Разработка веб-приложения для фотографирования и детектирования дорожных повреждений

Гатаулин Александр Денисович, Бабчинецкий Сергей Геннадиевич
Санкт-Петербургский государственный университет аэрокосмического приборостроения, Санкт-Петербург, Россия
aleksandrgataulin745@gmail.com
lnpt@guap.ru

Аннотация. Рассматривается вопрос обеспечения федеральных, муниципальных и частных компаний актуальной информацией о состоянии дорожного полотна для проведения предварительного анализа геопространственных данных с помощью веб-приложения для фотографирования гражданскими лицами дорожных повреждений, дальнейшего их детектирования и построения графической карты с размеченными обнаруженными дефектами. Отмечается возможность использования камеры смартфона для создания фотоснимков целевых объектов, сверточной нейронной сети и алгоритмов компьютерного зрения для обработки изображений, создания и просмотра графической карты с векторными слоями. Приведены примеры веб-приложений для использования гражданской позиции в сферах благоустройства городской местности, архитектура работы приложения, включающую основные бизнес-процессы, микросервисы и технологии взаимодействия. Также показаны результаты разработки и дальнейшие перспективы развития проекта.
Ключевые слова. Дорожные повреждения, микросервисы, геопространственные данные, графическая карта, компьютерное зрение, сверточная нейронная сеть.

Development of a web application for photographing and detecting road damage

Gataulin Alexander Denisovich, Babchinetsky Sergey Gennadievich
Saint-Petersburg State University of Aerospace Instrumentation, St. Petersburg, Russia
aleksandrgataulin745@gmail.com
lnpt@guap.ru

Abstract. The issue of providing federal, municipal and private companies with up-to-date information on the condition of the roadway for conducting a preliminary analysis of geospatial data using a web application for photographing road damage by civilians, further detecting them and building a graphic map with marked detected defects is being considered. The possibility of using a smartphone camera to create photographs of target objects, a convolutional neural network and computer vision algorithms for image processing, creating and viewing a graphic map with vector layers is noted. Examples of web applications for the use of citizenship in the areas of urban

improvement, the architecture of the application, which includes the main business processes, microservices and interaction technologies, are given. The results of the development and further prospects for the development of the project are also shown.

Keywords. Road damage, microservices, geospatial data, graphic map, computer vision, convolutional neural network.

Введение

Обслуживание дорожного полотна является важной и актуальной задачей по всему миру, в связи с развитием транспортной инфраструктуры и увеличением трафика движения из года в год. Диагностикой дорог занимаются определенные государственные и частные организации, включая федеральные, муниципальные, частные компании по инспекции дорожного покрытия, научно-исследовательские организации [1]. Геопростанственная информация о состоянии дорожного полотна может потребоваться вышеперечисленным субъектам, частным предпринимателям и владельцам земельных участков, транспортным компаниям и автопаркам. В связи с отсутствием предварительного анализа геопространственных данных правительство и дорожные компании не имеют общего представления и актуальной информации о транспортно-эксплуатационном состоянии дорожного покрытия города. Поэтому в труднодоступных и неприметных районах появляются опасные участки для водителей автомобилей, что влечет к возникновению ДТП и поломке транспортного средства. В связи с этим возникает потребность в разработке веб-приложения с общей графической картой с размеченными дорожными повреждениями, которые будут фиксироваться с помощью лиц с активной гражданской позицией. Цель исследования заключается таким образом в обеспечении правительственных организаций и дорожных служб актуальной информацией о количестве и характере повреждений с помощью городского населения.

Техническое задание и требуемая функциональность

Приложение должно иметь следующий функционал со стороны клиента:

- авторизация и регистрация пользователя под логином и email с данными физического или юридического лица [2],
- фотографирование дорожных повреждений и получение ответа о валидации изображения,
- просмотр общей карты города с размеченными дорожными повреждениями с возможностью просмотра местоположения дефекта, его класса, адреса и изображения,
- получение бонусов за верифицированное изображение для возможности обновления внешнего вида векторного слоя, получения доступа к другим

районам города и покупки за внутреннюю валюту подписки для расширения функционала.

Со стороны сервера веб-приложение должно иметь:

- грамотную систему авторизации и аутентификации пользователей с раздачей временных токенов для работы в приложении,
- модель машинного обучения для детекции и классификации дорожных повреждений [3],
- алгоритмы проверки схожести изображений для избавления от дубликатов записей и переполнения базы данных [4],
- меры безопасности персональных данных пользователей с шифрованием данных по сети и шифрованием паролей [5],
- оптимизированную базу данных пользователей, изображений и дорожных повреждений для эффективного хранения и гибкого взаимодействия с информацией,
- систему обработки координат и рендера растрового слоя графической карты с векторным слоем целевых объектов,
- реализацию бизнес-логики, включая расчет внутренней валюты, раздачу подписок и отслеживание окончания срока работы подписки,
- горизонтально-масштабируемую архитектуру для дальнейшего расширения и развития проекта,
- мониторинг и аналитику для отслеживания производительности и сбора данных о взаимодействии пользователей с веб-приложением.

