УДК 004.94

А.Б. Гомбоин, А.М. Кориков

Параллельная обработка географических данных в мобильном приложении «Мобильная ГИС»

Представлен подход к параллельной обработке географических данных в мобильных приложениях на языке программирования Dart. Предлагается способ эффективного распределения и обработки данных о газовых, нефтяных и т.п. месторождениях с использованием параллельных вычислений на смартфонах. Предлагаемый способ использует параллельные потоки мобильного устройства. Цель исследования – повышение производительности и скорости обработки географических данных, что является критически важным для геоинформационных приложений. Предложенный метод демонстрирует значительные улучшения по сравнению с обычными методами обработки географических данных и обеспечивает более быструю и эффективную работу мобильных приложений, что делает его важным инструментом для повышения эффективности и удобства использования мобильных геоинформационных приложений.

Ключевые слова: параллельное программирование, Flutter, Dart, Isolate, FutureGroup, географические данные, ГИС, мобильная ГИС.
DOI: 10.21293/1818-0442-2024-27-2-51-56

В наше время невозможно представить человека без мобильных устройств. Одним из таких устройств являются смартфоны. Смартфоны – портативные устройства, сочетающие в себе функции телефона и персонального компьютера (ПК). Они позволяют пользователям совершать звонки, отправлять сообщения, просматривать веб-страницы, запускать приложения и пользоваться ими. Современные мобильные приложения становятся все более функциональными и предлагают пользователям широкий спектр возможностей, включая геоинформационные системы (ГИС), которые обрабатывают и отображают географические данные [1].

В последние годы пользователи часто используют ГИС на ПК и смартфонах. ГИС стали неотъемлемой частью повседневной жизни и используются для навигации, локации объектов, построения маршрутов и т.п. Это обеспечивает экономию времени, ускоряет прибытие пользователя на место назначения. Существование многих пользователей сложно представить без ГИС на смартфонах: они обеспечивают удобство и продуктивность их жизни.

Увеличение объема информации, необходимой для отображения геоинформационных данных в приложениях ГИС, работающих в режиме реального времени, обнаруживает недостатки по скорости загрузки географических данных и производительности мобильных приложений, что доказывает актуальность разработки эффективных методов обработки геоинформационных данных.

В контексте вышесказанного одним из возможных вариантов эффективной обработки данных становится использование параллельной обработки географической информации в мобильных приложениях. Параллельная обработка позволяет распределить вычислительные задачи на независимые потоки исполнения, что может уменьшить общее время работы мобильных приложений, повысить их производительность и ускорить обработку данных. Поэтому актуально рассмотреть приложение, написанное на фреймворке Flutter [2]. Для реализации параллельной обработки данных этот фреймворк предоставляет Isolate – класс во Flutter для запуска кода в отдельных изолированных потоках.

Постановка задачи

Целью исследования является повышение производительности и скорости обработки географических данных на основе методов их параллельной обработки в мобильных приложениях на Flutter. Предлагается рассмотреть одно из перспективных приложений: «Мобильная ГИС», созданное благодаря фреймворку Flutter и использующее Isolate. На данном приложении поставлены и решены задачи: реализация параллельной обработки данных; сравнительный анализ результатов производительности мобильного приложения до и после внедрения разработанного метода параллельной обработки географических данных.

Географические данные

Географические данные представляют собой информацию, описывающую местоположение и характеристики объектов или явлений на Земле, и бывают в основном двух типов: векторные и растровые [3].

Векторные данные представляют объекты на карте с помощью точек, линий и полигонов [4]. Точка обозначает конкретное местоположение объекта, например банкомата или магазина. На рис. 1 представлен пример точки. Линия представляет собой соединенную последовательность точек, используемую для отображения линейных объектов (рис. 2), например, дорога или река. Полигон – замкнутая фигура, образованная линиями и используемая для обозначения областей (рис. 3), например, город, озеро или страна. Также стоит выделить ещё один объект – это текст, который показывает название чего-либо на карте.

