UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Gabriel Solis Corrêa

Análise Comparativa de Ferramentas para Desenvolvimento de Jogos Digitais

Uberlândia, Brasil

2022

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Gabriel Solis Corrêa

Análise Comparativa de Ferramentas para Desenvolvimento de Jogos Digitais

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. rer. nat. Daniel Duarte Abdala

Universidade Federal de Uberlândia – UFU

Faculdade de Ciência da Computação

Bacharelado em Ciência da Computação

Uberlândia, Brasil

2022

Gabriel Solis Corrêa

Análise Comparativa de Ferramentas para Desenvolvimento de Jogos Digitais

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Ciência da Computação.

Trabalho aprovado. Uberlândia, Brasil, 30 de Março de 2022:

Prof. Dr. rer. nat. Daniel Duarte Abdala
Orientador

Prof. Dr. André R. Backes

Prof. Dr. Marcelo Zanchetta do Nascimento

Uberlândia, Brasil
2022

Resumo

Este trabalho apresenta sete diferentes ferramentas e serviços comumente utilizados no desenvolvimento de jogos para manutenção e aprimoramento do produto final. Além disso, este trabalho realiza uma análise comparativa entre 14 diferentes implementações dos elementos apresentados considerando as utilidades, facilidade de implementação em um projeto, custo de aquisição e diferentes sinergias com outras implementações que se complementam. Uma análise de caso é apresentada a partir do jogo de celular Neko Dungeon, utilizado para exemplificar como é feita a utilização destas ferramentas e serviços em um jogo comercial.

Palavras-chave: Jogos Eletrônicos, Ferramentas de Suporte, Análise Comparativa.

Abstract

This work presents seven different tools and services commonly used in game development for maintain and improve the final product. In addition, this work performs a comparative analysis between 14 different implementations of the presented elements considering the utilities, ease of implementation in a project, acquisition cost and different synergies with other implementations that complement each other. A case analysis is presented from the mobile game Neko Dungeon, used to exemplify how these tools and services are used in a commercial game.

Keywords: Electronic Games, Support Tools, Services, Comparative Analysis.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

O mercado de jogos sempre foi atrativo para graduandos e profissionais da área da tecnologia da informação. No entanto, nas últimas décadas, o mercado de jogos digitais tornou-se incrivelmente grande, sendo atualmente maior que o mercado cinematográfico segundo (STATISTA, 2020), visto que o primeiro é um mercado que movimenta 160 bilhões de dólares por ano, enquanto a indústria cinematográfica é responsável por 45 bilhões. Devido a este crescimento, vários outros profissionais de outras áreas já estão sendo atraídos para este mercado, como artistas gráficos, especialistas em trilha sonora e efeitos sonoros e roteiristas. Embora a área seja bastante fomentada no contexto contemporâneo, jogos modernos necessitam muito mais que um bom motor de jogos, um visual atrativo e uma sonoridade envolvente. Dessa forma, esses jogos são extremamente complexos e requerem que várias tarefas sejam executadas em segundo plano, muitas das quais o jogador nem está ciente.

Nesse sentido, serviços como Analytics, Análise de Crashes e Localização são exemplos de recursos que hoje em dia são essenciais para o funcionamento de jogos eletrônicos modernos, mas que são de difícil visualização pelo jogador.

Muitas vezes, as maiores adversidades enfrentadas na produção de jogos vêm da implementação de conceitos amplamente conhecidos que exigem serviços ou ferramentas pouco conhecidas e/ou de difícil entendimento. Com poucos materiais disponíveis para o público geral, geralmente encontrados somente na língua inglesa, este trabalho tem como objetivo principal apresentar uma visão inicial comparativa de algumas dessas tecnologias de difícil acesso que são tão necessárias para o desenvolvimento, suporte, aprimoramento, análise e manutenção de jogos eletrônicos.

Para exemplo do tamanho deste mercado, a Figura 1 mostra o valor da Indústria de jogos atualmente e uma projeção para os próximos anos. Já na Figura 2 é mostrado o crescimento do número de jogadores em todo o mundo. A Figura 3 mostra o crescimento do número de empresas voltadas para o desenvolvimento de jogos nos Estados Unidos da América. E por último, a Figura 4 apresenta um exemplo de faixa salarial para um programador de jogos nos Estados Unidos da América no ano de 2022. As fontes destas Figuras podem ser encontradas em (STATISTA, 2021), (IBISWORLD, 2021) e (ZIPRECRUITER, 2022).

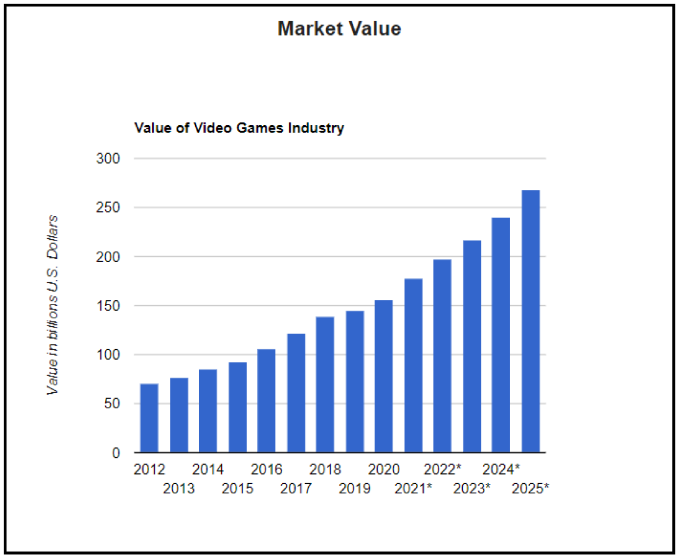

Figura 1 – Valor da Indústria de Jogos. Fonte: Statista, 2021

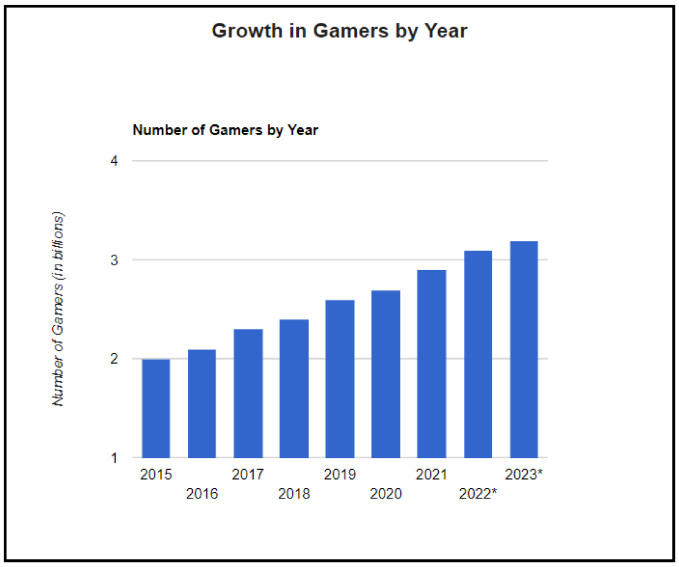

Figura 2 – Número de Jogadores. Fonte: Statista, 2021

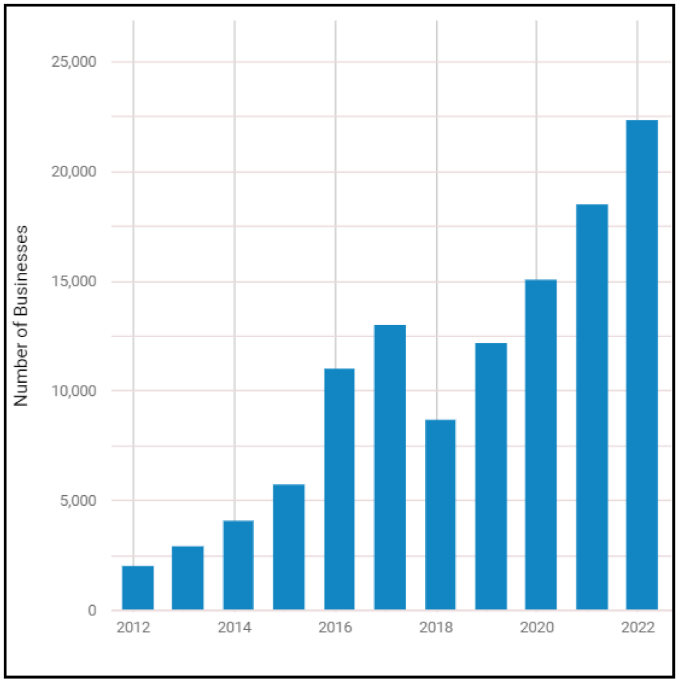

Figura 3 – Número de empresas voltadas para o desenvolvimento do jogos nos EUA. Fonte: IBISWorld, 2021

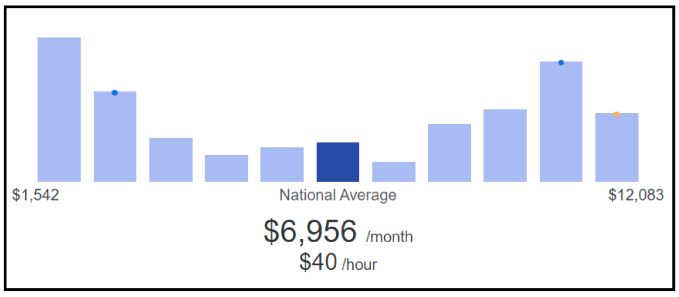

Figura 4 – Exemplo de Faixa Salarial para um programador de jogos nos EUA. Fonte: ZipRecruiter, 2022

Embora o número de empresas, profissionais e clientes na área estejam numa constante crescente ao se observar os últimos dez anos, a maioria das formações em desenvolvimento de jogos e de guias e livros disponíveis na internet acerca dessa área não abordam ou não tem como foco principal os serviços e as ferramentas, suas aplicações e como fazer uso deles. Em geral, a maioria dos jogos comerciais lançados fazem uso de pelo menos um dos serviços e ferramentas que foram abordados neste trabalho.

Como caso de estudo para exemplificar a utilização e a necessidade das ferramentas de suporte revisadas neste trabalho, utilizou-se um jogo comercial chamado Neko Dungeon desenvolvido pela empresa Gilp Studio (STUDIO, 2022). Este jogo está sendo desenvolvido há de um ano e já se encontra disponível para ser instalado nas principais plataformas de jogos para celular atualmente. Vale ressaltar que a empresa Gilp Studio deu permissão

para utilização deste jogo como caso de estudo e o autor desta monografia trabalhou e contribuiu ativamente no desenvolvimento deste jogo.

As ferramentas e serviços que foram abordadas neste trabalho tiveram como critérios de escolha.

- Popularidade da ferramenta/serviço no jogos comerciais lançados nos últimos anos;
- Disponibilidade da ferramenta/serviço nos motores de jogos mais populares;
- Disponibilidade da ferramenta/serviço tanto de forma gratuita quanto paga;
- Curva de aprendizado e aplicação da ferramenta/serviço.

1.1 Objetivos

Esta monografia tem como objetivo apresentar ao leitor uma visão geral de técnicas e sistemas de suporte para desenvolvimento de jogos que são de grande importância para o aprimoramento e manutenção do produto final.

Além disso, esta monografia tem como objetivos específicos:

- Apresentar serviços de Analytics;
- Apresentar serviços para Análise de Crashes;
- Apresentar ferramentas para Localização;
- Apresentar serviços para Notificações Locais e Remotas;
- Apresentar serviços para Microtransactions;
- Apresentar serviços para Advertising;
- Apresentar ferramentas para Anti-Cheat;
- Apresentar o uso de todas as ferramentas e serviços discutidos em um jogo comercial.

1.2 Justificativa

A proposta deste projeto é apresentar conhecimentos e técnicas que podem facilitar e acelerar o desenvolvimento da maioria dos tipos de jogos, dentre elas bibliotecas, ferramentas que cuidam de segurança e traduções e também de serviços.

Com isso, este trabalho busca informar sobre a área de serviços e ferramentas para desenvolvimentos de jogos. Atualmente, esta é uma área com esparsa literatura a qual,

em sua maioria, consiste de livros que não são mais relevantes na década de 2020, já que são antigos ou logo ficam defasados, em virtude dos constantes avanços nesse setor. É possível encontrar trechos relevantes em livros mais recentes, porém normalmente estes são focados em outras partes do desenvolvimento de jogos, de modo que os profissionais da área precisam buscar outras formas de se manterem informados.

Observa-se que na atual crescente deste mercado, a busca por expandir seu conhecimento na área se tornou uma preocupação de diversos desenvolvedores que também querem fazer parte deste mercado.

Muitas das ferramentas que serão discutidas nesta monografia são essenciais para o desenvolvimento de jogos e muitas vezes não são de conhecimento público. Ao analisar sob essa ótica, esse projeto busca apresentar/comparar quais tecnologias estão à disposição e como utilizá-las.

A organização deste trabalho segue a seguinte estrutura:

- o capítulo 2 apresenta a revisão do estado da arte, nele cada uma das ferramentas e serviços abordados são brevemente explicados, além das principais referências disponíveis associadas a cada um destas são elencadas e comparadas;
- o capítulo 3 apresenta a fundamentação teórica, com o intuito de esclarecer os principais conceitos e apresentar exemplos das diferentes ferramentas abordadas;
- o capítulo 4 refere-se ao caso de estudo deste trabalho, no qual a utilização das sete ferramentas e serviços abordadas são apresentadas e discutidas no contexto real do jogo comercial Neko Dungeon;
- o capítulo 5 finaliza o trabalho com a apresentação de uma discussão final sobre as ferramentas e a conclusão da monografia.

2 Revisão do Estado da Arte

Neste capítulo, as ferramentas consideradas pelo trabalho foram organizadas em ordem de utilidade em um maior número de projetos. As ferramentas/serviços Analytics, Análise de Crashes e Localização são utilizadas na maioria dos jogos comerciais em virtude das suas funções serem comumente necessitadas. As Notificações, Advertising e Microtransaction são normalmente utilizadas por jogos para aparelhos móveis, com exceção do Microtransaction que, em alguns casos, pode ter utilidade em jogos para outras plataformas. Por fim, a última ferramenta abordada é o Anti-Cheat, que depende bastante da natureza do jogo para se fazer necessário, por isso, normalmente é implementado em jogos de caráter multi-jogador ou em jogos em que o uso de trapaças pode impactar negativamente no modelo de monetização utilizado.

Nesse cenário de ferramentas e serviços essenciais para o desenvolvimento de jogos, o Analytics é um serviço amplamente empregado, que colhe informações sobre o dispositivo utilizado pelo jogador e sobre o seu comportamento durante a execução do jogo. Assim, essas informações facilitam e embasam análises para melhorar não só a experiência do usuário, como também a retenção dos jogadores. A primeira ferramenta de Analytics aplicada a jogos foi criada pela empresa Bioware e apresentada por Georg Zoeller em uma palestra do evento *Game Developers Conference* (GDC) em 2010 (ZOELLER, 2010), a ferramenta foi criada para ser utilizada nos jogos Dragon Age Origins e Mass Effect da mesma empresa. O primeiro trabalho que elucidou o uso dos Analytics para melhorar a experiência do usuário foi escrito por Ben Medler em (MEDLER, 2009). Para saber mais sobre Analytics e como implementá-lo em um jogo comercial da forma correta, sugere-se o livro escrito por Magy Seif El-Nasr em (EL-NASR; DRACHEN; CANOSSA, 2013) e o Capítulo 3 deste trabalho onde toda a parte de implementação associada ao Analytics é abordada. As implementações mais utilizadas desta ferramenta na atualidade são o Google Firebase Analytics (FIREBASE, 2022c) e o DeltaDNA (DELTADNA, 2022), onde o Firebase Analytics se destaca pelas suas múltiplas funcionalidades e pelo seu plano gratuito bastante abrangente, requisitando pagamento apenas se o número de jogadores exceder um limite determinado pela plataforma. Já o DeltaDNA se destaca por ter ainda mais funções comparadas ao Firebase Analytics, no entanto, o DeltaDNA sempre opera em um plano pago independente do número de jogadores.

Outro serviço amplamente utilizado é a Análise de Crashes, que consiste em um serviço para coleta de dados referente a quaisquer erros ou falhas críticas que possam acontecer durante a execução do jogo. Esse serviço é bastante importante para agilizar o processo de diagnóstico e correção de bugs, pois facilita este processo para a análise de bugs que acontecem raramente e para bugs que acontecem em dispositivos específicos no

qual a equipe de desenvolvimento não tem acesso. A Análise de Crashes foi concebida junto da ferramenta de Analytics por conta da sua forma de operação similar. É incerto qual a primeira vez que a Análise de Crashes foi implementada em um jogo comercial, porém é notável que a grande parcela dos maiores jogos comerciais atualmente fazem uso desta ferramenta. Por exemplo, o jogo League Of Legends, desenvolvido pela empresa Riot Games, faz uso da ferramenta Bugsplat, o jogo World of Warcraft, desenvolvido pela empresa Blizzard, faz uso da ferramenta Error Reporter. Já a plataforma de jogos para computador Steam implementa em todos os jogos hospedados na sua plataforma o Steam Error Reporting, incluído no SDK Steamworks, entre muitos outros. Para saber mais sobre Análise de Crashes e como implementá-la em um jogo comercial da forma correta, sugere-se a leitura do Capítulo 3 deste trabalho onde toda a parte de implementação associada a Análise de Crashes é abordada. As implementações desta ferramenta mais utilizadas na atualidade são o Google Firebase Crashlytics (FIREBASE, 2022a) e o Bugsnag (BUGSNAG, 2022), onde o Firebase Crashlytics se destaca pelas suas múltiplas funcionalidades e pelo seu plano gratuito bastante abrangente, requisitando pagamento apenas se o número de jogadores exceder um limite determinado pela plataforma, e o Bugsnag se destaca por ter ainda mais funções comparadas ao Firebase Analytics. No entanto, o Bugsnag sempre opera em um plano pago independente do número de jogadores.

No contexto de ampliação dos públicos de um jogo para diversos países, Localização é uma ferramenta para facilitar a internacionalização de um jogo, visto que implementa funcionalidades para tradução de textos e elementos visuais com base na língua do dispositivo do usuário ou de acordo com a língua escolhida por este. Sobre esse processo, os videogames passam pelo processo de tradução desde o começo da história dos jogos. A princípio, isto acontecia devido à necessidade da internacionalização dos jogos entre os mercados estadunidenses e japoneses que eram, e ainda são, líderes no setor. No entanto, naquela época e até a terceira geração dos consoles, este processo era realizado no código fonte do jogo, sem uso de nenhuma ferramenta para facilitar o processo. Como principais exemplos de jogos que passaram pelo processo de internacionalização, os jogos Space Invaders desenvolvido por Tomohiro Nishikado e Pac-Man desenvolvido por Tōru Iwatani, e comercializado pela empresa Namco, foram grandes sucessos no ocidente mesmo sendo jogos japoneses. Para saber mais sobre Localização e como implementá-la em um jogo comercial da forma correta, sugere-se o livro escrito por Heather Maxwell Chandler em (CHANDLER, 2011) e o Capítulo 3 deste trabalho onde toda a parte de implementação associada ao Localização é abordada. As implementações desta ferramenta mais utilizadas na atualidade são o Unity Localization (UNITY, 2022c) e o Lean Localization (WILKES, 2022), onde o Unity Localization se destaca pela sua facilidade de implementação e pelo seu sistema robusto de importação de tabelas de localização pela internet e o Lean Localization se sobressai pelo seu sistema de importações de tabelas, que é compatível com diversos formatos de arquivos.