Разработка и реализация веб-приложения

Процесс разработки приложения имеет несколько стадий:

1. ***Постановка задачи и анализ.***
   Определение целей, требований и целевой аудитории.
2. ***Проектирование архитектуры***.
   Определение основных микросервисов, технологий, протоколов обмена сообщениями [6]. Построение схемы базы данных.
3. ***Разработка.***
   Разработка серверной и клиентской части приложения, используя выбранные технологии. Разработка бизнес-логики, пользовательского интерфейса, взаимодействия с базой данных.
4. ***Тестирование.***
   Юнит-тестирование, интеграционное тестирование, системное тестирование, тестирование производительности [7].
5. ***Деплой и оптимизация.***
   Развертывание приложения на сервере, обеспечение безопасности и настройка базы данных [8].
6. ***Сопровождение и обновления.***
   Мониторинг и поддержка, отслеживание работы приложения, исправление ошибок, внедрение обновлений в систему.

Архитектура веб-приложения

В ходе работы была выбрана микросервисная архитектура разработки приложения, так как данный подход имеет следующие преимущества:

- *Масштабируемость*:
  Возможность масштабирования каждого микросервиса отдельно, что позволяет легко управлять ресурсами и поддерживать высокую производительность.
- *Гибкость и маневренность*:
  Независимость микросервисов позволяет разработчикам внедрять изменения, обновления и исправления отдельных служб, минимизируя воздействие на остальные компоненты системы.
- *Легкость разработки и поддержки*:
  Каждый микросервис может быть разработан и поддерживаться независимо, что упрощает жизненный цикл разработки и облегчает внесение изменений.
- *Технологическое разнообразие:*
  Возможность использования разных технологий и языков программирования для различных микросервисов, что позволяет выбирать наилучшие инструменты для конкретных задач.
- *Автономность и надежность:*
  Если один из микросервисов перестает работать, это не влияет на работоспособность других. Аварийные ситуации локализованы, что улучшает общую надежность системы.
- *Быстрое развертывание:*
  Возможность развертывания, обновления и отката каждого микросервиса независимо от остальных, что упрощает и ускоряет процесс разработки и обслуживания.
- Распределенная разработка:
  Команды могут параллельно работать над разными микросервисами, что способствует распределенной разработке и повышает эффективность работы.
- *Легкость внедрения новых технологий:*
  Возможность внедрения новых технологий в отдельные микросервисы без необходимости обновления всей системы.
- *Резистентность к сбоям:*
  В случае сбоя в одном микросервисе, другие продолжают функционировать, что обеспечивает стойкость системы в целом.
- *Эффективное использование ресурсов:*
  Возможность оптимизировать ресурсы, выделяемые каждому микросервису в зависимости от его нагрузки и требований.

Рис. 1 – Микросервисная архитектура серверной части приложения

Сервисы будут взаимодействовать друг с другом и обмениваться данными м помощью протокола HTTP, используя REST API посредством стандартных HTTP методов: (GET, POST, PUT, DELETE). В дальнейшем с развитием системы и повышения трафика пользователей будут использоваться брокеры сообщений, например RabbitMQ, Apache Kafka, Reddis и тд.

Данные о пользователях и о ямах будут храниться в базе данных под системой управления PostgreSQL. На рис. 2 показана схема связей таблиц друг с другом [9].

За хранение данных о пользователях отвечают 3 таблицы, одна из которых «client» содержит основную информацию о юридических и физических лицах. Также созданы две таблицы «fiz_property» и «enterprise_property», связанные с таблицей «client» по внешнему ключу «client_id». Также есть отдельная таблица, хранящая данные о подписках пользователей, соединенная с «fiz_property» по внешнему ключу «subscribe_id».

За хранение данных о ямах отвечает таблица «Pothole_Point», она хранит данные о дате добавления, улице, классе, пользователе, который добавил информацию, адрес картинки и геометрию объекта, тип которого предоставляется плагином PostGIS. Имеет 4 внешних ключа «district_id», «Pothole_class», «picture_adress», «client_id», которые связаны с таблицами данных об улице, классе, картинке и пользователе соответственно.

Типы связи таблиц:

- «client» - «fiz_property», «client» - «enterprise_property», «Pothole_Point» - «Picture»: 1 к 1,
- «District» - «Pothole_Point», « Pothole_Class» - «Pothole_Point», «Subscribe» - «fiz_property»: 1 ко многим,
- «Pothole_Point» - «client»: многие ко многим.