Растровые данные представляют собой сетку пикселей, которые содержат значение, соответствующее определенной местности, например, аэрофотосъемка, карта высот и т.п. [5]. Примером растровых

данных является фрагмент карты любого города из растровой карты OpenStreetMap.

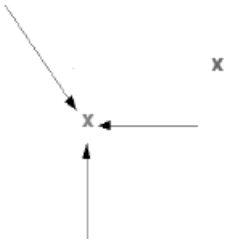

Рис. 1. Изображение точки в векторных данных

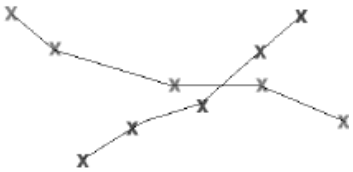

Рис. 2. Изображение линии в векторных данных

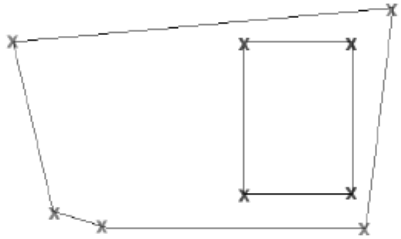

Рис. 3. Изображение полигонов в векторных данных

Для описанных выше типов географических данных существует множество форматов для хранения и обмена ими. Рассмотрим наиболее распространенные.

WKT (Well-Known Text) – текстовый формат для представления векторных геометрических объектов [6, 7]. Пример объектов:

1. POINT (30 10) – точка с координатами (долгота 30, широта 10).

2. LINESTRING (30 10, 10 30, 40 40) – линия, проходящая через три точки.

3. POLYGON ((30 10, 40 40, 20 40, 10 20, 30 10)) – полигон, образованный пятью точками.

Стоит отметить, что существуют ещё такие объекты, как MULTIPOINT, MULTILINE, MULTIPOLYGON. Различие данных объектов от ранее перечисленных в том, что они содержат геометрию нескольких фигур, но относятся к одному объекту на карте.

WKB (Well-Known Binary) – бинарный формат для предоставления тех же векторных геометрических объектов, что и WKT, но который является более компактным и эффективным [8, 9].

GeoJSON – расширение формата JSON [10, 11], которое разработано специально для представления географических объектов. Этот формат отличается простотой и читабельностью как для человека, так и для компьютера. Пример объектов представлен на рис. 4.

Shapefile – популярный формат, разработанный компанией Esri, состоящий из нескольких файлов, хранящих геометрию, атрибуты и метаданные [12, 13].

```
{
    "type": "Feature",
    "geometry": {
      "type": "Point",
      "coordinates": [125.6, 10.1]
    },
    "properties": {
      "name": "Париж"
    }
}
```

Рис. 4. Пример формата данных GeoJSON

GeoTIFF – формат растровых изображений, дополненный географической привязкой, позволяющей точно позиционировать изображение на земной поверхности [14, 15].

Описанные выше форматы географических данных предоставляют мощный инструмент для решения широкого круга прикладных задач. Понимание основы работы с ними, типов их форматов и возможностей использования становится необходимым для специалистов в областях, связанных с ГИС.

Параллельность во фреймворке Flutter

Для выполнения параллельной работы во фреймворке Flutter существует Isolate. Стоит отметить, что данный фреймворк работает на языке программирования Dart, который является однопоточным, поэтому используется Isolate для обеспечения параллельного выполнения операций.

Isolate во Flutter – независимый поток выполнения, который может работать параллельно с основным потоком приложения [16]. Он используется для выполнения задач CPU без блокировки основного потока приложения.

Для создания второстепенного изолята используется класс Isolate и метод (функция) spawn, также существует более упрощенный способ использования с помощью метода compute. Когда изолят создается, он получает свои собственные ресурсы, включая собственное пространство памяти, стек вызовов и регистры процессора. Это позволяет изолятам работать параллельно, не влияя друг на друга или на основной поток выполнения приложения.

