МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»

Кафедра информатики и программирования

РАЗРАБОТКА КРОССПЛАТФОРМЕННОЙ ИГРЫ НА UNITY С ЭЛЕМЕНТАМИ МУЛЬТИПЛЕЕРА НА .NET

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 441 группы

направления 02.03.03 Математическое обеспечение и администрирование информационных сетей

факультета компьютерных наук и информационных технологий

Новоярчикова Михаила Андреевича

Научный руководитель

старший преподаватель _____ А. А. Казачкова

Заведующий кафедрой

к.ф.-м.н., доцент _____ М. В. Огнева

Саратов 2025

ВВЕДЕНИЕ

Актуальность темы.

В современном мире игровая индустрия занимает особое место и является одной из самых перспективных и быстроразвивающихся отраслей [1]. Одним из наиболее интересных жанров в мире видеоигр являются стратегические игры. Они предлагают игрокам возможность принимать сложные решения, а также разрабатывать уникальные стратегии. Unity – один из самых популярных и мощных игровых движков [2], который позволяет разработчикам создавать игры для различных платформ, включая ПК, консоли и мобильные устройства.

Цель бакалаврской работы - является создание стратегической кроссплатформенной игры с элементами мультиплеера при помощи Unity и .Net.

Поставленная цель определила **следующие задачи**:

1. Провести обзор мобильных игр в жанре стратегий.
2. Изучить архитектуру компьютерных сетей, клиент-серверную архитектуру приложений и необходимые алгоритмы.
3. Изучить платформы Unity и .Net.
4. Спроектировать и создать расширяемую архитектуру игрового приложения.
5. Создать клиентскую часть с помощью Unity и адаптировать под различные устройства.
6. Разработать серверную часть на .Net.
7. Провести интеграцию клиентской части с сервером.
8. Автоматизировать развертывание серверной части.

Методологические основы.

Архитектура компьютерных сетей, разработка клиент-серверной архитектуры, разработка чистой и расширяемой архитектуры, разработка микросервисной и отказоустойчивой архитектуры представлена в работах:

Сталлингс У., Таненбаум Э., Куросэ Д., Питерсон Л., Рихтер С., Мартин Р., Ньюман С., Ричардсон К., Клеппман М., Меджорс Ч.

Практическая значимость работы.

Практическая значимость работы проявляется в создании полноценного продукта — стратегической игры, доступной на мобильных и десктопных устройствах, в которую люди будут с удовольствием играть и развивать своё стратегическое мышление. Реализованные механики АТВ-шкалы, генерации сеток и поиска пути обеспечивают глубину тактических решений и увлекательность процесса. Мультиплеер и надёжный бэкенд на .NET гарантируют стабильность соединения и масштабируемость проекта. Полученный результат готов к массовому запуску и дальнейшему совершенствованию на рынке.

Структура и объём работы.

Бакалаврская работа состоит из введения, 6 разделов, заключения, списка источников и 24 приложений. Общий объём работы – 195 страниц, из них 130 страниц – основное содержание, включая 40 рисунков, цифровой носитель в качестве приложения, список источников информации – 21 наименование.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Первый раздел «Обзор мобильных игр в жанре стратегий» посвящён анализу аналогичных мобильных игр, которые могут составить конкуренцию разрабатываемому проекту. Рассмотрены такие игры, как Heroes of Might and Magic III HD Edition, являющаяся платной и с ограниченным функционалом; «Герои войны и денег», где сильная тактическая составляющая сочетается с проблемами баланса и высоким порогом вхождения; Braveland, предлагающая PvP-сражения, однако имеющая слабую тактическую глубину и влияние внутриигровых покупок на баланс. Указанные аналоги помогут выделить сильные стороны и избежать слабых мест при разработке новой игры.

Второй раздел «Архитектура компьютерных сетей» посвящён рассмотрению базовых принципов и структур, на основе которых строятся современные компьютерные сети, обеспечивающие эффективное взаимодействие устройств. Рассматриваются такие важные концепции, как модель OSI, разделяющая сетевые функции на семь уровней для упрощения взаимодействия и совместимости устройств; стек протоколов TCP/IP, используемый в основе интернета и большинства современных сетей; прикладной протокол HTTP, обеспечивающий передачу веб-данных между клиентами и серверами; а также протоколы SSL/TLS, которые отвечают за безопасность передачи информации через шифрование, аутентификацию и защиту целостности данных. Изученные аспекты сетевой архитектуры позволяют эффективно спроектировать и реализовать сетевые решения, обеспечивающие надёжную и безопасную коммуникацию.

