сование или добавление текста. Entity Framework Core упрощает работу с базой данных через ORM, а MongoDb хранит неструктурированные данные, такие как JSON-объекты. Redis ускоряет доступ к часто используемым данным через кэширование, Swagger (OpenAPI) применяется для документирования и тестирования API.

Микросервисная архитектура back-end позволяет разбить систему на независимые сервисы, каждый из которых отвечает за конкретную бизнес-логику. Это обеспечивает масштабируемость, гибкость и устойчивость к сбоям. Например, сервис аутентификации управляет пользователями и их правами, SignalR синхронизирует действия в реальном времени, MongoDb хранит данные о компонентах доски, а Redis кэширует часто используемую информацию. Такая архитектура упрощает тестирование, развертывание и оптимизацию ресурсов, позволяя эффективно распределять нагрузку.

Использование микросервисной архитектуры делает back-end приложения более адаптивными к изменениям и нагрузкам, что особенно важно для такого динамичного продукта, как интерактивная доска.

А. В. Кирчук, В. С. Закревская
(*ГГУ имени Ф. Скорины, Гомель*)

ПРЕИМУЩЕСТВА ИСПОЛЬЗОВАНИЯ GODOT ENGINE ПРИ СОЗДАНИИ 2D-ИГР

При поиске оптимальных технологий для реализации проекта по созданию 2D игры был проведен небольшой анализ современных игровых движков с целью выявления как преимуществ, так и недостатков каждого.

Для проверки по заданным параметрам выбраны наиболее популярные на сегодняшний день игровые движки, а именно: Unreal Engine, Unity, Godot Engine, Cocos2d и GameMaker Studio.

В результате проведенного исследования был выбран Godot Engine, так как он обладает многими преимуществами, основные из которых приведем ниже:

1) встроенная поддержка 2D: в Godot Engine 2D игры реализованы как отдельная система, а не поверх 3D структуры, как в Unity или Unreal Engine [1]. Это означает, что 2D объекты работают быстрее, проще настраиваются и занимают меньше ресурсов;

2) простота и скорость разработки: редактор Godot Engine интуитивно понятен; система узлов позволяет легко создавать сложные сцены и их взаимодействие; имеет встроенный скриптовый язык GDScript, похожий на Python, который прост в освоении, а также есть поддержка C# и VisualScript [2];

3) лёгкость и производительность: очень легкий и быстро запускается, даже на слабых машинах; проекты занимают меньше места, чем на других игровых движках; готовые билды проектов обладают низким потреблением ресурсов;

4) бесплатность и лицензия: полностью бесплатен и имеет лицензию MIT, что дает полный контроль над проектом. Не нужно платить за использование движка, вне зависимости от доходов;

5) кроссплатформенность: поддерживает экспорт на множество платформ: Windows, macOS, Linux, Andriod, iOS, HTML5 и даже консоли. Процесс экспорта легкий с минимальной настройкой [3];

6) сообщество и документация: быстрорастущее сообщество, множество бесплатных уроков, активные форумы и каналы в Discord. Документация хорошо структурирована.

В заключение можно сказать, основываясь на перечисленных преимуществах и большом количестве доступных ресурсов для изучения и использования этого языка, что Godot Engine, с моей точки зрения, является лучшим выбором для разработки 2D игр.

Литература

1 Иванов, П. Godot Engine: от новичка до профессионала / П. Иванов. – СПб. : Питер, 2024.

2 Кузнецов, В. Разработка 2D игр на Godot: практическое руководство / В. Кузнецов. – Екатеринбург: Урал, 2023.

3 Петров, С. Современные движки для разработки игр: Godot и его возможности / С. Петров // Журнал «Игровая разработка». – 2024. – № 3. – С. 45–50.