As Notificações Locais e Remotas são implementadas por uma ferramenta de mesmo nome para mostrar notificações no aparelho celular do jogador após um tempo determinado durante a execução do jogo ou a partir de uma chamada remota realizada pela internet de um serviço específico para enviar Notificações remotas. Essa ferramenta tem como principal função aumentar a retenção dos jogadores. É incerto qual a primeira vez que a Análise de Crashes foi implementada em um jogo comercial, porém é evidente que a grande parcela dos maiores jogos comerciais de celular atualmente fazem uso desta ferramenta, por exemplo o jogo Clash of Clans, desenvolvido pela empresa Supercell, faz uso de notificações para notificar o jogador de que algum processo dentro do jogo foi concluído ou de que alguém atacou a aldeia do jogador. Já o jogo Candy Crush Saga, desenvolvido pela empresa King, faz uso de notificações para comunicar o jogador de que algum recurso foi reabastecido ou de que o jogador já está há demasiado tempo sem interagir com o jogo. Para saber mais sobre Notificações Locais e Remotas e como implementá-las em um jogo comercial da forma correta, sugere-se a leitura do Capítulo 3 deste trabalho onde toda a parte de implementação associada a Notificações é abordada. As implementações desta ferramenta mais utilizadas na atualidade são o Easy Mobile Pro (SGLIB, 2022a) e o One Signal (ONESIGNAL, 2022), onde o Easy Mobile Pro, que é uma ferramenta paga, se destaca pelas suas múltiplas funcionalidades e por ser capaz de implementar notificações locais e remotas, por mais que seja necessário realizar a conexão com um provedor de notificações para que o suporte a notificações remotas aconteça, e o One Signal, que é um serviço pago, se destaca pelo seu excelente suporte à notificações remotas, além de oferecer diversas ferramentas para analisar o impacto de cada notificação na retenção dos jogadores.

Além dos recursos de Analytics, análise de crashes e notificação, outro importante serviço é o Microtransaction, o qual permite realizar transações monetárias entre o jogador e a loja em que o jogo está hospedado, para que o jogador possa comprar qualquer produto virtual ou assinatura que desejar dentro do jogo. Esse serviço tem como principal função gerar receita para o jogo e viabilizar o modelo de monetização escolhido. Microtransactions foram implementadas pela primeira vez em 2006 no jogo Elder Scrolls IV: Oblivion, desenvolvido pela empresa Bethesda Studios, vendendo cosméticos para itens do jogo. Para saber mais sobre Microtransaction e como implementá-lo em um jogo comercial da forma correta, sugere-se o livro escrito por Tim Fields em (FIELDS, 2014) e o Capítulo 3 deste trabalho onde toda a parte de implementação associada ao Microtransactions é abordada. As implementações desta ferramenta mais utilizadas na atualidade são o Easy Mobile Pro (SGLIB, 2022c) e o Steamworks (VALVE, 2022a), onde ambas as ferramentas se mostram bastante eficientes no que propõe fazer. Válido ressaltar que a ferramenta Easy Mobile Pro deve ser adquirida para ser utilizada.

Seguindo a mesma linha de monetização, outra importante implementação foi o serviço de Advertising, que é um serviço para mostrar anúncios durante a execução do

jogo, sendo estes anúncios puláveis após um período de tempo ou não puláveis. Esse serviço tem como principal função gerar receita para o jogo e viabilizar o modelo de monetização escolhido. As primeiras implementações de anúncios em jogos aconteceu na década de 80, como o jogo Pepsi Invaders, desenvolvido pela empresa Atari, que foi um jogo patrocinado pela marca Coca-Cola no qual o jogador luta contra o principal competidor da marca, o jogo Cool Spot, desenvolvido pela empresa Virgin Interactive, que é um jogo patrocinado pela marca 7-Up no qual o jogador joga uma aventura em que o mascote da marca é o protagonista. Para saber mais sobre Advertising e como implementá-lo em um jogo comercial da forma correta, sugere-se o livro escrito por Heather Gérald Marolf em (MAROLF, 2007) e o Capítulo 3 deste trabalho onde toda a parte de implementação associada ao Advertising é abordada. As implementações desta ferramenta mais utilizadas na atualidade são o Easy Mobile Pro (SGLIB, 2022b) e o Unity Ads (UNITY, 2022b), onde o Easy Mobile Pro, que é uma ferramenta paga, se destaca pelas suas múltiplas funcionalidades e o Unity Ads se destaca por ser uma ferramenta gratuita.

E por último, o Anti-Cheat é uma ferramenta para dificultar a utilização de trapaças, principalmente daquelas que envolvem a manipulação de dados guardados tanto em memória volátil quanto em memória permanente. Esse serviço tem como principal função tentar impedir que um trapaceiro atrapalhe a experiência de outros jogadores em jogos multi-jogador, além de tentar impedir que o trapaceiro faça uso de trapaças que podem impactar negativamente no modelo de monetização escolhido. Não é possível constatar qual a primeira vez que o Anti-Cheat foi implementado em um jogo comercial, porém é notável que a grande parcela dos maiores jogos comerciais multi-jogadores atualmente fazem uso desta ferramenta, como por exemplo o jogo Valorant, desenvolvido pela empresa Riot Games, faz uso da ferramenta Easy Anti-Cheat, o jogo Tom Clancy's Rainbow Six Siege, desenvolvido pela empresa Ubisoft, faz uso da ferramenta BattlEye, a plataforma de jogos para computador Steam implementa em todos os jogos hospedados na sua plataforma o Valve Anti-Cheat, incluído no SDK Steamworks, entre muitos outros. Para saber mais sobre Anti-Cheat e como implementá-lo em um jogo comercial da forma correta, sugere-se o livro escrito por Steven Davis em (DAVIS, 2009) e o Capítulo 3 deste trabalho onde toda a parte de implementação associada ao Anti-Cheat é abordada. As implementações desta ferramenta mais utilizadas na atualidade são o Anti-Cheat Toolkit (CODESTAGE, 2022) e o SCUE4 Anti-Cheat Solution (LEITE, 2022), onde o Anti-Cheat Toolkit, que é uma ferramenta paga, se destaca por possuir funcionalidades tanto para criptografia de dados quanto para detecção de alguns tipos de dados e o SCUE4 Anti-Cheat Solution se destaca por possuir funcionalidades para criptografia de dados e por ser uma ferramenta gratuita.

Dentre os trabalhos e teses analisados, em repositórios institucionais de universidades por todo o Brasil foi possível observar diversos trabalhos envolvendo jogos, porém praticamente todos eles se enquadram em duas categorias que são diferentes do que este

trabalho propõe: Trabalhos relacionados a jogos sérios (Jogos sérios são quaisquer jogos onde o foco principal não é o entretenimento do jogador, como Jogos educativos, jogos voltados para a reabilitação física e/ou psicossocial de indivíduos, entre outros), sendo a implementação de tais jogos ou uma análise do impacto desses jogos em realizar o propósito pelo qual foram criados. A segunda categoria enquadra trabalhos relacionados à implementação de IAs (Inteligências Artificiais) de tipo agente jogador em jogos (IAs de tipo agente jogador são softwares que tentam imitar o comportamento de um jogador humano para tentar jogar um jogo da melhor forma possível).

Exemplos destes tipos de trabalhos são o trabalho de Jefferson Dias Cardoso em (CARDOSO; LOPES, 2019) que é um trabalho sobre a implementação de um jogo sério para ensinar robótica. O trabalho de Dayenne Godoy Pellucci Maciel em (MACIEL, 2020) também ilustra essa abordagem, visto que é um trabalho que propõe e analisa novos estilos de jogos no ensino da matéria de ciências e o trabalho de Guilherme Pacheco de Oliveira em (OLIVEIRA, 2018) que é um trabalho sobre a implementação de uma IA que implementa um agente jogador para jogar um jogo de turnos criado pelo próprio autor. Como exceção à regra, cita-se o trabalho de Gabriel Albuquerque Ferreira em (FERREIRA, 2021) em que o autor escreve sobre as dificuldades para realizar a localização de um jogo comercial, o Spelunky classic HD, no qual é mostrado o processo de localização da parte textual e da parte visual do jogo.

Acerca do uso de cada ferramenta em jogos comerciais, é possível dizer que a maioria dos jogos para celular que faz uso do estilo de negócio, no qual o jogo é grátis para jogar e realiza a sua monetização por outros meios (comumente chamado de “Freemium” na indústria) e utiliza das ferramentas Advertising e Microtransactions, uma vez que elas são essenciais para viabilizar este estilo de negócio. Todo jogo para celular pode fazer uso da ferramenta de Notificações Locais e/ou Remotas para aumentar a retenção do usuário. Além disso, todo jogo que possui uma opção nas suas configurações para que se possa mudar a língua do jogo faz uso da ferramenta de Localização. A maioria dos jogos que possuem interação com outros jogadores ou que possuem um modelo de negócio que pode ser impactado negativamente por trapaças faz uso da ferramenta de Anti-Cheat. E por último, qualquer jogo de qualquer gênero ou plataforma pode fazer uso do Analytics e da Análise de Crashes, pois são ferramentas que independem das características do jogo e sempre são benéficas para a manutenção e melhoramento do jogo.

Exemplos de jogos que fazem uso destas ferramentas citadas acima são: Archero, um jogo de celular gratuito para jogar, produzido pelo estúdio de jogos Habby que faz uso de todas as ferramentas discutidas neste trabalho; o Call of Duty: Black Ops Cold War, um jogo pago para computador e para os consoles PlayStation 4 e 5 e Xbox Series X, Series S e One produzido pelo estúdio de jogos Activision que faz uso das ferramentas Analytics, Análise de Crashes, Localização, Microtransaction e Anti-Cheat; o League

of Legends, um jogo gratuito para jogar para computador produzido pelo estúdio de jogos Riot Games que faz uso das ferramentas Analytics, Análise de Crashes, Localização, Microtransaction e Anti-Cheat. No contexto contemporâneo, é extremamente difícil encontrar jogos comerciais que não fazem uso de nenhuma das ferramentas discutidas, salvo alguns jogos menores.

No Capítulo 3, uma revisão pormenorizada com informações sobre funcionamento e instalação das diferentes ferramentas selecionadas é apresentada.

3 Fundamentação Teórica

Neste capítulo foram abordados e analisados um total de 14 diferentes implementações para as sete ferramentas/serviços consideradas para o trabalho. Foram escolhidas duas implementações de cada ferramenta com base na popularidade e na frequência de utilização no desenvolvimento de jogos. A seguir é mostrado um lista contendo cada uma das implementações escolhidas para cada um dos serviços e ferramentas considerados:

- Analytics:
  - Google Firebase (FIREBASE, 2022b);
  - deltaDNA (DELTADNA, 2022);
- Análise de Crashes:
  - Crashlytics (FIREBASE, 2022a);
  - Bugsnag (BUGSNAG, 2022).
- Localização:
  - Unity Localization (Nativa da Unity Engine) (UNITY, 2022c);
  - Lean Localization (WILKES, 2022).
- Notificações:
  - Easy Mobile Pro (SGLIB, 2022a);
  - One Signal (ONESIGNAL, 2022).
- Microtransactions:
  - Easy Mobile Pro (Para jogos móveis.) (SGLIB, 2022c);
  - Steamworks (Para jogos da Steam) (VALVE, 2022b).
- Advertising:
  - Easy Mobile Pro (SGLIB, 2022b);
  - Unity Ads (UNITY, 2022b).
- Anti-Cheat:
  - Anti-Cheat Toolkit (CODESTAGE, 2022);
  - SCUE4 Anti-Cheat Solution (LEITE, 2022).

Uma questão importante a ser pontuada sobre qualquer uma das ferramentas discutidas a seguir é que todas as ferramentas selecionadas para este trabalho respeitam a Lei Geral de Proteção de Dados Pessoais (LGPD) e a General Data Protection Regulation (GDPR), do inglês Regulação Geral de Proteção de Dados Pessoais, as quais dispõem de tratamentos para dados pessoais, com o objetivo de proteger os direitos fundamentais de liberdade e privacidade de cada pessoa natural, sendo a LGPD referente a cidadãos brasileiros e a GDPR cidadãos de países pertencentes à União Europeia. As ferramentas respeitam estas leis de forma a requisitar o consentimento do usuário na abertura do jogo, informar sobre os objetivos por trás da coleta de dados e não coletar dados de crianças sem o consentimento dos pais.

Na seção 3.1 inicia-se a revisão sistemática das ferramentas por meio da análise da ferramenta Analytics.

3.1 Analytics

Este é um serviço para coleta e análise de dados dos usuários. Seu principal uso refere-se ao estudo do comportamento do jogador perante inúmeros aspectos do jogo, em especial, o fluxo principal de jogabilidade, a interação com as interfaces do jogo e a interação com os elementos de monetização. A finalidade é melhorar a experiência do usuário através de mudanças de valores (e até em alguns casos modificando fluxos inteiros de interação). Outra utilidade não menos importante desta ferramenta é em aumentar a retenção do jogador realizando mudanças que o estimulem a jogar por mais tempo (especialmente importante em jogos para dispositivos móveis) e também melhorar a monetização do jogo (no caso de jogos em que o seu modelo de monetização depende das ações do jogador dentro do jogo).

Todas as informações coletadas por este serviço são enviadas a um servidor e por fim disponibilizadas em um painel de controle ou console, no qual estas informações são organizadas e preparadas para análise. Por conta disto, este serviço requer que o dispositivo no qual o jogo está sendo jogado esteja conectado com a internet para que as informações sejam enviadas. Caso o dispositivo esteja desconectado da internet, é comum que os dados sejam reservados para envio a posteriori ao servidor, assim que o aparelho retomar a conexão à internet.

Algumas informações genéricas são coletadas automaticamente bastando apenas que o serviço estar presente, como coletar o modelo do aparelho e informações básicas sobre o usuário (de forma que respeite a LGPD e a GDPR). No entanto o real potencial por trás do Analytics é a coleta de informações por meio do disparo de eventos. Eventos são uma forma de notificar o servidor que coleta as informações de que algum usuário realizou uma ação referente àquele evento. Estes são disparados pelo código do jogo a

medida que o desenvolvedor achar válido. Cada evento pode ser disparado múltiplas vezes se necessário e cada evento pode ou não possuir parâmetros quantitativos ou qualitativos para trazer detalhes. Por exemplo, um jogo com fluxo de jogabilidade de partidas pode ter um evento que é enviado toda vez que um jogador ganha ou perde uma partida e estes eventos podem trazer detalhes sobre o que aquele jogador possuía ao ganhar ou perder, qual a duração da partida e no caso das perdas seria possível detalhar o que fez o jogador perder. Com estas informações, é possível determinar o que pode estar atrapalhando a experiência do usuário e então realizar mudanças no jogo para corrigir o que foi observado como um problema.

Válido ressaltar que muitos dos painéis de controles implementados por estes serviços fornecem ferramentas para facilitar a análise dos dados, como filtros e funis que permitem selecionar um grupo demográfico do usuários e o analisar perante eventos, tempo de jogo, monetização, dentre outros dados.

Um exemplo de um possível uso desta ferramenta é na implementação de um evento que é chamado toda vez que o jogador realiza uma compra dentro do jogo de algum recurso. Esse evento coletaria a quantidade comprada daquele recurso e para qual personagem aquele recurso foi comprado, assim seria possível observar qual a média de recurso comprada por usuário e por transação, qual grupo demográfico que mais compra daquele recurso, para qual personagem tal recurso é mais vezes comprado (assim apontando se aquele personagem é mais popular ou se ele necessita mais daquele recurso comparado a outros personagens), entre outras métricas. Em posse dessas informações, os desenvolvedores do jogo podem realizar alterações nos valores de custo e recurso comprados ou até mesmo realizar uma mudança completa no fluxo de compra com a melhora do jogo em mente.

Neste trabalho foram consideradas duas diferentes implementações do Analytics para serem utilizadas em um jogo comercial, o Firebase Analytics (FIREBASE, 2022b) e o deltaDNA (DELTADNA, 2022). O levantamento acerca do Analytics se limitou a estas duas implementação pois elas são as principais implementações discutidas e recomendadas nos fóruns de desenvolvimento de jogos. É válido ressaltar que há outras implementações de Analytics que não foram consideradas para este trabalho como a Unity Analytics (UNITY, 2022d) e o Smartlook (SMARTLOOK, 2022).

3.1.1 Firebase Analytics

Firebase Analytics é um serviço de Analytics disponibilizado pela Google de forma gratuita para projetos pequenos e com um plano pago de custo variável para projetos maiores e mais complexos. Ele é compatível com jogos produzidos na Unity, com aplicativos nativos construídos para Android e iOS e com aplicativos construídos utilizando C++. Este serviço possui um plano pago com custo variável dependendo das demandas do jogo.

Este plano pode ser necessário para jogos de grande escala pois algumas funcionalidades do serviço são limitadas e que podem ser insuficientes se o número de jogadores for maior que o suportado.

Para uso do Firebase Analytics, primeiro é necessário criar um projeto no site do Google Firebase referente ao jogo. A Figura 5 mostra a tela de boas vindas.

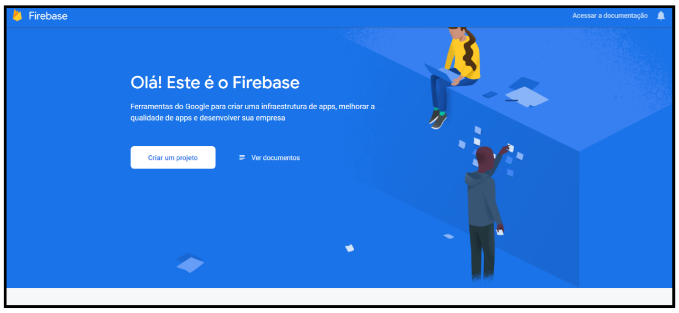

Figura 5 – Tela de criação de projetos do Painel de Controle do Google Firebase. Fonte: Documentação do Google Firebase, 2022

Com o projeto criado, é necessário realizar o download e a instalação do arquivo de configuração e do SDK do Google Firebase no diretório do jogo como mostrado nas Figuras 6 e 7. Este processo é diferente para cada uma das plataformas citadas na introdução da sessão (no caso deste trabalho, é mostrado o processo para adicionar o Firebase Analytics na Unity), caso seja necessário, a documentação sobre este processo pode ser encontrada em (FIREBASE, 2022b).

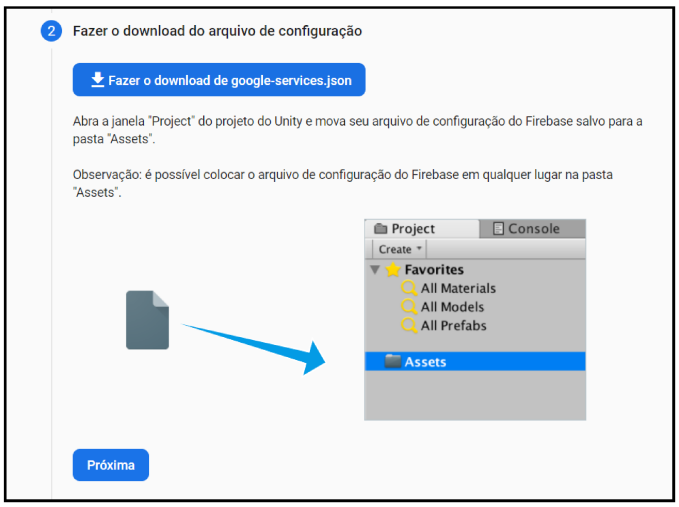

Figura 6 – Instruções e Download para o arquivo de configuração do Google Firebase. Fonte: Documentação do Google Firebase, 2022

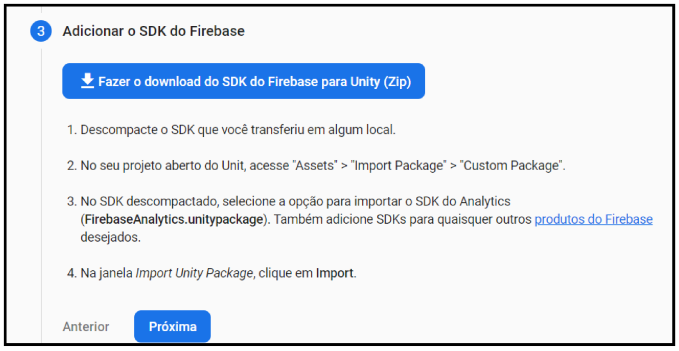

Figura 7 – Instruções e Download para o SDK do Google Firebase. Fonte: Documentação do Google Firebase, 2022

Após realizado a instalação, deve-se realizar a inicialização do serviço e então os eventos poderão ser disparados normalmente. Para poder visualizar os dados coletados basta acessar o painel de controle do projeto do jogo que foi criado no Google Firebase e verificar a sessão de Analytics na barra lateral como mostrado na Figura 8. Uma implementação do Firebase Analytics em um jogo comercial é apresentada no Capítulo 4.

