DOI 10.37882/2223-2966.2023.6-2.20

ПОСТРОЕНИЕ АРХИТЕКТУРЫ ПРОГРАММНОЙ СИСТЕМЫ ДЛЯ ГЕОИНФОРМАЦИОННОГО ПРИЛОЖЕНИЯ ДОПОЛНЕННОЙ РЕАЛЬНОСТИ

BUILDING A SOFTWARE SYSTEM ARCHITECTURE FOR A GEOINFORMATION AUGMENTED REALITY APPLICATION

S. Kitanin
A. Makarevich

Summary: The article deals with the design of software system architecture for a geoinformation augmented reality application. It gives a theoretical introduction to the C4 model for describing the architecture, namely for building software architecture diagrams. It also provides a theoretical introduction to describing the architecture of applications in a highly specialized field of software, namely for building software based on Unity development environment. The diagrams of C4 model, using the Domain specific language (DSL) specially developed for the construction of C4 diagrams, and the diagram of the component architecture of the application are constructed. To visualize the theoretical scenarios of the software solution BMPN diagrams are built.

Keywords: C4 model, unity application architecture, augmented reality, geographic information system.

Китанин Сергей Сергеевич
магистрант,
Российский Технологический Университет МИРЭА
kitanin.ser@mail.ru
Макаревич Артём Денисович
Аспирант,
Российский Технологический Университет МИРЭА
kitanin.ser@mail.ru

Аннотация. В статье рассматривается проектирование архитектуры программной системы для геоинформационного приложения дополненной реальности. Приводится теоретическое введение модели С4 для описания архитектуры, а именно для построения диаграмм архитектуры программного обеспечения. Также приводится теоретическое введение для описания архитектуры приложения применительно к узкоспециализированной области программного обеспечения, а именно для построения ПО на базе среды разработки Unity. Построены диаграммы модели С4, с использованием специально разработанного для построения С4 диаграмм языка Domain specific language (DSL), а также диаграмма компонентной архитектуры приложения. Для визуализации теоретических сценариев работы программного решения построены BMPN диаграмма.

Ключевые слова: модель С4, unity архитектура приложения, дополненная реальность, геоинформационная система.

Использование различных гаджетов и новых технологий расширяет область применения и влияние навигационных технологий в жизни человека. Дополненная реальность помогает человеку погрузиться в виртуальный мир, взаимодействовать с ним, а также помогать человеку в реальном мире за счет использования виртуальных объектов. Для реализации программных систем необходимо применять архитектурное проектирование для формирования общего обзора системы и планирования разработки.

Зачастую представление архитектуры программной системы с помощью диаграмм имеет непоследовательные обозначения (цветовое кодирование, формы, стили линий и т.д.), двусмысленные названия, не обозначенные отношения, специфичная терминология, отсутствующие варианты технологий, смешанные абстракции и т.д. В ИТ отрасли есть унифицированный язык моделирования (UML), ArchiMate и SysML, но многие команды уже отбросили их в пользу гораздо более простых диаграмм «коробки и линии». В гонке за гибкостью из-за отказа от языков моделирования многие команды разработчиков программного обеспечения утратили способность к визуальному общению. На рисунке 1 приведен пример диаграммы архитектуры.

Для решения вышеописанной проблемы была создана Модель С4. Модель С4 — это простой в освоении и удобный для разработчиков подход к построению диаграмм архитектуры программного обеспечения. Хорошие диаграммы архитектуры программного обеспечения помогают в общении внутри/вне команд по разработке программного обеспечения/продукта, эффективном принятии новых сотрудников, обзорах/оценках архитектуры, выявлении рисков (например, риск-штурм), моделировании угроз и т.д.

Характеристика модели С4:
- набор иерархических абстракций (программные системы, контейнеры, компоненты и код);
- набор иерархических диаграмм (контекст системы, контейнеры, компоненты и код);
- независимость от нотации.

Модель С4 была создана как способ помочь командам разработчиков программного обеспечения описать и передать архитектуру программного обеспечения, как во время предварительных сессий проектирования, так и при ретроспективном документировании существующей кодовой базы. Это способ создания карт вашего кода с различными уровнями детализации, подобно тому, как используется что-то вроде Google Maps для

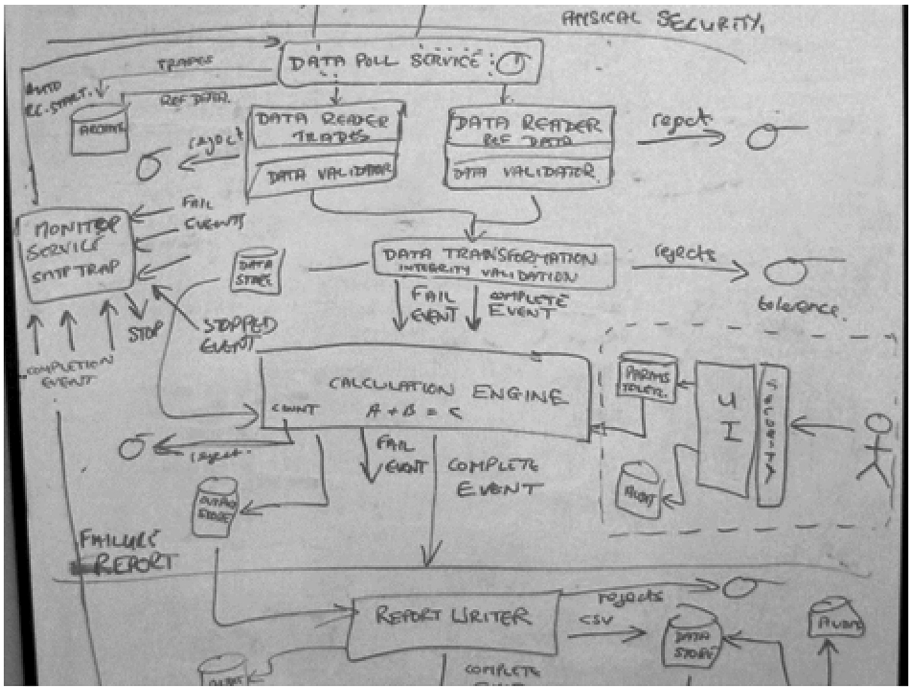

Рис. 1. Пример диаграммы архитектуры

увеличения или уменьшения масштаба интересующей области [1].