Класс Isolate позволяет разделить сложные задачи на более мелкие и выполнять их параллельно, например, обработка больших объемов данных, выполнение интенсивных вычислений или загрузка данных из сети могут быть выполнены в изоляте, чтобы не замедлять основной поток приложения.

Для обмена данными между изолятами в Flutter можно использовать механизмы сообщений. Это позволяет эффективно передавать данные между различными изолятами и основным потоком.

Работа с изолятами требует некоторых знаний о многопоточном программировании и управлении ресурсами, например, необходимо учитывать синхронизацию доступа к данным из разных изолятов, обработку исключений и управление жизненным циклом изолята. Упрощенная схема работы изолятов показана на рис. 5.

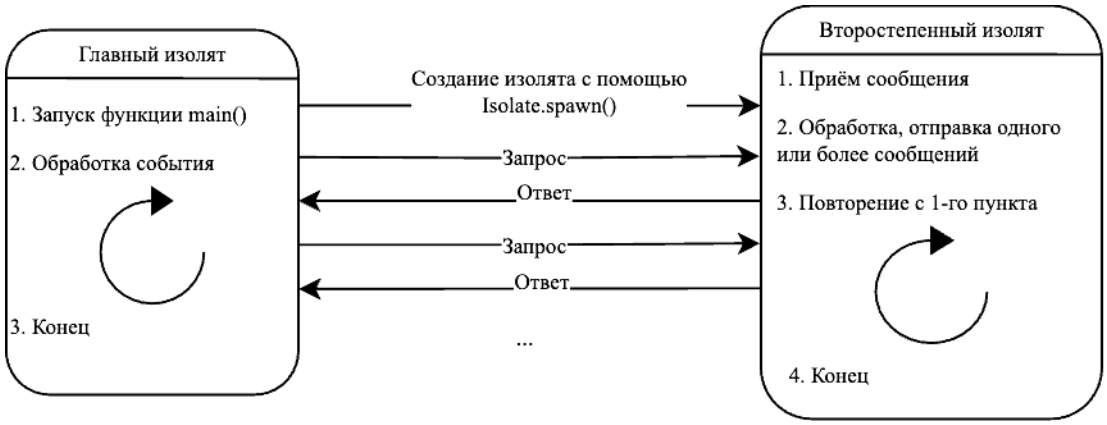

Рис. 5. Схема работы изолятов

Описание приложения «Мобильная ГИС»

В данном разделе рассмотрим, как внедрить изоляты в мобильное приложение. Для примера будет использовано приложение «Мобильная ГИС». Данное приложение, как уже упоминалось ранее, написано с использованием Flutter. Оно создано для помощи специалистам, к примеру геологам. Они могут находиться в экспедиции и не иметь доступа к компьютеру, но им необходимо просмотреть информацию, связанную с нефтяным или газовым месторождением, и т.п.

Приложение принимает данные в виде специального зашифрованного пакета, который ранее был загружен на телефон. В пакете хранится географическая информация о различных участках, лесах, зданиях, дорогах и т.д. Для хранения геометрии об объектах используется формат данных WKB, описание которого приведено выше. Выбор пал на данный тип, потому что он более компактен и удобен для хранения в базе данных SQLite и занимает меньше места в памяти.

У данного приложения имеются проблемы по длительности загрузки данных в него с помощью зашифрованного пакета. Проблемы были выявлены в ходе эксплуатации приложения. Длительная загрузка происходила из-за расшифровки пакета данных. Далее выполняется загрузка расшифрованных данных в оперативную память для последующего отображения этих данных на карту в приложении. Обнаружено также недостаточное быстродействие по скорости построения дорог, которые приходят в пакете.

Применение Isolate в приложении

Для случая с расшифровкой пакета был использован Isolate с методом spawn и портами для сообщения между изолятами. Было создано два порта: первый – для приема-передачи сообщений ошибок; второй порт создан для приема-передачи сообщения с данными, успешно прошедших обработку. Был создан класс DTO (Data Transfer Object), который необходим для передачи данных в другой изолят. Данный класс содержит множество полей, среди них отметим самые важные: поля для портов, о которых выше уже упоминалось; ссылка на файл с данными.