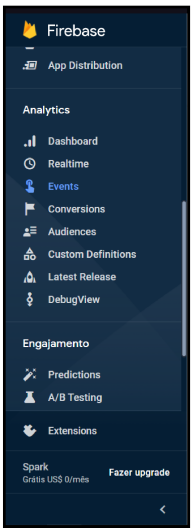

Figura 8 – Barra lateral do painel de controle do Google Firebase. Fonte: Google Firebase, 2022

3.1.2 deltaDNA

deltaDNA é um serviço de Analytics pago compatível com jogos produzidos na Unity e para aplicativos nativos construídos para Android e iOS.

Para uso do deltaDNA, primeiro é necessário criar uma conta no site do deltaDNA como é feito na Figura 9.

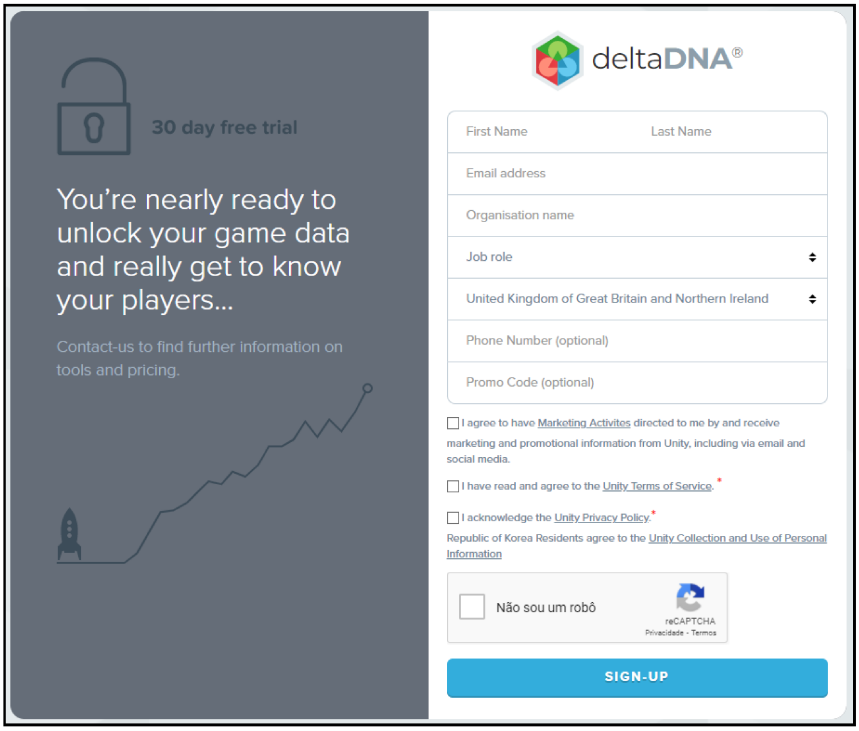

Figura 9 – Tela de criação de conta do deltaDNA. Fonte: Documentação do deltaDNA, 2022

Feito isto, é necessário adicionar o jogo na conta que acabou de ser criada como é feito na Figura 10.

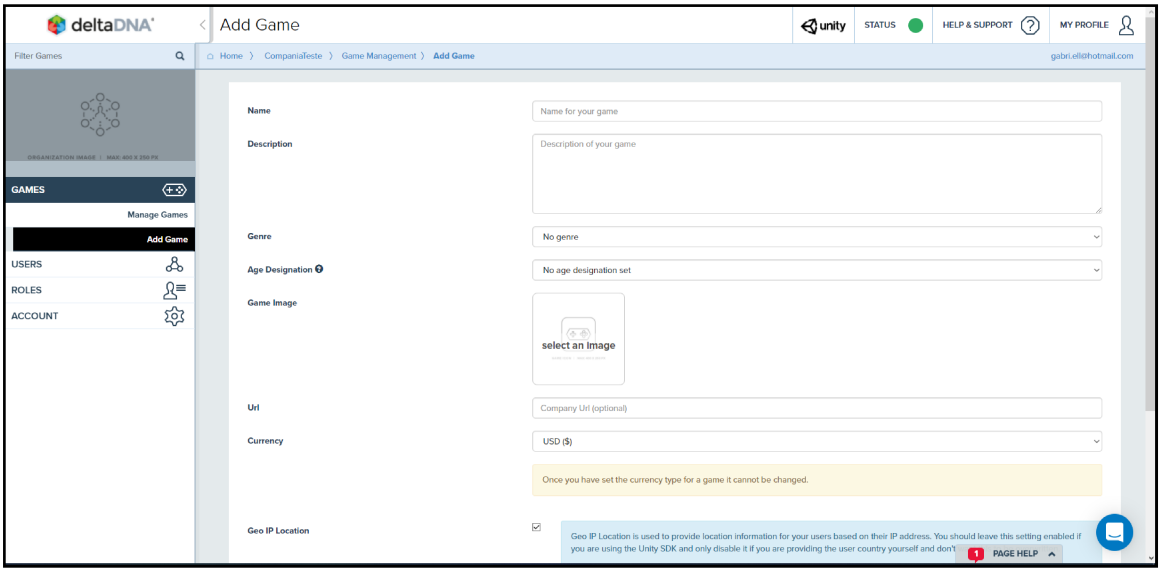

Figura 10 – Tela de criação do projeto do jogo no deltaDNA. Fonte: Documentação do deltaDNA, 2022

Após isto, é necessário realizar o download e a instalação do SDK do deltaDNA no diretório do jogo como mostrado na Figura 11. Este processo é diferente para cada uma

das plataformas citadas na introdução da sessão, caso seja necessário, a documentação sobre este processo pode ser encontrada em (DELTADNA, 2022).

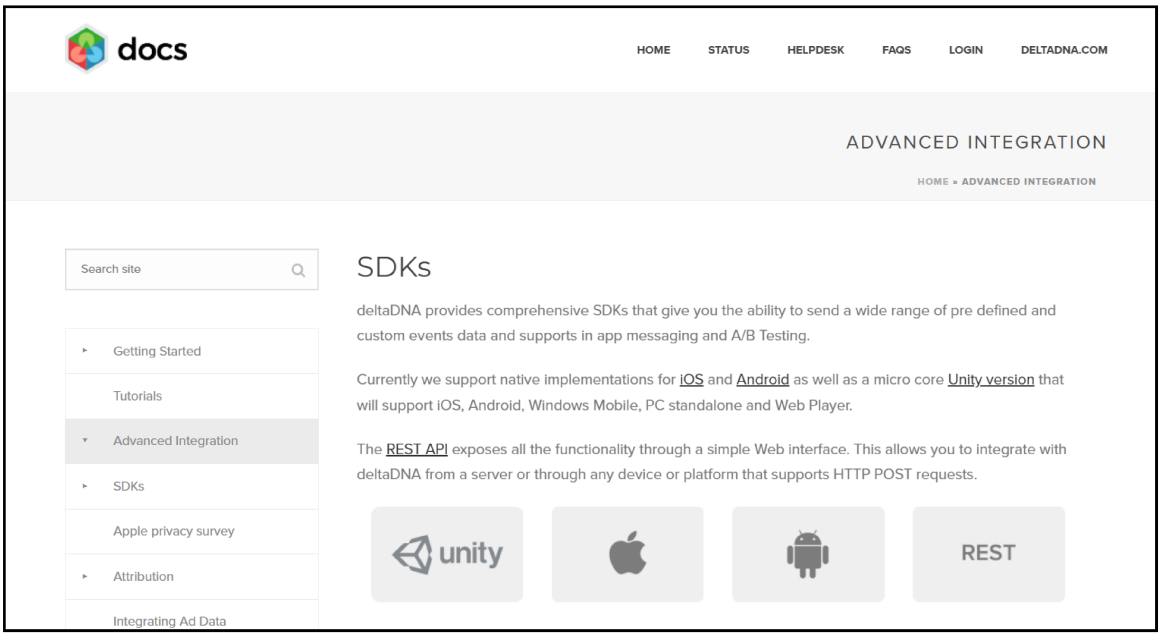

Figura 11 – Instruções e Download para o SDK do deltaDNA. Fonte: Documentação do deltaDNA, 2022

3.1.3 Análise

Ambas as ferramentas operam de forma muito similar, elas possuem um painel de controle que mostra informações sobre o fluxo de usuários e de seus eventos, assim como dados detalhados sobre faturamento e retenção. Ambas também oferecem bastante controle sobre o registro dos eventos e análise com filtros e funis, porém é notável que o deltaDNA oferece mais opções e detalhes sobre as ações do jogador. Entretanto, o Firebase Analytics é uma ferramenta gratuita enquanto o deltaDNA é uma ferramenta paga que cobra por usuário ativo no jogo (o que pode escalar para grandes valores em projetos maiores). Outro ponto é que somente o Firebase Analytics facilita em muito a integração dos dados de Microtransactions e Advertising para jogos feitos para a plataforma Android hospedados na Google Play Store e que fazem uso de anúncios vindo da provedora Google AdMob, uma vez que tanto a Google Play Store e Google AdMob são serviços da própria Google.

Tabela 1 – Tabela comparativa das implementações de Analytics

	Firebase Analytics	**DeltaDNA**
Custo	Gratuito para jogos menores, e Pago com custo baseado e demanda para jogos maiores	Pago com custo baseado em demanda
Compatibilidade	Unity, Apps. nativos de Android e iOS e Apps. nativos feitos em C++	Unity e Apps. nativos de Android e iOS
Sinergia com outros serviços da Google	Sim	Não
Painel de controle com mais opções	Não	Sim

3.2 Análise de Crashes

Este é um serviço que coleta informações sobre erros e falhas críticas. Seu principal uso refere-se no diagnóstico de quaisquer falhas que ocorram durante a execução do jogo no dispositivo final, tudo com a finalidade de corrigir erros que podem ocorrer em dispositivos específicos que a equipe de desenvolvimento responsável pelo jogo não tem acesso ou que ocorrem muito raramente.

Assim como no serviço de Analytics, todas as informações coletadas por este serviço são enviadas a um servidor e por fim disponibilizadas em um painel de controle ou console. Tais informações são subsequentemente organizadas e preparadas para análise pela própria ferramenta. Por conta disto, este serviço requer que o dispositivo no qual o jogo está sendo jogado esteja conectado com a internet para que as informações sejam enviadas. Nas ocasiões nas quais o dispositivo esteja desconectado da internet, é comum que os dados sejam reservados para envio posterior ao servidor automaticamente pela própria ferramenta, assim que o aparelho retornar a conexão à internet.

As principais informações coletadas por este serviço são a mensagem de erro que o serviço foi capaz de coletar, assim como toda a pilha de chamada de função (stack trace) relacionada ao erro para facilitar encontrar qual componente do código do jogo foi responsável pelo erro, e informações sobre o dispositivo em que o erro ocorreu, como especificações de hardware, o modelo do dispositivo no caso de aparelhos móveis e consoles de videogame, a versão do sistema operacional e a versão do jogo instalado.

Neste trabalho foram consideradas duas diferentes implementações da Análise de Crashes para serem utilizadas em um jogo comercial, o Firebase Crashlytics (FIREBASE, 2022a) e o Bugsnag (BUGSNAG, 2022). O levantamento acerca da Análise de Crashes se limitou a estas duas implementação pois elas são as principais implementações discutidas e recomendadas nos fóruns de desenvolvimento de jogos, é válido ressaltar que há outras implementações de Análise de Crashes que não foram consideradas para este trabalho como o Instabug, Raygun e o UXCam.

3.2.1 Firebase Crashlytics

Firebase Crashlytics é um serviço de Análise de Crashes disponibilizado gratuitamente pela Google compatível com jogos produzidos na Unity, com aplicativos nativos construídos para Android e iOS e com aplicativos construídos utilizando C++.

Para uso do Firebase Crashlytics, primeiro é necessário criar um projeto no site do Google Firebase referente ao jogo como é realizado na Figura 5.

Com o projeto criado, é necessário realizar o download e a instalação do SDK do Google Firebase Crashlytics no diretório do jogo como mostrado na Figura 12. Este processo é diferente para cada uma das plataformas citadas na introdução da sessão (no caso deste trabalho, é mostrado o processo para adicionar o Firebase Analytics na Unity), caso seja necessário, a documentação sobre este processo pode ser encontrada em (FIREBASE, 2022a). Uma implementação do Firebase Crashlytics em um jogo comercial é apresentada no Capítulo 4.

Figura 12 – Instruções e Download para o SDK do Firebase Crashlytics. Fonte: Documentação do Firebase Crashlytics, 2022

3.2.2 Bugsnag

Bugsnag é um serviço de Análise de Crashes disponibilizado de forma gratuita para projetos menores e com um plano pago de custo variável para projetos maiores e mais complexos, compatível com jogos produzidos na Unity, com aplicativos nativos construídos para Android e iOS e com aplicativos construídos utilizando C++. Este serviço possui três planos pago com custo variável dependendo das demandas do jogo, estes planos são necessários para jogos de grande escala, pois algumas funcionalidades do serviço são limitadas e que podem ser insuficientes se o número de jogadores for maior que o suportado.

Para uso do Bugsnag, primeiro é necessário criar uma conta no site do Bugsnag como é feito na Figura 13.

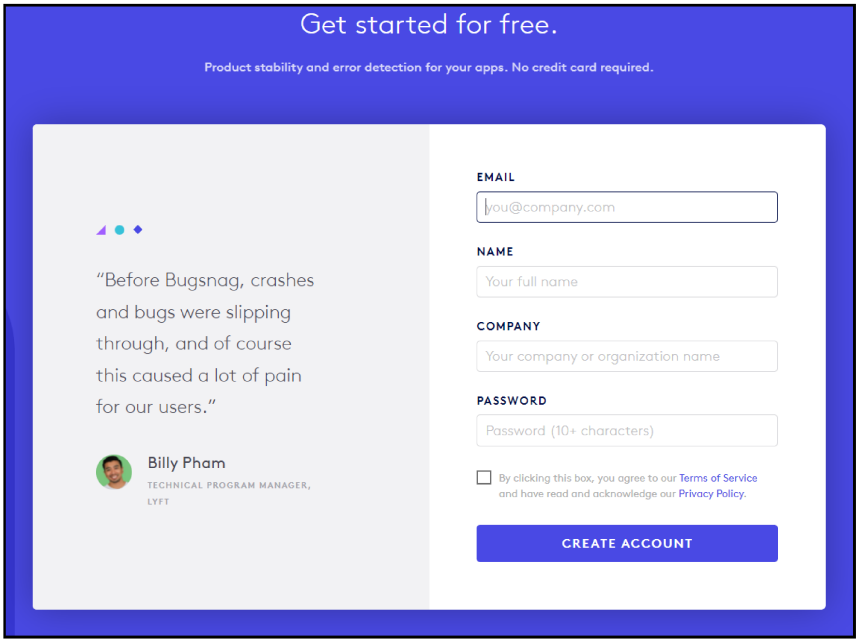

Figura 13 – Tela de criação de conta do Bugsnag. Fonte: Bugsnag, 2022

Feito isto, é necessário adicionar o jogo na conta que acabou de ser criada como é feito na Figura 14.

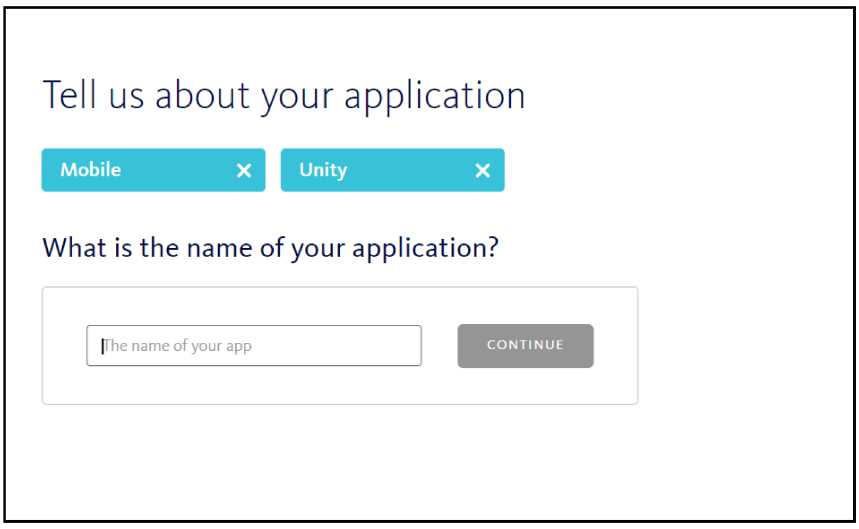

Figura 14 – Tela de criação do projeto do jogo no Bugsnag. Fonte: Bugsnag, 2022

Após isto, é necessário realizar o download e a instalação do SDK do Bugsnag no diretório do jogo como mostrado na Figura 15. Este processo é diferente para cada uma das plataformas citadas na introdução da sessão, caso seja necessário, a documentação sobre este processo pode ser encontrada em (BUGSNAG, 2022).

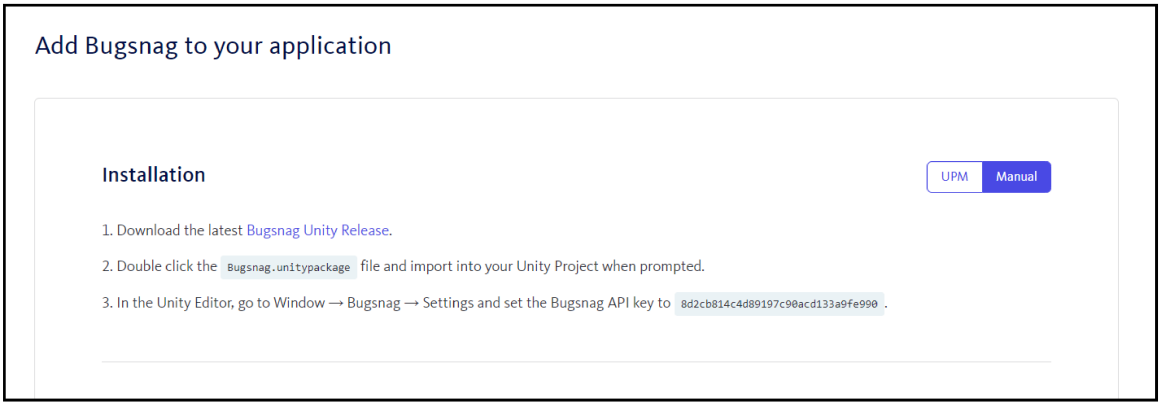

Figura 15 – Instruções e Download para o SDK do Bugsnag. Fonte: Documentação do Bugsnag, 2022

3.2.3 Análise

Ambas as ferramentas operam de forma muito similar. Elas possuem um painel de controle que mostra informações sobre os diferentes erros coletados recentemente e oferecem opções para filtrar os erros com base no tipo, no modelo do dispositivo, na versão do sistema operacional e versão do jogo. Entretanto, é notável que o Bugsnag oferece muito mais opções e detalhes sobre os erros e falhas. Porém, o Firebase Crashlytics é uma ferramenta totalmente gratuita enquanto o Bugsnag é uma ferramenta paga que cobra com base na demanda (o que pode escalar para grandes valores em projetos maiores).

