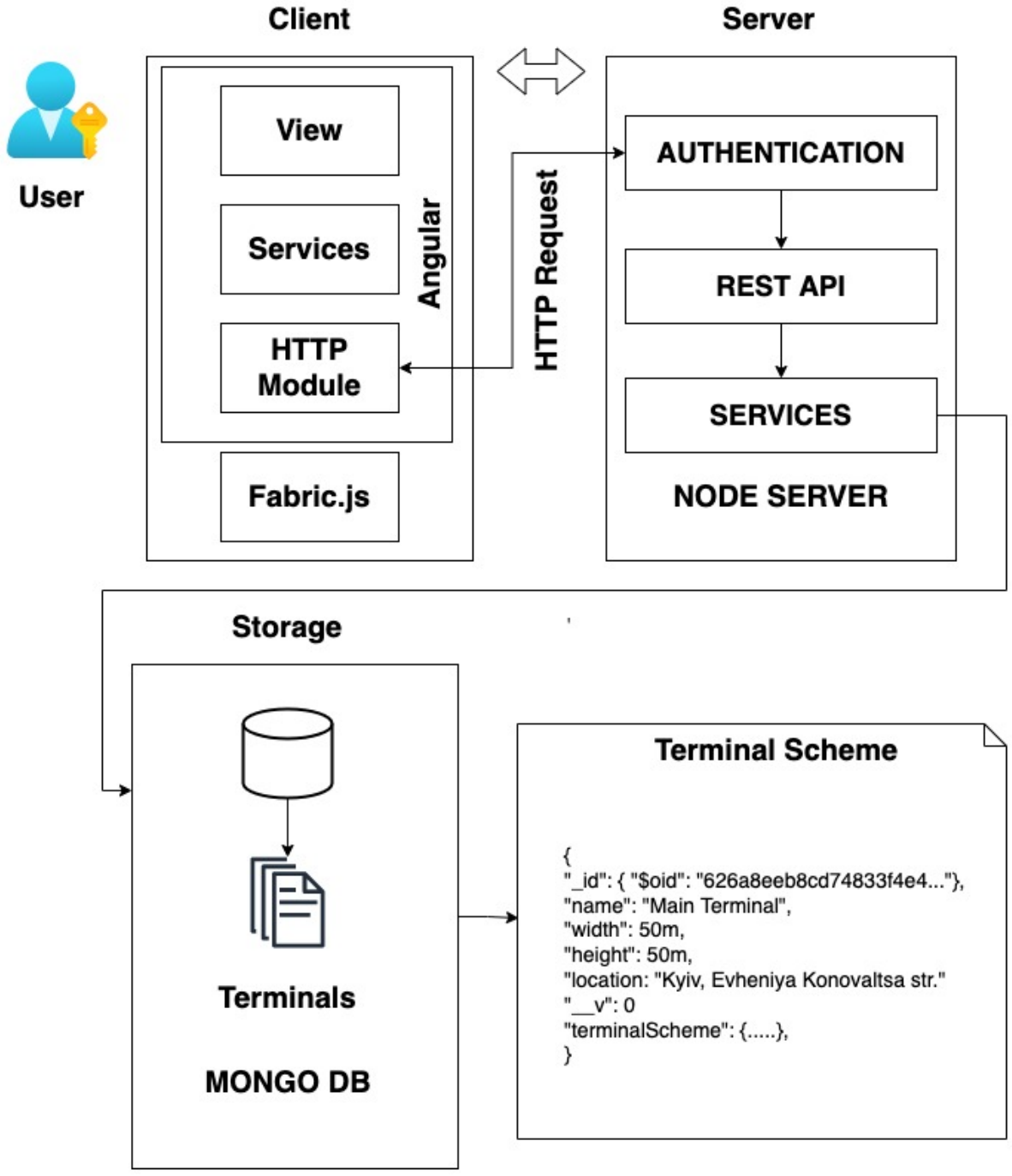

Рисунок 5.1 – Загальна архітектура веб-додатку parking constructor

Рівні архітектури додатку:

- База даних, рівень на якому фізично зберігаються дані у визначеному форматі.
- Сервер, де зосереджена бізнес-логіка, логіка доступу до бази даних та безпека даних в загальному.
- Клієнтський додаток, за допомогою якого користувач взаємодіє зі всією системою.

5.2 Реалізація серверної частини

5.2.1 База даних

Для зберігання, швидкого завантаження та вивантаження схеми терміналу, нам потрібна легка та ефективна база даних. Також бажано було обрати СУБД яка не потребує багато додаткових налаштувань та постійної підтримки програмного коду при зміні моделі об’єктів. Для цих задач ідеально підходять не реляційні СУБД. Саме тому було вирішено обрати MongoDB.

MongoDB — це документно-орієнтована база даних NoSQL, яка використовується для зберігання даних великого обсягу. Замість використання таблиць і рядків, як у традиційних реляційних базах даних, MongoDB використовує колекції та документи. Документи складаються з пар ключ-значення, які є основною одиницею даних у MongoDB. Колекції містять набори документів і функцій, що є еквівалентом таблиць реляційної бази даних.

Нереляційна база даних, або база даних NoSQL, зберігає дані, на відміну від реляційної бази даних, не використовуючи таблиці, рядки, первинні або зовнішні ключів. Замість цього нереляційна база даних використовує модель зберігання, оптимізовану для конкретних вимог типу даних, що зберігаються.

У “Додатку А” наведений приклад зберігання даних схеми терміналу у веб-застосунку parking-system.

5.2.2 Веб-сервіс

Серверний додаток має добре працювати з великим об‘ємом даних та мати можливість працювати з сторонніми бібліотеками для обробки. Ідеальний кандидат це Node.js. Можливість запускати Javascript у віртуальній машині, значно економить системні ресурси у порівняння з додатками написані на платформі .Net або Java.

Node.js інтерпретує код JavaScript за допомогою движку JavaScript V8 від Google. Цей механізм компілює код JavaScript безпосередньо у машинний код. Це полегшує та пришвидшує ефективне впровадження коду.

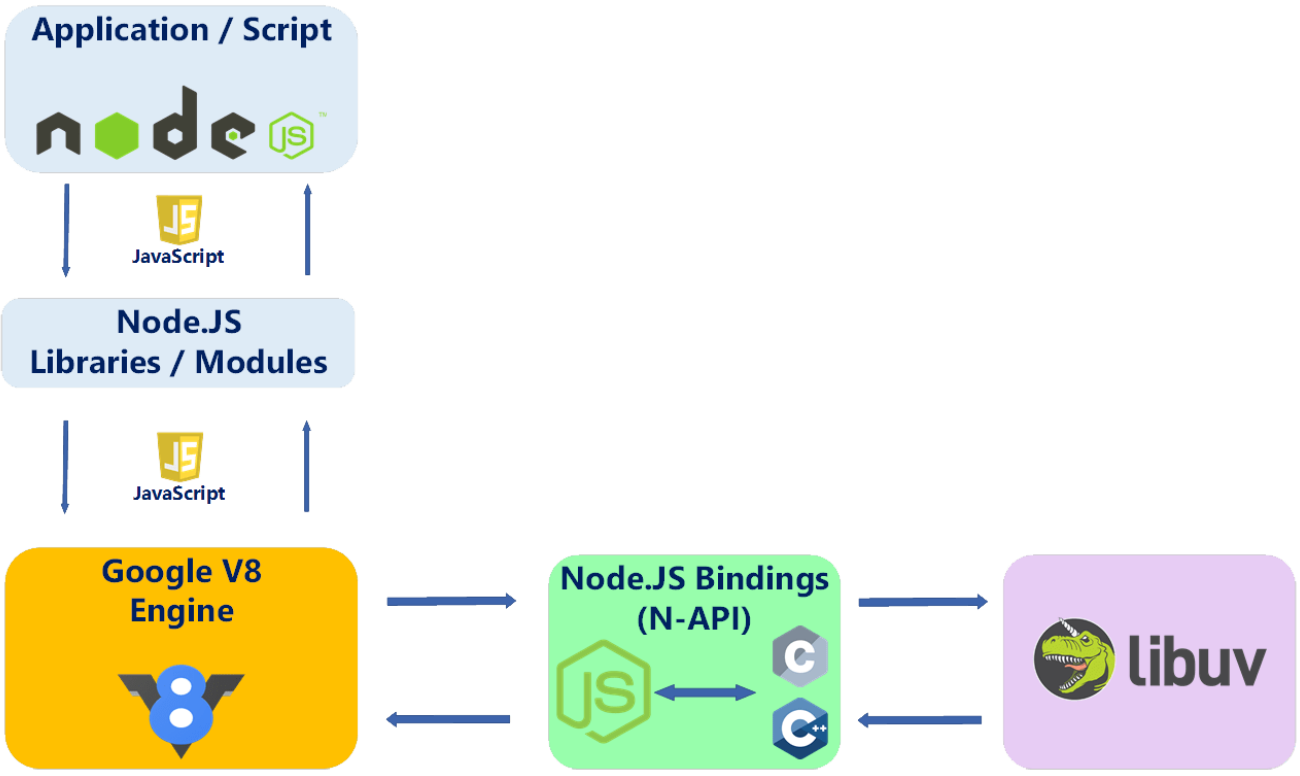

Рисунок 5.2 – Загальна архітектура Node.js

Node.js використовує однопотокову модель із механізмом Event Loop, який відповідає за виконання коду, збір і обробку подій, а також виконання підзадач у черзі. Ця модель значно відрізняється від моделей на інших мовах, таких як C і Java.