В методе spawn передаются два обязательных параметра: в первом обязательном параметре – статический метод класса или функция, которые выполняются в другом изоляте, а во втором – аргументы передаваемого метода или функции. Метод должен быть обязательно статичным, внутри данного метода присутствует блок обработки исключений try-catch, в котором при успешной работе порт отправляет сообщение главному изоляту данные. При ошибке работы срабатывает блок обработки catch, который отправляет по другому порту ошибку.

В главном изоляте настроены на слушание два уже упомянутых порта. При ошибке срабатывает блок кода, который обрабатывает ошибку и делает соответствующие манипуляции и далее закрывает порты. При успешной работе порт принимает данные и обрабатывает их, и далее с помощью метода close закрывает соединение с портами и методом kill уничтожает второстепенный изолят, чтобы он не расходовал вычислительные ресурсы процессора.

Данное решение позволило ускорить работу по расшифровке пакета данных. Пакет размером 1,5 Мб до применения Isolate расшифровывался 75 с. После применения изолятов время расшифровки уменьшилось до 5 с. Время расшифровки зависит от размера пакета данных. Также стоит отметить, что была решена проблема работы пользовательского интерфейса и расшифровки. Визуальная работа приложения останавливалась из-за того, что основной изолят начинал вычисления по расшифровке данных и тем самым занимал все ресурсы.

Применение метода compute и FutureGroup в приложении

Для случая с загрузкой данных в оперативную память применялась более упрощенная технология изолятов в виде простой функции compute. Также был применен ещё один инструмент под названием FutureGroup.

Функция compute является более простым решением, поскольку не требуется создания портов для обмена данными между изолятами. Она может принимать статический метод или функцию в качестве аргументов, а также их параметры. По завершении работы возвращает асинхронные данные, тип которых определяется возвращаемым значением переданной функции.

В приложении «Мобильная ГИС» после расшифровки файла получается база данных в формате SQLite. Делается проход по всем таблицам и, чтобы получить данные из базы, выполняются запросы к ней, а чтобы ускорить данный процесс, используется упомянутая ранее функция compute, и далее запросы делаются параллельно. Запросы вынесены в отдельный метод, чтобы в последующем передать его в функцию compute. Для передачи параметров в метод с запросами используется тип данных Map. Стоит отметить один из параметров – BackgroundIsolateBinaryMessenger [17]. BackgroundIsolateBinaryMessenger – это механизм, который позволяет обмениваться данными между различными изолированными потоками исполнения в фоновом режиме. BackgroundIsolateBinaryMessenger предоставляет удобный и типизированный способ обмена данными между главным Isolate и фоновыми задачами. Он позволяет регистрировать обработчики сообщений. Вы можете зарегистрировать функции-обработчики, которые будут вызываться при получении сообщений определенного типа от фоновой задачи. BackgroundIsolateBinaryMessenger использует каналы для идентификации различных типов сообщений. Каждый канал ассоциируется с функцией-обработчиком в главном изоляте и функцией-отправителем в фоновом изоляте.

Конкретно в нашем случае BackgroundIsolateBinaryMessenger решает проблему взаимодействия второстепенного изолята обращаться к базе данных. Без него не будет работать метод по отправке запросов к базе данных.

После полученных данных из базы происходит их обработка и, чтобы ускорить данный процесс, используется FutureGroup.

FutureGroup – это пакет в языке программирования Flutter, который обеспечивает возможности для работы с асинхронными операциями [18]. Этот пакет предоставляет класс FutureGroup, который позволяет собирать несколько объектов Future в одну группу и выполнять контроль над их выполнением. Использование FutureGroup предлагает ряд преимуществ по повышению отзывчивости. Обработка результатов по мере поступления может ускорить выполнение кода, не дожидаясь завершения всех операций. Улучшается читаемость кода. FutureGroup позволяет писать более декларативный и лаконичный код по сравнению с ручным управлением множеством фьючерсов. Динамическое добавление фьючерсов и централизованная обработка ошибок создают мощный инструмент для управления асинхронным кодом.