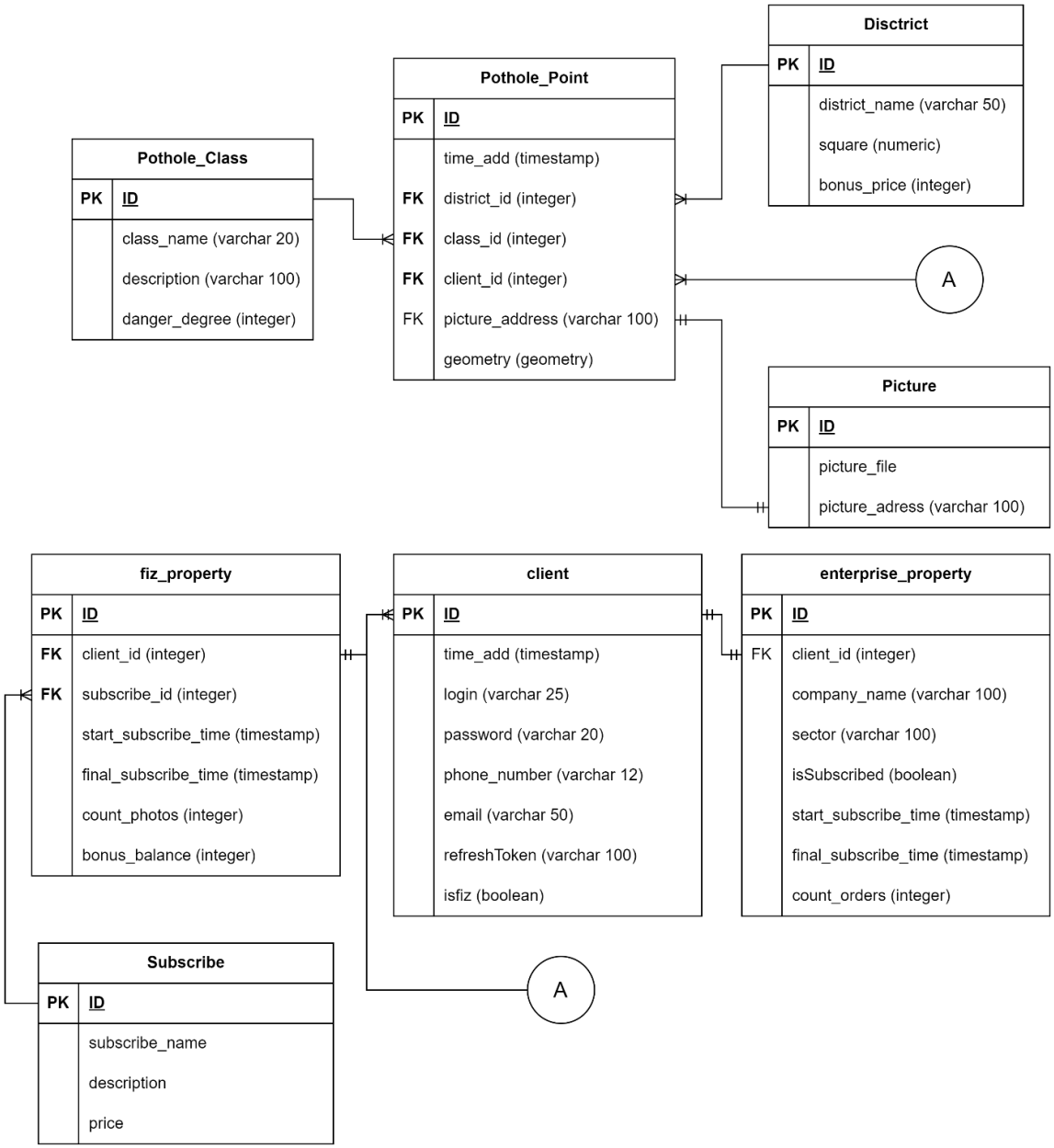

Рис. 2 – Схема базы данных

Языки программирования и библиотеки

Для серверной части была выбрана среда выполнения для JavaScript – Node.js [10], имеющая эффективные ввод и вывод, использующая асинхронную и событийно-ориентированную модель, поддержку модулей, WebSocket и пакетный менеджер npm. Также был выбран фреймворк для разработки серверных приложений Express, предоставляющий Midddlewares [11], работу с WebSocket, грамотное управление и взаимодействие с микросервисами, поддержка обработки файлов и изображений, а также возможность проксирования запросов на сервер.

Для клиентской части был выбран язык программирования TypeScript и фреймворк для построения клиентских приложений Angular, предоставляющий компонентную архитектуру, двустороннее связывание

данных, модульность, механизм Dependency Injection (DI), роутинг, шаблоны и директивы [12].

Для решения задач компьютерного зрения был выбран язык программирования Python и фреймворк OpenCV [13], так имеется большое количество решений тех или иных задач CV с помощью этой библиотеки. Плюс, данный фреймворк прост в изучении и его функциональность достаточна для реализации проекта.

