



**Escola Superior  
de Tecnologia  
e Gestão**

Politécnico de Coimbra

# **Relatório do Terceiro Sprint**

## **Project Akrasia**

**Autor(es):**

Afonso Vitório, Fabian Nunes e Hugo Henriques

**Data:** Junho de 2021



# Resumo

Este projeto consiste em desenvolver um jogo RPG de ação, 2D no estilo cyberpunk.

No primeiro sprint tivemos como objetivo desenvolver a totalidade do prólogo do jogo.

Ao longo do segundo sprint tivemos como objetivo desenvolver a jogabilidade e ferramentas que intervém neste âmbito.

Neste terceiro sprint tivemos como missão continuar o desenvolvimento da jogabilidade, melhorar algumas das ferramentas iniciadas no sprint anterior e continuar o desenvolvimento do modo multiplayer.

O motor de jogo utilizado é o RPG Maker MV. Apesar de alguma dificuldade inicial na escolha deste motor de jogo ficamos satisfeitos, com a decisão de usar este motor.

Ao longo do primeiro sprint tínhamos vindo a questionar-nos se este motor de jogo nos possibilita a liberdade criativa que pretendemos, mas ao longo dos sprints demonstramos que recorrendo a plugins em javascript, temos a liberdade necessária.

No primeiro sprint, conseguimos cumprir o objetivo inicial e o prólogo ficou concluído, inclusivamente fizemos algumas funcionalidades extra não planeadas.

No segundo sprint, também cumprimos o que nos tínhamos proposto, tendo ficado concluído um sistema de ataque, um plugin de criação de mapas de forma automática, uma máquina de estados para animar os inimigos, mini jogos de casino e muitos outros plugins. Iniciamos também o desenvolvimento do modo multiplayer.

Neste terceiro sprint melhoramos as ferramentas desenvolvidas no sprint anterior, adicionamos ao plugin que anima os inimigos a possibilidade de fazer pathfinding, continuamos o desenvolvimento do modo multijogador, tendo criado um sistema de registo, login e bónus de login diário, recorrendo a uma API. Continuamos também o desenvolvimento das websockets para o modo multijogador propriamente dito. Este sprint criamos o nosso próprio servidor em Ubuntu Server num Raspberry Pi que tratará de armazenar todos as ferramentas necessárias para o modo multi-jogador (Base de Dados, API, WebSockets).

## Palavras-chave

RPG Maker MV, RPG, Cyberpunk, 2D, videogame, indie game, cyberpunk game, RPG Maker MV Plugins, Javascript, Cellular Automata, Finite State Machine

# Índice

<b>1.</b>	<b>Lista de Figuras .....</b>	<b>5</b>
	7	
<b>2.</b>	<b>Lista de Tabelas .....</b>	<b>8</b>
<b>1.</b>	<b>Introdução.....</b>	<b>1</b>
<b>2.</b>	<b>Estado da Arte .....</b>	<b>3</b>
<b>3.</b>	<b>Objetivos e Metodologias.....</b>	<b>4</b>
3.1.	Ferramentas e Tecnologias .....	4
3.2.	Planeamento .....	7
<b>4.</b>	<b>Trabalho Desenvolvido .....</b>	<b>8</b>
4.1.	Prólogo .....	9
4.1.1.	Cutscenes .....	9
4.1.2.	Parallax Mapping.....	11
4.1.3.	Minijogos.....	12
4.1.4.	Qualidade e interação .....	14
4.1.5.	Multimédia .....	15
4.1.6.	Uso de Plugins externos .....	16
4.2.	Plugin de geração de mapas.....	17
4.3.	Plugins de Animação dos Inimigos .....	21
4.4.	Plugins para Active Combat System.....	23
4.5.	Plugins Gráficos HUD, Menus e Título.....	30
4.6.	Plugin Utilitários .....	37
4.7.	Plugins Minijogos .....	43
4.8.	Plugins Multiplayer.....	51
4.8.1.	REST API.....	52
4.8.2.	Base de Dados .....	55
4.8.3.	Autenticação .....	56
4.8.3.1.	Registo .....	57
4.8.3.2.	Login.....	58
4.8.3.3.	Reset .....	59
4.8.3.4.	Front end .....	61
4.8.4.	Lobby .....	62
4.8.4.1.	Daily Reward.....	62
4.8.4.2.	Atualizar Armadura.....	63
4.8.4.3.	Rank .....	64
4.8.5.	Jogabilidade e Comunicação .....	67
4.8.5.1	Movimentação das personagens .....	67
4.8.5.2	Sistema de disparo .....	67