Уровень 1: Контекстная диаграмма системы, приведенная на рисунке 2 обеспечивает отправную точку, показывая, как рассматриваемая программная система вписывается в окружающий мир.

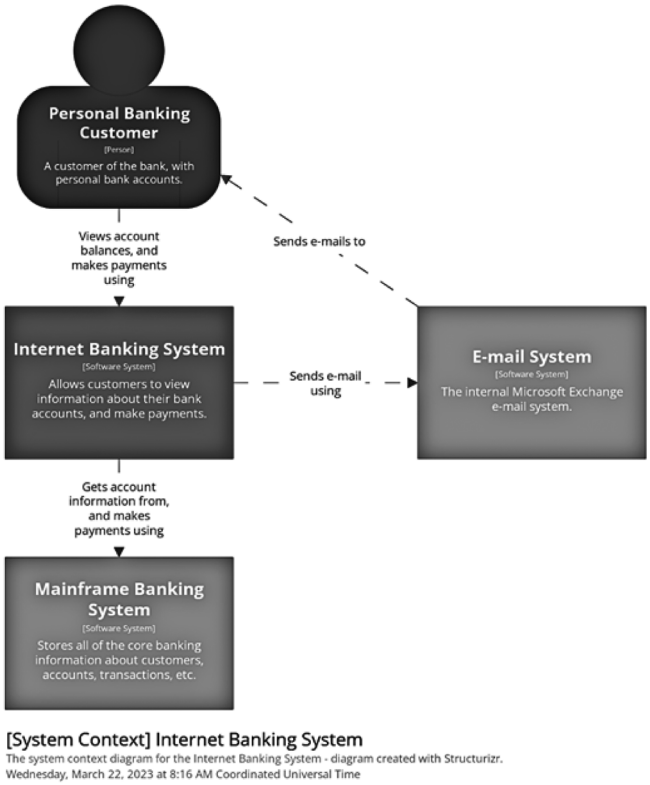

Рис. 2. Уровень 1: Контекстная диаграмма системы

Уровень 2: Диаграмма контейнера, приведенная на рисунке 3 увеличивает масштаб программной системы, показывая высокоуровневые технические строительные блоки.

Уровень 3: Диаграмма компонентов, приведенная на рисунке 4 увеличивает масштаб отдельного контейнера, показывая компоненты внутри него.

Уровень 4: Диаграмма кода, приведенная на рисунке 5 (например, класса UML) может быть использована для увеличения масштаба отдельного компонента, показывая, как этот компонент реализован.

На основе вышеописанной модели была составлена архитектура системы с использованием специального языка для описания диаграмм Domain specific language (DSL) [2]. Описание диаграммы представлено на рисунке 6. Диаграмма контекста представлена на рисунке 7, диаграмма контейнеров на рисунке 8.

NodeJs сервер общается с Google Directions API посредством протокола HTTP для получения маршрута до определенной геопозиции.

Directions (Направления) API — это веб-служба, которая использует HTTP-запрос для возврата маршрутов в формате JSON или XML между местоположениями. Направления доступны в нескольких формах:

— как отдельный API;
— как часть клиентского API JavaScript Карт;
— для использования на стороне сервера как часть клиентских библиотек для веб-служб Google Maps [3].