Модулем в Angular, предоставляющим работу с веб-картами, будет Leaflet. Он имеет простой и удобный API интерфейс, широкий функционал работы с веб-картами, возможность установки слоя карты, векторных слоев, проецирования в различных системах координат.

Сверточная нейронная сеть

В качестве сверточной нейронной сети выбор пал на современную модель YoloV8 [14], так как она имеет ряд преимуществ:

- Новейшая модель в своей линейке
- Использование в качестве вычислительного оборудования видеокарт NVIDIA с ядрами CUDA, на которые падает вся нагрузка (на данный момент команда оснащена видеокартами NVIDIA GeForce RTX 3060)
- Точное выполнение задачи детекции дорожных дефектов
- Простое взаимодействие с нейросетью через CLI панель и через язык программирования Python
- Подробная документация по использованию модели
- Встроенная возможность трекинга объектов, что позволит не делать запись координат одной ямы в таблице каждый кадр

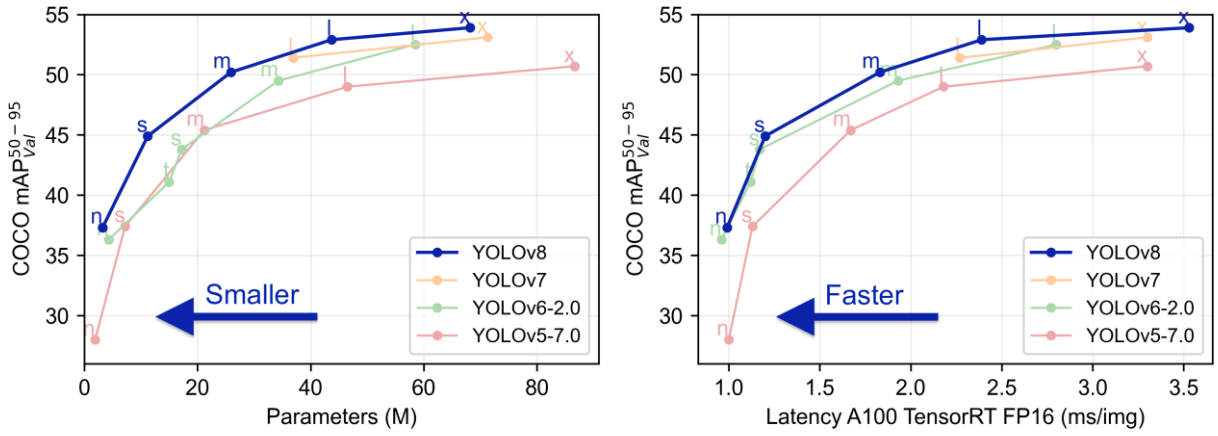

Рис. 3 – Графики сравнения показателей моделей семейства Yolo

Таблица 1

Технические характеристики моделей YoloV8

Model	Size (pixels)	mAP (50-95)	Speed (CPU ONNX ms)	Speed (A100 TensorRT ms)	Params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2

Клиентская часть

Веб-приложение разделяются на два основных интерфейса. Первый представляет собой набор обзорных веб-страниц и личный кабинет пользователя. Второй представляет собой внешний вид приложения с самим функционалом фотографирования и просмотра графической карты.

Веб-страницы обзорного интерфейса [15]:

1. Главная страница, содержащая основную информацию о проекте,
2. Страница «О нас», повествующая о компании и ее деятельности,
3. Набор страниц «Проекты», в которой содержится информация о проектах компании,
4. Авторизация и регистрация,
5. Личный кабинет.

Веб-страницы функционального приложения:

1. Страница профиля пользователя с просмотром данных логина, почты, количества верифицированных записей, текущей подписки, баланса внутренней валюты,
2. Страница графической карты с размеченными целевыми объектами,
3. Страница для создания записи с функционалом фотографирования дефекта, привязки изображения к географическим координатам,
4. Страница «Магазин» для покупки подписок, доступов к районам, внешнего вида векторной графики.