Tabela 2 – Tabela comparativa das implementações de Análise de Crashes

	Firebase Crashlytics	**Bugsnag**
Custo	Gratuito	Gratuito para jogos menores, e Pago com custo baseado e demanda para jogos maiores
Compatibilidade	Unity, Apps. nativos de Android e iOS e Apps. nativos feitos em C++	Unity, Apps. nativos de Android e iOS e Apps. nativos feitos em C++
Painel de controle com mais opções	Não	Sim

3.3 Localização

Esta é uma ferramenta para modificar elementos textuais e visuais com base na idioma nativo do dispositivo do usuário ou com base nas preferências deste. Seu principal uso refere-se na tradução de textos e de imagens que contém textos para vários idiomas, tudo com a finalidade de melhorar a internacionalização do jogo, permitindo que ele seja apreciado por pessoas de diversos países principalmente daqueles em que a maioria da população não faz uso da língua inglesa, que é a língua padrão utilizada nos jogos lançados recentemente.

Todos os textos e imagens preparados para serem utilizadas nas línguas suportadas ficam salvas em tabelas dentro do jogo que podem ser acessadas durante a execução ao realizar uma chamada assíncrona passando como parâmetro o nome da tabela e a chave do elemento textual ou visual que deseja acessar. É normal que algumas destas ferramentas possuam formas de sincronizar as tabelas internas do jogo com tabelas externas salvas em outras aplicações, como Google Spreadsheets e softwares de banco de dados, como por exemplo MySQL e Oracle, além disso algumas destas ferramentas permitem a importação de arquivos de extensão CSV que são arquivos comumente utilizados por softwares de banco de dados e manejamento de tabelas para exportação de tabelas.

Neste trabalho foram consideradas duas diferentes implementações de Localização para serem utilizadas em um jogo comercial, o Unity Localization (UNITY, 2022c) e o Lean Localization (WILKES, 2022). O levantamento acerca da Localização se limitou a estas duas implementação pois elas são as principais implementações discutidas e recomendadas nos fóruns de desenvolvimento de jogos voltados para o motor de jogos Unity, é válido ressaltar que há outras implementações de Localização que não foram consideradas para este trabalho como o I2 Localization (INTERILLUSION, 2022) e o SmartCAT (SMARTCAT, 2022).

3.3.1 Unity Localization

Unity Localization é uma ferramenta de localização disponibilizado gratuitamente pela própria Unity, compatível somente com jogos produzidos no motor de jogos Unity.

Para uso da Unity Localization, primeiro é necessário realizar a importação do pacote da Unity Localization no Package Manager da Unity, para mais informações sobre esse processo que pode vir a mudar em versões futuras da Unity, acesse (UNITY, 2022a).

Realizado a instalação, deve-se criar o arquivo de configuração principal da Unity Localization, as tabelas de localização que serão necessárias e os idiomas que serão alvos da localização (deve ser adicionado todos os idiomas, inclusive o idioma atual que se encontra o jogo) como é feito nas Figuras 16, 17 e 18. Uma implementação do Unity Localization em um jogo comercial é apresentada no Capítulo 4.

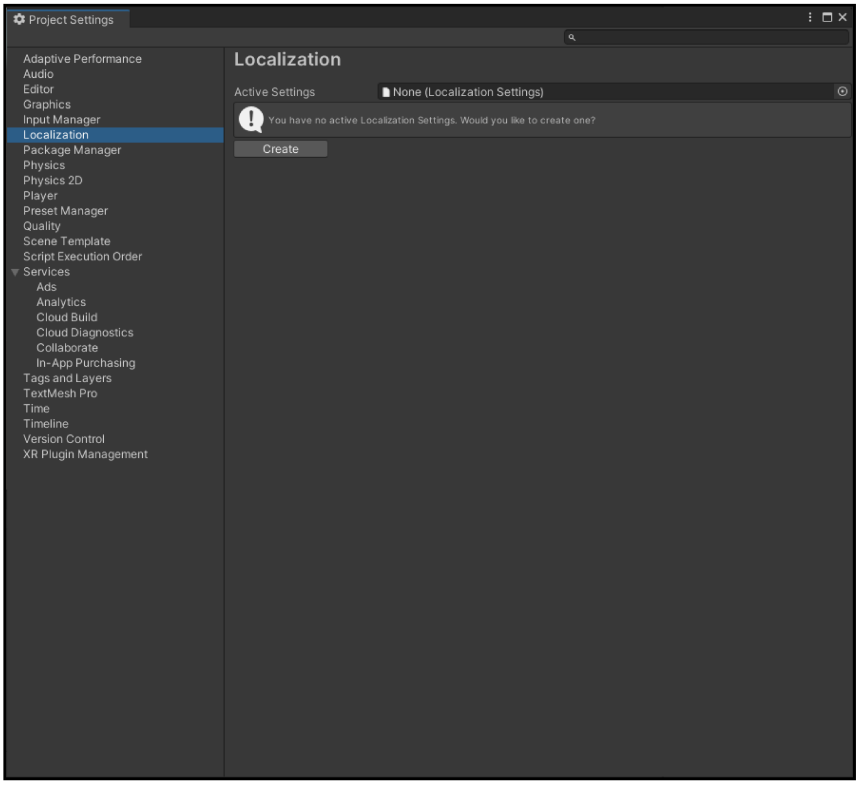

Figura 16 – Criação do Arquivo de Configuração da Unity Localization. Fonte: Documentação da Unity, 2020

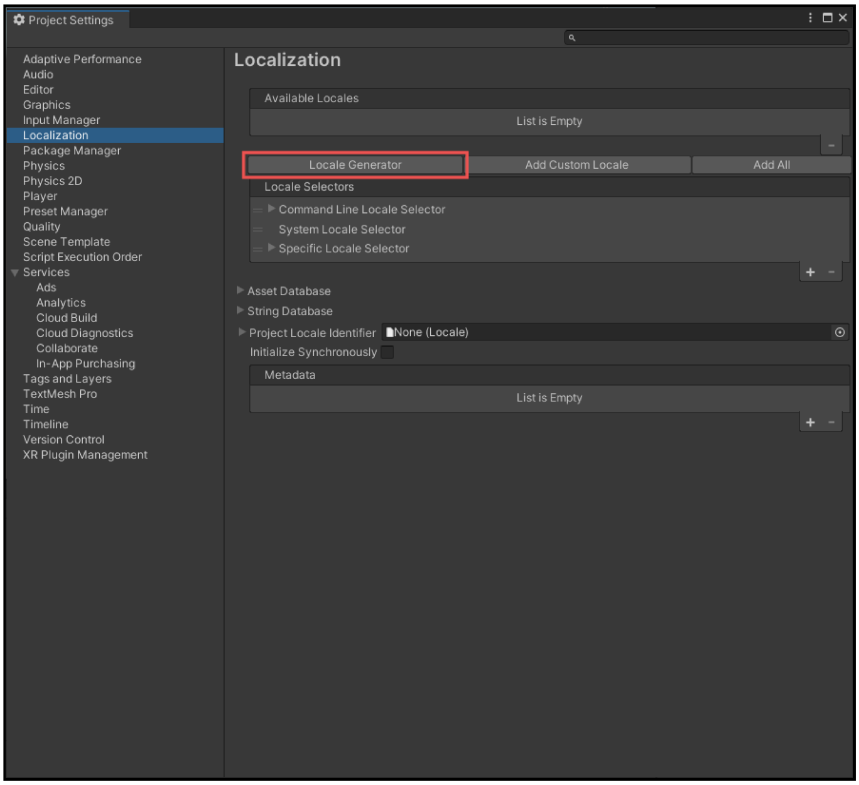

Figura 17 – Geração das Localizações da Unity Localization. Fonte: Documentação da Unity, 2022

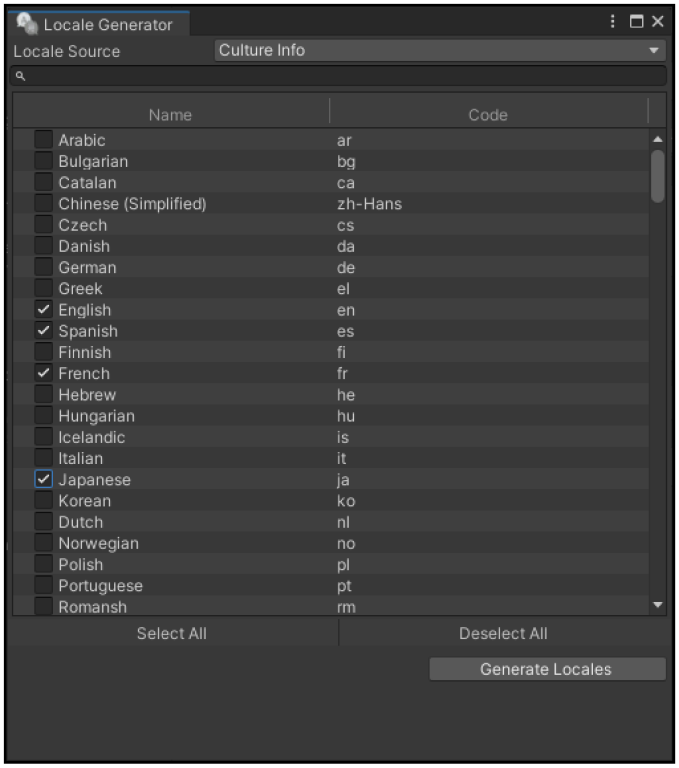

Figura 18 – Seleção das Localizações da Unity Localization. Fonte: Documentação da Unity, 2022

Com estes arquivos criados, basta realizar a configuração do arquivo principal de configuração como é feito na Figura 19, note que é necessário deixar indicado qual o idioma que será utilizado caso o sistema seletor de idiomas falhe em encontrar a língua nativa do dispositivo ou caso a língua nativa do dispositivo não seja um dos idiomas suportados pelo jogo.

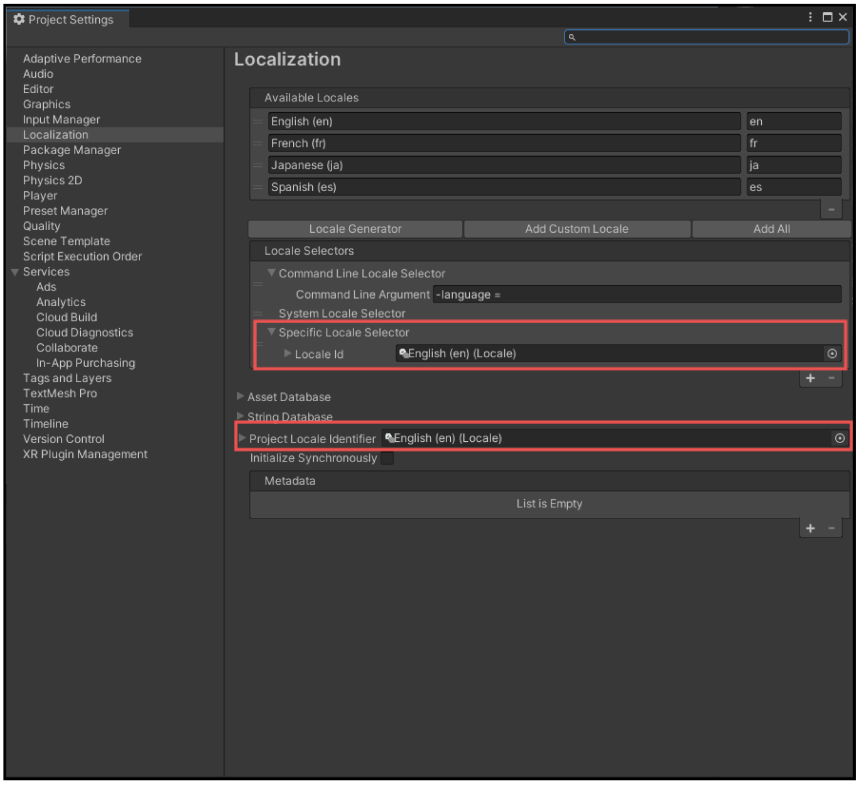

Figura 19 – Configuração da Unity Localization. Fonte: Documentação da Unity, 2022

Realizado a configuração, deve-se preencher as tabelas criadas com as chaves relevantes do jogo e adicionar os textos e imagens que serão implementadas em cada um dos idiomas suportados. Para mais informações sobre como fazer uso da ferramenta, acesse (UNITY, 2022c).

3.3.2 Lean Localization

Lean Localization é uma ferramenta de localização criado pro Carlos Wilkes e disponibilizado gratuitamente na loja Unity Asset Store, compatível somente com jogos produzidos no motor de jogos Unity.

Para fazer uso do Lean Localization, primeiro é necessário realizar a instalação do pacto pelo Package Manager da Unity.

Realizado a instalação, basta adicionar um GameObject com o componente Lean-Localization na primeira cena do jogo e a configurar as línguas que deseja ser suportadas como é feito nas Figuras 20 e 21.

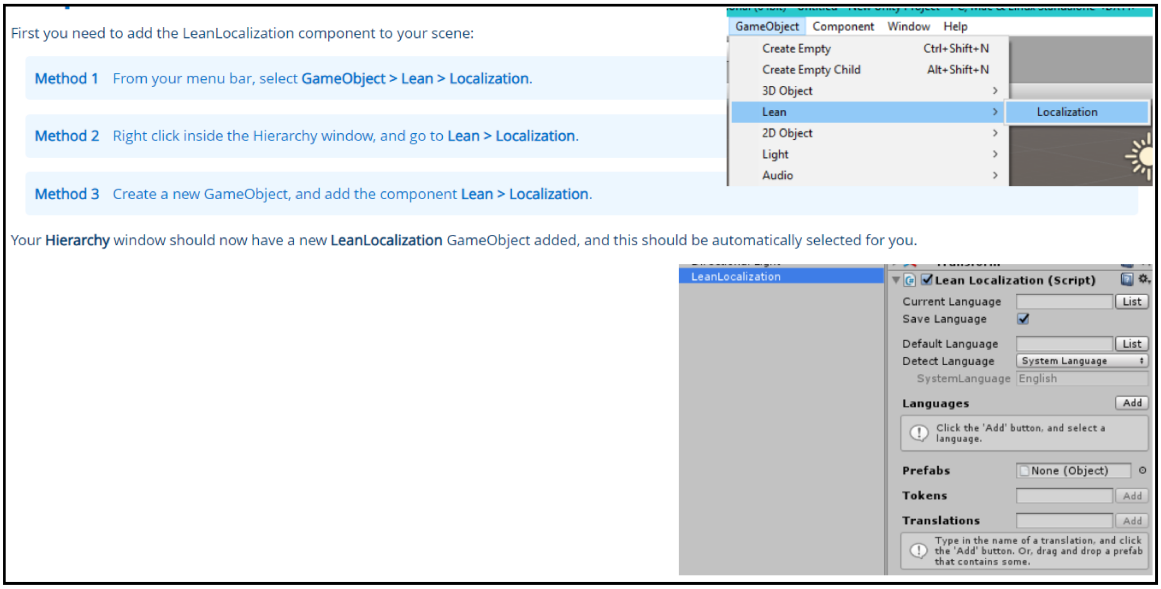

Figura 20 – Configuração do Lean Localization. Fonte: Documentação do Lean Localization, 2022

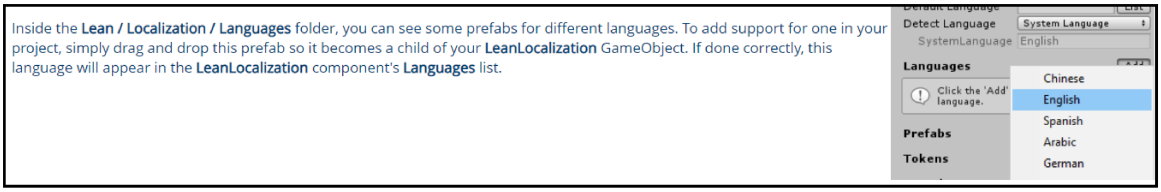

Figura 21 – Configuração do Lean Localization. Fonte: Documentação do Lean Localization, 2022

Realizado a configuração, deve-se preencher a configuração criada com as chaves relevantes do jogo e adicionar os textos e imagens que serão implementadas em cada um dos idiomas suportados. Para mais informações sobre como fazer uso da ferramenta, acesse (WILKES, 2022).

3.3.3 Análise

As ferramentas operam de formas distintas, a Unity Localization faz o armazenamento das localizações por meio de diferentes tabelas e o Lean Localization faz o armazenamento em uma única tabela, o Lean Localization exige que exista múltiplos objetos instanciados na cena para realizar o armazenamento das localizações e funcionamento da ferramenta e a Unity Localization não exige que exista nenhum objeto instanciado para o seu funcionamento (tudo pode ser operado por código se for necessário). Sobre a sincronização das localizações internas do jogo com localizações externas, a Unity Localization oferece uma função mais rigida porém pronta para uso realizando a sincronização com tabelas do Google Spreadsheets enquanto o Lean Localization oferece uma função mais

livre porém que exige trabalho extra que é através de importação de arquivos CSV, o que permite o Lean Localization receber localizações armazenadas da maioria dos softwares e aplicações web de banco de dados.

Tabela 3 – Tabela comparativa das implementações de Localização

	Unity Localization	**Lean Localization**
Custo	Gratuito	Gratuito
Compatibilidade	Unity	Unity
É compatível com mais de uma tabela	Sim	Não
Forma de realizar a importação e a exportação de tabelas	Através do Google Spreadsheets	Através de arquivos de extensão .csv

3.4 Notificações Locais e Remotas

Esta ferramenta é implementada somente em aparelhos móveis e são chamadas com a finalidade de aumentar a retenção dos jogadores. Seu principal uso refere-se em chamar a atenção do usuário para diversas coisas, pode ser utilizado para avisar que uma nova atualização está disponível, que algum bonus está disponível se o jogador abrir o jogo e também pode ser utilizado para incentivar aquele jogador que está a bastante tempo sem jogar.

As Notificações devem ser chamadas para serem mostradas, no caso das Notificações locais estas são programadas no jogo previamente para serem mostradas após um tempo determinado e podem ser programadas para repetirem com determinada frequência definida pelo desenvolvedor. No caso das Notificações remotas estas são enviadas em tempo real através de um painel de controle para todos os dispositivos que tem o jogo instalado e que estejam conectados à internet. Nas ocasiões nas quais o dispositivo esteja desconectado da internet, é comum que o servidor aguarde para enviar a notificação programada assim que o aparelho retornar a conexão à internet.

Neste trabalho foram consideradas duas diferentes implementações de Notificações para serem utilizadas em um jogo comercial, o Easy Mobile Pro (SGLIB, 2022a) e o One Signal (ONESIGNAL, 2022). O levantamento acerca das Notificações se limitou a estas duas implementação pois elas são as principais implementações discutidas e recomendadas nos fóruns de desenvolvimento de jogos voltados para aparelhos móveis, é válido ressaltar que há outras implementações de Localização que não foram consideradas para este trabalho como o Firebase Cloud Messaging (FIREBASE, 2022d) e o Pushwoosh (PUSHWOOSH, 2022).