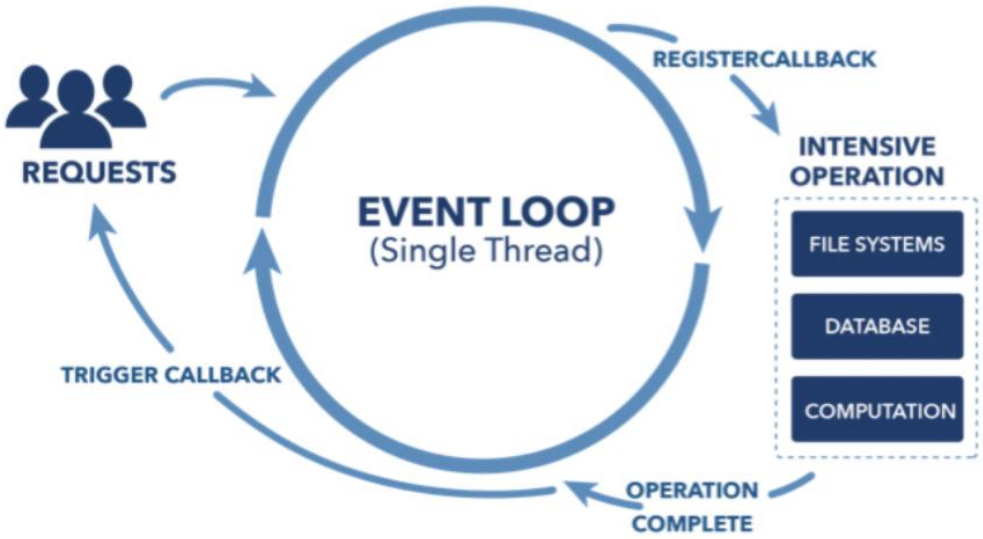

Рисунок 5.3 – Архітектура обробки подій всередині V8

Ще однією перевагою Node.js є вбудований пакетний менеджер – NPM. Node Package Manager (NPM) – менеджер пакетів JavaScript який дає винятковий контроль над залежностями проекту.

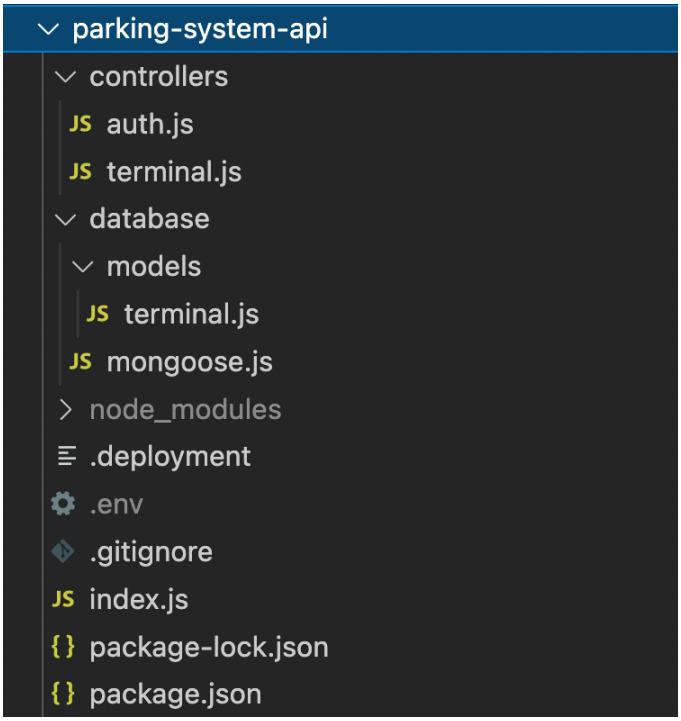

Рисунок 5.4 – Файлова структура серверного додатку

У папці controllers описуються функції, які відокремлюють код для маршрутизації запитів від коду, який фактично обробляє запити.

Приклад функції контролеру:

```
router.patch('/terminals/:id', (req, res) => {
    Terminal.findOneAndUpdate({ _id: req.params.id }, { $set: req.body })
            .then(() => {
                res.status(204).send();
            })
            .catch(error => console.log(error));
});
```

У папці database описується конфігурація роботи з базою даних та наявні моделі.

Опис конфігурації бази даних:

```
mongoose.connect(process.env.DB_CONNECTION_STRING)
        .then(() => console.log('Database connected!'))
        .catch((error) => console.log(error));
module.exports = mongoose;
```

Файл index.js – точка входу для додатку, де описується конвеєр обробки HTTP запиту:

```
/**
 *   Variables Configuration
 */
const express = require('express');
const app = express();
const cors = require("cors");
require('./database/mongoose');
const jwtCheck = jwt.expressjwt({
    secret: jwks.expressJwtSecret({
        cache: false,
        rateLimit: false,
        jwksRequestsPerMinute: 5,
        jwksUri: '…'
  }),
  audience: '…',
  issuer: '…',
  algorithms: ['RS256']
});
const authRouter = require("./controllers/auth");
const terminalRouter = require("./controllers/terminal");
```

```
/**
 *   App Configuration
 */

app.use(cors());
app.use(express.json({limit: '50mb'}));
app.use("/", authRouter);
app.use("/", jwtCheck, terminalRouter);
app.listen(3000, () => console.log('App is listening on port 3000'))
```

5.3 Впровадження механізму автентифікації

Впровадження автентифікації на основі фреймворку OAuth2 буде реалізовано за домогою сервісу auth0.

Auth0 — це гнучке рішення для додавання служб автентифікації та авторизації до додатків. Воно дозволяє уникнути витрат, часу та ризиків, пов’язаних із створенням власного рішення для автентифікації та авторизації користувачів.

Діаграма роботи механізму автентифікації на основі фреймворку OAuth2 була розглянута у підрозділі 1.2.4.

Нижче наведені кроки для впровадження механізму використовуючи сервіс auth0:

1. Підключення модулю AuthModule з конфігурацією серверу авторизації та налаштуваннями для інтерсептору:

```
AuthModule.forRoot({
      ...environment.auth,
```

```
      httpInterceptor: {
        allowedList: [`${environment.api.parkingSystem}/*`],
      }
    })
```

2. Додаваня інтерсептору до провайдерів який буде додавати токен авторизації до будь-якого запиту до серверу:

```
 {
      provide: HTTP_INTERCEPTORS,
      useClass: AuthHttpInterceptor,
      multi: true
 }
```

3. Додавання AuthGuard до конфігурації маршрутів для контролю доступу до відповідних компонентів:

```
{
    path: 'terminals',
    component: TerminalListComponent,
    canActivate: [AuthGuard]
}
```

4. Реалізація компоненту AuthButton для авторизації та закінчення сессії.
5. Встановлення механізму валідації токену в конвеєр обробки запиту у серверній частині:

```
const jwtCheck = jwt.expressjwt({
    secret: jwks.expressJwtSecret({
        cache: false,
        rateLimit: false,
        jwksRequestsPerMinute: 5,
        jwksUri: '…'
  }),
  audience: '…',
  issuer: '…',
  algorithms: ['RS256']
});
app.use("/", jwtCheck, terminalRouter);
```

5.4 Розробка клієнтського додатку

Для реалізації відображення та редагування 2-d моделей терміналу, а саме односторінкового застосунку було обрано фреймворк Angular.

Якщо створювати веб-додаток лише з використанням класичного Javascript, він вийде досить громіздким та зі складною архітектурою. Протягом останніх років було випущено декілька бібліотек і фреймворків які значно спрощують написання та масштабування сучасних веб-додатків. Найбільш сучасні, були розглянуті у 2 розділі цієї дипломної роботи.