В приложении во FutureGroup добавляется методом add асинхронный метод с аргументами, в котором происходит обработка данных. Добавление методом add должно происходить внутри цикла, в приложении цикл проходит по таблицам базы данных, следовательно, получается, что происходит добавление одного и того же метода несколько раз, но для разных таблиц с их объектами. Цикл не будет ждать выполнения каждого метода, а отдаст результат сразу для всех таблиц либо может отдавать результаты по мере их выполнения. Всё зависит от метода, который будет вызван после конца добавления элементов в FutureGroup. Конкретно в данном случае сделано так, чтобы ожидался результат выполнения всех методов. После цикла указывается, что добавление закончено с помощью метода close и ожидается результат выполнения всех функций.

Для базы данных размером 5 Мб до применения параллельности время загрузки в оперативную память составляло 316 с. После применения упомянутых выше способов распараллеливания время загрузки стало существенно меньше: 15 с. В зависимости от размера базы данных это время может изменяться, но соответствующие изменения по времени загрузки становятся очень заметными.

При построении дорог был применен тот же подход, что и для первого случая. В результате применения изложенного выше подхода время построений маршрутов, существенно сократилось.

Заключение

Описанные алгоритмы параллельной работы, основанные на технологии Isolate, были добавлены в мобильное приложение «Мобильная ГИС». Предложенный подход значительно повысил скорость загрузки данных в приложение. Благодаря этому улучшению пользоваться данным приложением стало на порядок удобнее, а само оно стало более эффективным и более быстрым, что благоприятно сказывается на работе пользователя с приложением.

В заключение подчеркнем, что исследование темы распараллеливания обработки географических данных позволит разработчикам мобильных приложений на Flutter значительно улучшить производительность своих продуктов, улучшится также работа приложения «Мобильная ГИС».

Перспектива дальнейших исследований связана с применением алгоритмов, базирующихся на предложенном подходе, ускоряющих загрузку приложения «Мобильная ГИС» и улучшающих показ данных на карте при различных масштабах и большом количестве объектов на карте.

Литература

1. Геоинформационные системы [Электронный ресурс]. – Режим доступа: https://bibl.nngasu.ru/electronic-resources/uchmetod/geodesy/847228.pdf, свободный (дата обращения: 11.06.2024).

2. Framework Flutter [Электронный ресурс]. – Режим доступа: https://flutter.dev, свободный (дата обращения: 11.06.2024).

3. Геоинформационные системы [Электронный ресурс]. – Режим доступа: https://kpfu.ru/portal/docs/F1685823411/GIS.pdf, свободный (дата обращения: 11.06.2024).

4. Tiled vector data model for the geographical features of symbolized maps PloS one [Электронный ресурс]. – Режим доступа: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0176387, свободный (дата обращения: 02.07.2024).

5. Holroyd F. Raster GIS: Models of raster encoding / F. Holroyd, S.B.M. Bell // Computers & Geosciences. – 1992. – Vol. 18, No. 4. – P. 419–426.

6. Meyer W. Geometry and its applications. – New York: Chapman and Hall/CRC, 2022. – 489 p.

7. Birkhoff G.D. Basic geometry / G.D. Birkhoff, R. Beatley. – New York: American Mathematical Society, 2000. – 295 p.

8. Moffat A. Large-alphabet semi-static entropy coding via asymmetric numeral systems / A. Moffat, M. Petri // ACM Transactions on Information Systems (TOIS). – 2020. – Vol. 38, No. 4. – P. 1–33.

9. Nguyen D.T. Learning-based lossless compression of 3d point cloud geometry / D.T. Nguyen, M. Quach, G. Valenzise, P. Duhamel // ICASSP–2021. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). – 2021. – P. 4220–4224.

10. Rahmatulloh A. GeoJSON Implementation for Demographic and Geographic Data Integration Using RESTful Web Services / A. Rahmatulloh, B.T. Handoko, R.N. Shofa, I. Darmawan // 2022 Seventh International Conference on Informatics and Computing (ICIC). – 2022. – P. 1–6.