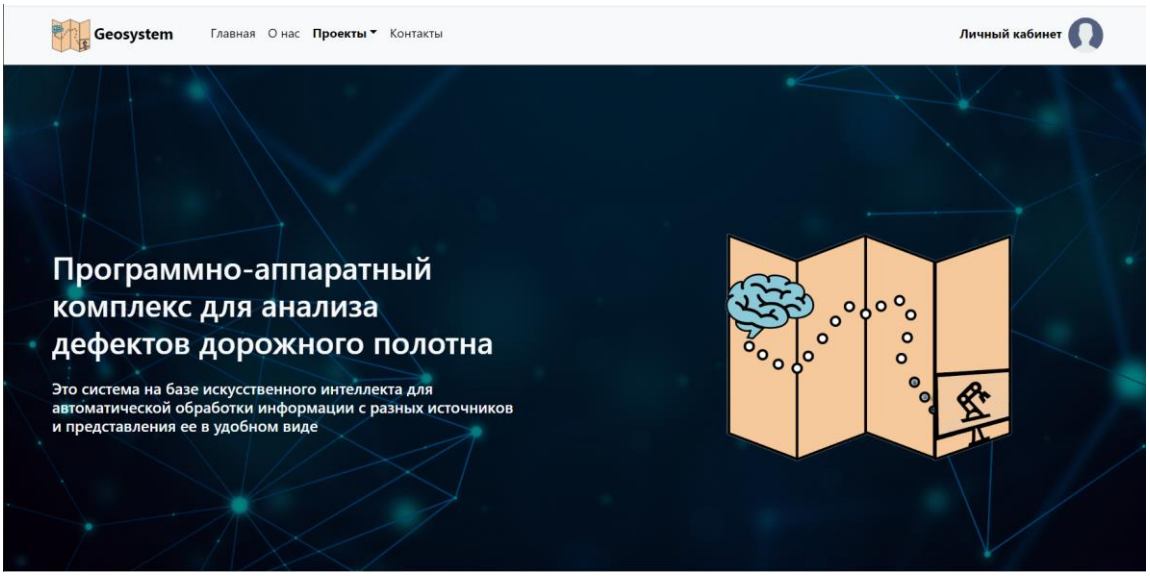

Рис. 4 – Главная страница интерфейса обзорного сайта

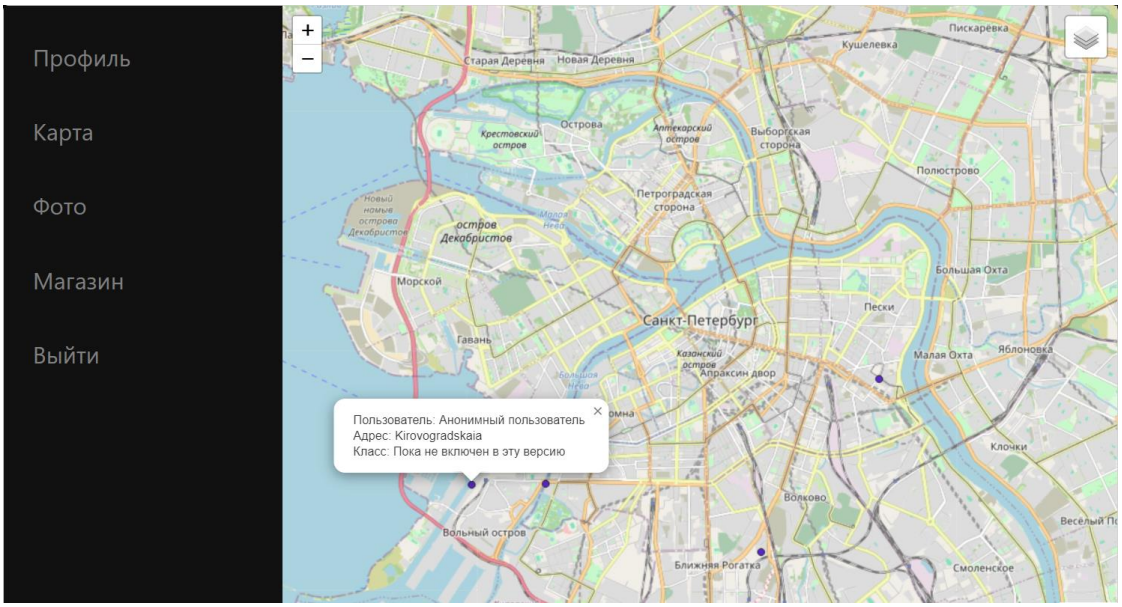

Рис. 5 – Страница графической карты функционального интерфейса

Интерактивная графическая карта

Реализация графической карты происходит с помощью модуля Leaflet [16], который с помощью консольной команды «npm install leaflet» добавляется в проект в папку «node_modules» и обозначается в конфигурационном файле проекта Angular в «package.json». Вся логика работы происходит в компоненте map-page.component.ts и в сервисе map.service.ts.

Импортирование модуля происходит в следующем виде: «import * as L from 'leaflet'». Далее поля и методы используются через обращение к статическим фукнциям – L.*action*.

Функционал работы сервиса и компонента:

1) Отображение подложки карты.

a) Создается объект карты: «new L.Map(…properties).setView(center, zoom)»,

b) L.tileLayer(*путь до слоя подложки*).addTo(*объект карты*);

2) Добавление маркеров в различные группы слоев, классифицируемых по районам города, чтобы была возможность отображать нужные слои в зависимости от подписки и выкупленных районов.

a) const marker = L.marker([potholeData.geometry.coordinates[0], potholeData.geometry.coordinates[1]])

b) Далее маркер добавляется в набор объектов LayerGroup. Прототип метода добавления маркера в группу на TypeScript: private getPotholeDistrict(districtValue): L.LayerGroup ();

3) Переключение между слоями подложки и слоями районов для гибкого взаимодействия с картой.
Реализуется с помощью создания контроля слоев new L.Control.Layer, в который добавляются нужные слои tileLayer и LayerGroup, что даст возможность переключать виды отображения карты.