Angular - фреймворк, головне призначення якого - створення односторінкових додатків. Завдяки патерну MVC (Model-view-controller) додатки на Angular є більш структурованими, що полегшує процеси тестування та розробки. Фреймворк імперативно декларує архітектуру побудови веб додатку та його розширення, що дозволяє розробнику зосередитися на більш прикладних завданнях, таких як побудова користувацького інтерфейсу, списків форм та ін.

Ключ до застосування шаблону MVC полягає у реалізації ключової передумови поділу проблем, у якому модель даних у додатку відокремлена від бізнес-логіки та логіки презентації. У веб-розробці клієнта це означає розділення даних, логіки, яка оперує цими даними, і елементів HTML, які використовуються для відображення даних. Результатом є програма на стороні клієнта, яку легше розробляти, підтримувати та тестувати.

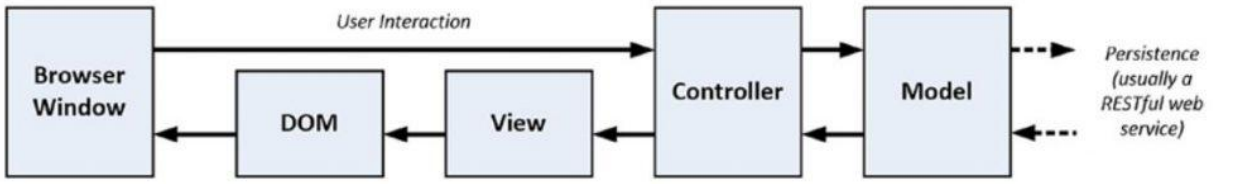

Рисунок 5.4 – MVC архітектура Angular додатку

5.4.1 Розробка конструктора для відображення та редагування 2-d моделі терміналу.

У 3 розділі було розглянуто наявні технології для 2-d візуалізації використовуючи HTML елемент Canvas. Для поставленої задачі, було вирішено використовувати бібліотеку Fabric.js. Завдяки об'єктно-орієнтованому підходу для створення та маніпулюванням елементами Canvas бібліотеки Fabric.js, можна побудувати зручну систему класів та описати поведінку усіх складових автомобільного терміналу.

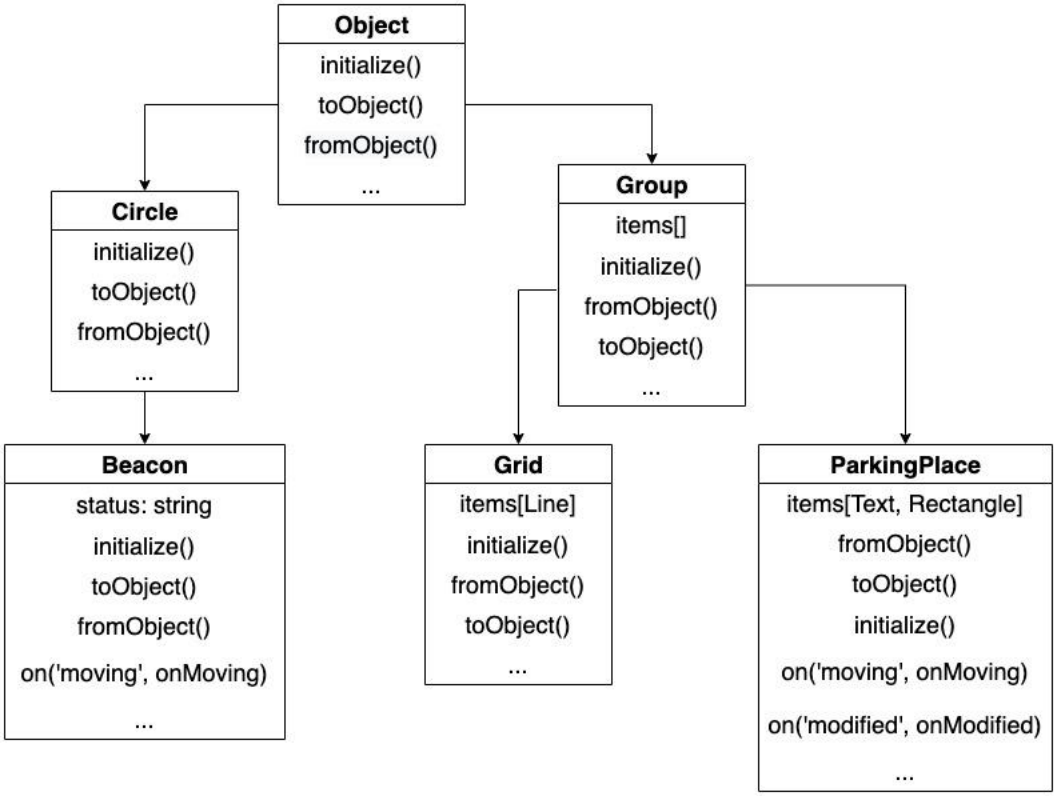

Рисунок 5.5 – Діаграма класів для реалізації 2-d конструктора терміналу

Object – базовий клас, від якого наслідуються усі інші класи. Він містить у собі значну кількість функцій, які можна розширювати в залежності від потреб при наслідуванні. Так, на діаграмі виділено 3 основні функції які будуть розширені у нащадках для реалізації необхідного функціоналу.

- initialize – ініціалізує об’єкт перед рендерингом;
- fromObejct – логіка сериалізації;
- toObject – логіка десериалізації.

Circle, це також клас вбудований у бібліотеку fabric.js, який є нащадком Object та дозволяє створювати коло на Canvas.

Group – вбудований клас, який дозволяє групувати об'єкти та маніпулювати ними як одним цілим.

Beacon – клас для відображення блютуз маячку. Він містить поле status, в залежності від якого коло змінює свій колір (червоний, якщо статус DISABLED та зелений, якщо ENABLED), а також розширює функції initialize, fromObject та toObject.

Розширення функції initialize:

```
initialize: function(options) {
    const fill = BeaconStatus[options.status] == BeaconStatus.DISABLED.toString() ? "red" : "green";
    const overriddenOptions = {
      radius: 10,
      fill: fill,
      hasControls: false,
      hasBorders: false,
      top: options.top ?? 0,
```

```
        left: options.left ?? 0,
        rfid: options.rfid,
        status: options.status
      };
      this.callSuper('initialize', overriddenOptions);
      this.on('moving', onMoving);
    }
```

Grid – клас для відображення сітки. Він наслідується від класу Group та містить у собі масив ліній, які формують сітку конструктору, а також розширює функції initialize, fromObject та toObject.

ParkingPlace – клас для відображення сітки. Він наслідується від класу Group та містить текстовий елемент класу Text, який призначений для відображення ідентифікатору паркомісця та Rectangle для відображення самого паркомісця, а також розширює функції initialize, fromObject, toObject. Крім цього перевизначає обробники подій on(‘moving’, onMoving) та on(‘modified’, onModified).

Перевизначення обробника подій on(‘modified’, onModified):

```
function onModified(options) {
  const group = options.transform.target;
  const parkingPlace = group._objects[0];
  const minWidth = 50;
  const minHeigth = 50;
  const calculatedWidth = (Math.round(group.getScaledWidth() / 10)) * 10;
  const calculatedHeight = (Math.round(group.getScaledHeight() / 10)) * 10;
```

```
        const newWidth = calculatedWidth < minWidth ? minWidth : calculatedWidth;
        const newHeight = calculatedHeight < minHeigth ? minHeigth : calculatedHeight;
        let newTop = -newHeight * 0.5;
        let newLeft = -newWidth * 0.5;
        group.set({
          width: newWidth,
          height: newHeight,
          scaleX: 1,
          scaleY: 1
        });

        parkingPlace.set({
          width: newWidth,
          height: newHeight,
          scaleX: 1,
          scaleY: 1,
          top: newTop,
          left: newLeft
        });
      }
```

Повний опис класу ParkingPlace наведений у Додатку «Б».