4.8.5.3 Gestão da vida .....	68
4.9. Plugin de Eventos Random.....	69
4.10. Servidor Ubuntu Server em Raspberry Pi.....	73
4.11.1 Instalação do Servidor HTTP.....	75
4.11.2 Instalação do PHP.....	76
4.11.3 Instalação do Composer.....	76
4.13. Requisitos Implementados.....	77
<b>5. Plugins Externos .....</b>	<b>80</b>
<b>6. Conclusões .....</b>	<b>83</b>
6.1. Forças .....	83
6.2. Limitações .....	85
6.3. Trabalho Futuro.....	85
<b>7. Referências.....</b>	<b>86</b>
7.1. Lista de Referências.....	86
<b>8. Anexos .....</b>	<b>89</b>
8.1. Jogo da Vida e Autómato Celular .....	89

## 1. Lista de Figuras

Figura 1 - Cutscene do Jogo.....	10
Figura 2 - Cutscene do Jogo.....	11
Figura 3 - Cutscene do Jogo.....	12
Figura 4 - Hacking System Shock 2 .....	13
Figura 5 - Minigame de Hacking .....	14
Figura 6 - Cidade durante dia .....	15
Figura 7 - Email usado durante jogo.....	16
Figura 8 - Cidade Lighting Noite .....	17
Figura 9 - Plugin Parameters .....	18
Figura 10 - Informação do Plugin In Editor.....	18
Figura 11 - ID do mapa .....	19
Figura 12 - Invocar Plugin no jogo .....	19
Figura 13 - Geração de Mapa aleatório.....	20
Figura 14 – Geração de Mapa aleatório com diferentes tiles .....	20
Figura 15 - Geração de Mapa aleatório com NPC .....	21
Figura 16 - Invocar Plugin no jogo .....	22
Figura 17 - Problema da abordagem da Máquina de Estados para animação dos inimigos .....	22
Figura 18 - Combate default do RPG maker.....	24
Figura 19 - Jogo usando ABS.....	24