3.4.1 Easy Mobile Pro

O Easy Mobile Pro é uma ferramenta para Notificações Locais e Remotas paga disponibilizada na Unity Asset Store e é compatível com jogos para aparelhos móveis, criados no motor de jogos Unity. Válido ressaltar que é comum que esta ferramenta receba um desconto temporariamente de forma sazonal de no mínimo 50%.

Para fazer uso do Easy Mobile Pro, primeiro é necessário realizar o download e instalação do pacote pelo Package Manager.

Realizado a instalação, deve-se habilitar a auto-inicialização do pacote (como feito na Figura 22) e caso seja necessário notificações locais, realizar o agendamento destas por código. Caso seja necessário mais informações, acesse (SGLIB, 2022a). Uma implementação do Easy Mobile Pro em um jogo comercial é apresentada no Capítulo 4.

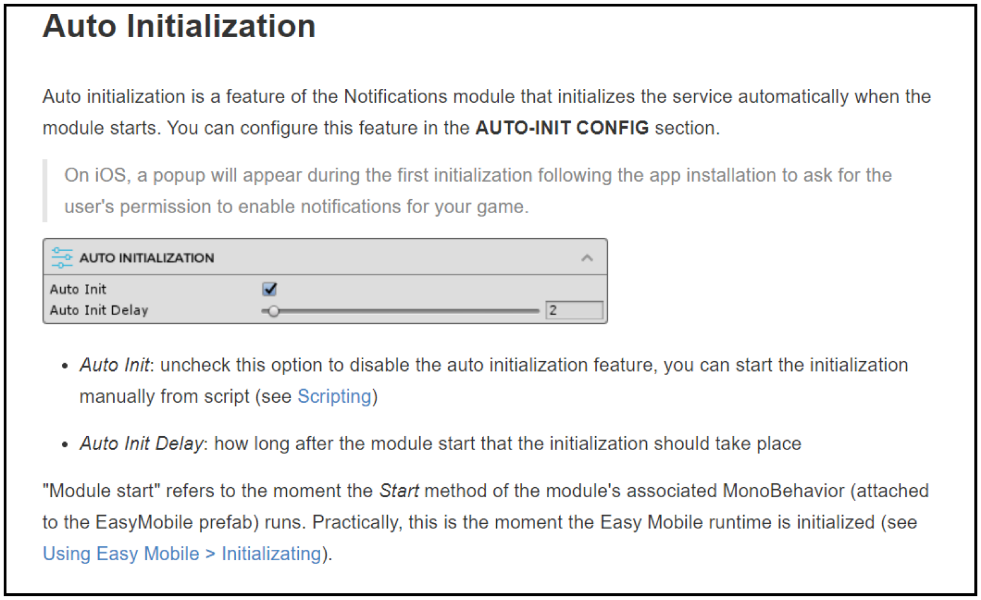

Figura 22 – Configuração de auto-inicialização das Notificações Easy Mobile Pro. Fonte: Documentação do Easy Mobile Pro, 2022

3.4.2 One Signal

O One Signal é um serviço para Notificações Remotas com planos gratuitos para menores demandas e pagos para maiores demandas e é compatível com aplicativos nativos para Android e iOS, para jogos produzidos na Unity, para aplicações web e para muitos outros tipos de aplicações.

Para fazer uso do One Signal é necessário criar uma conta no site do One Signal e criar um projeto referente ao jogo e realizar as configurações indicadas pelo site como feito nas Figuras 23 e 24.

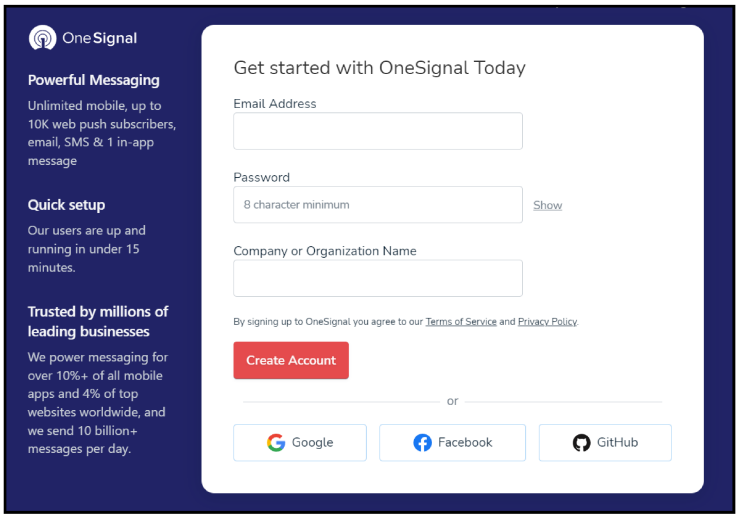

Figura 23 – Criação de conta do One Signal. Fonte: One Signal, 2022

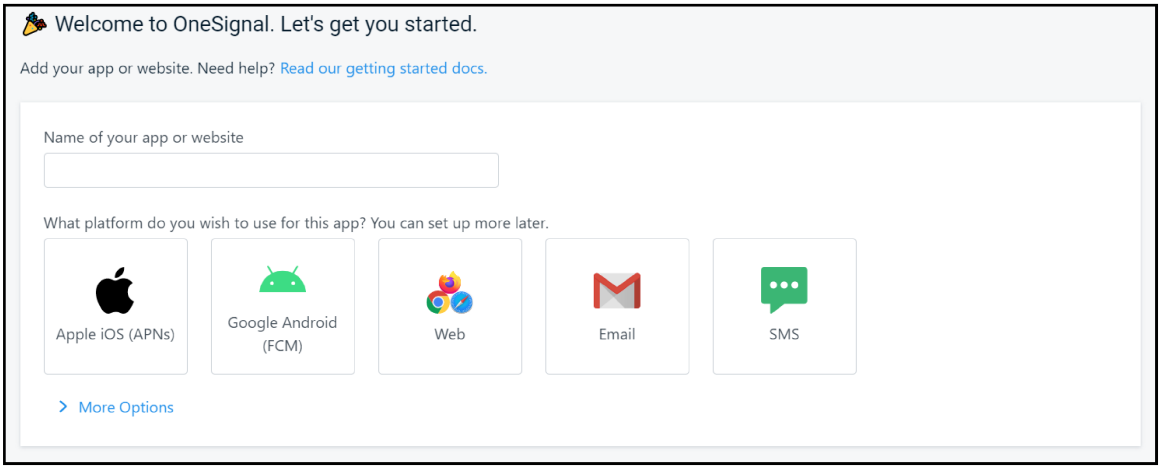

Figura 24 – Criação do projeto do jogo no One Signal. Fonte: One Signal, 2022

Realizado a criação da conta e do projeto, deve-se realizar o download e instalação do SDK do One Signal (como feito na Figura 25) e o jogo estará pronto para receber notificações que o desenvolvedor achar necessário pelo painel de controle do One Signal. Caso seja necessário mais informações sobre o funcionamento do One Signal, acesse (ONESIGNAL, 2022).

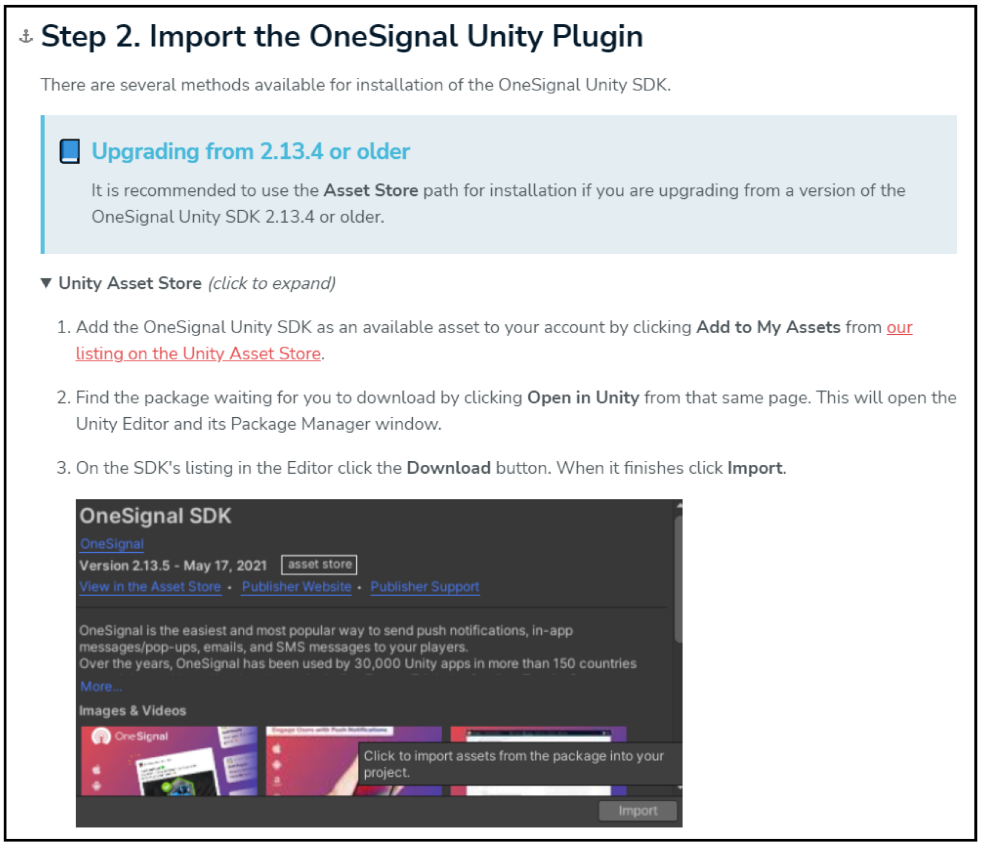

Figura 25 – Instruções e Download para o SDK do One Signal. Fonte: Documentação do One Signal, 2022

3.4.3 Análise

Ambas as ferramentas operam de formas diferentes, o Easy Mobile Pro possui suporte para ambas notificações locais e remotas enquanto o One Signal suporta apenas notificações remotas, o Easy Mobile Pro por mais que suporte notificações remotas é necessário que ele se conecte com um serviço de notificações remotas, como o Firebase ou o próprio One Signal, para poder exercer esta função. O Easy Mobile Pro acima de tudo implementa diversas outras ferramentas em seu pacote que normalmente são necessárias para o desenvolvimento de jogos, tornando-o mais completo quando comparado ao One Signal.

Tabela 4 – Tabela comparativa das implementações de Notificações locais e remotas

	Easy Mobile Pro	**OneSignal**
Custo	Pago para aquisição	Pago com custo baseado em demanda
Compatibilidade	Unity	Unity, Apps. nativos de Android e iOS, Apps. Web, entre outros
Suporte à notificações locais	Suporte completo	Não oferece suporte
Suporte à notificações remotas	Suporte inicial que requer uso de serviço externo para realizar o disparo de notificações	Suporte completo

3.5 Microtransactions

Este é um serviço de monetização que permite a implementação de compras dentro de um jogo comercial com dinheiro real a partir de uma interface separada que cuida de toda a transação. Seu principal uso refere-se na monetização de jogos comerciais, principalmente daqueles que são de caráter gratuito e que depende de outros meios para rentabilizar o jogo produzido.

O Processo da compra deve ser chamada por código, por exemplo quando algum botão da interface é pressionado, feito isto uma interface de transação da loja aparece na tela requisitando do usuário que insira um forma de pagamento e então realize a transação. Após o usuário decidir comprar ou não, a interface da loja é fechada e um dos callbacks, um pedaço de código executável que é passado como argumento para outro código para que possa ser executado um um momento oportuno, implementados para receber resultado da transação é chamado e no caso da compra for bem sucedida é necessário entregar o produto comprado ao jogador.

Este serviço é extremamente dependente da plataforma ao qual pertence o jogo, as ferramentas e o fluxo de funcionamento deste serviço varia bastante entre as plataformas, por isso foram considerados uma ferramenta para o tratamento de Microtransactions para jogos produzido para as plataformas Android e iOS, O Easy Mobile Pro (SGLIB, 2022c), e outra ferramenta para o tratamento de Microtransactions em jogos hospedados na loja de jogos para computador Steam, o Steamworks (VALVE, 2022b), que é atualmente a maior plataforma de jogos de computador. É valido ressaltar que existe muitas outras ferramentas que não foram citadas neste trabalho como o Unity IAP (UNITY, 2022e).

3.5.1 Easy Mobile Pro

O Easy Mobile Pro é uma ferramenta para Microtransactions paga disponibilizada na Unity Asset Store e é compatível com jogos para aparelhos móveis, criados no motor de jogos Unity. Válido ressaltar que é comum que esta ferramenta receba um desconto temporariamente de forma sazonal de no mínimo 50%.

Para fazer uso do Easy Mobile Pro, primeiro é necessário realizar o download e instalação do pacote pelo Package Manager.

Realizado a instalação, deve-se habilitar a auto-inicialização do pacote (como feito na Figura 22) e preparar o jogo para que este possa realizar chamadas de transações e que os seus callbacks sejam retornados apropriadamente. Caso seja necessário mais informações, acesse (SGLIB, 2022c). Uma implementação do Easy Mobile Pro em um jogo comercial é apresentada no Capítulo 4.

3.5.2 Steamworks

O Steamworks é uam ferramenta para Microtransactions gratuita disponibilizada pelo loja de jogos Steam para dar suporte a diversas funcionalidades necessárias para lançar um jogo nesta loja.

Para fazer uso do Steamworks, primeiro é necessário realizar a criação de conta e o cadastramento como um parceiro Steamworks, isto exige dados bancários, tributários e da empresa para ser realizado. Após a criação e o cadastro no plano de parceiros for concluídos, deve-se realizar o download e instalação do SDK do Steamworks no diretório do jogo como é mostrado na Figura 26. Caso mais informações sejam necessárias, acesse (VALVE, 2022b) para informações sobre a instalação e criação de contas e acesse (VALVE, 2022a) para informações sobre a implementação do serviço de Microtransactions.

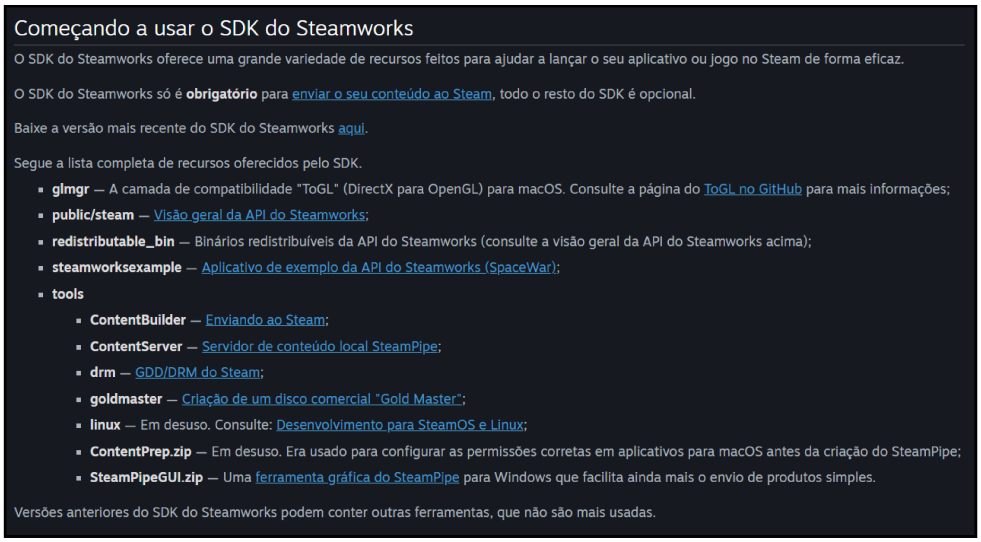

Figura 26 – Instruções e Download para o SDK do Steamworks. Fonte: Documentação do Steamworks, 2022

3.5.3 Análise

Ambas as ferramentas implementam o fluxo de interface de transação de formas parecidas, uma chamada é realizada e a transação é mostrada na tela do jogador sobre o jogo, com o Easy Mobile Pro fazendo uso das interfaces de compra da Google Play Store para jogos produzidos para aparelhos Androids e da App Store para jogos produzidos para aparelhos Apple e o Steamworks fazendo uso da interface da loja Steam. Porém o tratamento do resultado da compra é realizado de forma diferente entre eles, enquanto o Easy Mobile Pro faz uso de callbacks passados como argumento na chamada da transação, o Steamworks faz uso de uma variável de estado para informar qual o estado atual da transação que está acontecendo. Fora esta diferença, ambas as ferramentas se mostram bastante eficientes no que propõe fazer.

Tabela 5 – Tabela comparativa das implementações de Microtransactions

	Easy Mobile Pro	**Steamworks**
Custo	Pago para aquisição	Gratuito
Compatibilidade de plataforma	Jogos para celular feitos no Unity	Jogos para computador
Lojas suportadas	Google Play Store e App Store	Steam
Forma de notificar ao jogo que a transação foi concluída	Através de callbacks	Através da mudança de uma variável pública

3.6 Advertising

Este é um serviço de monetização que permite a implementação de anúncios dentro de um jogo comercial a partir de interfaces pré-produzidas que são posicionadas sobre o jogo. Seu principal uso refere-se na monetização de jogos comerciais, principalmente daqueles que são de caráter gratuito e que dependem de outros meios para rentabilizar o jogo produzido.

Este serviço depende de provedoras de anúncio para poder ter anúncios para mostrar, cada implementação do Advertising deve possuir suporte para grande parte das provedoras de anúncios. Para funcionamento correto, é necessário que o desenvolvedor faça a configuração da provedora escolhida no seu painel de configuração da implementação do Advertising utilizada.

Existem 3 diferentes tipos de anúncios atualmente: O anúncio intersticial, o anúncio com recompensa e o anúncio em Banner. O anúncio Intersticial é um anúncio que é mostrado sem o consentimento do usuário, que cobre toda a tela e que pode ser pulado após um pequeno determinado período de tempo, normalmente 5 segundos. O anúncio com recompensa é um anúncio que é mostrado com o consentimento do jogador, deve ser avisado na interface do jogo que um anúncio com recompensa será mostrado e o usuário deve concordar em assisti-lo, que concede uma recompensa a este ao final da duração do anúncio, o usuário pode optar por pular o anúncio antes do final da sua duração mas fazendo com que o usuário não receba a recompensa que foi oferecida antes do anúncio ser mostrado. E por último, o anúncio em Banner é um anúncio de caráter permanente que cobre uma parcela da tela do jogo para mostrar um anúncio, animado ou não, o jogo normalmente pode ser jogador enquanto este anúncio está sendo mostrado, ao contrário dos outros anúncios anteriormente citados.

Para implementação, cada um dos tipos de anúncio citados segue uma forma diferente de funcionamento. O anúncio Intersticial e o anúncio com recompensa devem ser carregados anteriormente, um processo que normalmente é automático, e então devem ser chamados quando o desenvolvedor achar oportuno, lembrando que o anúncio com recompensa exige que o usuário seja avisado sobre o anúncio e sua recompensa antes de ser chamado, essa chamada normalmente recebe como argumentos callbacks para indicar o

término do anúncio e no caso do anúncio com recompensa há 2 diferentes callbacks, um indicando que o usuário assistiu o anúncio e deve receber a recompensa e outro indicando que o usuário pulou o anúncio.