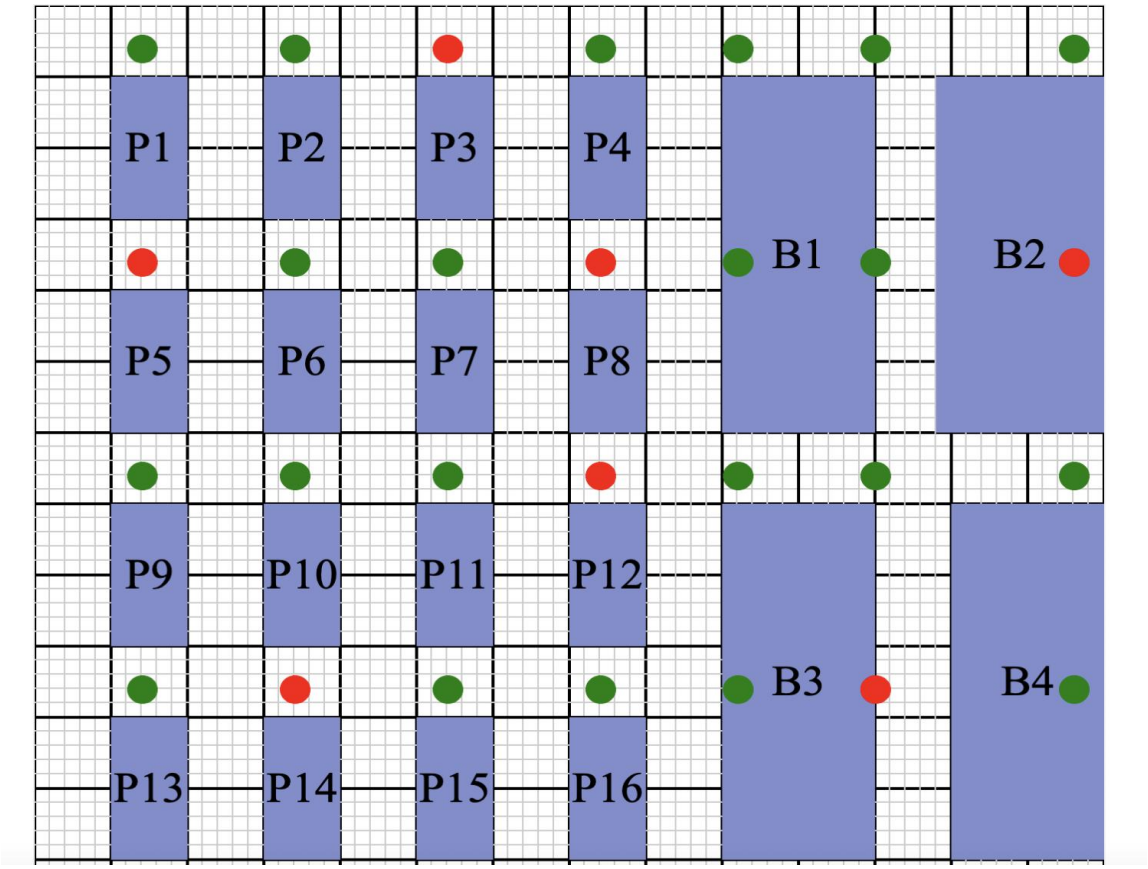

Рисунок 5.6 – Приклад створеної моделі терміналу

5.4.2 Розробка сторінок веб-додатку.

Після постановки задачі було створено прототипи вигляду веб-застосунку за допомогою програмного застосунку Figma та пізніше розроблено їх у додатку.

Так, при першому вході в додаток відображається стартова сторінка, на якій відображаються усі наявні термінали.

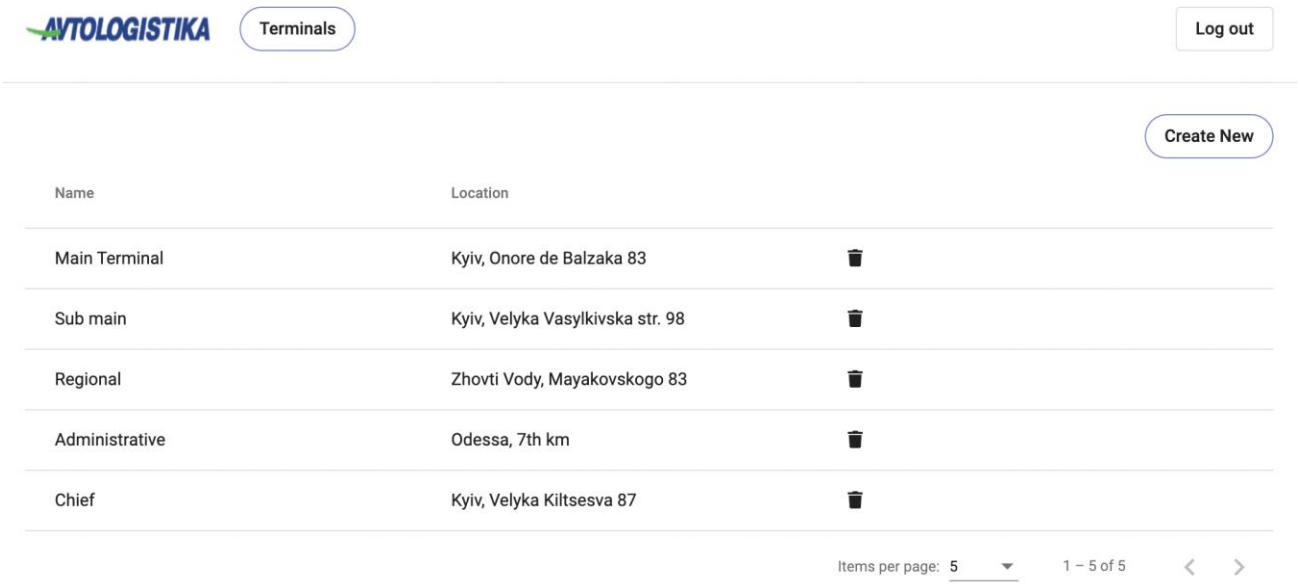

Рисунок 5.7 – Стартова сторінка, на якій відображаються усі наявні термінали.

При автентифікації, впровадження якої було розглянуто (див. підрозділ 4.2) сторінка виглядає, як показано на рисунку нижче.

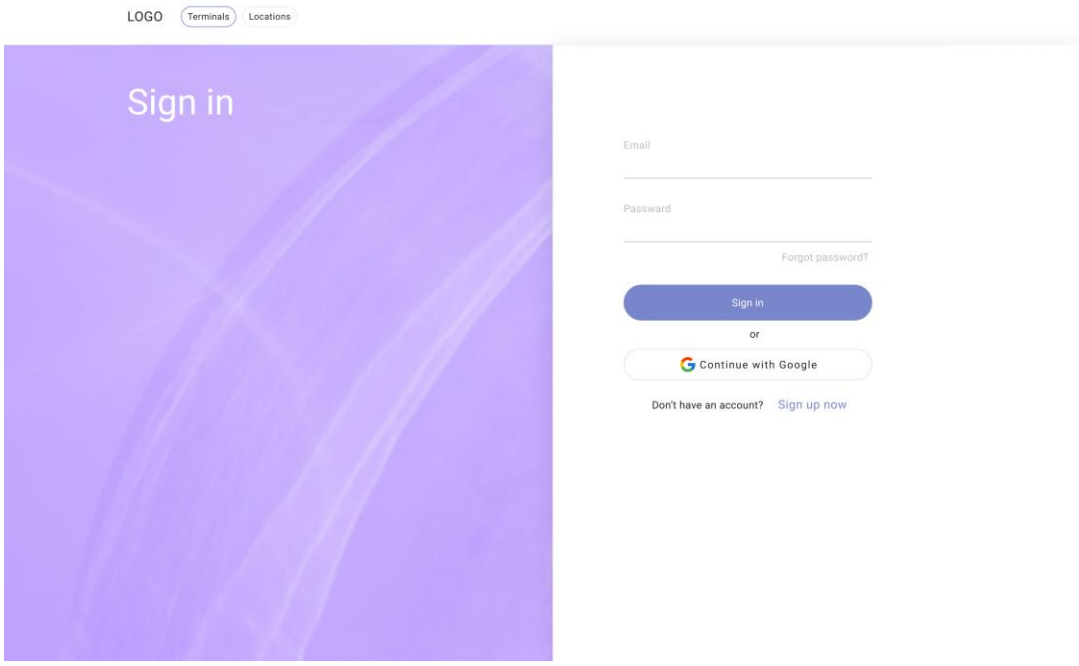

Рисунок 5.8 – Вигляд сторінки автентифікації

Нижче наведені сторінки створення нової моделі терміналу.