Figura 20 - ABS Ataque Melee .....	26
Figura 21 - ABS Pistola .....	27
Figura 22 - ANS Shotgun .....	28
Figura 23 - ABS Assault Rifle .....	29
Figura 24 - Sprite Sheet Armas .....	30
Figura 25 - Hierarquia de uma scene no maker .....	31
Figura 26 - Default menu no maker.....	32
Figura 27 - Hud do jogo Legend of Zelda.....	32
Figura 28 - Hud do Plugin .....	33
Figura 29 - Default Titlescreen Maker .....	34
Figura 30 - Titlescreen com o Plugin .....	35
Figura 31 - Default Main menu do maker .....	36
Figura 32 - Main Menu usando o Plugin.....	37
Figura 33 - Mudar comandos no jogo Assassins Creed .....	38
Figura 34 - Mudar de comandos Com o plugin .....	39
Figura 35 - 2º Configuração de Teclado .....	40
Figura 36 - Default Save System do Maker.....	41
Figura 37 - AutoSave em um jogo .....	42
Figura 38 - Auto Save utilizando o Plugin .....	42
Figura 39 - Jogo de SlotMachines.....	43
Figura 40 - Slot Machine Plugin do Maker .....	43
Figura 41 - SlotMachine Plugin Criado por nós .....	44
Figura 42 - SlotMachine em caso de perda .....	45
Figura 43 - Plugin de Jogo de Dados.....	46
Figura 44 - Plugin de Jogo de dados contra Npc .....	47
Figura 45 - Jogo de BlackJack.....	48
Figura 46 - Plugin de Jogo de Blackjack .....	48
Figura 47 - Plugin de Jogo Blackjack .....	49
Figura 48 - Plugin de jogo de blackjack .....	50
Figura 49 - Ciclo de Vide de um componente .....	51
Figura 50 - MVC Spring .....	53
Figura 51 - Endpoints.....	54
Figura 52 - Modelo Conceptual .....	56
Figura 53 - Menu de entrada de multiplayer .....	57
Figura 54 - Demonstração no postman .....	59
Figura 55 - Email de Reset .....	60
Figura 56 - Página de Reset de password.....	60
Figura 57 - Formulário dentro do jogo .....	61
Figura 58 - Confirmação do servidor .....	62
Figura 59 - Reward diário .....	63
Figura 60 - Atualizar armadura .....	64
Figura 61 - Ranking System.....	65
Figura 62 - Informação do utilizador .....	66
Figura 63 - Dois clientes a comunicarem.....	67
Figura 64 - Disparo entre jogadores .....	68
Figura 65 - UI de Multiplayer.....	69
Figura 66 - Plugin de Eventos Randomizados.....	70

Figura 67 – Plugin de Eventos Randomizados .....	71
Figura 68 - Plugin de Eventos Randomizados.....	72
Figura 69 - Plugin de Eventos Randomizados.....	73
Figura 70 - Especificações do Raspberry Pi 3B+ .....	74
Figura 71 - Teste ao WebSocket através do IP Público do router .....	74
Figura 72 - Interface Cockpit no IP público do Servidor com os utilizadores.....	75
Figura 73 - Raspberry Pi no seu habitat.....	75

## **2. Lista de Tabelas**

Tabela 1 - Comparação de Jogos .....	3
Tabela 2 - Comparação de Jogos .....	5
Tabela 3 - Ferramentas utilizadas.....	7
Tabela 4 - Linguagens utilizadas .....	7
Tabela 5 - Resultado do Registo .....	58
Tabela 6 - Estado do user .....	58
Tabela 7 - Requesitos.....	77
Tabela 8 - Plugins Externos.....	81
Tabela 9 - Matéria abordada .....	84

# **Lista de Acrónimos**

<b>RPG</b>	Role-Playing Game
<b>ABS</b>	Action Battle System
<b>IA</b>	Inteligência Artificial
<b>NPC</b>	Non-Player Character
<b>PVP</b>	Player Versus Player
<b>MMORPG</b>	Massive Multiplayer Online Role-Playing Game



## 1. Introdução

O nosso projeto consiste em desenvolver um jogo do tipo RPG de ação a duas dimensões. O jogo a desenvolver tem como inspiração o estilo cyberpunk.

O que nos levou a escolher este projeto é o facto de considerarmos que o desenvolvimento de jogos é uma área bastante importante e interessante das ciências informáticas. Os vários aspetos do desenvolvimento têm vindo a receber mais interesse e investigações pelos meios académicos. Os desenvolvimentos nos videojogos têm impacto não só para o desenvolvimento de jogos, como para as mais diversas áreas científicas. [1][3]

Para além disso, o desenvolvimento de videojogos tem vindo a ter cada vez mais relevância financeira. Em 2020, a indústria dos videojogos gerou cerca de 159 milhares de milhões de dólares em receita. [4]

Temos interesse em tornar uma história por nós idealizada realidade através de um videojogo e no final desta unidade curricular gostávamos de ter uma prova de conceito do nosso jogo, que fosse interessante de jogar.

A história do nosso jogo passa-se num universo distópico, em que um partido chamado Global People's Party (GPP) governa de forma totalitária os habitantes do Planeta Terra.