Todas as informações sobre os anúncios e a receita geradas são pertinentes ao painel de controle da provedora de anúncios escolhida, caso esteja fazendo uso da provedora Google AdMob e o serviço de Analytics Google Firebase, a coleta destes dados para o painel de controle do Google Firebase se torna automático, possibilitando a unificação destes dados de anúncios e de Analytics em um único painel.

Neste trabalho foram consideradas duas diferentes implementações de Advertising para serem utilizadas em um jogo comercial, o Easy Mobile Pro (SGLIB, 2022b) e o Unity Ads (UNITY, 2022b). O levantamento acerca das ferramentas de Advertising se limitou a estas duas implementação pois elas são as principais implementações discutidas e recomendadas nos fóruns de desenvolvimento de jogos voltados para aparelhos móveis, é válido ressaltar que há outras implementações de Localização que não foram consideradas para este trabalho como o IronSource (IRONSOURCE, 2022) e o Mobile Ads (GLEY, 2022).

3.6.1 Easy Mobile Pro

O Easy Mobile Pro é uma ferramenta para Advertising paga disponibilizada na Unity Asset Store e é compatível com jogos para aparelhos móveis, criados no motor de jogos Unity. Válido ressaltar que é comum que esta ferramenta receba um desconto temporariamente de forma sazonal de no mínimo 50%.

Para fazer uso do Easy Mobile Pro, primeiro é necessário realizar o download e instalação do pacote pelo Package Manager.

Realizado a instalação, deve-se habilitar a auto-inicialização do pacote (como feito na Figura 22), realizar a configuração (e instalação do Plugin referent) da provedora de anúncios escolhida no jogo e preparar o jogo para que este possa realizar chamadas de anúncios e que os seus callbacks sejam retornados apropriadamente. Caso seja necessário mais informações, acesse (SGLIB, 2022b). Uma implementação do Easy Mobile Pro em um jogo comercial é apresentada no Capítulo 4.

3.6.2 Unity Ads

O Unity Ads é uma ferramenta para Advertising gratuita disponibilizada pela Unity e é compatível com jogos para aparelhos móveis criados no motor de jogos Unity.

Para fazer uso do Unity Ads, deve-se primeiro ativar o componente do Unity Ads dentro do projeto do jogo e o configurar (como mostrado na Figura 27), feito isto deve-se configurar o painel de controle da Unity Monetization no site da Unity, realizar a

configuração e instalação do plugin da provedora de anúncios escolhidas e preparar o jogo para que este possa realizar chamadas de anúncios e que os seus callbacks sejam retornados apropriadamente. Caso seja necessário mais informações, acesse (UNITY, 2022b).

Figura 27 – Ativação do componente de Ads da Unity. Fonte: Documentação da Unity, 2022

3.6.3 Análise

Ambas as ferramentas operam de formas muito similares, ambas possuem suporte para anúncios intersticias, anúncios com recompensa e anúncios em Banner, ambas as ferramentas possuem um fluxo de implementação parecidos: Primeiro acontece a inicialização da ferramenta, então ocorre o carregamento dos anúncios e a ferramenta fica pronta para receber chamadas de anúncios, ao receber uma chamada, o anúncio é mostrado e o resultado do processo é retornado em forma de callbacks. As diferenças notada é que o Unity Ads não possui a função de carregamento automático de anúncio que deve ser implementada, porém o Unity Ads é uma ferramenta gratuita que oferece praticamente todo que o Easy Mobile Pro, que é uma ferramenta paga, oferece quando o quesito é Advertising.

Tabela 6 – Tabela comparativa das implementações de Advertising

	Easy Mobile Pro	**Unity Ads**
Custo	Pago para aquisição	Gratuito
Compatibilidade	Unity	Unity
Suporte à anúncios intersticias	Sim	Sim
Suporte à anúncios com recompensa	Sim	Sim
Suporte à anúncios em banner	Sim	Sim
Opção para carregamento automático de anúncios	Sim	Não

3.7 Anti-Cheat

Estas ferramentas têm como função ocultar e mascarar dados críticos para o funcionamento do jogo. Seu principal uso refere-se na tentativa de impedir que usuários façam uso de trapaças que podem acabar com a própria experiência e com a experiência de outros jogadores e que façam uso de trapaças que podem impactar negativamente no modelo de monetização escolhido.

A principal forma que estas ferramentas utilizam para tentar impedir trapaças é a partir da criptografia dos dados ao serem guardados fazendo uso de algum tipo de criptografia escolhida. A criptografia pode ser aplicada tanto em dados guardados em memória volátil ou em memória permanente, e regularmente são implementados formas de se tentar detectar tentativas de modificação em tais dados.

Neste trabalho foram consideradas duas diferentes implementações de Anti-Cheat para serem utilizadas em um jogo comercial, o Anti-Cheat Toolkit (CODESTAGE, 2022) e o SCUE4 Anti-Cheat Solution (LEITE, 2022). O levantamento acerca das ferramentas de Anti-Cheat se limitou a estas duas implementação pois elas são as principais implementações discutidas e recomendadas nos fóruns de desenvolvimento de jogos, é válido ressaltar que há outras implementações de Localização que não foram consideradas para este trabalho como o Easy Anti-Cheat (EPICGAMES, 2022) e o Valve Anti-Cheat (VALVE, 2022c).

3.7.1 Anti-Cheat Toolkit

O Anti-Cheat Toolkit é uma ferramenta para Anti-Cheat paga disponibilizada na Unity Asset Store e é compatível com jogos criados no motor de jogos Unity. Válido ressaltar que é comum que esta ferramenta receba um desconto temporariamente de forma sazonal de no mínimo 50%.

Esta ferramenta oferece funções de criptografia para variáveis, para o relógio interno do jogo, para dados salvos no registro do dispositivo e para dados salvos em arquivo, além de oferecer detecção para uso de trapaças que não envolvem adulteração de dados como o “WallHack”.

Para fazer uso do Anti-Cheat Toolkit, primeiro é necessário realizar o download e instalação do pacote pelo Package Manager.

Realizado a instalação, deve-se implementar as alternativas para armazenamento de dados implementadas pela ferramenta no lugar das alternativas de armazenamento de dados que estão sendo utilizadas atualmente no jogo. As formas de armazenamento de dados implementados pela ferramenta automaticamente recebem o tratamento de criptografia durante uso. Caso seja necessário mais informações, acesse (CODESTAGE, 2022).

Uma implementação do Anti-Cheat Toolkit em um jogo comercial é apresentada no Capítulo 4.

3.7.2 SCUE4 Anti-Cheat Solution

O Anti-Cheat Toolkit é uma ferramenta para Anti-Cheat gratuita disponibilizada na Unreal Engine Marketplace e é compatível com jogos criados no motor de jogos Unreal.

Esta ferramenta oferece funções de criptografia para variáveis, para o relógio interno do jogo, para dados salvos no registro do dispositivo e para dados salvos em arquivo.

Para fazer uso do Anti-Cheat Toolkit, primeiro é necessário realizar o download e instalação do pacote pelo Unreal Engine Marketplace.

Realizado a instalação, deve-se implementar as alternativas para armazenamento de dados implementadas pela ferramenta no lugar das alternativas de armazenamento de dados que estão sendo utilizadas atualmente no jogo. As formas de armazenamento de dados implementados pela ferramenta automaticamente recebem o tratamento de criptografia durante uso. Caso seja necessário mais informações, acesse (LEITE, 2022).

3.7.3 Análise

Ambas as ferramentas são extremamente eficazes no que propõe fazer, oferecem criptografia para todas alternativas de armazenamento de dados implementadas pelo motor de jogos que suportam. As diferenças notadas é que o Anti-Cheat Toolkit também oferece algumas funções para detecção de outros tipos de dados enquanto o SCUE4 Anti-Cheat Solution se restringe a somente a criptografia de dados, porém o SCUE4 Anti-Cheat Solution é uma ferramenta gratuita que oferece praticamente tudo que o Anti-Cheat Toolkit, que é uma ferramenta paga, oferece quando o quesito é Anti-Cheat.

Tabela 7 – Tabela comparativa das implementações de Anti-Cheat

	Anti-Cheat Toolkit	**SCUE4 Anti-Cheat Solution**
Custo	Pago para aquisição	Gratuito
Compatibilidade	Unity	Unreal Engine
Criptografia de memória	Sim	Sim
Detecção de outros tipos de trapaça	Sim	Não

4 Estudo de caso de uso

Neko Dungeon é um jogo desenvolvido pela Gilp Studio do qual o autor deste trabalho faz parte. Este é um jogo desenvolvido para dispositivos móveis feito no motor de jogos digitais Unity. Ele pertence aos gêneros RPG (“Role-playing Game”), aventura e quebra-cabeça. É ambientado na temática de gatos personificados que lutam contra monstros para salvar o reino dos gatos. Um breve panorama do visual do jogo é exposto nas Figuras 28.

Figura 28 – A esquerda, Tela de menu do jogo Neko Dungeon e, a direita, Tela de batalha do jogo Neko Dungeon. Fonte: Neko Dungeon, 2022

Este jogo faz uso de todas as ferramentas e serviços anteriormente discutidas no Capítulo 3 e estas foram implementadas até certo ponto, já que aprimoram a usabilidade do jogo e também são essenciais para o funcionamento correto.

As motivações para a implementação de cada ferramenta e serviço foram as seguintes: O **Analytics** foi implementado com a análise de comportamento dos usuários

em mente, já a **Análise de Crashes** foi implementada para diagnosticar e ajudar resolver certas falhas críticas que podiam ocorrer durante a execução em certos modelos de dispositivos os quais os desenvolvedores não tinham acesso ou para ajudar a resolver falhas críticas que não retornavam uma mensagem de erro. A **Localização** foi implementada com a internacionalização do jogo em mente pois era necessário localizar todos os textos do jogo para isso, as **Notificações Locais** foram implementadas para aumento de retenção dos jogadores, de forma a chamar a atenção deles para quando alguma coisa se tornasse disponível ou se eles não interagem com o jogo após um determinado período de tempo. Além disso, as **Microtransactions** foram implementadas para a possibilitar compras dentro do jogo, o **Advertising** foi implementado para mostrar propagandas dentro do jogo e a **Anti-Cheat** foi implementada para dificultar o uso de trapaças dentro do jogo (através de edição de chaves de registro e modificação da memória volátil do dispositivo).

A seguir será descrito como foi realizado a implementação de cada uma dessas ferramentas e serviços no jogo Neko Dungeon.

4.1 Analytics

O Analytics foi utilizado no Neko Dungeon devido a necessidade de analisar o comportamento do usuário para que a equipe possa alterar o jogo com a melhora da experiência do jogador como foco, por exemplo, com esta ferramenta foi possível descobrir quais fases estavam fáceis ou difíceis demais quando comparados com o que o Designer responsável pelo projeto planejou.

Este serviço foi implementado utilizando a implementação do Google Firebase Analytics, a escolha desse serviço deve-se a confiança na sua eficiência, na preferência pela forma como gera os relatórios de dados e que além de ser gratuita ainda obedece à LGPD. A implementação ocorreu da seguinte forma:

Primeiro foi criado e configurado o projeto do jogo dentro da painel de controle do Google Firebase e então adicionado como um App o jogo e seu id, veja Figura 5.

Depois disso, foi realizado o download do arquivo de configuração e do SDK do Firebase Analytics e então foram propriamente adicionados ao projeto do jogo na Unity, veja as Figuras 6 e 7.

E com isso feito, o projeto está quase pronto para coletar dados acerca do comportamento dos usuários, algumas informações já podem ser coletadas, no entanto desenvolvimento adicional é necessário para realizar o disparo de eventos. Primeiro, é necessário realizar a inicialização do serviço do Google Firebase e também do Firebase Analytics ao se iniciar o jogo como é feito nas Figuras 29 e 30.

```
private void Init()
{
    if(!initted)
    {
        initted = true;

        Firebase.FirebaseApp.CheckAndFixDependenciesAsync().ContinueWith(task => {
            var dependencyStatus = task.Result;
            if (dependencyStatus == Firebase.DependencyStatus.Available) {
                Debug.Log("Firebase SUCCESS, readying analytics");

                firebaseIsOk = true;
                AnalyticsController.ReadyAnalytics();

            } else {
                Debug.LogError(
                    "Could not resolve all Firebase dependencies: " + dependencyStatus);
            }
        });
    }
}
```

Figura 29 – Código de inicialização do serviço do Google Firebase. Fonte: elaboração própria, 2022

```
public static void ReadyAnalytics()
{
    RemoteConfigManager.Instance.StartCoroutine(Init());
}

private static IEnumerator Init()
{
    yield return new WaitForSeconds(1);

    Instance.app = Firebase.FirebaseApp.DefaultInstance;

    FirebaseAnalytics.SetAnalyticsCollectionEnabled(true);
    FirebaseAnalytics.SetUserProperty(FirebaseAnalytics.UserPropertySignUpMethod,"Google");

    Instance.ready = true;
    Instance.invokeWhenReady.Invoke();
    Instance.invokeWhenReady.RemoveAllListeners();
    Debug.Log("Analytics Ready");
}
```

Figura 30 – Código de inicialização do serviço de Analytics do Firebase. Fonte: elaboração própria, 2022

O código mostrado na Figura 29 opera da seguinte forma: realiza-se uma checagem e correção das dependências do Firebase através do método assíncrono “CheckAndFixDependencies” e ,caso nenhum erro ocorra durante essa checagem, a inicialização do serviço do Firebase é dada como concluída e então inicia-se a inicialização do serviço de Analytics como mostrado na Figura 30. Nesta Figura é mostrado o código que opera da seguinte forma: aguarda-se um segundo para que qualquer resquício de processamento da inicialização do Firebase termine de executar, então guarda-se uma referência do FirebaseApp padrão para ser utilizado a posteriori e por final define-se o método pelo qual o usuário logou no serviço (no caso do Neko Dungeon, o usuário sempre loga pelo método da Google).

Válido pontuar que é necessário realizar um callback para todas as tentativas de disparar eventos que aconteceram antes da inicialização do serviço de Analytics acabar,

como é realizado pelo método "instance.invokeWhenReady.Invoke()".

Após a inicialização, é possível disparar eventos fazendo uso do método e das sobrecargas mostras na Figura 31.

```
public static void LogEvent(string eventName, Parameter[] parameters)
{
    if(!Instance.ready)
        Instance.invokeWhenReady.AddListener(() => LogEvent(eventName,parameters));
    else
        FirebaseAnalytics.LogEvent(eventName,parameters);
}

public static void LogEvent(string eventName, string paramName, string paramValue)
{
    if(!Instance.ready)
        Instance.invokeWhenReady.AddListener(() => LogEvent(eventName,paramName,paramValue));
    else
        FirebaseAnalytics.LogEvent(eventName,paramName,paramValue);
}

public static void LogEvent(string eventName, string paramName, long paramValue)
{
    if(!Instance.ready)
        Instance.invokeWhenReady.AddListener(() => LogEvent(eventName,paramName,paramValue));
    else
        FirebaseAnalytics.LogEvent(eventName,paramName,paramValue);
}

public static void LogEvent(string eventName, string paramName, double paramValue)
{
    if(!Instance.ready)
        Instance.invokeWhenReady.AddListener(() => LogEvent(eventName,paramName,paramValue));
    else
        FirebaseAnalytics.LogEvent(eventName,paramName,paramValue);
}

public static void LogEvent(string eventName, string paramName, int paramValue)
{
    if(!Instance.ready)
        Instance.invokeWhenReady.AddListener(() => LogEvent(eventName,paramName,paramValue));
    else
        FirebaseAnalytics.LogEvent(eventName,paramName,paramValue);
}

public static void LogEvent(string eventName)
{
    if(!Instance.ready)
        Instance.invokeWhenReady.AddListener(() => LogEvent(eventName));
    else
        FirebaseAnalytics.LogEvent(eventName);
}
```

Figura 31 – Código para disparar eventos do serviço de Analytics do Firebase. Fonte: elaboração própria, 2022

O método "LogEvent" e suas sobrecargas mostrados na Figura 31 operam da seguinte forma: Caso o serviço de Analytics ainda não estiver sido inicializado, reserva-se o evento a ser disparado no callback "invokeWhenReady" para ser enviado após a inicialização ser concluída, caso contrário, o evento é disparado junto de quaisquer parâmetros passados fazendo uso do método do Firebase Analytics "FirebaseAnalytics.LogEvent()".

As sobrecargas são necessárias uma vez que existem eventos que não possuem parâmetros ou que possuem um ou mais parâmetros, podendo ser de diferentes tipos

(string, int, long e double).

4.2 Análise de Crashes

A Análise de Crashes foi necessária devido a falta de formas de se diagnosticar erros e falhas críticas que estavam ocorrendo em tipos e marcas dispositivos que a equipe não tinha acesso ou que estavam ocorrendo muito raramente.

Este serviço foi implementado a partir Google Firebase Crashlytics, um serviço de análise de crashes disponibilizado gratuitamente pela Google, a escolha desse serviço deve-se a confiança na sua eficiência, na preferência pela forma como gera os relatórios de dados e que além de ser gratuita ainda obedece à LGPD. A implementação ocorreu da seguinte forma:

Assim como Analytics, foi criado e configurado o projeto do jogo dentro da painel de controle do Google Firebase e então adicionado como um App o jogo e seu id, veja a Figura 5 .

Depois disso, foi realizado o download do SDK do Firebase Crashlytics e então foram propriamente adicionados ao projeto do jogo na Unity, veja a Figura 12.

E para finalizar e ativar o funcionamento do Firebase Analytics, basta inicializar o Google Firebase como na Figura 29.

4.3 Localização

A Localização foi necessária para a disponibilização do jogo em diversos países pois sem esta ferramenta seria extremamente complexo realizar o armazenamento e aplicação das diferentes traduções pelos textos do jogo.

Esta ferramenta foi implementada a partir da Unity Localization, uma ferramenta de localização disponibilizado gratuitamente pela própria Unity, além de também fazer uso do Google Spreadsheets para fazer armazenamento remoto das tabelas de tradução das diversas línguas suportadas, a escolha desta ferramenta deve-se pela confiança na sua eficiência, na sua compatibilidade com o Google Spreadsheets e por ser gratuita. A implementação ocorreu da seguinte forma:

Para realizar a instalação foi necessário realizar a importação do pacote da Unity Localization pelo Package Manager da própria Unity. Note que para instalar o pacote da Unity Localization em versões da Unity que antecedem a versão 2021.2, é necessário usar a opção “Add package from git URL” e digitar “com.unity.localization” no campo disponibilizado.

Após é realizado toda a configuração da Unity Localization, veja as Figuras 16, 17, 18 e 19.

Para fazer uso das Localizações, foi implementado um controlador que implementa métodos que permitem localizar um componente de Text (Componente padrão de texto da Unity) ou de TMP_Text (Componente padrão de texto do Text Mesh Pro, um pacote gratuito disponibilizado no Package Manager da Unity).