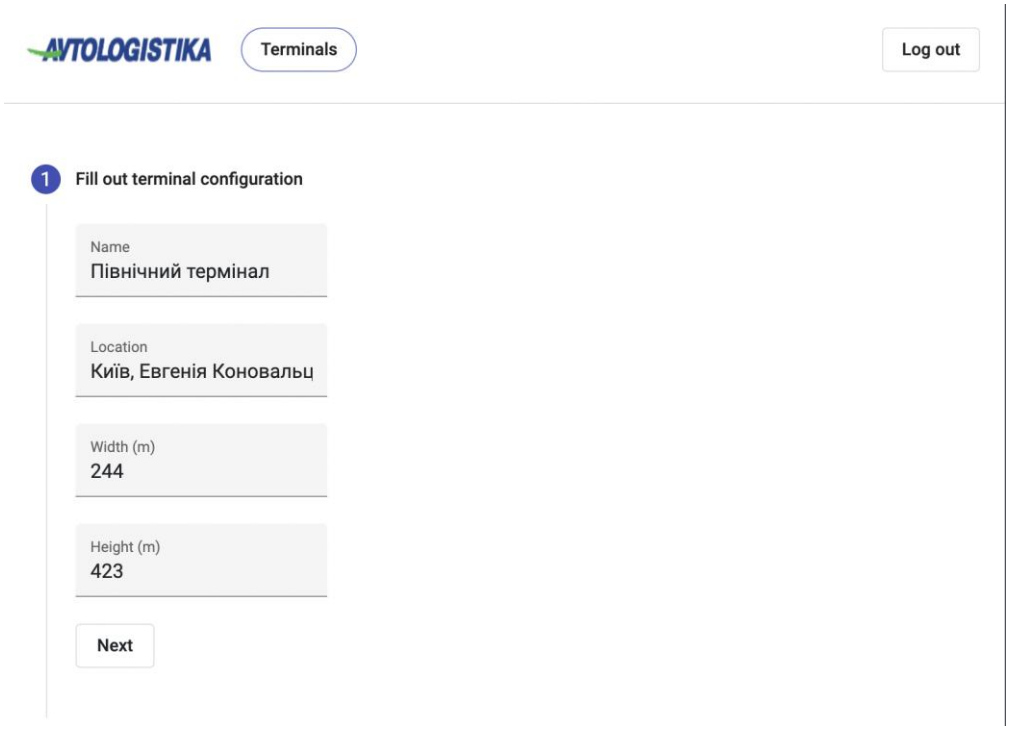

Рисунок 5.9 – Перший крок створення моделі терміналу

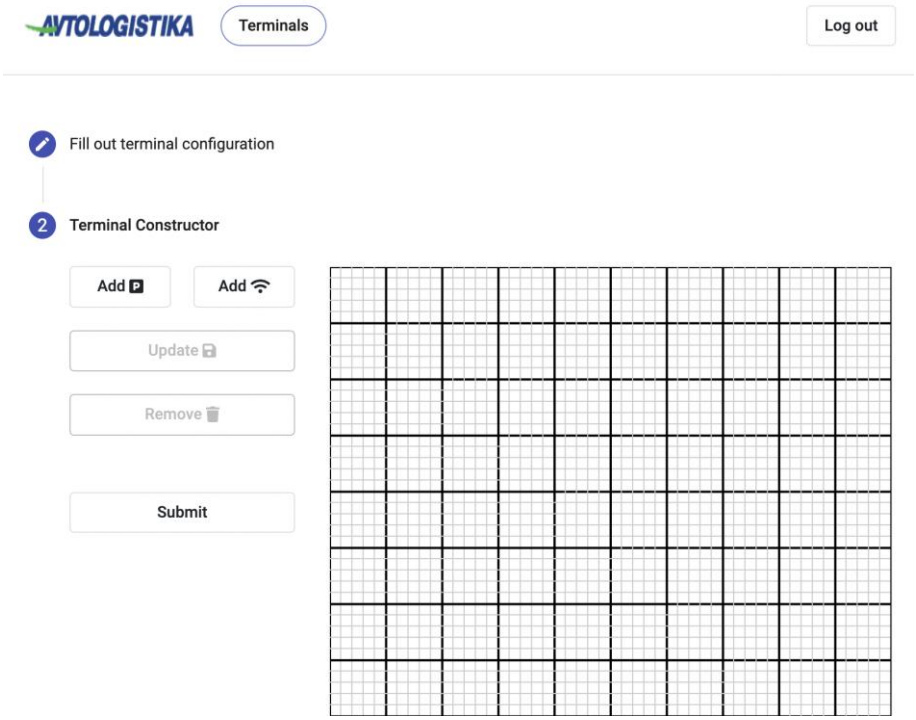

Рисунок 5.10 – Другий крок створення моделі терміналу

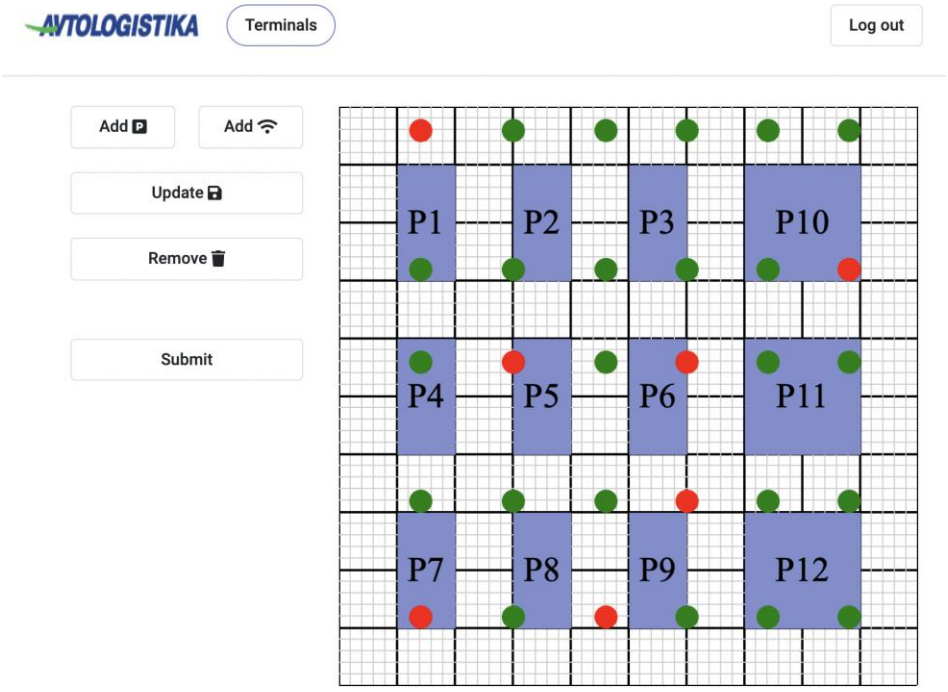

Рисунок 5.11 – Створена модель терміналу

Частина програмного коду, за допомогою якого було реалізовано компонент конструктора терміналу можна переглянути у “Додатку В”.

5.5 Інфраструктура неперервного розгортання

У 2 розділі було розглянуто доступні провайдери хмарних послуг для реалізації неперервного розгортання. Для поставленої задачі, було вирішено використовувати бібліотеку Azure, а саме сервіс Azure Pipelines.

Azure Pipelines поєднує в собі безперервну інтеграцію (CI) і безперервну доставку (CD) для тестування, збірки та розгортання проекту на будь-які хмарні ресурси [16].

Безперервна інтеграція (CI) — це практика розробки, коли розробники часто інтегрують код у спільне сховище, бажано кілька разів на день. Кожну інтеграцію потім можна перевірити за допомогою автоматизованої збірки та автоматизованих тестів. Хоча автоматичне тестування не є суворо частиною CI, воно зазвичай мається на увазі.

Безперервна доставка (CD) — це практика постійного випуску вашого програмного забезпечення за допомогою автоматизації процесів якості, безпеки та розгортання протягом усього життєвого циклу доставки програмного забезпечення. Постійно проводячи тести безпеки, нефункціональні та функціональні тести в продуктах або виробничих середовищах, програмне забезпечення поступово перевіряється та стає придатним до випуску.