4) Кластеризация маркеров в зависимости от масштаба карты (потребуется отдельное импортирование модуля «npm install leaflet.markercluster»)

a) В полях класса добавлена переменная private markerClusterGroup: L.MarkerClusterGroup,

b) Добавление маркера в группу кластеров происходит в следующей строчке: this.markerClusterGroup.addLayer(marker);

5) Смена рисунка маркеров в зависимости от приобретенного изображения на странице «Магазин». На данный момент еще не написана логика работы с магазином.

6) Запрос на получение географических координат ям и их данных происходит через сервис с помощью модуля «HttpClient», импортированного из «@angular/common/http» из node_modules. Посылая запрос с помощью асинхронного метода «this.http.get<any>(*API URL*)» [17], получаем объект типа «Observable<any>», у которого, вызвав метод «subscribe». Внутри данного метода идет дальнейшая обработка полученной информации.

Реализация обработки фотографий

На стороне клиента за обработку фотографий отвечает компонент media-processing.component.ts и сервис media.service.ts. В данных классах происходит прием загруженной фотографии, отображение ее в элементе UI Canvas, отправка изображения в Backend, откуда модуль express, являясь в данном случае прокси сервером, перенаправляет запрос в микросервис для обработки изображений.

За обработку фотографий отвечает микросервис, написанный на языке Python. Он содержит следующие библиотеки для обработки запросов и изображений:

- Flask [18]. Нужен для принятия http запросов и отправки ответа в формате JSON [19], содержащего закодированное обработанное изображение, данные координат, статус выполнения запроса и сообщение об успехе или неудаче.

- Opencv-python. Нужен для проверки изображения на наличие в базе данных, его обработки сверточной нейросетью.
- Ultralytics. Фреймворк для выполнения задач компьютерного зрения, содержащий предобученные архитектуры моделей серточных нейронных сетей. В данном случае используется обученная на датасете платформы Roboflow [20] модель нейросети YoloV8.

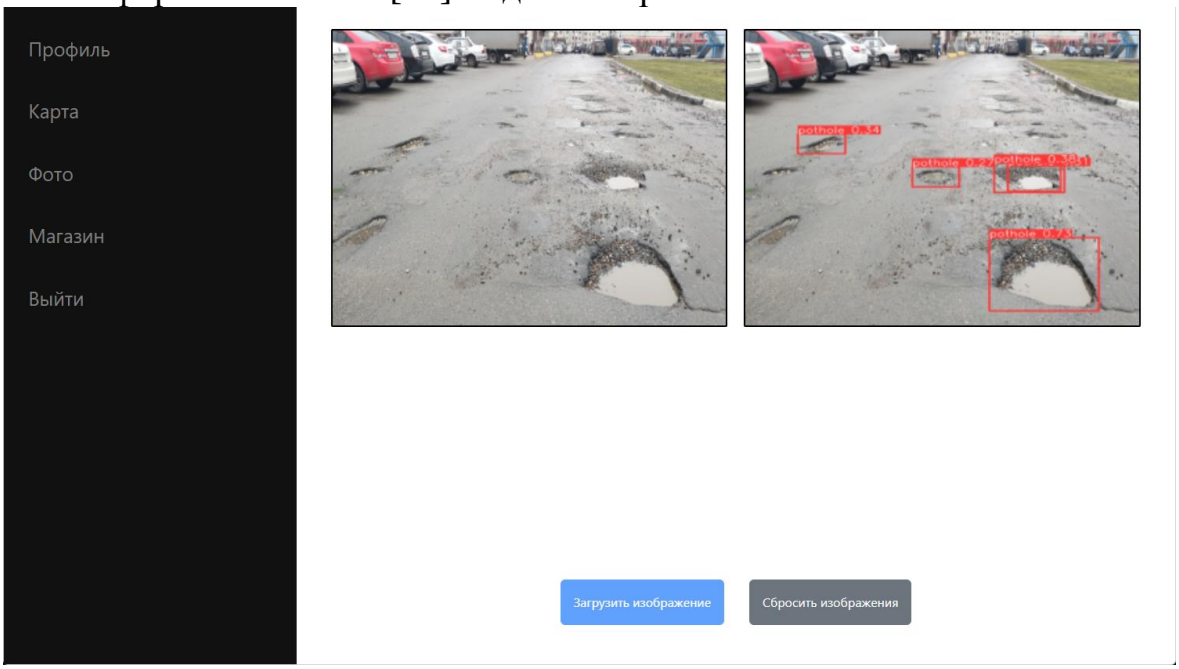

Рис. 6 – Страница обработки изображений

Заключение

В контексте развития веб-приложения для фотографирования дорожных повреждений гражданскими лицами и дальнейшего детектирования их с использованием сверточных нейронных сетей и алгоритмов компьютерного зрения, полученные результаты являются обнадеживающими. Разработанное приложение предоставляет эффективное средство для предварительного анализа геопространственных данных о состоянии дорожного полотна.