Третий раздел «Игровой движок Unity» посвящён рассмотрению ключевых возможностей, компонентов и подходов, используемых при разработке игр на платформе Unity. Приведён анализ основных преимуществ Unity, таких как низкий порог входа, мощная поддержка кроссплатформенной разработки и большой набор готовых ресурсов (asset-ов и плагинов), а также рассмотрены ограничения движка в проектах с низкой

нагрузкой на графику или необходимостью глубокого низкоуровневого управления.

Отдельное внимание уделено фреймворку Mono, обеспечивающему поддержку сценариев в Unity, а также ключевым методам жизненного цикла компонентов (Awake, Start, Update, LateUpdate, FixedUpdate). Подробно описаны способы создания, копирования и уничтожения объектов, а также методы временного отключения их активности, что позволяет эффективно управлять игровыми ресурсами и логикой приложения.

Рассматриваются механизмы работы с игровым временем (Time.deltaTime), обеспечивающие плавность и стабильность поведения игровых элементов независимо от частоты кадров. Особое внимание уделено асинхронному программированию: подробно описаны подходы на основе корутин, Task и специализированного решения UniTask, которое оптимизировано для использования в Unity и обеспечивает высокую производительность при минимальном расходе ресурсов.

Анализируются две основные системы построения пользовательского интерфейса в Unity: традиционная Canvas с простотой и удобством проектирования, и новая система UI Toolkit, обеспечивающая высокую производительность и гибкость за счёт декларативного подхода и разделения логики и представления.

Описаны популярные инструменты и библиотеки, используемые в разработке на Unity: плагин DoTween, позволяющий удобно реализовывать сложные анимации с минимальной нагрузкой на ресурсы, и фреймворк Zenject, реализующий принцип инверсии управления и внедрения зависимостей для улучшения архитектуры проекта, повышения модульности, тестируемости и снижения связанности кода.

Подробно разобраны инструменты управления ресурсами: AssetBundle и Addressable Asset System, обеспечивающие динамическую загрузку контента и эффективное управление памятью, а также более простые

подходы на основе папки Resources и Sprite Atlas, оптимального для мобильных приложений.

Дополнительно рассмотрены концепции делегатов и событий, позволяющие реализовывать гибкое событийно-ориентированное взаимодействие компонентов, паттерн «наблюдатель», а также реактивное программирование на основе библиотек Reactive Extensions, UniRx и R3, оптимизированных под специфику Unity и обеспечивающих эффективную работу с асинхронными событиями и потоками данных.

Также подробно описаны механизмы обработки ввода: система событий Unity EventSystems, старая и новая системы ввода (Input Manager и Input System), которые предоставляют удобный и эффективный способ реализации взаимодействия пользователя с игрой.

В разделе подробно рассматриваются решения для хранения данных в Unity: от простых встроенных хранилищ (PlayerPrefs) до надёжных и защищённых решений на основе SQLite и Realm DB с возможностью аппаратного шифрования и высокой скоростью работы.

Наконец, рассмотрена концепция Assembly Definition Files (asmdef), предназначенная для оптимизации процесса сборки и структурирования больших проектов, и детально разобраны шаблоны архитектуры пользовательских интерфейсов семейства MV* (MVC, MVP, MVVM), позволяющие чётко разделить обязанности компонентов и повысить гибкость и удобство поддержки игрового приложения.

Четвёртый раздел «Платформа .NET и ASP.NET» охватывает широкий спектр тем, связанных с созданием современных и высокопроизводительных приложений. В нём описаны ключевые преимущества платформы .NET, включая поддержку различных языков программирования, универсальность и производительность, обеспечиваемую JIT-компиляцией и сборщиком мусора в CLR. Подробно рассмотрены принципы SOLID, которые являются основой устойчивого и легко расширяемого кода.

Важное внимание уделено различным архитектурным стилям, от EBI и ECB до Clean Architecture, которые способствуют четкому разделению обязанностей, инверсии зависимостей и созданию масштабируемых систем. Также затронуты подходы Domain Driven Design и Hexagonal Architecture, обеспечивающие гибкость и простоту поддержки приложений.

Отдельно описана микросервисная архитектура, её достоинства и недостатки, а также ключевые паттерны, такие как Mediator, CQRS и Event Sourcing, позволяющие эффективно управлять распределёнными сервисами. Также рассмотрены паттерны отказоустойчивости (Retry, Circuit Breaker, Timeout, Rate Limiter, Fallback, Hedging), повышающие надёжность микросервисов.