5.5.1 Огляд схеми пайплайну хмарної платформи Azure

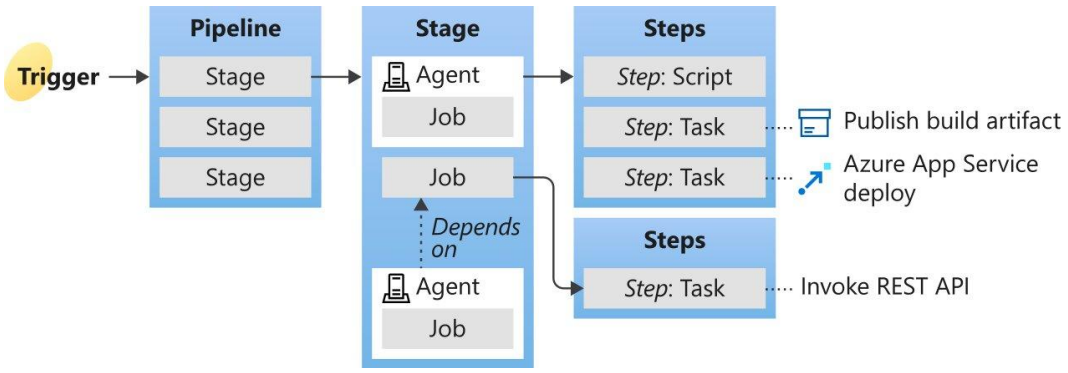

Рисунок 5.12 – Схеми пайплайну хмарної платформи Azure

- Тригер повідомляє конвеєр про необхідність запуску;
- Конвеєр складається з одного або кількох етапів;
- Етап групує завдання;
- Завдання виконуються агентами;
- Кожне завдання містить щонайменше один крок;
- Крок є найменшим будівельним блоком конвеєра та безпосередньо виконує дію, наприклад, викликає REST API або публікує артефакт збірки.

Для реалізації неперервного розгортання було побудовано 2 конвеєри, для клієнтської та серверної частин.

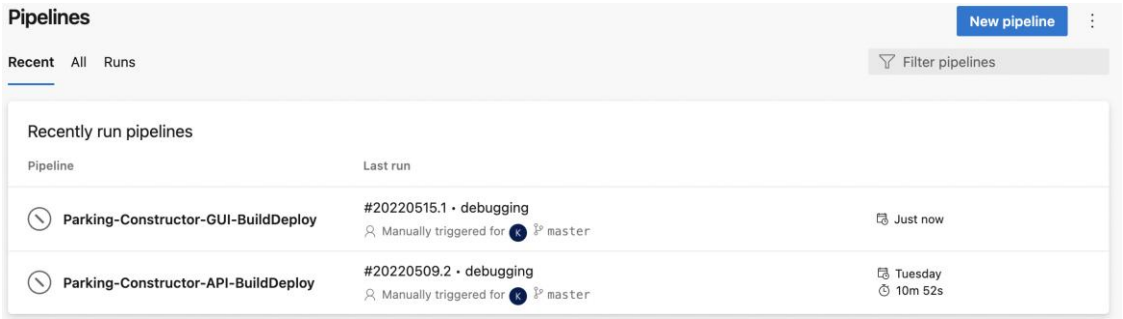

Рисунок 5.13 – Ковеєри для серверної та клієнтської частин

5.5.2 Огляд конвеєру для клієнтської частини

Стадія збірки

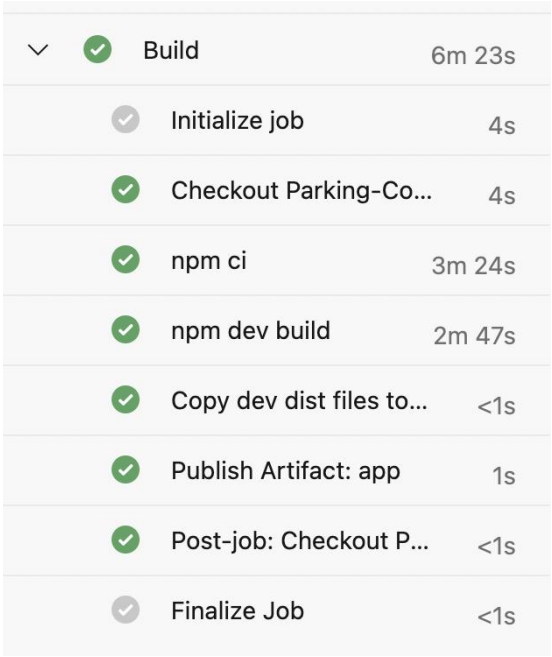

Рисунок 5.14 – Кроки стадії збірки

Кроки стадії збірки:

1. Завантаження останньої версії коду.
2. Встановлення усіх необхідних сторонніх бібліотек.
3. Збірка Angular додатку для середовища dev.
4. Копіювання збірки до папки з артефактами.
5. Публікація артефакту.

Стадія розгортання

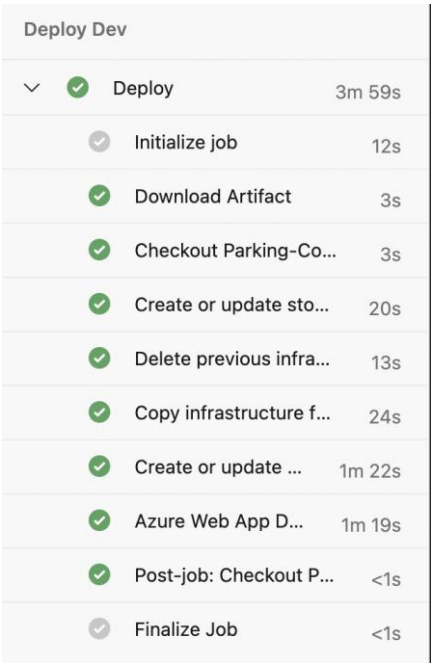

Рисунок 5.15 – Кроки стадії розгортання

1. Завантаження артефакту збірки.
2. Завантаження останньої версії коду.
3. Створення або оновлення ресурсу сховища.
4. Видалення минулої інфраструктури розгортання зі сховища.
5. Копіювання нової інфраструктури розгортання в сховище.
6. Створення ресурсів веб-додатку за допомогою ARM шаблонів.
7. Розгортання додатку в створених на минулому кроці ресурсах.

Слід зазначити що усі ресурси створюються з використанням ARM шаблонів. ARM розшифровується як Azure Resource Manager, що є службою розгортання та керування Microsoft для Azure [16]. У “Додатку Д” наведений приклад ARM шаблону для сховища.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було розроблено інтегровану систему, до якої входить веб-сервер для зберігання та обробки даних та веб-інтерфейс для візуалізації та маніпуляції 2-d моделями автомобільних терміналів.

Проведено аналіз можливостей 2-d візуалізації у сучасних веб-додатках, опановано методику та етапи створення односторінкового веб-додатку використовуючи фреймворк Angular. Побудовано систему класів на основі бібліотеки fabric.js для вирішення задачі побудови 2-d моделі терміналу

Також було реалізовано аутентифікацію для забезпечення надійності даних і розроблені конвеєри для неперервного розгортання клієнтського та серверного додатків, що дозволяє автоматично створювати нові середовища з усіма необхідними хмарними ресурсами та доставляти код на вже існуючі.

За результатами кваліфікаційної роботи було проаналізовано технології створення сучасних веб-додатків, розглянуто різні бібліотеки та фреймворки для реалізації веб-застосування, на основі чого було створено веб-додатк, що дає змогу користувачу зручним чином вказати розташування усіх паркомісць та наявних BLE маяків.

Даний веб-додаток може бути використаний користувачами у галузі автомобільного лізингу для впровадження системи навігації всередині автомобільного терміналу, що дає змогу автоматизувати наявні бізнес процеси та зменшити вплив людського фактору.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Мобільні інформацій технології навігації користувача в приміщеннях, [Електронний ресурс] – Режим доступу до ресурсу : https://science.lpnu.ua/sites/default/files/journal-paper/2018/jun/12974/13119-137.pdf
2. Система авторизації мікросервісів на основі KeyCloak для захисту середовищ хмарних обчислень, [Електронний ресурс] – Режим доступу до ресурсу : https://ela.kpi.ua/bitstream/123456789/27207/1/Pryzhkov_magistr.pdf
3. The Auth Schemes of REST, [Електронний ресурс] – Режим доступу до ресурсу : https://datatracker.ietf.org/doc/html/rfc7519.
4. What is OAuth 2.0? [Електронний ресурс] – Режим доступу до ресурсу : https://auth0.com/intro-to-iam/what-is-oauth-2/
5. Angular vs React vs Vue.js: Which is the Best Choice for 2022?, [Електронний ресурс] – Режим доступу до ресурсу : https://javascript.plainenglish.io/angular-vs-react-vs-vue-js-which-is-the-best-choice-for-2022-5ef83f2257ab
6. Angular vs Vue.js: що обрати? [Електронний ресурс] – Режим доступу до ресурсу : https://dou.ua/lenta/columns/angular-vs-vuejs/
7. Angular vs. React vs. Vue.js – Choosing a JavaScript Framework for Your Project? [Електронний ресурс] – Режим доступу до ресурсу : https://relevant.software/blog/angular-vs-react-vs-vue-js-choosing-a-javascript-framework-for-your-project/
8. Basic usage of canvas [Електронний ресурс] – Режим доступу до ресурсу : https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Basic_usage