11. Fosci P. Soft Querying Features in GeoJSON Documents: The GeoSoft Proposal / P. Fosci, G. Psaila // International Journal of Computational Intelligence Systems. – 2023. – Vol. 16, No. 1. – P. 163–203.

12. Gabriele M. HBIM-GIS integration with an IFC-to-shapefile approach: The palazzo trotti vimercate pilot case study / M. Gabriele, M. Previtali // ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences. – 2021. – Vol. 8. – P. 167–174.

13. Pagou E.S. Shapefile-based multi-agent geosimulation and visualization of building evacuation scenario / E.S. Pagou, V.C. Kamla, I. Tchappi, Y. Mualla, A. Najjar, S. Galland // Procedia Computer Science. – 2023. – Vol. 220. – P. 519–526.

14. Zhuk E. Using GIS technology for visualizing marine environmental data in netCDF format // Ninth International Conference on Remote Sensing and Geoinformation of the Environment (RSCy–2023). – 2023. – Vol. 12786. – P. 675–680.

15. The IMERG multi-satellite precipitation estimates reformatted as 2-byte GeoTIFF files for display in a Geographic Information System (GIS) [Электронный ресурс]. – Режим доступа: https://gpm.nasa.gov/sites/default/files/2023-12/IMERG_GIS-GeoTIFF_ReadMe_6_27_23.pdf, свободный (дата обращения: 02.07.2024).

16. Flutter Isolate [Электронный ресурс]. – Режим доступа: https://docs.flutter.dev/perf/isolates, свободный (дата обращения: 12.06.2024).

17. BackgroundIsolateBinaryMessenger [Электронный ресурс]. – Режим доступа: https://api.flutter.dev/flutter/services/BackgroundIsolateBinaryMessenger-class.html, свободный (дата обращения: 12.06.2024).

18. FutureGroup [Электронный ресурс]. – Режим доступа: https://api.flutter.dev/flutter/async/FutureGroup-class.html, свободный (дата обращения 12.06.2024).

Гомбоин Александр Булатович
Инженер-программист отд. развития геоинформационного программного обеспечения (ОРГПО) Томского научного-исследовательского и проектного института нефти и газа (ТомскНИПИнефть)
Мира пр-т, 72, г. Томск, Россия, 634027
Аспирант каф. автоматизированных систем управления (АСУ) Томского государственного университета систем управления и радиоэлектроники (ТУСУР)
Ленина ул., 40, г. Томск, Россия, 634050
Тел.: +7-991-509-92-93
Эл. почта: a.gomboin@mail.ru

Кориков Анатолий Михайлович
Д-р техн. наук, профессор каф. АСУ ТУСУРа
Ленина пр-т, 40, г. Томск, Россия, 634050
ORCID: 0000-0001-8070-7476
Тел.: +7-913-869-96-37
Эл. почта: anatolii.m.korikov@tusur.ru

Gomboin A.B., Korikov A.M.

Parallel processing of geographical data in a mobile application

This article presents an approach to parallel processing of geographical data in mobile applications using advanced technologies. A technique for the efficient distribution and processing of data on gas and oil fields, etc., using parallel computing on mobile devices, is considered. This method involves the use of parallel streams on a mobile device. The research is carried out in order to improve the performance and speed of processing geographical data, which is critically important for real-time applications. The impact of this approach on response time and data processing quality is also analyzed. The proposed approach demonstrates significant improvements in comparison with traditional methods of processing geographical data, ensuring faster and more efficient operation of mobile applications. This makes this approach an important tool for improving the efficiency and usability of mobile geoinformation applications.

Keywords: parallel programming, Flutter, Dart, Isolate, geodata, GIS.
DOI: 10.21293/1818-0442-2024-27-2-51-56

References

1. Geoinformation systems. Available at: https://bibl.nngasu.ru/electronicresources/uchmetod/geodesy/847228.pdf, free (Accessed: June 11, 2024).