Проект успешно интегрирует современные технологии, такие как использование камер смартфонов, сверточных нейронных сетей и графических карт для визуализации обнаруженных дефектов. Микросервисная архитектура обеспечивает гибкость и масштабируемость системы, а использование современных методов обработки изображений снижает вероятность ошибок при детектировании дефектов.

В перспективах развития проекта стоит внедрение системы бонусов и подписок, привязка географических координат к изображениям, обеспечение взаимодействия с магазином, доработка приложения и проведение Unit тестов для окончательного деплоя приложения.

Библиографический список

1.Жилин, Н. С. Диагностика автомобильных дорог для повышения безопасности движения // Мир дорог. – 2020. – № 127. – С. 58-59.

2.von Holst C. et al. The work of the European Union Reference Laboratory for Food Additives (EURL) and its support for the authorisation process of feed additives in the European Union: a review //Food Additives & Contaminants: Part A. – 2016. – Т. 33. – №. 1. – pp. 66-77.

3.В. К. Козыревский, А. И. Веселов. Способ уменьшения вычислительной сложности процедуры обучения детектора лиц на базе метода Виолы-Джонса // Труды МАИ. – 2017. – № 93. – С. 24.

4.Jhamtani H., Berg-Kirkpatrick T. Learning to describe differences between pairs of similar images //arXiv preprint arXiv:1808.10584. – 2018.

5.М. А. Иконников, И. Н. Карманов. Меры и требования к защищенным веб-приложениям // Интерэкспо Гео-Сибирь. – 2019. – Т. 6, № 2. – С. 13-19. – DOI 10.33764/2618-981X-2019-6-2-13-19.

6.А. А. Рыбалко, А. В. Наумов. Модель обеспечения отказоустойчивости контейнерных виртуальных сервисов в центрах обработки данных // Труды МАИ. – 2017. – № 97. – С. 20.

7.E. Habibi, S. H. Mirian-Hosseinabadi. Sharif-TaaWS: a tool to automate unit testing of web services // Automated Software Engineering. – 2022. – Vol. 30, No. 1. – pp. 1-30. – DOI 10.1007/s10515-022-00368-4.

8.Nichols J., Hua Z., Barton J. Highlight: a system for creating and deploying mobile web applications //Proceedings of the 21st annual ACM symposium on User interface software and technology. – 2008. – pp. 249-258.

9.Дудаков, Н. С. Методика проектирования баз данных для автоматизированных систем управления специального назначения / Н. С. Дудаков, К. В. Макаров, А. В. Тимошенко // Труды МАИ. – 2016. – № 90. – С. 25.

10. Bangare S. L. et al. Using Node. Js to build high speed and scalable backend database server //International Journal of Research in Advent Technology. – 2016. – Т. 4. – pp. 19.

11. M. Reza Selim, Yu. Goto, J. Cheng. A Resilient Message Queuing Middleware // Computer Sciences and Telecommunications. – 2008. – No. 2(16). – pp. 77-92.

12. Т. А. Алешина, В. Ю. Белаш. Актуальные возможности фреймворка Angular/AngularJS для web-разработок // Научные труды Калужского государственного университета имени К.Э. Циолковского, 2019. – С. 507-510.

13. P. Bailke, S. Divekar. Real-time moving vehicle counter system using opencv and Python // International Journal of Engineering Applied Sciences

and Technology. – 2022. – Vol. 6, No. 11. – pp. 190-194. – DOI 10.33564/ijeast.2022.v06i11.036.

14. Shen L., Lang B., Song Z. DS-YOLOv8-Based Object Detection Method for Remote Sensing Images //IEEE Access. – 2023. – Т. 11. – pp. 125122-125137.

15. К. О. Атеев. Анализ трендов современных технологий разработки пользовательского интерфейса в веб-приложениях // Тенденции развития науки и образования. – 2020. – № 68-1. – С. 6-9. – DOI 10.18411/lj-12-2020-01.

16. Kaluža M., Troskot K., Vukelić B. Comparison of front-end frameworks for web applications development //Zbornik Veleučilišta u Rijeci. – 2018. – Т. 6. – №. 1. – pp. 261-282.