Особое место отведено паттерну Saga и способам координации распределённых транзакций (оркестрация и хореография), а также роли Middleware в микросервисной экосистеме для обеспечения безопасности, наблюдаемости и надёжности.

Подробно описаны подходы REST API и gRPC, асинхронное и синхронное программирование, включая примитивы синхронизации и особенности многопоточности. Рассмотрены технологии реального времени (WebSocket, SSE, long polling, polling) и их преимущества в разных сценариях использования.

Затронуты инструменты обработки данных и обмена сообщениями, такие как RabbitMQ и Redis, а также средства фоновой обработки задач в .NET (Hangfire). Подробно описаны теоретические основы работы с базами данных, включая реляционную модель, SQL, транзакции и ACID-гарантии, а также подходы ORM и NoORM.

Важным блоком является инфраструктурная часть, охватывающая контейнеризацию (Docker и Docker Compose), оркестрацию контейнеров (Kubernetes) и подходы непрерывной интеграции и доставки (CI/CD) на базе GitLab.

Заключительная часть посвящена тестированию приложений, включая важность Unit-тестов и популярные инструменты (xUnit, NUnit), подходы к валидации данных (Fluent Validation), механизмам маршрутизации и конфигурации приложений в ASP.NET, а также системам наблюдаемости и мониторинга с использованием Elastic Stack (Elasticsearch, Logstash, Kibana), что обеспечивает глубокий контроль и понимание работы серверных систем.

Пятый раздел «Разработка клиента игры «Tactical Heroes» с помощью Unity» посвящен созданию клиентской части игры «Tactical Heroes» с использованием игрового движка Unity, вдохновленной пошаговой стратегией Heroes of Might and Magic. Игра построена на квадратной сетке с юнитами и героями, каждый из которых имеет уникальные характеристики, способности и анимации. Предусмотрены режимы одиночной и многопользовательской игры с матчмейкингом на основе рейтинга Эло.

Архитектура проекта разделена на слои (доменная логика, репозитории, сервисы и UI), которые взаимодействуют через интерфейсы и DI-контейнер Zenject. Центральным элементом управления игровым процессом является машина состояний, отвечающая за генерацию сетки, инициализацию юнитов, ход боя, проверку его завершения и финализацию.

Для хранения данных используются базы SQLite (для структурированных данных) и RealmDb (для гибких и масштабируемых структур). Разработаны универсальные сервисы для сетевых запросов, авторизации (с использованием JWT-токенов), обработки ошибок и кэширования изображений. Доменная логика включает расчет инициативы по ATB-шкале, алгоритм Дейкстры для поиска пути, генерацию игровой сетки и валидацию атак и перемещений.

Шестой раздел «Разработка серверной составляющей игры TacticalHeroes» раскрывает разработку серверной части игры Tactical Heroes, реализованной в виде микросервисной архитектуры с использованием REST API, HTTP 1.1 и gRPC. Центральным звеном является APIGateway, который

отвечает за маршрутизацию запросов, авторизацию через JWT, валидацию и обработку ошибок.

Функциональность разделена на микросервисы:

1. FrontendService (MVC + Vue.js) отображает сайт.
2. GamePlayService отвечает за игровой процесс через UDP-протокол для минимизации задержек, с собственным механизмом гарантированной доставки сообщений.
3. ChatsService обеспечивает работу чатов через WebSocket.
4. EmailService реализован через RabbitMQ и отправляет письма пользователям.
5. GameDataService, GamesService, PlayerBuildsService, ProfileService и ImagesService управляют данными игроков и игровых сущностей.
6. Хранение данных организовано в PostgreSQL (реляционные данные) и Amazon S3 (медиафайлы). Инфраструктура развернута через Docker Compose, автоматизировано развертывание с помощью GitHub Actions, реализовано логирование через стек ELK с уведомлениями об ошибках в Telegram. Все сервисы размещены на хостинге Timeweb с маршрутизацией и HTTPS через Nginx и SSL-сертификаты.

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломной работы была реализована поставленная цель — создана кроссплатформенная стратегическая игра «Tactical Heroes» с элементами мультиплеера на базе игрового движка Unity и платформы .NET.

Для достижения этой цели решены все задачи, сформулированные во введении.