2. Framework Flutter. Available at: https://flutter.dev, free (Accessed: June 11, 2024).

3. Geoinformation systems. Available at: https://kpfu.ru/portal/docs/F1685823411/GIS.pdf, free (Accessed: June 11, 2024).

4. Tiled vector data model for the geographical features of symbolized maps. Available at: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0176387, free (Accessed: June 11, 2024).

5. Holroyd F., Bell S.B.M. Raster GIS: Models of raster encoding. *Computers & Geosciences*, 1992, v. 18, no. 4, pp. 419–426.

6. Meyer W. Geometry and its applications. New-York, Chapman and Hall/CRC, 2022, 489 p.

7. Birkhoff G.D., Beatley R. Basic geometry. New-York, American Mathematical Society, 2000, 295 p.

8. Moffat A., Petri M., Large-alphabet semi-static entropy coding via asymmetric numeral systems. *ACM Transactions on Information Systems (TOIS)*, 2020, vol. 38, no. 4, pp. 1–33.

9. Nguyen D.T., Quach M., Valenzise G., Duhamel P. Learning-based lossless compression of 3d point cloud geometry. *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 4220–4224.

10. Rahmatulloh A., Handoko B.T., Shofa R.N., Darmawan I. GeoJSON Implementation for Demographic and Geographic Data Integration Using RESTful Web Services. *2022 Seventh International Conference on Informatics and Computing (ICIC)*, 2022, pp. 1–6.

11. Fosci P., Psaila G. Soft Querying Features in GeoJSON Documents: The GeoSoft Proposal. *International*

Journal of Computational Intelligence Systems. 2023, vol. 16, no. 1, pp. 163–203.

12. Gabriele M., Previtali M. HBIM-GIS integration with an IFC-to-shapefile approach: The palazzo trotti vimercate pilot case study. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences.* 2021, vol. 8, pp. 167–174.

13. Pagou E.S., Kamla V.C., Tchappi I., Mualla Y., Najjar A., Galland S. Shapefile-based multi-agent geosimulation and visualization of building evacuation scenario. *Procedia Computer Science.* 2023, vol. 220, pp. 519–526.

14. Zhuk E. Using GIS technology for visualizing marine environmental data in netCDF format. *Ninth International Conference on Remote Sensing and Geoinformation of the Environment (RSCy2023)*. 2023, vol. 12786, pp. 675–680.

15. The IMERG multi-satellite precipitation estimates reformatted as 2-byte GeoTIFF files for display in a Geographic Information System (GIS). Available at: https://gpm.nasa.gov/sites/default/files/2023-12/IMERG_GIS-GeoTIFF_ReadMe_6_27_23.pdf, free (Accessed: June 11, 2024).

16. Flutter Isolate. Available at: https://docs.flutter.dev/perf/isolates, free (Accessed: June 12, 2024).

17. BackgroundIsolateBinaryMessenger. Available at: https://api.flutter.dev/flutter/services/BackgroundIsolateBinaryMessenger-class.html, free (Accessed: June 12, 2024).

18. FutureGroup. Available at: https://api.flutter.dev/flutter/async/FutureGroup-class.html, free (Accessed: June 12, 2024).

Aleksander B. Gomboin
Software Engineer, Department of Geoinformation Software Development, Tomsk Scientific Research and Design Institute of Oil and Gas
72, Mira st., Tomsk, Russia, 634027
Postgraduate student, Department of Automated Control Systems, Tomsk State University of Control Systems and Radioelectronics (TUSUR)
40, Lenin pr., Tomsk, Russia, 634050
Phone: +7-991-509-92-93
Email: a.gomboin@mail.ru

Anatoly M. Korikov
Doctor of Science in Engineering, Professor,
Department of Automated Control Systems, TUSUR
40, Lenin pr., Tomsk, Russia, 634050
ORCID: 0000-0001-8070-7476
Phone: +7-913-869-96-37
Email: anatolii.m.korikov@tusur.ru