Recentemente neste mundo, um hacker chamado V, entrou nos servidores de uma empresa de mineração da Lua Titã e revelou em fóruns na internet oculta, informações sobre a corrupção do governo, o genocídio dos habitantes de titã pelo GPP e as práticas pouco éticas protagonizadas pela empresa de mineração do governo, na lua Titã. O Governo tenta dissuadir os cidadãos passando a imagem de que estas informações eram teorias da conspiração.

A nossa história é experienciada por Zeke, um especialista em cibersegurança que trabalha para uma empresa chamada WhiteSafe, que foi contratada pelo governo para investigar a fuga de informação e apurar o culpado do ataque, ou seja, desmascarar o hacker V.

Temos vindo a desenvolver o projeto ao longo dos vários sprints, no primeiro sprint focamo-nos em desenvolver o prólogo, o mundo onde se passa o jogo, a arte, a história, minijogos, roteiros das falas do jogador com jogadores não controláveis e roteiros daquilo que se passa nas cenas.

No segundo sprint o nosso foco foi em desenvolver jogabilidade, para tal desenvolvemos mecanismos de luta. Um plugin que permite gerar mapas de forma aleatória, recorrendo a autómatos celulares. Vários minijogos, uma máquina de estados que anima os inimigos. E iniciamos também o desenvolvimento do modo de multi-jogador. De notar que nenhum dos desenvolvimentos deste sprint é final, mas um início daquilo que pretendemos trabalhar nos próximos sprints.

Ao longo do terceiro sprint, foram realizadas diversas tarefas distintas, foi realizado um sistema de autenticação no engine e uma lobby para o multiplayer, foi realizada a comunicação entre dois jogadores na rede com websockets. Foi criado um servidor web e rest para comunicar com o jogo e para guardar os dados numa base de dados. Foi também melhorada a animação dos inimigos passando a esta a ter uma animação mais orgânica e uma forma de fazer pathfinding até ao jogador. Também foi realizada a configuração de um servidor Linux para fazer hosting do servidor rest e das websockets.

## 2. Estado da Arte

O nosso jogo encontra-se em duas categorias diferentes no universo dos videojogos. Por um lado, dada a dimensão e estilo do nosso jogo, este enquadrar-se entre os jogos do estilo indie. Por outro lado, dado o estilo cyberpunk do jogo enquadramos este também entre os jogos cyberpunk disponíveis no mercado.

No que toca aos jogos indie existem vários a partir dos quais fomos buscar inspiração, principalmente Legend of Zelda e Hyperlight Drifter no que toca ao gameplay.

No que toca ao estilo da arte, fomos buscar inspiração a jogos como Half-Life, Cyberpunk 2077 e Else Heart.Break().

Relativamente à narrativa fomos buscar inspiração a obras literárias como o 1984 de George Orwell, a jogos também anteriormente referidos e ao filme Matrix escrito e dirigido pelas irmãs Lana e Lilly Wachowskis.

Tabela 1 - Comparação de Jogos

Jogo	RPG de Ação	2D	Cyberpunk/Sci-Fi	História através de interações ao critério do jogador
Legend of Zelda	X	X		
Hyperlight Drifter	X	X		
Half-Life			X	X
Cyberpunk 2077	X		X	
Else Heart.Break()		X	X	X
Project Akrasia	X	X	X	X

## 3. Objetivos e Metodologias

Ao longo deste capítulo indicamos as ferramentas utilizadas e o porquê da sua escolha. E também quais as metodologias que estamos a utilizar para realizar o projeto e porquê.

### 3.1. Ferramentas e Tecnologias

A escolha das ferramentas foi uma tarefa que requereu algum pensamento. Tivemos de escolher um motor de jogo e ferramentas para edição de sprites, áudio e imagem. Existem muitas opções interessantes sobre as quais tivemos de ponderar.