Para começar, é necessário criar uma referência ao tipo LocalizeStringDatabase, para que este possa realizar as chamadas assíncronas de carregamento das localizações como é feito nas Figura 32.

```
private static LocalizedStringDatabase database
{
    get
    {
        if(donotuse_database == null)
        {
            donotuse_database = new LocalizedStringDatabase();
        }

        return donotuse_database;
    }
}
```

Figura 32 – Código para geração de referência ao LocalizeStringDatabase. Fonte: elaboração própria, 2022

Após isto, basta implementar as funções de localização mostradas nas Figuras 33 e 34.

```
public static void UpdateLocalizer(int tableCode,Text tx,string key)
{
    LocalizeStringEvent behaviour = tx.gameObject.GetComponent<LocalizeStringEvent>();
    if(behaviour == null)
    {
        behaviour = tx.gameObject.AddComponent<LocalizeStringEvent>();
        behaviour.OnUpdateString.AddListener(st => tx.text = st);
    }
    behaviour.StringReference.SetReference(GetTable(tableCode),key);
}

public static void UpdateLocalizer(int tableCode,TMP_Text tx,string key)
{
    LocalizeStringEvent behaviour = tx.gameObject.GetComponent<LocalizeStringEvent>();
    if(behaviour == null)
    {
        behaviour = tx.gameObject.AddComponent<LocalizeStringEvent>();
        behaviour.OnUpdateString.AddListener(st => tx.text = st);
    }

    behaviour.StringReference.SetReference(GetTable(tableCode),key);
}
```

Figura 33 – Código das funções UpdateLocalizer. Fonte: elaboração própria, 2022

```
public static void GetLocalizedString(int tableCode,string key,Text tx,System.Action<AsyncOperationHandle<string>> eve)
{
    AsyncOperationHandle<string> async = database.GetLocalizedStringAsync(GetTable(tableCode),key);
    LocalizeStringEvent locstring = tx.GetComponent<LocalizeStringEvent>();
    if(locstring) Destroy(locstring);
    if(async.IsDone)
    {
        eve(async);
    }
    else
    {
        async.Completed += eve;
    }
}

public static void GetLocalizedString(int tableCode,string key,TMP_Text tx,System.Action<AsyncOperationHandle<string>> eve)
{
    AsyncOperationHandle<string> async = database.GetLocalizedStringAsync(GetTable(tableCode),key);
    LocalizeStringEvent locstring = tx.GetComponent<LocalizeStringEvent>();
    if(locstring) Destroy(locstring);
    if(async.IsDone)
    {
        eve(async);
    }
    else
    {
        async.Completed += eve;
    }
}

public static void GetLocalizedString(int tableCode,string key,System.Action<AsyncOperationHandle<string>> eve)
{
    AsyncOperationHandle<string> async = database.GetLocalizedStringAsync(GetTable(tableCode),key);
    if(async.IsDone)
    {
        eve(async);
    }
    else
    {
        async.Completed += eve;
    }
}
```

Figura 34 – Código das funções GetLocalizedString. Fonte: elaboração própria, 2022

Existem 2 tipos de métodos para localizar os componentes de texto, um que localiza o componente de texto diretamente chamado “UpdateLocalizer” e outro que antes de localizar o componente de texto disponibiliza um callback para que o texto localizado possa ser alterado antes de ser passado para o componente de texto (como por exemplo adicionar um número ou o nome do jogador num texto) chamado “GetLocalizedString”. O Primeiro destes métodos, para fins de simplicidade, faz uso de um componente padrão do Unity Localization, o Localize String Event que localiza o componente de texto que está contido no mesmo GameObject a partir de uma chave dada.

O método “UpdateLocalizer” opera da seguinte forma: Primeiro realiza-se uma checagem se o componente de texto passado como parâmetro ja possui o componente Localize String Event, caso não possua, cria-se este componente e o configura para alterar a propriedade de texto do componente de texto, após isso configura-se qual a tabela a ser utilizada e de qual chave desta tabela a localização deve ser retirada.

O método “GetLocalizedString” opera da seguinte forma: Primeiro inicia-se uma operação assíncrona para retirar da tabela utilizada a localização referente a chave passada como parâmetro, então caso o componente de texto ja possua um componente de Localize

String Event, remove-se este componente, e por final realiza-se uma última checagem, se a operação assíncrona iniciada no início deste método ja foi concluída, é chamado o callback passado como parâmetro do método, caso contrário, configura-se na operação assíncrona o callback que deve ser chamado assim que a operação for concluída.

Os nomes das tabelas são providenciadas pelo método "GetTable" que retorna uma "string" com o nome da tabela dado o código de tabela provido na chamada das funções.

4.4 Notificações Locais

As Notificações Locais foram necessárias para aumenta a retenção e o engajamento do usuário com o jogo de forma a chamar a sua atenção com notificações quando algum bonus estiver disponível ou quando o jogador ficar demasiado tempo sem jogar o jogo.

Esta ferramenta foi implementada a partir do Easy Mobile Pro, uma ferramenta paga para a Unity com utilidades diversas voltada para jogos desenvolvidos para aparelhos móveis e desenvolvida pela SgLib Games, a escolha dessa ferramenta deve-se pela confiança na sua eficiência, nas suas múltiplas funções, na facilidade de implementação e que obedece à LGPD. A implementação ocorreu da seguinte forma:

Primeiro, após adquirido, o pacote do Easy Mobile Pro foi importado através do Package Manager. Para funcionamento correto das Notificações, é necessário realizar a inicialização do Plugin de Notificações a partir do seguinte código mostrado na Figura 35.

```
private void Init()
{
    if(alreadyInitted) return;
    alreadyInitted = true;

    Notifications.Init();
}
```

Figura 35 – Código para inicialização das Notificações do Easy Mobile Pro. Fonte: elaboração própria, 2022

E para finalizar foi adicionado o código que permite realizar o agendamento das Notificações Locais mostrado nas Figuras 36 e 37.

```
public void ScheduleCoinRewardNotif()
{
    Notifications.GetPendingLocalNotifications((nots) => {
        bool shouldRemove = false;

        Tools.SaveSystem.TryGetValue("CoinRewardID",out string coinRewardRequest,string.Empty);

        foreach(NotificationRequest req in nots)
        {
            if(req.id == coinRewardRequest)
            {
                shouldRemove = true;
            }
        }

        if(shouldRemove) Notifications.CancelPendingLocalNotification(coinRewardRequest);

        float realValue = FindObjectOfType<RewardsButton>().CooldownInHours*3600;
        int hours = Mathf.FloorToInt(realValue/3600);
        int minutes = Mathf.FloorToInt((realValue - (hours*3600))/60);
        int seconds = Mathf.FloorToInt(realValue - (hours*3600) - (minutes*60));

        coinRewardRequest = Notifications.ScheduleLocalNotification(new System.TimeSpan(hours,minutes,seconds),GenerateContentCoinReward());

        Tools.SaveSystem.SetKeyValue("CoinRewardID",coinRewardRequest);
    });
}

public void ScheduleDailyRewardNotif(System.TimeSpan Span)
{
    Notifications.ScheduleLocalNotification(Span,GenerateContentDailyReward());
}
```

Figura 36 – Código para agendamento das Notificações do Easy Mobile Pro. Fonte: elaboração própria, 2022

```
private NotificationContent GenerateContentCoinReward()
{
    NotificationContent content = new NotificationContent();

    content.title = "Ooooh?!";

    content.body = "A new coin reward is available in Neko Dungeon!";

    return content;
}

private NotificationContent GenerateContentDailyReward()
{
    NotificationContent content = new NotificationContent();

    content.title = "Nya!";

    content.body = "Your daily reward is ready to be collected in Neko Dungeon!";

    return content;
}
```

Figura 37 – Código para gerar o conteúdo das Notificações do Easy Mobile Pro. Fonte: elaboração própria, 2022

É valido pontuar que o primeiro método de agendamento “ScheduleCoinRewardNotif” faz uso do método assíncrono “Notifications.GetPendingLocalNotifications” para checar se a última notificação do tipo “CoinReward” ainda está pendente, se ainda estiver, esta última notificação é removida antes de agendar a nova notificação do tipo “CoinReward”.

4.5 Microtransactions

As Microtransactions foram necessárias devido a necessidade da implementação de compras dentro do jogo.

Este serviço foi implementado a partir do Easy Mobile Pro, uma ferramenta paga para a Unity com utilidades diversas voltada para jogos desenvolvidos para aparelhos móveis e desenvolvida pela SgLib Games, a escolha dessa ferramenta deve-se pela confiança na sua eficiência, nas suas múltiplas funções, na facilidade de implementação e que obedece à LGPD. A implementação ocorreu da seguinte forma:

Primeiro, após adquirido, o pacote do Easy Mobile Pro foi importado através do Package Manager. Após importação, o arquivo EM_SETTINGS.asset é criado na pasta "Assets/EasyMobile/Resources/" e neste arquivo foi realizado a seguinte configuração mostrada na Figura 38.

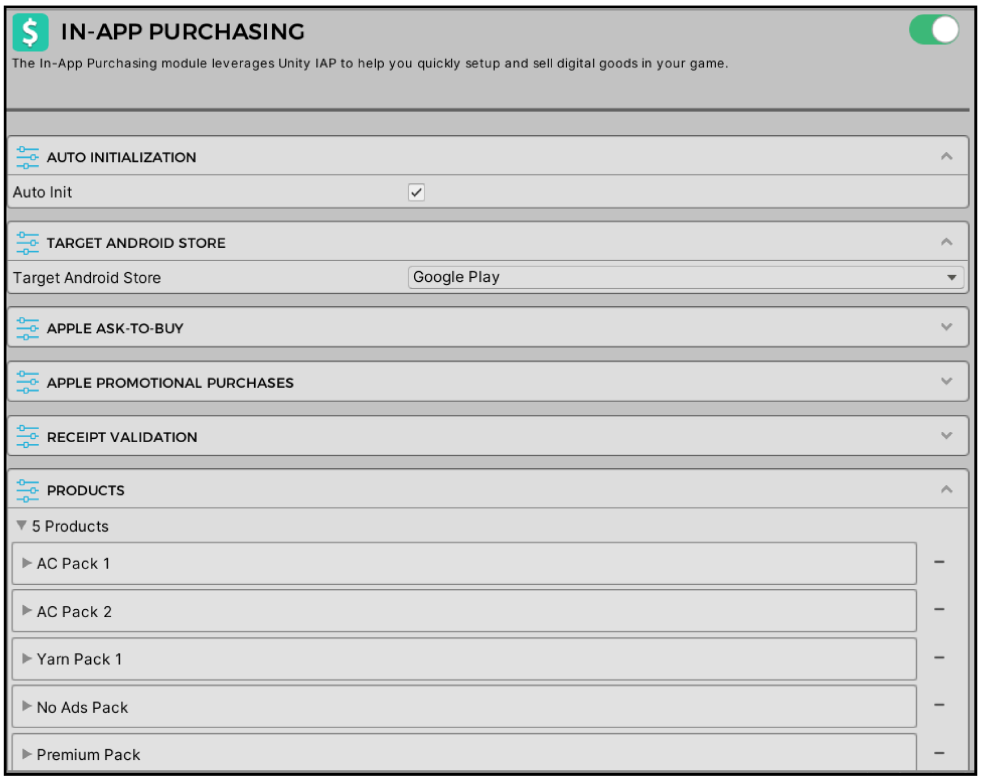

Figura 38 – Configuração dos Microtransactions no Easy Mobile Pro. Fonte: elaboração própria, 2022

Note os produtos criados devem possuir o mesmo id que os seus respectivos produtos na loja de aplicativos que o jogo será hospedado, além disso após todos os produtos serem adicionados é necessário pressionar o botão no final da página "Generate Constants Class" para gerar as constantes necessárias.

Para funcionamento correto das Microtransactions, é necessário realizar a inicia-

lização do Plugin de Microtransactions a partir do seguinte código mostrado na Figura 39.

```
private void Start()
{
    InAppPurchasing.InitializePurchasing();

    StartCoroutine(WaitForInit(() => {
        if(InAppPurchasing.IsProductOwned(EM_IAPConstants.Product_No_Ads_Pack))
        {
            if(!Advertising.IsAdRemoved())
            {
                RemoveAdsPack();
            }
        }
    }));
}
```

Figura 39 – Código para inicialização dos Microtransactions no Easy Mobile Pro. Fonte: elaboração própria, 2022

É valido pontuar que após a inicialização, de forma assíncrona é realizado a checagem da compra do produto "No_Ads_Pack" pois este é um produto de compra única que não deve ser disponibilizado novamente para compra se o usuário já o possuir.

E para finalizar são implementadas as chamadas para realizar o processo de compra e também as chamadas para configurar os callbacks necessários a partir dos seguintes códigos mostrados nas Figuras 40 e 41.

```
public void BuyAC1()
{
    if(InAppPurchasing.IsInitialized())
    {
        SetButtonsInteractable(false);
        SetHandlers(true);
        InAppPurchasing.Purchase(EM_IAPConstants.Product_AC_Pack_1);
    }
}

public void BuyAC2()
{
    if(InAppPurchasing.IsInitialized())
    {
        SetButtonsInteractable(false);
        SetHandlers(true);
        InAppPurchasing.Purchase(EM_IAPConstants.Product_AC_Pack_2);
    }
}

public void BuyNoAds()
{
    if(InAppPurchasing.IsInitialized() && !InAppPurchasing.IsProductOwned(EM_IAPConstants.Product_No_Ads_Pack))
    {
        SetButtonsInteractable(false);
        SetHandlers(true);
        InAppPurchasing.Purchase(EM_IAPConstants.Product_No_Ads_Pack);
    }
}
```

Figura 40 – Código para compra dos Microtransactions no Easy Mobile Pro. Fonte: elaboração própria, 2022

```
private void SetHandlers(bool target)
{
    if(target)
    {
        InAppPurchasing.PurchaseCompleted += PurchaseCompletedHandler;
        InAppPurchasing.PurchaseFailed += PurchaseFailedHandler;
        InAppPurchasing.PurchaseDeferred += PurchaseFailedHandler;
    }
    else
    {
        InAppPurchasing.PurchaseCompleted -= PurchaseCompletedHandler;
        InAppPurchasing.PurchaseFailed -= PurchaseFailedHandler;
        InAppPurchasing.PurchaseDeferred -= PurchaseFailedHandler;
    }
}
```

Figura 41 – Código para os callbacks dos Microtransactions no Easy Mobile Pro. Fonte: elaboração própria, 2022

Após o método "InAppPurchasing.Purchase" é chamado, o balão de compra da loja de aplicativos que o jogo está hospedado é mostrado e dependendo do resultado da transação um dos callbacks configurados pelo método "SetHandlers" é chamado para que o jogo possa atualizar a tela e, no caso de compra sucedida, entregar o item comprado para o usuário.

4.6 Advertising

O Advertising foi implementado para mostrar propagandas dentro do jogo.

Este serviço foi implementado a partir do Easy Mobile Pro, uma ferramenta paga para a Unity com utilidades diversas voltada para jogos desenvolvidos para aparelhos móveis e desenvolvida pela SgLib Games, a escolha dessa ferramenta deve-se pela confiança na sua eficiência, nas suas múltiplas funções, na facilidade de implementação e que obedece à LGPD. A implementação ocorreu da seguinte forma:

Primeiro, após adquirido, o pacote do Easy Mobile Pro foi importado através do Package Manager. Após importação, o arquivo EM_SETTINGS.asset é criado na pasta "Assets/EasyMobile/Resources/" e neste arquivo foi realizado a seguinte configuração mostrada nas Figuras 42 e 43.

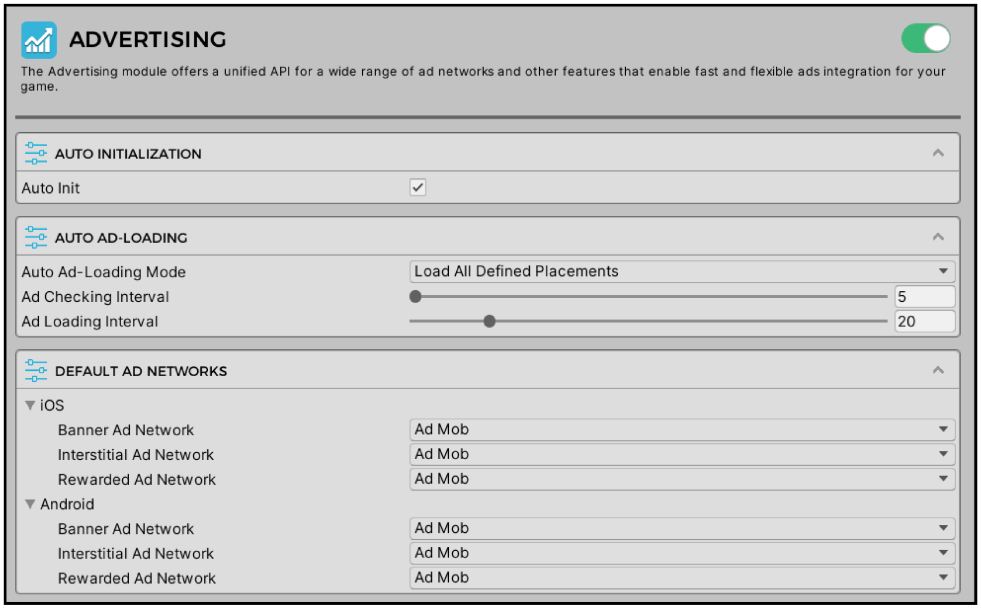

Figura 42 – Configuração do Advertising no Easy Mobile Pro. Fonte: elaboração própria, 2022

Figura 43 – Configuração do AdMob no Easy Mobile Pro. Fonte: elaboração própria, 2022

Válido que a rede provedora de anuncios escolhida para o aplicativo, no caso do Neko Dungeon é a Google AdMob, deve ser escolhida na sessão de "Default Ad Networks" e então configurada na sua respectiva sessão logo abaixo como na Figura 43, nesta sessão deve ser importado o seu plugin, então deve-se ser adicionado os respectivos IDs dos Ads gerados no painel de controle da rede provedora de anuncios escolhida e por fim deve-se clicar no botão de "Setup" para concluir a configuração.

Para funcionamento correto do Advertising, é necessário realizar a inicialização do Plugin principal do Easy Mobile Pro a partir do seguinte código mostrado nas Figuras 44, 45 e 46.

```
private void Initialize()
{
    if(initialized) return;

    initialized = true;

    if(!RuntimeManager.IsInitialized())
    {
        RuntimeManager.Init();
    }

    playing = false;

    Advertising.RewardedAdCompleted += (net,adplace) => Instance.playing = false;
    Advertising.RewardedAdSkipped += (net,adplace) => Instance.playing = false;

}
```

Figura 44 – Código para inicialização do Advertising no Easy Mobile Pro. Fonte: elaboração própria, 2022

Neste código é realizado a configuração de dois callbacks para implementar o fluxo de funcionamento relacionado ao anúncio com recompensa, para caso ele seja assistido por inteiro ou caso ele seja pulado.

E para finalizar são implementadas as chamadas para mostrar os anuncios a partir dos seguintes códigos:

```
public static void ShowInterstitialAd()
{
    if(!RuntimeManager.IsInitialized()) RuntimeManager.Init();

    if(Advertising.IsInterstitialAdReady())
    {
        Advertising.ShowInterstitialAd();
    }
}

public static void ShowRewardedAd(System.Action<RewardedAdNetwork, AdPlacement> completed, System.Action<RewardedAdNetwork, AdPlacement> skipped)
{
    if(!RuntimeManager.IsInitialized()) RuntimeManager.Init();

    AdPlacement ad = GetNextAdPlacement();

    if(ad != null)
    {
        Instance.ClearRewardAdCallbacks();

        Advertising.ShowRewardedAd(ad);
        Advertising.RewardedAdCompleted += completed;
        Advertising.RewardedAdSkipped += skipped;
        Instance.nowCompleted = completed;
        Instance.nowSkipped = skipped;

        Instance.playing = true;
    }
}

public static bool IsAdPlaying()
{
    return Instance.playing;
}
```

Figura 45 – Código para chamada do Advertising no Easy Mobile Pro. Fonte: elaboração própria, 2022

```
private void ClearRewardAdCallbacks()
{
    if(nowCompleted != null)
    {
        Advertising.RewardedAdCompleted -= nowCompleted;
    }

    if(nowSkipped != null)
    {
        Advertising.RewardedAdSkipped -= nowSkipped;
    }

    nowCompleted = null;
    nowSkipped = null;
}
```

Figura 46 – Código para limpeza dos callbacks do Advertising no Easy Mobile Pro. Fonte: elaboração própria, 2022

Uma vez que o método "ShowRewardedAd" chama um anúncio com recompensa, é necessário passar como parâmetro callbacks para caso o anúncio seja completado ou pulado para que o jogo possa atualizar a interface e, caso o anúncio seja completado, entregar Ao usuário a sua recompensa. Note que é necessário limpar os callbacks de retorno, como feito no método "ClearRewardAdCallbacks()", dos anúncios com recompensa antes de chamar um novo anúncio com recompensa uma vez que se não limpado ocorrerá a chamada indevida de callbacks passados que provavelmente não serão válidos.