1. В первом разделе был проведён систематический анализ современного рынка мобильных стратегий.
2. Во втором разделе рассмотрены базовые модели OSI и TCP/IP, проанализированы принципы работы протоколов HTTP и WebSocket, методы шифрования SSL/TLS и механизмы обеспечения безопасности. Отдельное внимание уделено клиент-серверной парадигме и её влиянию на производительность и масштабируемость игровых систем.
3. В третьем разделе детально изучены: архитектура Unity, система компонентов, жизненный цикл скриптов Mono, средства асинхронного программирования, управления ресурсами и пользовательскими интерфейсами. Описаны внедрение зависимостей с помощью Zenject, реактивный подход (UniRx / R3) и применение шаблонов семейства MV*.
4. В четвертом разделе проанализированы принципы SOLID, архитектурные стили (монолит, SOA, микросервисы), паттерны CQRS, Mediator, Saga, а также механизмы надёжности (Circuit Breaker, Retry, Bulkhead). Освещены возможности ASP.NET Core, RabbitMQ, Redis, PostgreSQL и Hangfire. Рассмотрены вопросы тестирования, валидации и наблюдаемости.
5. В пятом разделе описана проектная архитектура клиентской части на Unity, иерархия сервисов и доменных сущностей. Представлены реализация машины состояний боя, алгоритмы генерации сетки, поиска пути (A*) и ATB-шкалы, механизмы сохранения данных (SQLite, Realm) и адаптация интерфейса под различные устройства.

6. В шестом разделе спроектирована микросервисная экосистема (API Gateway, GamePlay, Chats, GameData и др.) с использованием DDD, CQRS и middleware на .Net. Описаны схемы маршрутизации и взаимодействия сервисов, интеграция через HTTP-API и WebSocket, реализация очередей RabbitMQ и кэша Redis, автоматическое применение миграций. Приведены решения по CI/CD (Docker Compose, GitHub Actions) и мониторингу (ELK-стек, Telegram-бот с алертами).

Планы на будущее:

1. Вынести общую бизнес логику в Nuget пакеты, чтобы изменения затрагивали одновременно и клиент, и сервер.
2. Подключить автоматические тесты на сервере и клиенте.
3. Автоматизировать процесс сборки и публикации клиентской части.
4. Разделить разные сервисы на разные репозитории и запускать их на разных серверах с единой сетью, чтобы уменьшить нагрузку на сервер и ускорить процес развертывания на сервере.
5. Перенести оркестрацию в Kubernetes.
6. Подключить анализ метрик Prometheus/VictoriaMetrics + Grafana
7. Для анализа утечек на сервере.
8. Переписать клиентскую часть браузера на реактивный фреймворк (Angular) и сделать несколько отдельных сервисов, один для административного редактирования баз данных игры в реальном времени, другой – публичный сайт, где люди смогут ознакомиться с игрой.
9. Разделить сервер на несколько окружение (Production, Staging, Testing, Development.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Исследование развития игровой индустрии [Электронный ресурс]. – URL: https://www.grandviewresearch.com/industry-analysis/video-game-market (дата обращения: 27.05.2025). - Загл. с экрана. - Яз. англ.
2. Исследование развития Unity [Электронный ресурс]. – URL: https://create.unity.com/gaming-report-2022 (дата обращения: 27.05.2025). - Загл. с экрана. - Яз. англ.
3. Stallings, W. Data and Computer Communications / W. Stallings // Pearson. — 2013. — 784 p.
4. Tanenbaum, A. S. Computer Networks / A. S. Tanenbaum, D. J. Wetherall // Pearson. — 2011. — 912 p.
5. Kurose, J. F. Computer Networking: A Top-Down Approach / J. F. Kurose, K. W. Ross // Addison-Wesley. — 2013. — 896 p.
6. Peterson, L. L. Computer Networks: A Systems Approach / L. L. Peterson, B. S. Davie // Morgan Kaufmann. — 2015. — 800 p.
7. Goldstone, J. Unity in Action: Multiplatform Game Development in C# (2nd ed.) / J. Goldstone, P. Buttfield-Addison, T. Nugent // Manning Publications. — 2018. — 384 p.
8. Richter, J. CLR via C# (4th ed.) / J. Richter // Microsoft Press. — 2012. — 826 p.
9. Martin, R. C. Clean Code: A Handbook of Agile Software Craftsmanship / R. C. Martin // Prentice Hall. — 2008. — 464 p.
10. Martin, R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design / R. C. Martin // Prentice Hall. — 2017. — 432 p.