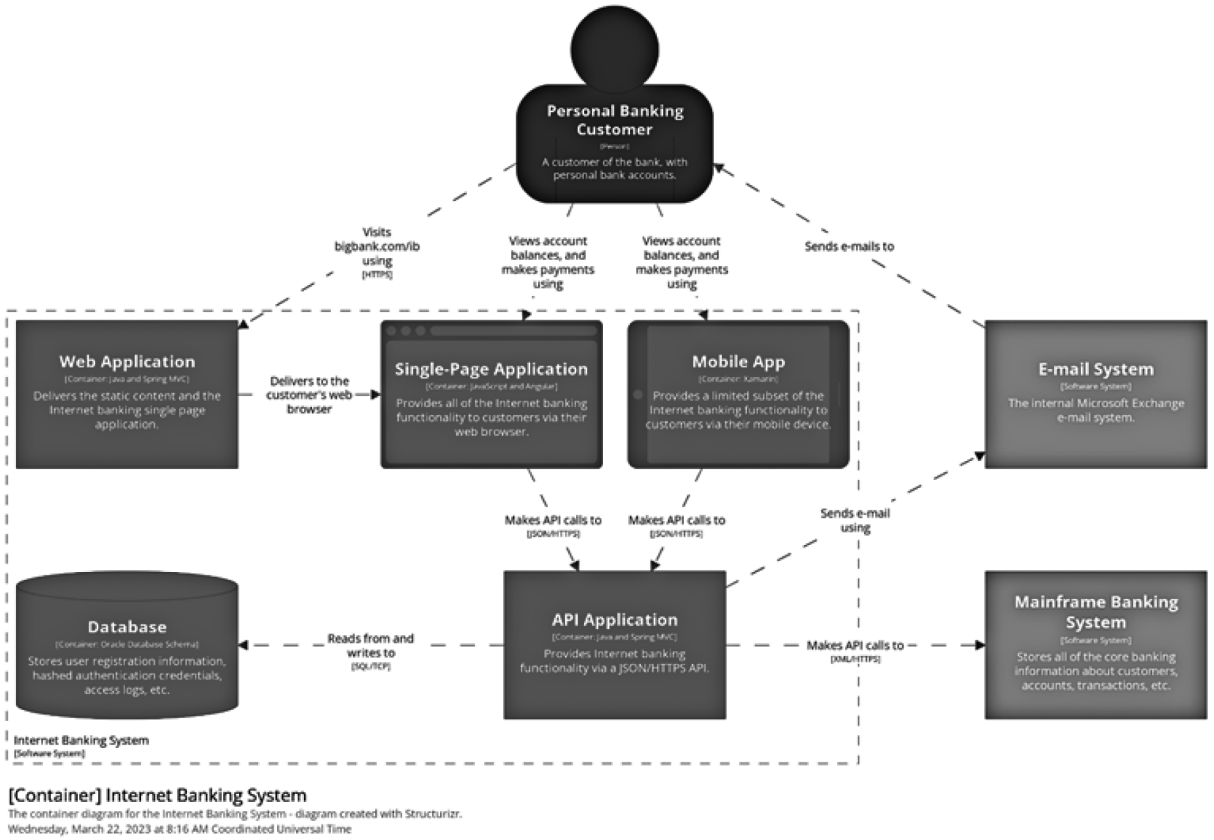

Рис. 3. Уровень 2: Диаграмма контейнера

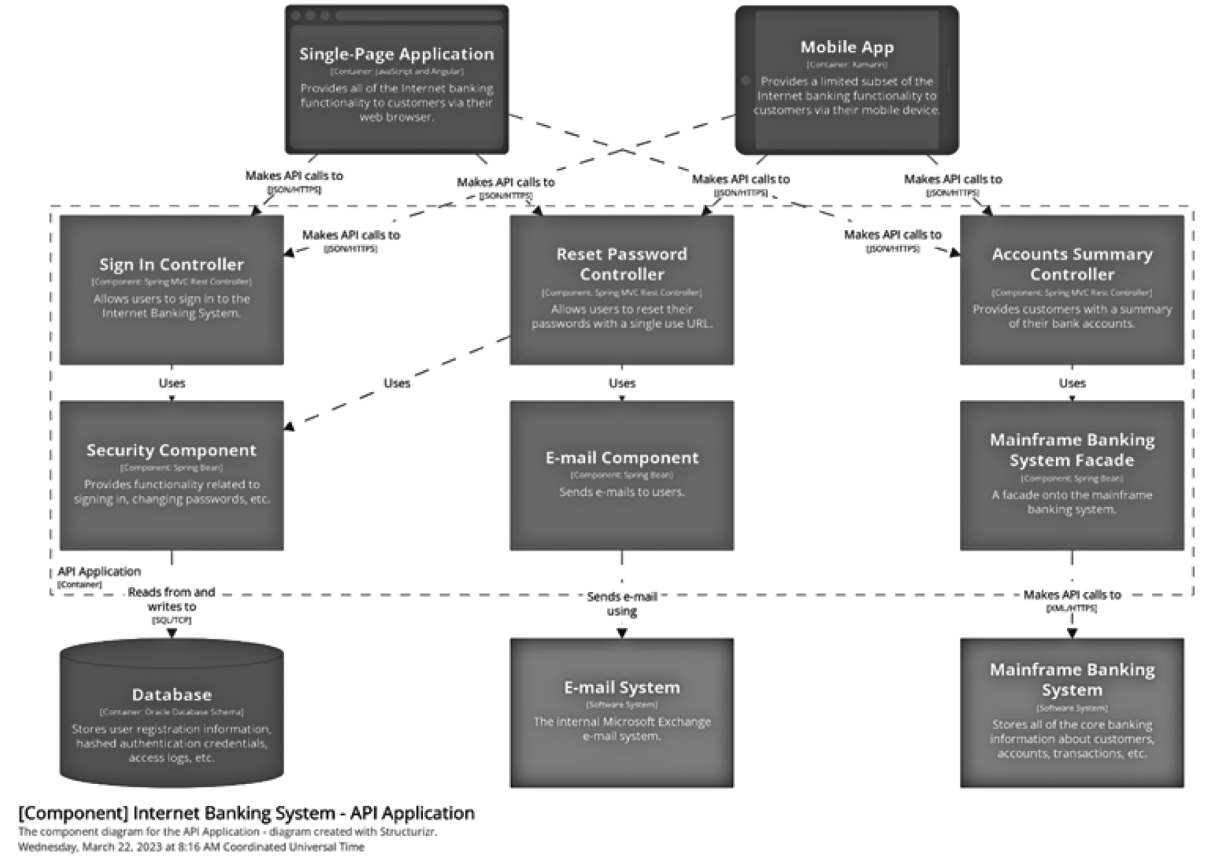

Рис. 4. Уровень 3: Диаграмма компонентов

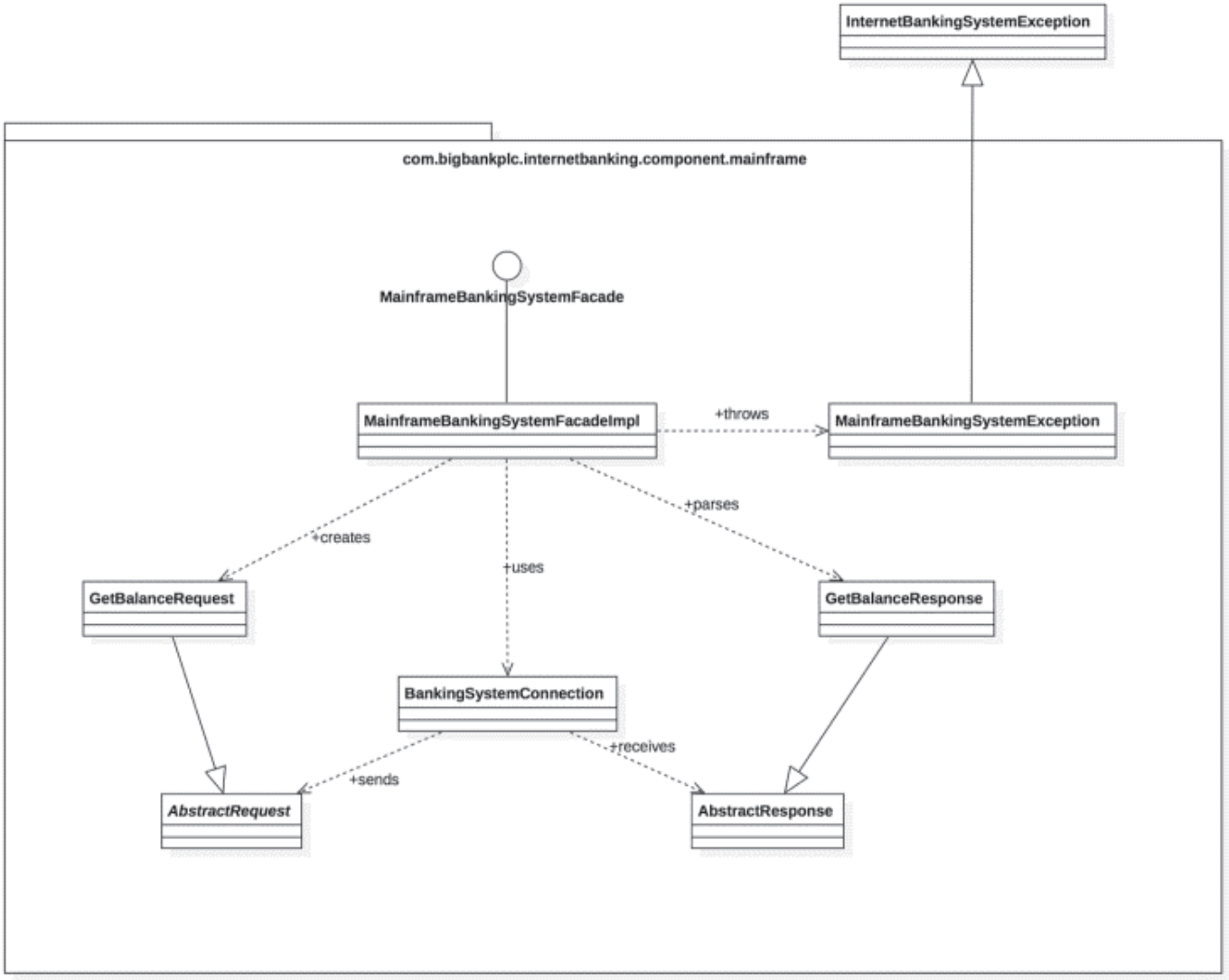

Рис. 5. Уровень 4: Диаграмма кода

```
workspace {

    model {
        user = person User {
            tags fontSize35
        }
        admin = person Admin {
            tags fontSize35
        }
        softwareSystem = softwareSystem Software System {
            tags fontSize35
            description "Provide AR GeoObjects"
            webapp = container Web Application {
                admin -> this "Uses"
                technology "Vue, Js"
                tags Vue, fontSize35
            }
            mobileApp = container Mobile Application {
                user -> this "Uses"
                technology Unity
                tags Unity, fontSize35
            }
            server = container Server {
                tags NodeJs, fontSize35
                technology "NodeJs, ExpressJs, Js"
                webapp -> this "Uses HTTPS"
                mobileApp -> this "Uses HTTPS, Websockets"
            }
            container Database {
                tags MongoDb, fontSize35
                technology MongoDb
                server -> this "Reads from and writes to MongoDB"
            }
        }
        directionsApi = softwareSystem DirectionsApi {
            tags "Google Cloud Platform - Cloud APIs" DirectionsApi, fontSize35
            description "Api providing an array of waypoints"

        }
        server -> directionsApi "Get array of user waypoints HTTPS"
    }

    views {
        styles {
            element Vue {
                icon https://upload.wikimedia.org/wikipedia/commons/f/f1/Vue.png
            }
            element NodeJs {
                icon https://upload.wikimedia.org/wikipedia/commons/thumb/d/d9/Node.js_logo.svg/1200px-Node.js_logo.svg.png
            }
            element MongoDb {
                icon https://upload.wikimedia.org/wikipedia/commons/thumb/9/93/MongoDB_Logo.svg/220px-MongoDB_Logo.svg.png
            }
            element Unity {
                icon https://upload.wikimedia.org/wikipedia/commons/thumb/c/c4/Unity_2021.svg/218px-Unity_2021.svg.png
            }
            element DirectionsApi {
                background #cccccc
            }
            element fontSize35 {
                fontSize 35
            }
        }
        systemContext softwareSystem {
            include *
            autolayout lr
        }

        container softwareSystem {
            include *
            autolayout lr
        }

        themes default https://static.structurizr.com/themes/google-cloud-platform-v1.5/theme.json
    }

}
```