4.7 Anti-Cheat

A Anti-Cheat foi necessário para dificultar tentativas de trapaças dentro do jogo (através de edição de chaves de registro e modificação da memória volátil do dispositivo).

Esta ferramenta foi implementada a partir do Anti-Cheat Toolkit, uma ferramenta paga para a Unity com utilidades diversas voltada para segurança e proteção contra trapaças e desenvolvida pela Code Stage, a escolha desta ferramenta deve-se a confiança na sua eficiência, nas suas múltiplas funções e nas avaliações na sua respectiva página na Unity Asset Store. A implementação ocorreu da seguinte forma:

Para realizar a instalação foi necessário realizar a importação do pacote do Anti-Cheat Toolkit pelo Package Manager do própria Unity.

Feito isto, foi necessário apenas implementar o código necessário mostrado nas Figuras 47 e 48.

```
public static bool TryGetKey(string key, out object result, int type, object defaultValue)
{
    result = defaultValue;

    if (string.IsNullOrEmpty(key))
    {
        return false;
    }
    switch(type)
    {
        case 0:
            result = ObscuredPrefs.GetInt(key, (int)defaultValue);
            break;
        case 1:
            result = ObscuredPrefs.GetFloat(key, (float)defaultValue);
            break;
        case 2:
            result = ObscuredPrefs.GetString(key, (string)defaultValue);
            break;
    }
    return true;
}

public static bool SetKey(string key, object value, int type)
{
    if (string.IsNullOrEmpty(key))
    {
        return false;
    }
    switch(type)
    {
        case 0:
            ObscuredPrefs.SetInt(key, (int)value);
            break;
        case 1:
            ObscuredPrefs.SetFloat(key, (float)value);
            break;
        case 2:
            ObscuredPrefs.SetString(key, (string)value);
            break;
    }
    return true;
}
```

Figura 47 – Código para o “ObscuredPrefs” do Anti-Cheat Toolkit. Fonte: elaboração própria, 2022

```
private ObscuredInt m_coins = -1;
private ObscuredInt m_ac = -1;
private ObscuredInt m_yarnballs = -1;
private ObscuredInt m_artifacts = -1;
```

Figura 48 – Código para os “ObscuredTypes” do Anti-Cheat Toolkit. Fonte: elaboração própria, 2022

As “ObscuredPrefs” é utilizado no lugar do “PlayerPrefs”, o “PlayerPrefs” é uma ferramenta nativa da Unity que permite salvar informações persistentes nas chaves de registro do dispositivo de forma desprotegida. O “ObscuredPrefs” possibilita o desenvolvedor fazer a mesma coisa que faria com o “PlayerPrefs” mas de forma obscura, realizando criptografia e unificando todas as chaves de informação numa única chave principal e também possibilitando a detecção e correção de tentativas de modificação a essa chave principal. Caso seja necessário, recomenda-se o guia de Anthony Uccello em (UCCELLO, 2017) que explica como realizar a implementação de um sistema de salvamento de dados persisten-

tes no Unity fazendo uso do "PlayerPrefs", o que se mostra bastante relevante para a implementação do "ObscuredPrefs" em um jogo comercial pois ambas as ferramentas são utilizadas da mesma forma.

As "ObscuredTypes" agem como as variáveis convencionais para o desenvolvedor mas elas são guardadas na memória volátil do dispositivo de forma obscura, realizando criptografia das suas informações e possibilitando a detecção e correção de tentativas de modificação dessa variável. No exemplo é mostrado apenas a versão obscura do tipo "Int", mas esta ferramenta também implementa versões obscuras das seguintes variáveis mostradas na Figura 49.

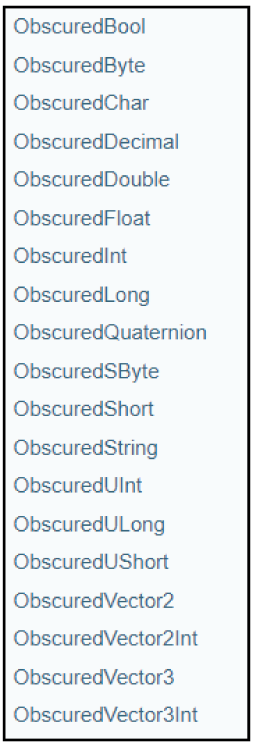

Figura 49 – Listagem de todos os "ObscuredTypes" do Anti-Cheat Toolkit. Fonte: elaboração própria, 2022

Ainda que algumas das ferramentas escolhidas tenham um preço elevado para um desenvolvedor brasileiro que está começando, o valor total delas normalmente não passa de aproximadamente menos de 5% do custo total do projeto, mesmo nos casos de projetos menores. Válido ressaltar que atualmente o Easy Mobile Pro e Anti-Cheat Toolkit podem ser encontrados sazonalmente na Unity Asset Store com descontos de no mínimo 50%, podendo assim facilitar a aquisição dessas ferramentas.

5 Conclusão

Em um contexto de consideráveis avanços na indústria de desenvolvimento de jogos, essas ferramentas de suporte a desenvolvimento de jogos não são primordiais para o funcionamento do jogo propriamente dito, no entanto elas são cruciais para o modelo de negócio do jogo. Por isso, não se desenvolve jogos modernos sem suporte a maioria das ferramentas citadas neste trabalho por que elas que habilitam diversos modelos de negócios necessários para viabilizar diferentes tipos de jogos e também elas que possibilitam e facilitam a melhoria e manutenção contínua do jogo.

Essa questão da implementação dessas ferramentas e serviços é validada, quando os jogos de celular começaram a popularizar, visto que os desenvolvedores de jogos na época perceberam que cobrar um preço para a aquisição de um jogo de celular não era uma estratégia válida para viabilizar este estilo de negócio, como era para os jogos de console e de computador. No caso destes jogos, os serviços de Microtransactions e Advertising se tornaram essenciais para viabilizá-los, pois assim era possível disponibilizar o jogo de forma gratuita e, uma vez que o jogador esteja jogando, seria possível então rentabilizar o jogo através de anúncios e de vendas dentro do aplicativo. Este estilo de jogo por sequentemente recebeu o nome de “Freemium”.

No entanto, é possível notar uma escassez significativa de materiais científicos e literatura técnica sobre o assunto. Este trabalho apresentou e elencou o estado da arte das principais ferramentas necessárias para dar suporte para o desenvolvimento de jogos e atualmente serve como um bom ponto de partida para qualquer desenvolvedor que deseja ingressar na área. Esperou-se, com a pesquisa resultante neste trabalho, contribuir, ainda que de forma introdutória, com uma boa fonte de referências sobre as diferentes ferramentas abordadas, pois condensa diversas informações sobre as ferramentas para que desenvolvedores pudessem observar na prática a aplicação desses conceitos, além de forma a tornar o acesso a este tipo de informação fácil e ágil.

Por mais que o foco da faculdade de ciência da computação não seja o desenvolvimento de jogos, é notável a relevância de diversas disciplinas deste curso no preparo de um profissional da área. A grande maioria das matérias relacionadas à programação são importantes para a formação de um programador de jogos, como a programação orientada a objetos, o algoritmos e estruturas de dados, a análise de algoritmos, a computação científica e otimizada e a inteligência Artificial. Algumas disciplinas da área da matemática também são relevantes, como a geometria analítica e álgebra linear, a física para computação e a estatística. E por último, quaisquer matérias da área de computação gráfica são extremamente importante, como a própria disciplina de computação gráfica e a disciplina

de multimídia.

Vale ressaltar que esta monografia é apenas um dos olhares sobre um tema amplo que está em constante evolução, que ainda apresenta diversas possibilidades de enfoque e caminhos a serem seguidos. Dessa forma, espera-se que os achados expressos nesta monografia possam cooperar com futuras pesquisas sobre serviços e ferramentas aplicados a modelos de negócios no mercado de desenvolvimento de jogos.

Para trabalhos futuros, pode ser realizado a análise de ainda mais implementações buscando abordar pelo menos uma implementação de cada ferramenta e serviço em cada uma das diferentes plataformas relevantes atualmente. Além disso, pode ser feito a abordagem de outras ferramentas e serviços que não foram abordadas neste trabalho como o game services, o cloud saving, a configuração remota, o sharing e o store review.

Referências

BUGSNAG. *Documentação de Instalação do Bugsnag*. 2022. Disponível em: <https://docs.bugsnag.com/platforms/>. Acesso em: 22 de Fevereiro de 2022. Citado 4 vezes nas páginas 16, 21, 28 e 30.

CARDOSO, J. D.; LOPES, R. d. A. Fl1pfl0p - um serious game para o ensino de robótica. 2019. Disponível em: <https://repositorio.ufu.br/handle/123456789/30205>. Citado na página 19.

CHANDLER, H. M. *The Game Localization Handbook*. [S.l.]: Jones & Bartley Leaning, 2011. Acesso em: 10 de Março de 2022. Citado na página 16.

CODESTAGE. *Documentação de uso do Anti-Cheat Toolkit*. 2022. Disponível em: <https://codestage.net/uas_files/actk/api/index.html>. Acesso em: 26 de Fevereiro de 2022. Citado 3 vezes nas páginas 18, 21 e 47.

DAVIS, S. *Protecting Games: A Security Handbook for Game Developers and Publishers*. Massachusetts: Cengage Learning, 2009. ISSN 978-1584506706. Citado na página 18.

DELTADNA. *Documentação de Instalação do deltaDNA*. 2022. Disponível em: <https://docs.deltadna.com/advanced-integration>. Acesso em: 22 de Fevereiro de 2022. Citado 4 vezes nas páginas 15, 21, 23 e 27.

EL-NASR, M.; DRACHEN, A.; CANOSSA, A. *Game Analytics: Maximizing the Value of Player Data*. [S.l.]: Springer, 2013. Acesso em: 10 de Março de 2022. Citado na página 15.

EPICGAMES. *Página principal do Easy Anti-Cheat*. 2022. Disponível em: <https://www.easy.ac/en-us/>. Acesso em: 19 de Março de 2022. Citado na página 47.

FERREIRA, G. A. Spelunky classic hd em pt-br: desafios da tradução e localização de jogos eletrônicos. 06 2021. Disponível em: <https://repositorio.ufu.br/handle/123456789/32043>. Citado na página 19.

FIELDS, T. *Mobile & Social Game Design: Monetization Methods and Mechanics*. [S.l.]: A K Peters/CRC Press, 2014. Acesso em: 10 de Março de 2022. Citado na página 17.

FIREBASE. *Documentação de Instalação do Firebase Crashlytics*. 2022. Disponível em: <https://firebase.google.com/docs/crashlytics/get-started?platform=unity>. Acesso em: 22 de Fevereiro de 2022. Citado 4 vezes nas páginas 16, 21, 28 e 29.

FIREBASE. *Documentação de Instalação do Google Firebase*. 2022. Disponível em: <https://firebase.google.com/docs/guides>. Acesso em: 22 de Fevereiro de 2022. Citado 3 vezes nas páginas 21, 23 e 24.

FIREBASE. *Documentação do Firebase Analytics*. 2022. Disponível em: <https://firebase.google.com/docs/analytics?hl=pt-br>. Acesso em: 17 de Março de 2022. Citado na página 15.

FIREBASE. *Página principal do Firebase Cloud Messaging*. 2022. Disponível em: <https://firebase.google.com/docs/cloud-messaging>. Acesso em: 19 de Março de 2022. Citado na página 38.

GLEY. *Página principal do Mobile Ads*. 2022. Disponível em: <https://assetstore.unity.com/packages/tools/integration/mobile-ads-gdpr-ios-14-5-compliant-102892>. Acesso em: 19 de Março de 2022. Citado na página 45.

IBISWORLD. *Video Games Software Developers in the US - Number of Businesses*. 2021. Disponível em: <https://www.ibisworld.com/industry-statistics/number-of-businesses/video-games-software-developers-united-states/>. Acesso em: 17 de Março de 2022. Citado na página 10.

INTERILLUSION. *Página principal do I2 Localization*. 2022. Disponível em: <http://inter-illusion.com/assets/I2LocalizationManual/I2LocalizationManual.html>. Acesso em: 19 de Março de 2022. Citado na página 32.

IRONSOURCE. *Página principal do Iron Source*. 2022. Disponível em: <https://www.is.com>. Acesso em: 19 de Março de 2022. Citado na página 45.

LEITE, B. X. *Documentação de uso do SCUE4 Anti-Cheat Solution*. 2022. Disponível em: <https://forums.unrealengine.com/t/plugin-anti-cheat-system/213948>. Acesso em: 26 de Fevereiro de 2022. Citado 4 vezes nas páginas 18, 21, 47 e 48.

MACIEL, D. G. P. Contribuições do jogo didático na aprendizagem de ciências: uma estratégia que exercita as habilidades cognitivas e sociais e promove a motivação. 08 2020. Disponível em: <http://hdl.handle.net/1843/34886>. Citado na página 19.

MAROLF, G. *Advergaming and In-Game Advertising: An Approach to the next Generation of Advertising*. Saarbrücken: VDM Verlag Dr. Mueller E.K, 2007. ISSN 978-3836402859. Citado na página 18.

MEDLER, B. Generations of game analytics, achievements and high scores. *Eludamos: Journal for Computer Game Culture*, v. 3, n. 2, p. 177–194, Oct. 2009. Disponível em: <https://www.eludamos.org/index.php/eludamos/article/view/vol3no2-4>. Citado na página 15.

OLIVEIRA, G. P. d. Métodos de inteligência artificial aplicados em jogos baseados em turnos. 07 2018. Disponível em: <https://repositorio.ufu.br/handle/123456789/22184>. Citado na página 19.

ONESIGNAL. *Documentação de uso do One Signal*. 2022. Disponível em: <https://documentation.onesignal.com/docs/mobile-sdk-setup>. Acesso em: 23 de Fevereiro de 2022. Citado 4 vezes nas páginas 17, 21, 38 e 40.

PUSHWOOSH. *Página principal do PushWoosh*. 2022. Disponível em: <https://www.pushwoosh.com>. Acesso em: 19 de Março de 2022. Citado na página 38.

SGLIB. *Documentação de uso do Easy Mobile Pro*. 2022. Disponível em: <https://www.easymobile.sglibgames.com/docs/pro/chapters/getting-started/using.html#overview>. Acesso em: 23 de Fevereiro de 2022. Citado 3 vezes nas páginas 17, 21 e 38.

SGLIB. *Documentação de uso do Easy Mobile Pro Advertising*. 2022. Disponível em: <https://www.easymobile.sglibgames.com/docs/pro/chapters/advertising/module-configuration.html>. Acesso em: 25 de Fevereiro de 2022. Citado 3 vezes nas páginas 18, 21 e 45.

SGLIB. *Documentação de uso do Easy Mobile Pro In-App Purchasing*. 2022. Disponível em: <https://www.easymobile.sglibgames.com/docs/pro/chapters/in-app-purchasing/module-configuration.html>. Acesso em: 24 de Fevereiro de 2022. Citado 3 vezes nas páginas 17, 21 e 42.

SMARTCAT. *Página principal do SmartCAT*. 2022. Disponível em: <https://www.smartcat.com>. Acesso em: 19 de Março de 2022. Citado na página 32.

SMARTLOOK. *Página principal do SmartLook*. 2022. Disponível em: <https://www.smartlook.com>. Acesso em: 19 de Março de 2022. Citado na página 23.

STATISTA. *Fontes Statista*. 2020. Disponível em: <https://www.statista.com/statistics/1132706/media-revenue-worldwide/>. Acesso em: 17 de Março de 2022. Citado na página 10.

STATISTA. *Fontes Statista*. 2021. Disponível em: <https://www.statista.com/topics/868/video-games/>. Acesso em: 17 de Março de 2022. Citado na página 10.

STUDIO, G. *Página do Neko Dungeon na Google Play Store*. 2022. Disponível em: <https://play.google.com/store/apps/details?id=studio.gilp.nekodungeon&hl=pt_BR&gl=US>. Acesso em: 6 de Abril de 2022. Citado na página 12.

UCCELLO, A. How to Save and Load a Game in Unity. 2017. Disponível em: <https://www.raywenderlich.com/418-how-to-save-and-load-a-game-in-unity>. Acesso em: 20 de Março de 2022. Citado na página 64.

UNITY. *Documentação de Instalação do Unity Localization*. 2022. Disponível em: <https://docs.unity3d.com/Packages/com.unity.localization@1.0/manual/Installation.html>. Acesso em: 23 de Fevereiro de 2022. Citado na página 32.

UNITY. *Documentação de uso do Unity Ads*. 2022. Disponível em: <https://learn.unity.com/tutorial/getting-started-with-unity-monetization#5feb8d15edbc2a28d93d2222>. Acesso em: 25 de Fevereiro de 2022. Citado 4 vezes nas páginas 18, 21, 45 e 46.

UNITY. *Documentação de uso do Unity Localization*. 2022. Disponível em: <https://docs.unity3d.com/Packages/com.unity.localization@1.0/manual/QuickStartGuide.html>. Acesso em: 23 de Fevereiro de 2022. Citado 4 vezes nas páginas 16, 21, 32 e 35.

UNITY. *Página principal do Unity Analytics*. 2022. Disponível em: <https://docs.unity.com/analytics/UnityAnalytics.html>. Acesso em: 19 de Março de 2022. Citado na página 23.

UNITY. *Página principal do Unity IAP*. 2022. Disponível em: <https://docs.unity3d.com/Manual/UnityIAP.html>. Acesso em: 19 de Março de 2022. Citado na página 42.

VALVE. *Documentação de uso do Microtransactions Steamworks*. 2022. Disponível em: <https://partner.steamgames.com/doc/features/microtransactions/implementation>. Acesso em: 24 de Fevereiro de 2022. Citado 2 vezes nas páginas 17 e 43.

VALVE. *Documentação de uso do Steamworks*. 2022. Disponível em: <https://partner.steamgames.com/doc/gettingstarted>. Acesso em: 24 de Fevereiro de 2022. Citado 3 vezes nas páginas 21, 42 e 43.

VALVE. *Página principal do Valve Anti-Cheat*. 2022. Disponível em: <https://partner.steamgames.com/doc/features/anticheat/vac_integration>. Acesso em: 19 de Março de 2022. Citado na página 47.

WILKES, C. *Documentação de uso do Lean Localization*. 2022. Disponível em: <https://carloswilkes.com/Documentation/LeanLocalization>. Acesso em: 23 de Fevereiro de 2022. Citado 4 vezes nas páginas 16, 21, 32 e 36.

ZIPRECRUITER. *Video Game Programmer Salary*. 2022. Disponível em: <https://www.ziprecruiter.com/Salaries/Video-Game-Programmer-Salary>. Acesso em: 17 de Março de 2022. Citado na página 10.

ZOELLER, G. *Development Telemetry in Video Games Projects*. 2010. Disponível em: <https://www.gdcvault.com/play/1012227/Development-Telemetry-in-Video-Games>. Acesso em: 17 de Março de 2022. Citado na página 15.