No que toca ao motor de jogo, dispúnhamos várias opções pelas quais optar para este projeto. Inicialmente escolhemos o motor de jogo Godot, que é uma ferramenta gratuita, de código aberto e disponível para vários sistemas operativos distintos. Possibilita muita liberdade, mas também requer que implementemos muitas mecânicas de raiz.

No entanto, depois de um diálogo com os docentes da unidade curricular, verificámos que o mais importante neste projeto era ter algo a mostrar rapidamente e no Godot iríamos demorar muito tempo até ter algo a mostrar, pois teríamos de implementar os vários mecanismos (controlo de personagem, interação, transição, etc), praticamente de raiz.

Então optamos pelo **RPG Maker MV**, que é um motor de jogos proprietário, que traz várias mecânicas de RPG 2D já implementadas e possibilita também liberdade criativa no gameplay através do uso de javascript, que permite modificar essencialmente tudo dentro deste motor de jogo. De forma a termos a escolha mais acertada pesamos os prós e contras que foram de novo avaliados no segundo sprint chegando ao seguinte resultado descrito na tabela abaixo.

**Tabela 2 - Comparação de Jogos**

	<b>Unity (engine mais popular atualmente do mercado)</b>	<b>Godot</b>	<b>Game Maker</b>	<b>Rpg Maker MV</b>
<b>Funcionalides</b>				
Linguagem	C#	GDScript (Python Like)	GML (java/c# like) e Blocos	Javascript e Blocos
Preço	Gratis até um certo valor	Gratis open source	Pago (80€) por cada versão	Pago (70€)
Assets	Grande variedade de Assets	Pouca variedade de assets	Grande variedade de Assets	Grande variedade de assets porém proprietários
Comunidade	Grande comunidade	Comunidade ainda pequena	Grande comunidade	Grande comunidade porém com pouca experiência em aspectos técnicos
Tipo de jogos para o engine	3D (pouco suporte para 2D)	3D/2D	2D (pouco suporte para 3D)	2D
Implementação de Features	Fácil, devido a linguagem, comunidade e asset store.	Difícil devido a linguagem proprietária e ainda existir poucos desenvolvedores de godot	Intermediário, devido a linguagem e limitações do engine, porém tem uma comunidade muito grande.	Intermediário, devido a comunidade ser maioritariamente artistas e não programadores, porém devido a utilização de javascript é possível ultrapassar certas barreiras do engine

Depois de avaliar os prós e contras de todos estes engines nós fomos para o RPG Maker MV pois apesar de ser em si o “Outsystems” dos game engines seria o engine que nos

iria possibilitar ter um jogo com qualidade e dar espaço para a elaboração de aspectos mais técnicos.

Apesar de não sermos o público-alvo deste engine, nenhum outro engine nos daria tanto trabalho artístico já realizado. Podemos também implementar funcionalidades que não existem no engine e que ninguém na comunidade ainda implementou. Fazendo assim trabalho que poderá ser de algum forma valioso para próximos desenvolvedores.