9. D3.js - Introduction [Електронний ресурс] – Режим доступу до ресурсу : https://www.tutorialspoint.com/d3js/d3js_introduction.htm
10. EaselJS Module [Електронний ресурс] – Режим доступу до ресурсу : https://createjs.com/docs/easeljs/modules/EaselJS.html
11. PixiJS features [Електронний ресурс] – Режим доступу до ресурсу : https://pixijs.com/
12. Introduction to Fabric.js. Part 1. [Електронний ресурс] – Режим доступу до ресурсу : http://fabricjs.com/fabric-intro-part-1
13. AWS vs Azure vs Google Cloud: Choosing the Right Cloud Platform [Електронний ресурс] – Режим доступу до ресурсу : https://intellipaat.com/blog/aws-vs-azure-vs-google-cloud/
14. Azure DevOps vs AWS DevOps vs GCP DevOps? [Електронний ресурс] – Режим доступу до ресурсу : https://devcontentops.io/post/2021/05/azure-devops-vs-aws-devops-vs-gcp-devops
15. Three-Tier Architecture [Електронний ресурс] – Режим доступу до ресурсу : https://www.ibm.com/cloud/learn/three-tier-architecture
16. What are ARM templates? [Електронний ресурс] – Режим доступу до ресурсу : https://docs.microsoft.com/en-us/azure/azure-resource-manager/templates/overview

ДОДАТКИ

Додаток А

Приклад зберігання даних схеми терміналу


```
{
  "_id": {
    "$oid": "626f108a65d3a652383d381f"
  },
  "name": "Main terminal",
  "location": "Kyiv,",
  "width": 70,
  "height": 70,
  "terminalScheme":
"{{\"version\":\"5.0.0\",\"objects\":[{\"type\":\"Grid\",\"version\":\"5.
0.0\",\"originX\":\"left\",\"originY\":\"top\",\"left\":0,\"top\":0,\"wid
th\":\"70\",\"height\":\"70\",\"fill\":\"rgb(0,0,0)\",\"stroke\":null,\"s
trokeWidth\":0,\"strokeDashArray\":null,\"strokeLineCap\":\"butt\",\"stro
keDashOffset\":0,\"strokeLineJoin\":\"miter\",\"strokeUniform\":false,\"s
trokeMiterLimit\":4,\"scaleX\":1,\"scaleY\":1,\"angle\":0,\"flipX\":false
,\"flipY\":false,\"opacity\":1,\"shadow\":null,\"visible\":true,\"backgro
undColor\":\"\",\"fillRule\":\"nonzero\",\"paintFirst\":\"fill\",\"global
CompositeOperation\":\"source-
over\",\"skewX\":0,\"skewY\":0,\"objects\":[{\"type\":\"line\",\"version\
":\"5.0.0\",\"originX\":\"left\",\"originY\":\"top\",\"left\":-
351,\"top\":-
351,\"width\":0,\"height\":700,\"fill\":\"rgb(0,0,0)\",\"stroke\":\"#000\
",\"strokeWidth\":2,\"strokeDashArray\":null,\"strokeLineCap\":\"butt\",\
"strokeDashOffset\":0,\"strokeLineJoin\":\"miter\",\"strokeUniform\":fals
e,\"strokeMiterLimit\":4,\"scaleX\":1,\"scaleY\":1,\"angle\":0,\"flipX\":
false,\"flipY\":false,\"opacity\":1,\"shadow\":null,\"visible\":true,\"ba
ckgroundColor\":\"\",\"fillRule\":\"nonzero\",\"paintFirst\":\"fill\",\"g
lobalCompositeOperation\":\"source-
over\",\"skewX\":0,\"skewY\":0,\"x1\":0,\"x2\":0,\"y1\":-
350,\"y2\":350},{\"type\":\"line\",\"version\":\"5.0.0\",\"originX\":\"le
ft\",\"originY\":\"top\",\"left\":-351,\"top\":-
351,\"width\":700,\"height\":0,\"fill\":\"rgb(0,0,0)\",\"stroke\":\"#000\
```

```
",\"strokeWidth\":2,\"strokeDashArray\":null,\"strokeLineCap\":\"butt\",\
"strokeDashOffset\":0,\"strokeLineJoin\":\"miter\",\"strokeUniform\":fals
e,\"strokeMiterLimit\":4,\"scaleX\":1,\"scaleY\":1,\"angle\":0,\"flipX\":
false,\"flipY\":false,\"opacity\":1,\"shadow\":null,\"visible\":true,\"ba
ckgroundColor\":\"\",\"fillRule\":\"nonzero\",\"paintFirst\":\"fill\",\"g
lobalCompositeOperation\":\"source-
over\",\"skewX\":0,\"skewY\":0,\"x1\":-
350,\"x2\":350,\"y1\":0,\"y2\":0},{\"type\":\"line\",\"version\":\"5.0.0\
",\"originX\":\"left\",\"originY\":\"top\",\"left\":-341,\"top\":-
351,\"width\":0,\"height\":700,\"fill\":\"rgb(0,0,0)\",\"stroke\":\"#ccc\
",\"strokeWidth\":1,\"strokeDashArray\":null,\"strokeLineCap\":\"butt\",\
"strokeDashOffset\":0,\"strokeLineJoin\":\"miter\",\"strokeUniform\":fals
e,\"strokeMiterLimit\":4,\"scaleX\":1,\"scaleY\":1,\"angle\":0,\"flipX\":
false,\"flipY\":false,\"opacity\":1,\"shadow\":null,\"visible\":true,\"ba
ckgroundColor\":\"\",\"fillRule\":\"nonzero\",\"paintFirst\":\"fill\",\"g
lobalCompositeOperation\":\"source-
over\",\"skewX\":0,\"skewY\":0,\"x1\":0,\"x2\":0,\"y1\":-
350,\"y2\":350},{\"type\":\"line\",\"version\":\"5.0.0\",\"originX\":\"le
ft\",\"originY\":\"top\",\"left\":-351,\"top\":-
341,\"width\":700,\"height\":0,\"fill\":\"rgb(0,0,0)\",\"stroke\":\"#ccc\
",\"strokeWidth\":1,\"strokeDashArray\":null,\"strokeLineCap\":\"butt\",\
"strokeDashOffset\":0,\"strokeLineJoin\":\"miter\",\"strokeUniform\":fals
e,\"strokeMiterLimit\":4,\"scaleX\":1,\"scaleY\":1,\"angle\":0,\"flipX\":
false,\"flipY\":false,\"opacity\":1,\"shadow\":null,\"visible\":true,\"ba
ckgroundColor\":\"\",\"fillRule\":\"nonzero\",\"paintFirst\":\"fill\",\"g
lobalCompositeOperation\":\"source-
over\",\"skewX\":0,\"skewY\":0,\"x1\":-
350,\"x2\":350,\"y1\":0,\"y2\":0},{\"type\":\"line\",\"version\":\"5.0.0\
",\"originX\":\"left\",\"originY\":\"top\",\"left\":-331,\"top\":-
351,\"width\":0,\"height\":700,\"fill\":\"rgb(0,0,0)\",\"stroke\":\"#ccc\
",\"strokeWidth\":1,\"strokeDashArray\":null,\"strokeLineCap\":\"butt\",\
"strokeDashOffset\":0,\"strokeLineJoin\":\"miter\",\"strokeUniform\":fals
e,\"strokeMiterLimit\":4,\"scaleX\":1,\"scaleY\":1,\"angle\":0,\"flipX\":
false,\"flipY\":false,\"opacity\":1,\"shadow\":null,\"visible\":true,\"ba
ckgroundColor\":\"\",\"fillRule\":\"nonzero\"….}",
  "__v": 0
}
```