Рис. 6. Описание диаграммы

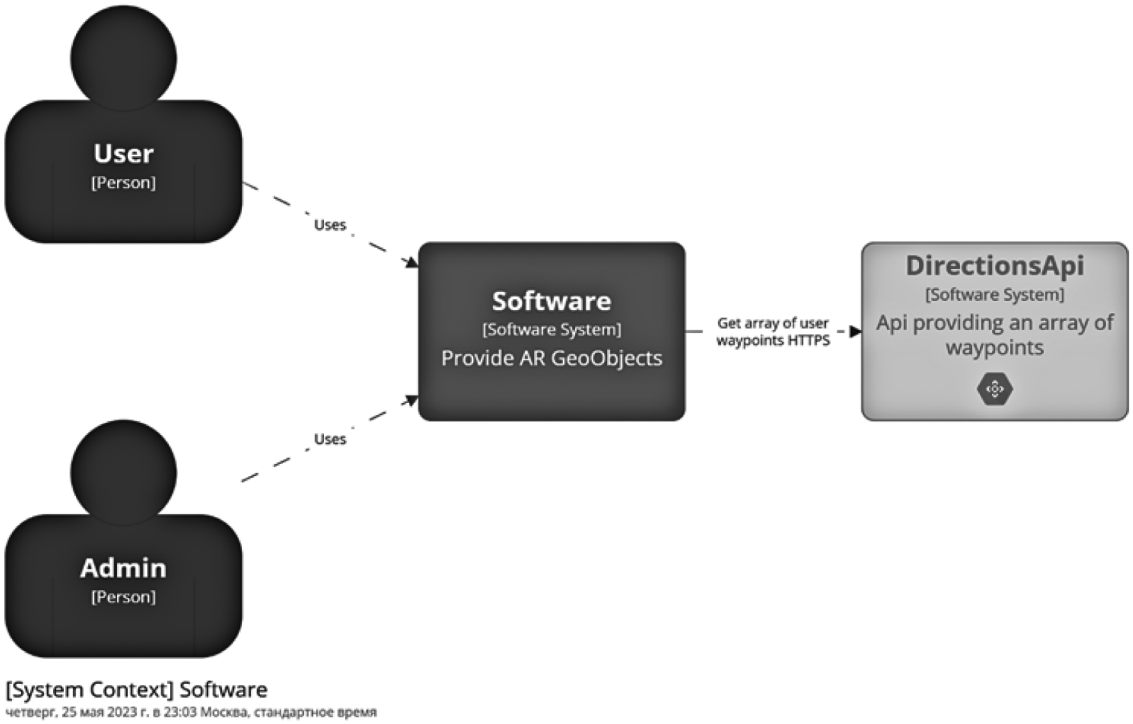

Рис. 7. Диаграмма контекста

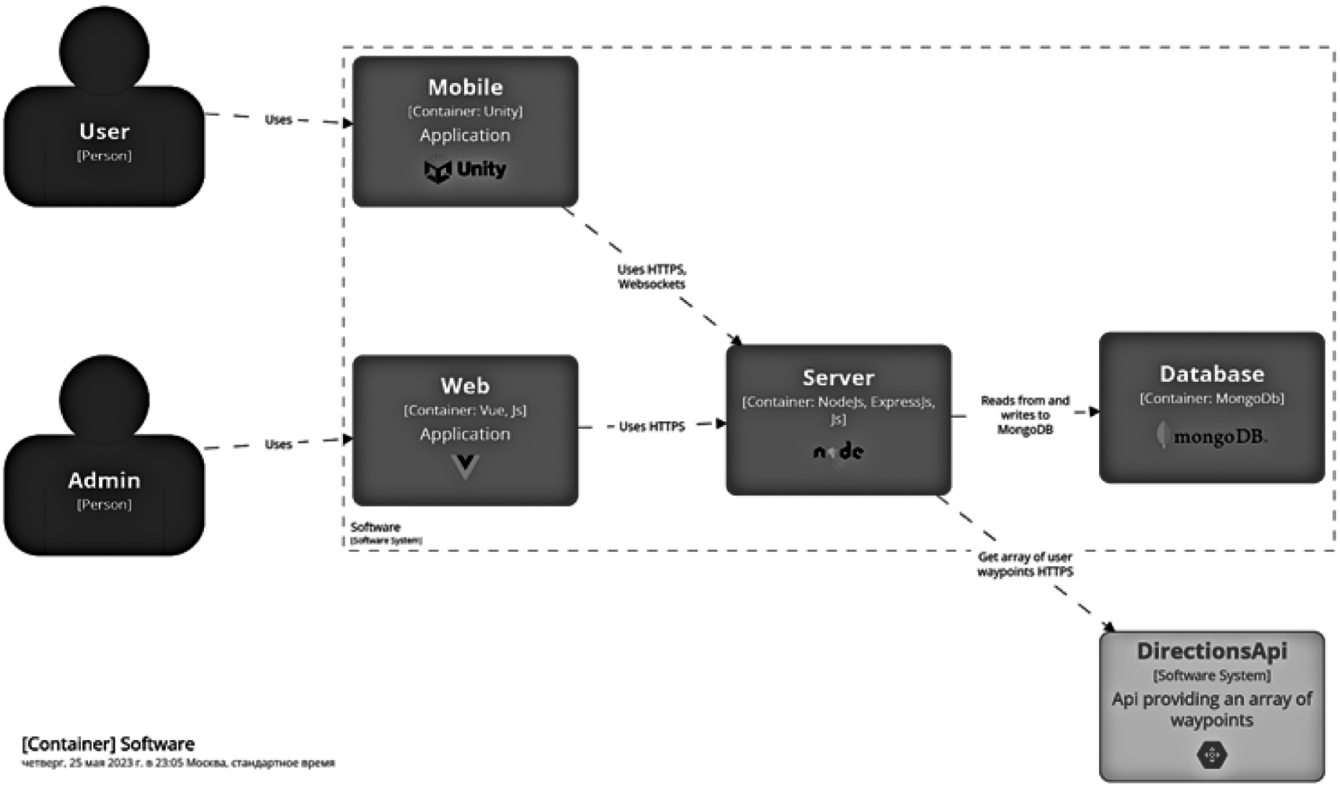

Рис. 8. Диаграмма контейнеров

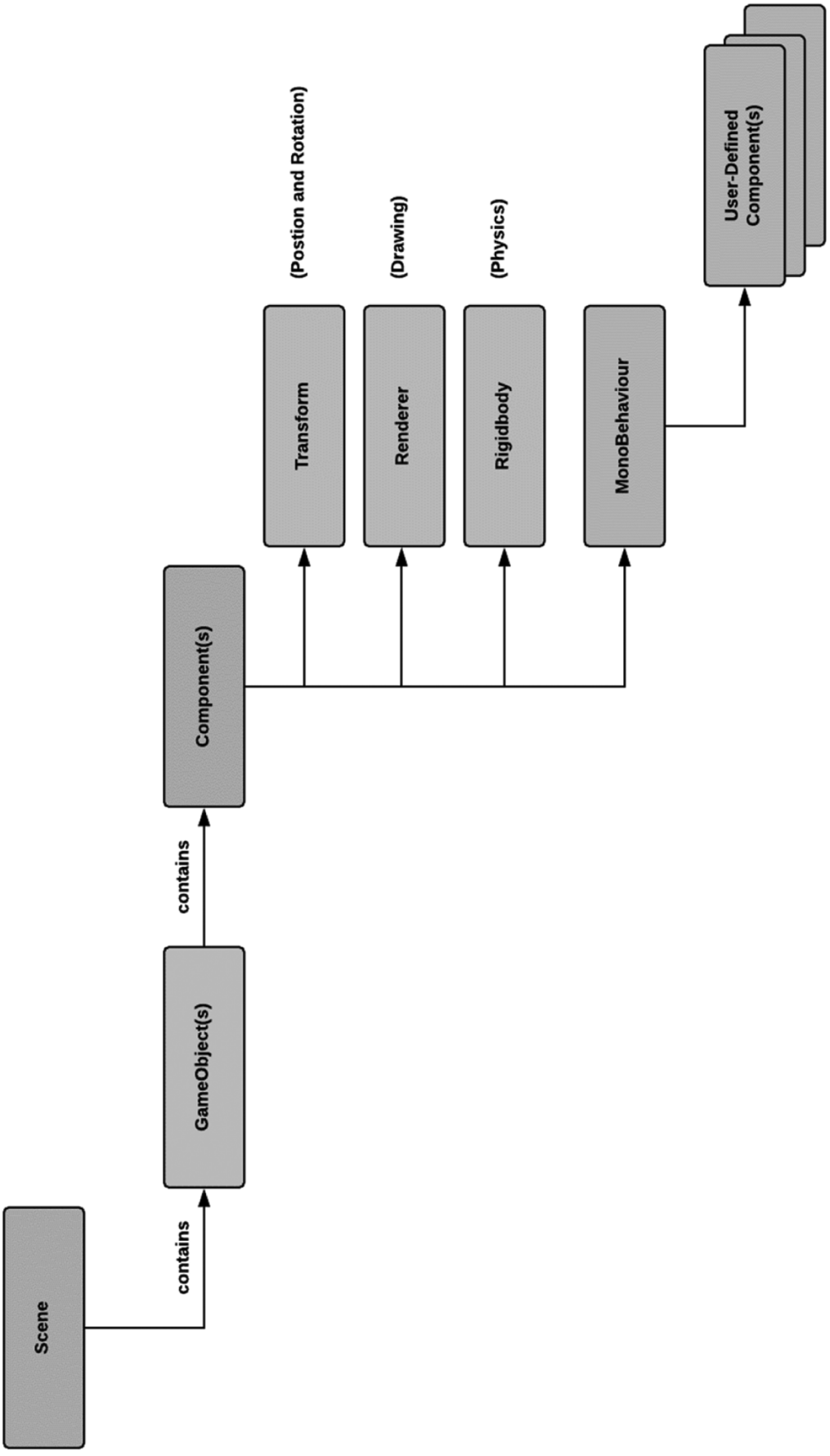

Рис. 9. Общая архитектура unity приложения

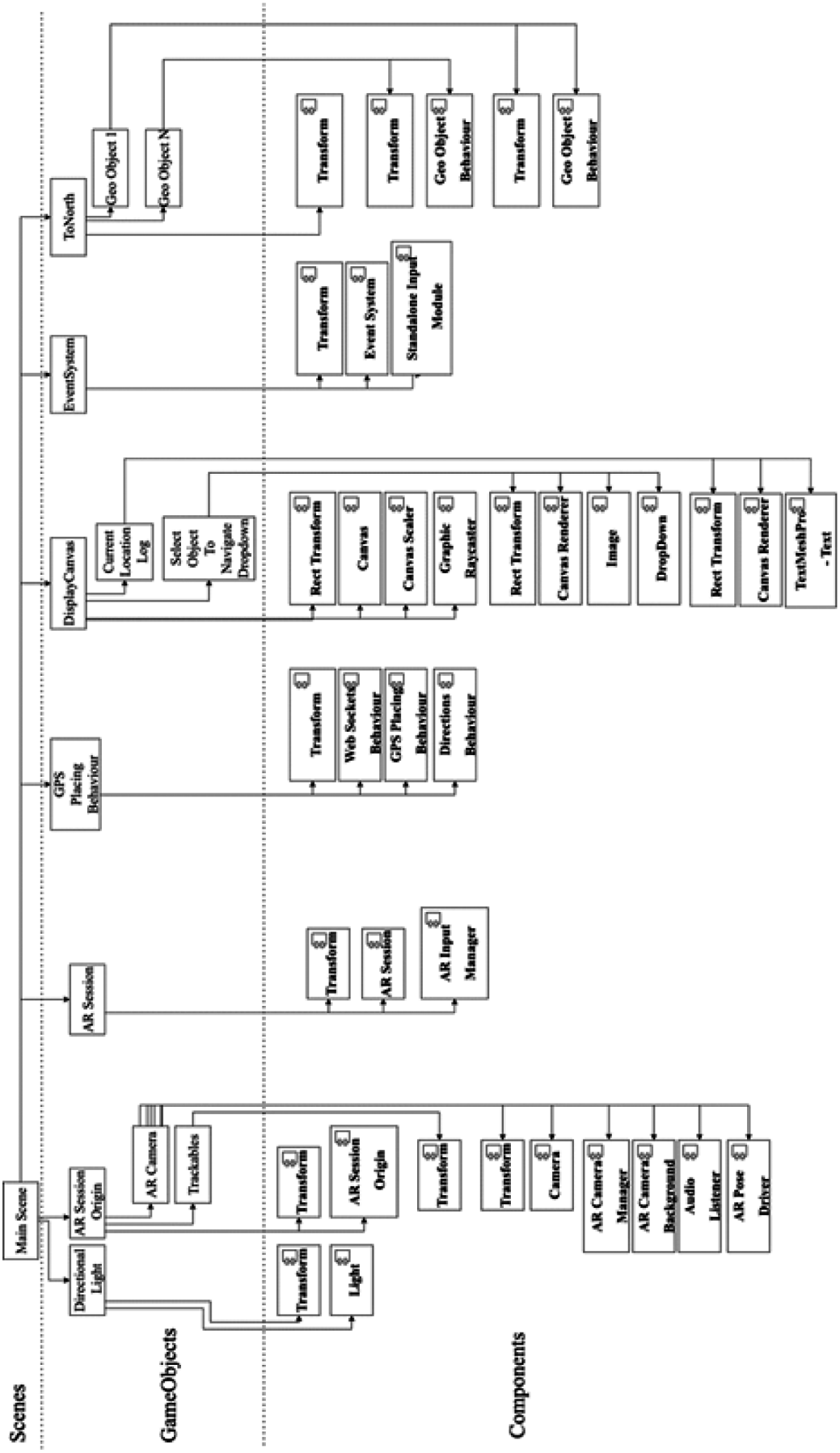

Рис. 10. Архитектура программного продукта

В текущей реализации мобильное приложение отправляет запрос на сервер, он в свою очередь на Directions API отправляется запрос с координатами начала и конца маршрута, в ответ API содержатся точки с координатами широты и долготы, которые сервер передает в мобильное приложение и в мобильном приложении отрисовывается путь по переданным точкам.

Для построения диаграммы компонентов и кода для описания контейнера мобильного приложения необходимо подстраиваться под специфику конкретной части системы и принятые стандарты при разработке специфичного ПО. Двумя основными столпами архитектуры ядра движка Unity являются следующие:

— components;
— scripting API.