Para implementar o Multiplayer foram utilizadas diversas tecnologias, para realizar o webserver que contém a REST API e guarda os dados do utilizador no multiplayer foi utilizada a framework Spring do Java, o webserver podia ser implementado em qualquer linguam como por exemplo no PHP com a framework Slim ou no NodeJS com a framework Sails, porém depois de experimentar essas a escolha caiu sobre **Spring** por causa da sua boa documentação, fácil curva de aprendizagem e semelhanças a outras frameworks dadas no curso (Glassfish), a versão do Java usada foi o **Java 8** que é uma das versões mais estáveis e oferece tudo o que é necessário. A base de dados escolhida foi **MySQL**, esta escolha deveu-se a ter sido a base de dados com que mais trabalhamos e termos um extenso conhecimento dela, porém no âmbito deste projeto uma base de dados não relacional como Firebase ou MongoDB faria mais sentido devido a simplicidade da base de dados, porém MySQL oferece-nos diversas vantagens como facilidade de leitura e eventos. Para a comunicação entre dois jogadores foi utilizada a tecnologia das Websockets, esta tecnologia é possível ser implementada em várias linguagens sendo que foi escolhido o **Ratchet** em PHP por já estar a ser desenvolvida em sprints anteriores e pela facilidade de implementar, também existem outras tecnologias como o Pusher que são “plug and play” porém são pagas e não oferecem a mesma flexibilidade. Para fazer host do servidor estamos a utilizar um **Raspberry** com o sistema operativo **ubuntu** onde foi configurado um webserver para ter acesso do exterior, nós já utilizamos em trabalhos anteriores esta solução pois é uma solução que não custa dinheiro e permite implementar conhecimentos de cadeiras dadas no curso. Para a realização de testes a Rest API foi utilizado o software **Postman** para fazer requisições HTTP, este software já foi utilizado em outras cadeiras e é um dos softwares mais completos para este fim, tendo outros exemplos como o **CRUD** e **Insomnia** e **RESTPie**.

No que toca a outras ferramentas, escolhemos usar o **asesprite** para a edição dos sprites que é uma ferramenta paga muito poderosa, porém muito amigável para utilizadores com pouca experiência. Também era possível utilizar outras ferramentas como o GIMP, porém essas ferramentas não são tão simples de utilizar para este propósito. Para o áudio utilizamos **audacity** que é a ferramenta gratuita mais utilizada, existem outros softwares como Sony forge, Adobe audition e FL Studio porém esses software são pagos e nós não precisamos de tal complexidade para o nosso projeto. Utilizamos também outras ferramentas de edição como **photoshop** e o **Da Vinci Resolve** para vídeos, a escolha dessas ferramentas deve-se ao à vontade à priori que os elementos no grupo possuem com elas. Estamos a utilizar IDEs para **javascript** como o **WebStorm**, **Atom** e o **VSCode** por terem sido os IDE utilizados já para unidades curriculares anteriores onde foi utilizado Javascript, como também o **InteliJ** para a criação do webserver e o **Datagrip** para gerir a base de dados. Para gerir o código utilizamos o **Github**, devido a termos já utilizado esta plataforma para os mais diversos trabalhos em grupo.

Para demonstrar mais facilmente o que foi explicado anteriormente decidimos fazer uma tabela com as tecnologias utilizadas e com as linguagens utilizadas.

**Tabela 3 - Ferramentas utilizadas**

<b>Utilidade</b>	<b>Ferramenta</b>
Engine do Jogo	RPG Maker MV
Edição de Imagem	Aseprite, Adobe Photoshop
Edição de Som	Audacity
Edição de Vídeo	Da Vinci Resolve
IDE	Webstorm, IntelliJ, Atom, VsCode, DataGrip
Version Control	GitHub (Sourcetree e Git Bash)
Servidor Web	Raspberry (Ubuntu 20.04 server)
Ferramenta para testes HTTP	Postman

**Tabela 4 - Linguagens utilizadas**

<b>Utilidade</b>	<b>Linguagem</b>
Jogo	Javascript
Servidor REST	Java 8 (Spring)
Websockets	PHP 8 (Ratchet)
Base de Dados	MySQL

## **3.2. Planeamento**

Devido à natureza do projeto, optamos por nos organizar seguindo os princípios do SCRUM e das metodologias ágeis, minimizando ao máximo a documentação e maximizando o desenvolvimento do jogo, sempre que possível.

Ao longo do primeiro sprint não houve propriamente uma atribuição de tarefas linear em que cada pessoa fazia uma ação específica, todos fomos desempenhando ações semelhantes.

Ao longo do segundo e terceiro sprint, de forma a maximizar a produtividade, dividimos mais as tarefas e cada membro do grupo trabalhou num problema específico.