Додаток Б

Програмний код класу ParkingPlace

```
import 'fabric';
declare var fabric;
fabric.ParkingPlace = fabric.util.createClass(fabric.Group, {
  type: 'ParkingPlace',
  initialize: function(options) {
    const defaultWidth = 50;
    const defaultHeight = 50;
    const items = [];
    const overriddenOptions = {
      width: options.width ?? defaultWidth,
      height: options.height ?? defaultHeight,
      top: options.top ?? 50,
      left: options.left ?? 50,
      label: options.label
    }
    if (options.objects && options.objects.find(obj => obj.type == 'rect')) {
      items.push(new fabric.Rect(options.objects.find(obj => obj.type == 'rect')));
    }
    else {
      items.push(new fabric.Rect({
        top: 0,
        left: 0,
        fill: "#808ece",
        width: defaultWidth,
        height: defaultHeight,
```

```
      }));
    }
    this.callSuper('initialize', items, overriddenOptions);
    this.on('modified', onModified)
    this.on('moving', onMoving);
    this.add(new fabric.Text(options.label, {
      fontSize: 30,
      originX: 'center',
      originY: 'center',
      fill: "#000000",
    }));
  },
  toObject: function() {
    return fabric.util.object.extend(this.callSuper('toObject'), {
      label: this.label
    });
  }
})
fabric.ParkingPlace.fromObject = function(object, callback) {
  const parkingPlace = new fabric.ParkingPlace(object);
  return callback(parkingPlace);
};
function onModified(options) {
  const group = options.transform.target;
  const parkingPlace = group._objects[0];
  const minWidth = 50;
  const minHeigth = 50;
  const calculatedWidth = (Math.round(group.getScaledWidth() / 10)) * 10;
```

```
  const calculatedHeight = (Math.round(group.getScaledHeight() / 10)) *
10;
  const newWidth = calculatedWidth < minWidth ? minWidth :
calculatedWidth;
  const newHeight = calculatedHeight < minHeigth ? minHeigth :
calculatedHeight;
  let newTop = -newHeight * 0.5;
  let newLeft = -newWidth * 0.5;
  group.set({
    width: newWidth,
    height: newHeight,
    scaleX: 1,
    scaleY: 1
  });
  parkingPlace.set({
    width: newWidth,
    height: newHeight,
    scaleX: 1,
    scaleY: 1,
    top: newTop,
    left: newLeft
  });
}
function onMoving(options) {
  const canvas = options.transform.target.canvas;
  let newTop = Math.round(options.transform.target.top / 10) * 10;
  let newLeft = Math.round(options.transform.target.left / 10) * 10;
  const minTop = 0;
  const maxTop = canvas.height - options.transform.target.height;
  const minLeft = 0;
```

```
  const maxLeft = canvas.width - options.transform.target.width;
  if (newTop < minTop)
    newTop = minTop
  if (newTop > maxTop)
    newTop = maxTop
  if (newLeft < minLeft)
    newLeft = minLeft
  if (newLeft > maxLeft)
    newLeft = maxLeft
  options.transform.target.set({
    left: newLeft,
    top: newTop
  });
}
```

Додаток В

Програмний код конструктора терміналу

```
import { Component, Input, Output, EventEmitter } from '@angular/core';

import 'fabric';

import { Terminal } from 'src/app/services/api/parking-system/models/terminals.model';

import { ParkingPlaceConfigurationFormGroup } from 'src/app/services/api/parking-system/models/terminal-form.model';

import { faTrash, faParking, faWifi, faSave } from '@fortawesome/free-solid-svg-icons';

import { MatDialog } from '@angular/material/dialog';

import { ParkingPlaceDialogComponent } from "./dialogs/parking-place-dialog.component/parking-place-dialog.component";

import { BeaconDialogComponent } from './dialogs/beacon-dialog.component/beacon-dialog.component';

import { resourceUsage } from 'process';

declare const fabric;

@Component({

  selector: 'terminal-constructor',

  templateUrl: './terminal-constructor.component.html',

  styleUrls: ['./terminal-constructor.component.scss']

})

export class TerminalConstructorComponent {

  private _terminal;

  public trashIcon = faTrash;

  public parkingIcon = faParking;

  public beaconIcon = faWifi;

  public updateIcon = faSave;


  public canvas;
```

```
  public parkingPlaceLabelFormGroup = new
ParkingPlaceConfigurationFormGroup();
  @Output() submitEventEmitter = new EventEmitter();
  @Input('terminal') set terminal(terminal: Terminal) {
    if (terminal) {
      this._terminal = terminal;
      this.initTerminalConstructor();
    }
  }
  get terminal() {
    return this._terminal;
  }
  constructor(private dialog: MatDialog) {}
  public readonly gridSize = 50;
  public readonly unitScale = 10;
  initTerminalConstructor() {
    this.canvas = new fabric.Canvas('canvas', {
      selection: false,
      borderColor: '#0f0',
      preserveObjectStacking: true
    });
    this.setSize();
    if (this.terminal.terminalScheme) {
      this.canvas.loadFromJSON(this.terminal.terminalScheme);
    }
    else {
      this.setGrid();
    }
  }
```

```
setSize() {
  const gridWidth = this.terminal.width * this.unitScale;
  const gridHeight = this.terminal.height * this.unitScale;
  this.canvas.setWidth(gridWidth);
  this.canvas.setHeight(gridHeight );
}
setGrid() {
  const grid = new fabric.Grid({
    width: this.terminal.width,
    height: this.terminal.height,
    gridSize: this.gridSize,
    unitScale: this.unitScale,
    selectable: false
  });
  this.canvas.add(grid);
}
addBeacon() {
  const dialogRef = this.dialog.open(BeaconDialogComponent);
  dialogRef.afterClosed().subscribe(result => {
    if (result) {
      const beacon = new fabric.Beacon(result);

      this.canvas.add(beacon);
    }
  })
}
addParkingPlace() {
  const dialogRef = this.dialog.open(ParkingPlaceDialogComponent);
  dialogRef.afterClosed().subscribe(result => {
```

```
      if (result) {
        const parkingPlace = new fabric.ParkingPlace({
          label: result.label
        });
        this.canvas.add(parkingPlace);
      }
    });
  }
  updateBeacon(beacon: any, data: any) {
    const dialogRef = this.dialog.open(BeaconDialogComponent, {data: data});
    dialogRef.afterClosed().subscribe(result => {
      if (result) {
        beacon.set(result);
        beacon.initialize(beacon);
        this.canvas.renderAll();
      }
    })
  }
  updateParkingPlace(parkingPlace:any, data: any) {
    const dialogRef = this.dialog.open(ParkingPlaceDialogComponent, {data: data});
    dialogRef.afterClosed().subscribe(result => {
      if (result) {
        parkingPlace.set(result);
        parkingPlace.initialize(parkingPlace);
        this.canvas.renderAll()
      }
    })
```

```
  }

  updateAvtiveObject() {
    const activeObject = this.canvas.getActiveObject();

    if (activeObject.type === fabric.Beacon.prototype.type) {
      this.updateBeacon(activeObject, activeObject)
    }
    if (activeObject.type === fabric.ParkingPlace.prototype.type) {
      this.updateParkingPlace(activeObject, activeObject);
    }
  }
  removeActiveObject() {
    this.canvas.remove(this.canvas.getActiveObject());
  }
  submit() {
    this.terminal.terminalScheme = JSON.stringify(this.canvas);
    this.submitEventEmitter.emit();
  }
}
```

Додаток Д

Приклад ARM шаблону для сховища

```
{
     "$schema":     "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
     "contentVersion": "1.0.0.0",
     "parameters": {
          "Environment": {
               "type": "string"
          }
     },
     "variables": {
          "location": "[resourceGroup().location]",
          "storageAccountName":                          "[concat('stacc', toLower(parameters('Environment')), 'pguif')]",
          "accountType": "Standard_LRS",
          "kind": "StorageV2",
          "accessTier": "Cool",
          "minimumTlsVersion": "TLS1_2",
          "supportsHttpsTrafficOnly": true,
          "allowBlobPublicAccess": true,
          "networkAclsBypass": "AzureServices",
          "networkAclsDefaultAction": "Allow"
     },
```

```
    "resources": [
        {
            "name": "[variables('storageAccountName')]",
            "type": "Microsoft.Storage/storageAccounts",
            "apiVersion": "2019-06-01",
            "location": "[variables('location')]",
            "properties": {
                "accessTier": "[variables('accessTier')]",
                "minimumTlsVersion":
"[variables('minimumTlsVersion')]",
                "supportsHttpsTrafficOnly":
"[variables('supportsHttpsTrafficOnly')]",
                "allowBlobPublicAccess":
"[variables('allowBlobPublicAccess')]",
                "networkAcls": {
                        "bypass":
"[variables('networkAclsBypass')]",
                        "defaultAction":
"[variables('networkAclsDefaultAction')]",
                        "ipRules": []
                }
            },
            "dependsOn": [],
            "sku": {
                "name": "[variables('accountType')]"
            },
            "kind": "[variables('kind')]",
            "tags": {
                "Environment": "[parameters('Environment')]"
            }
```

```
            }
        ],
        "outputs": {}
}
```