Unity — это движок, управляемый компонентами, и именно с помощью комбинации компонентов мы создаем нашу игру. Общая архитектура unity приложения приведена на рисунке 9. Если проанализировать архитектуру, то становится очевидно, что существует иерархия высокого уровня, в которой сущности содержат другие сущности. Существенными элементами этой структуры являются Компоненты, они являются строительными блоками игры.

Простой способ визуализировать эту архитектуру — это рассмотреть, что сцена представляет собой набор GameObjects, а GameObjects — это набор компонентов, которые могут реализовывать и включать в себя следующее:

— камеры и физика (systems);
— конфигурации, анимации, и текстуры (data);
— игровые механики и сценарии событий (behaviors) [14].

Учитывая специфику построения архитектуры unity приложения, была спроектирована архитектура программного продукта, представленная на рисунке 10.

Основным объектом для взаимодействия с геообъектами является объект GPS Placing Behaviour. Он включает в себя компоненты:

— web sockets behavior отвечает за соединение с сервером по веб сокетам и является сервисом для получения информации об объектах, доступных в зонах видимости этих объектов;
— gps placing behavior отвечает за создание, удаление, перемещение экземпляров геообъектов в сцене;
— directions behavior отвечает за создание, удаление, изменение экземпляра пути следования до геообъекта в сцене.

Все экземпляры геообъектов создаются и присваиваются объекту ToNorth для подстройки расположения геообъктов из геоокоординат к координатам unity и правильного расположения объектов соответственно их углу на север.

Объекты Geo Object 1 и Geo Object N представляют собой геообъекты разных типов: текстового, аудио, 3D. Написание имени объекта с буквой N подразумевает, что геообъектов может быть неограниченное количество. Диаграмма классов геообъектов представлена на рисунке 11.

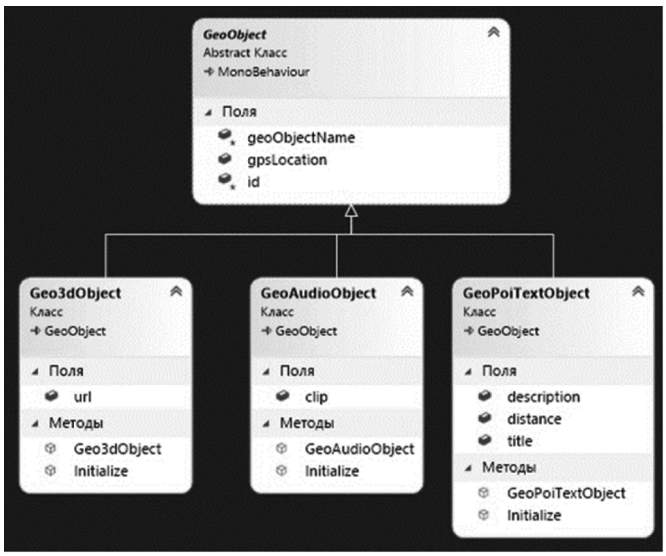

Рис. 11. Диаграмма классов геообъектов

Для разработки алгоритма функционирования программного продукта необходимо учесть, что у Unity приложения существует собственный цикл жизни [5], а также принцип определения координат пользователя, и необходимость перевода геоокординат в координаты Unity.