17. Кузнецова, С. В. Особенности кросс-платформенной разработки мобильных приложений с использованием Xamarin / С. В. Кузнецова // Труды МАИ. – 2022. – № 125. – DOI 10.34759/trd-2022-125-21.

18. Grinberg M. Flask web development: developing web applications with python. – " O'Reilly Media, Inc.", 2018.

19. Д. В. Кучуганов. Разработка библиотеки обработки JSON-форм для веб-приложений // Информационные технологии в науке, промышленности и образовании : Сборник трудов региональной научно-технической конференции, Ижевск, 2018. – С. 141-144.

20. Звайгзне А. Ю. Импортирование, проверка готовой модели нейронной сети, обучение новой нейронной сети на основе готового датасета //Постулат. – 2023. – №. 7 июль.

References

1. Zhilin, N. S. Diagnostics of highways to improve traffic safety//World of roads. – 2020. – № 127. - pp. 58-59.
2. von Holst C. et al. The work of the European Union Reference Laboratory for Food Additives (EURL) and its support for the authorisation process of feed additives in the European Union: a review //Food Additives & Contaminants: Part A. – 2016. – Т. 33. – №. 1. – pp. 66-77.
3. V. K. Kozyrevsky, A. I. Veselov. A method for reducing the computational complexity of a face detector training procedure based on the Viola-Jones method // Proceedings of MAI. – 2017. – No. 93. – pp. 24.
4. Jhamtani H., Berg-Kirkpatrick T. Learning to describe differences between pairs of similar images //arXiv preprint arXiv:1808.10584. – 2018.
5. M. A. Ikonnikov, I. N. Karmanov. Measures and requirements for secure web applications//Interexpo Geo-Siberia. – 2019. - T. 6, NO. 2. - pp. 13-19. – DOI 10.33764/2618-981X-2019-6-2-13-19.

6. A. A. Rybalko, A. V. Naumov. Model for ensuring fault tolerance of container virtual services in data centers // Proceedings of MAI. – 2017. – No. 97. – pp. 20.
7. E. Habibi, S. H. Mirian-Hosseinabadi. Sharif-TaaWS: a tool to automate unit testing of web services // Automated Software Engineering. – 2022. – Vol. 30, No. 1. – pp. 1-30. – DOI 10.1007/s10515-022-00368-4.
8. Nichols J., Hua Z., Barton J. Highlight: a system for creating and deploying mobile web applications //Proceedings of the 21st annual ACM symposium on User interface software and technology. – 2008. – pp. 249-258.
9. Dudakov, N. S. Methodology for designing databases for automated control systems for special purposes / N. S. Dudakov, K. V. Makarov, A. V. Timoshenko // Proceedings of MAI. – 2016. – No. 90. – pp. 25.
10. Bangare S. L. et al. Using Node. Js to build high speed and scalable backend database server //International Journal of Research in Advent Technology. – 2016. – T. 4. – pp. 19.
11. M. Reza Selim, Yu. Goto, J. Cheng. A Resilient Message Queuing Middleware // Computer Sciences and Telecommunications. – 2008. – No. 2(16). – pp. 77-92.
12. T. A. Aleshina, V. Yu. Belash. Current features of the Angular/AngularJS framework for web development//Scientific works of Kaluga State University named after K.E. Tsiolkovsky, 2019. - pp. 507-510.
13. P. Bailke, S. Divekar. Real-time moving vehicle counter system using opencv and Python // International Journal of Engineering Applied Sciences and Technology. – 2022. – Vol. 6, No. 11. – pp. 190-194. – DOI 10.33564/ijeast.2022.v06i11.036.
14. Shen L., Lang B., Song Z. DS-YOLOv8-Based Object Detection Method for Remote Sensing Images //IEEE Access. – 2023. – T. 11. – pp. 125122-125137.
15. K.O. Ateev. Analysis of trends in modern technologies for developing a user interface in web applications//Trends in the development of science and education. – 2020. – № 68-1. - pp. 6-9. – DOI 10.18411/lj-12-2020-01.
16. Kaluža M., Troskot K., Vukelić B. Comparison of front-end frameworks for web applications development //Zbornik Veleučilišta u Rijeci. – 2018. – T. 6. – №. 1. – pp. 261-282.
17. Kuznetsova, S.V. Features of cross-platform development of mobile applications using Xamarin / S.V. Kuznetsova // Proceedings of MAI. – 2022. – No. 125. – DOI 10.34759/trd-2022-125-21.
18. Grinberg M. Flask web development: developing web applications with python. – " O'Reilly Media, Inc.", 2018.
19. D.V. Kuchuganov. Development of a library for processing JSON forms for web applications//Information technologies in science, industry and education: Collection of works of the regional scientific and technical conference, Izhevsk, 2018. - pp. 141-144.

20. Zvaigzne A. Yu. Importing, testing a ready-made neural network model, training a new neural network based on a ready-made dataset // Postulate. – 2023. – No. 7 July.