Сценарии функционирования программного продукта изображены на рисунке 12. Концептуальное описание сценариев:

— пользователь, не прибегая к прокладыванию маршрута до геообъекта, получает визуальную информацию о направлении и текстовую информацию об удаленности геообъекта, в случае текстового геообъекта, звуковую информацию в зонах видимости геообъектов, в которых находится пользователь, в случае звукового объекта;
— пользователь, прибегая к прокладыванию маршрута до геообъекта, получает визуальную информацию о направлении и текстовую информацию об удаленности геообъекта, в случае текстового геообъекта, звуковую информацию в зонах видимости геообъектов, в которых находится пользователь, в случае звукового объекта, а также 3D маршрут до выбранного геообъекта.

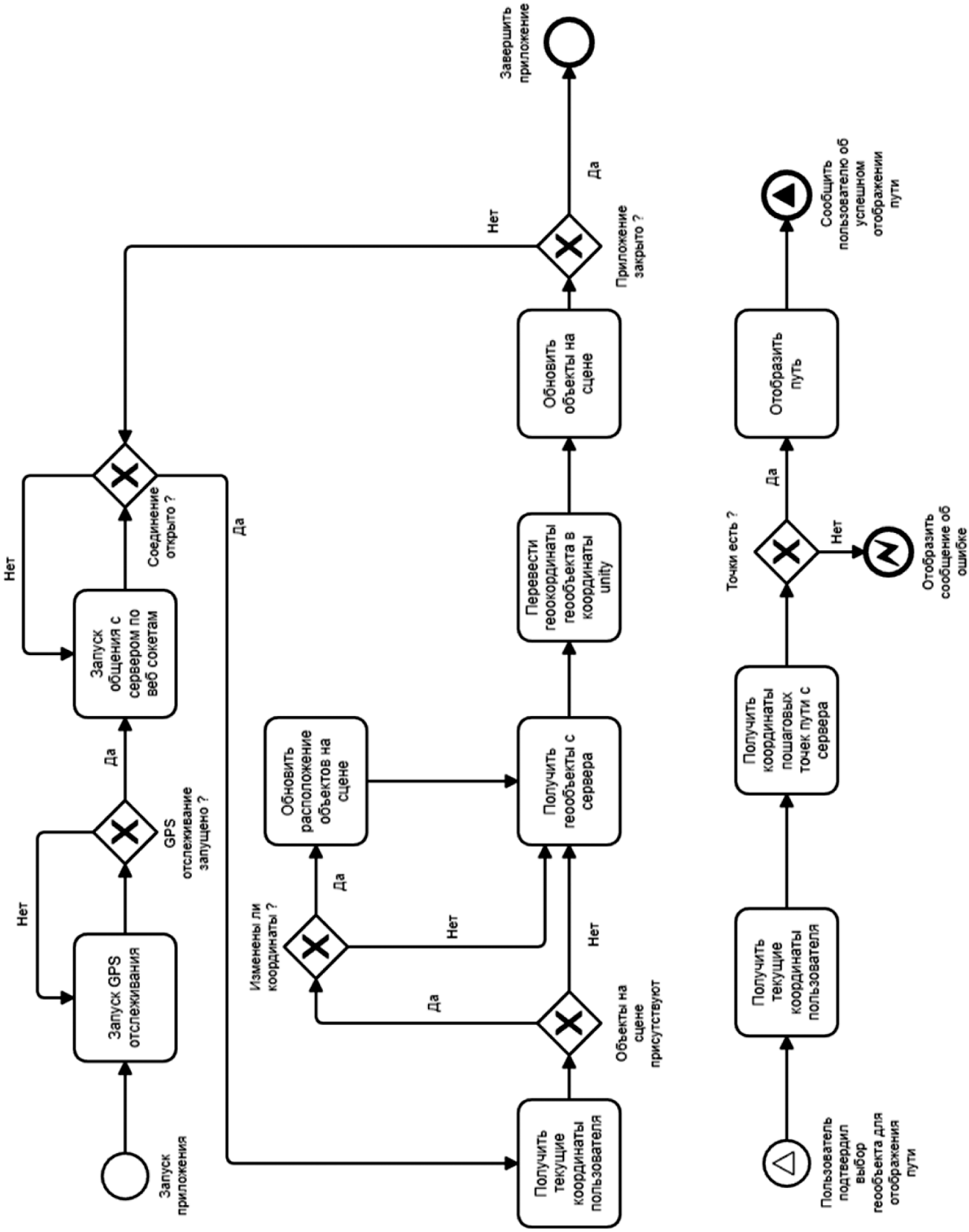

Рис. 12. Алгоритмы функционирования программного продукта

Вывод: рассмотрев подход к проектированию архитектуры «Модель С4» и применив данный подход к построению архитектуры программной системы для геоинформационного приложения дополненной реальности можно с уверенностью утверждать о его применимости для построения похожих систем и о применимости подхода в целом. В части построения архитектуры компонентов и кода присутствует необходимость адаптации под специфику реализации систем с использованием среды разработки Unity. Исходя из полученного результата проектирования можно с уверенностью сказать о том, что данная система может быть реализована на практике, в том числе с использованием уже существующих компонентов программного обеспечения и их аналогов с открытым исходным кодом.

ЛИТЕРАТУРА

1. Модель C4 [Электронный ресурс]. URL: https://c4model.com/ (дата обращения: 09.05.2023).
2. Structurizr DSL [Электронный ресурс]. URL: https://github.com/structurizr/dsl (дата обращения: 09.05.2023).
3. Directions API overview [Электронный ресурс]. URL: https://developers.google.com/maps/documentation/directions/overview (дата обращения: 09.05.2023).
4. Unity's architecture [Электронный ресурс]. URL: https://subscription.packtpub.com/book/game_development/9781789349337/1/ch01lvl1sec11/unity-s-architecture (дата обращения: 09.05.2023).
5. Order of execution for event functions [Электронный ресурс]. URL: https://docs.unity3d.com/Manual/ExecutionOrder.html (дата обращения: 09.05.2023).

Журнал «Современная наука: актуальные проблемы теории и практики»