No primeiro e no segundo sprint, todos os dias da semana exceto à quarta e ao domingo, realizamos uma reunião, às 9 da manhã, em que discutimos o que cada um ia fazer nesse dia e tinha feito no dia anterior. Resultando dessa reunião uma planilha com as tarefas realizadas no dia anterior, as tarefas a realizar no dia presente e as dificuldades sentidas.

Ao longo do terceiro sprint a realização destas reuniões não se justificou, pois, os membros do grupo encontravam-se presencialmente com bastante regularidade.

A distribuição das tarefas foi feita através da plataforma HacknPlan. No trabalho realizado podemos ver na Tabela 7 - Requisitos, os requisitos implementados e as pessoas associadas ao requisito.

## 4. Trabalho Desenvolvido

Ao longo do primeiro sprint tínhamos como objetivo desenvolver o prólogo do jogo. O capítulo que dá a introdução ao jogador para o universo onde o jogo se vai desenrolar.

O trabalho no primeiro sprint consistiu em desenvolver os contornos da história, do mundo e do protagonista. Nesse sprint foram criados ou editados todos os mapas, personagens, interações, diálogos, áudio, sistema de menus e outros componentes que constituem o jogo e o prólogo.

Foram criados 14 mapas diferentes (metro, 2 estações de metro, cidade, casino, WhiteSafe, entrada do escritório, escritório, sala de servidores, mapa do mini-jogo do hacking, escritório do patrão, elevador e casa do protagonista). Foram criados os diálogos e cenas da história do prólogo e diálogos extra que ajudam a complementar a história e caracterizar o universo. Foram também criados sons para os passos do protagonista, sons de ambiente com efeitos espaciais, efeitos de interação com o ambiente, efeitos que proporcionam a noção de um ambiente não-estático. Foi também criado um mini-jogo de hacking e vários mini jogos de casino (máquinas de slots, dados e blackjack). Também foram criados efeitos para mostrar a orgânica do mundo e das personagens, como mecânicas para beber/comer e sentar.

Ao longo do segundo sprint pretendemos iniciar o desenvolvimento da jogabilidade, para tal elaboramos os mecanismos de luta, composto por ataque corpo-a-corpo através de uma arma branca e ataque à distância, recorrendo a armas de disparo, sistema de munição e respetivas animações.

Elaboramos também neste sprint um sistema de menus. Uma inteligência artificial simples, recorrendo a uma máquina de estados. Um plugin que permite gerar mapas de

forma aleatória, recorrendo a autómatos celulares, que falaremos em maior detalhe adiante. E iniciamos o desenvolvimento do modo multi-jogador.

De notar que neste sprint e em futuros sprints, pretendemos que todas as ferramentas por nós desenvolvidas para o desenvolvimento do jogo, sejam disponibilizadas como plugin para este engine, de forma gratuita e em código aberto, para que futuros desenvolvedores de jogos neste engine, possam usar essas ferramentas e estudar/melhorar o seu funcionamento.

No terceiro sprint foi aperfeiçoada o sistema de animação dos inimigos, como também continuamos o desenvolvimento do multijogador tendo implementado um sistema de autenticação e uma lobby para realizar diversas ações como atualizar armadura, ver o stats ou entrar em uma partida conta outros jogadores (funcionalidade ainda não implementada).

## 4.1. Prólogo

Durante o primeiro Sprint decidimos-nos focar na história e no entendimento geral do engine e das suas capacidades, então criamos o primeiro capítulo da história recorrendo a alguns plugins externos e tentamos criar diversas interações que pudessem melhorar a experiência do jogador. Primeiro recorremos principalmente a eventos e javascript para partes que era impossível realizar por eventos, porém no segundo sprint alteramos alguns aspectos do prólogo que foram melhorados ao criar scripts em javascript para esse efeito.

### 4.1.1. Cutscenes

Um dos aspectos importantes de um jogo é a história. A principal forma de transmitir a história é através de cutscenes. Por isso tentamos realizar várias cutscenes no jogo para que conseguíssemos dar a entender ao jogador a história do mundo onde ele se situa.

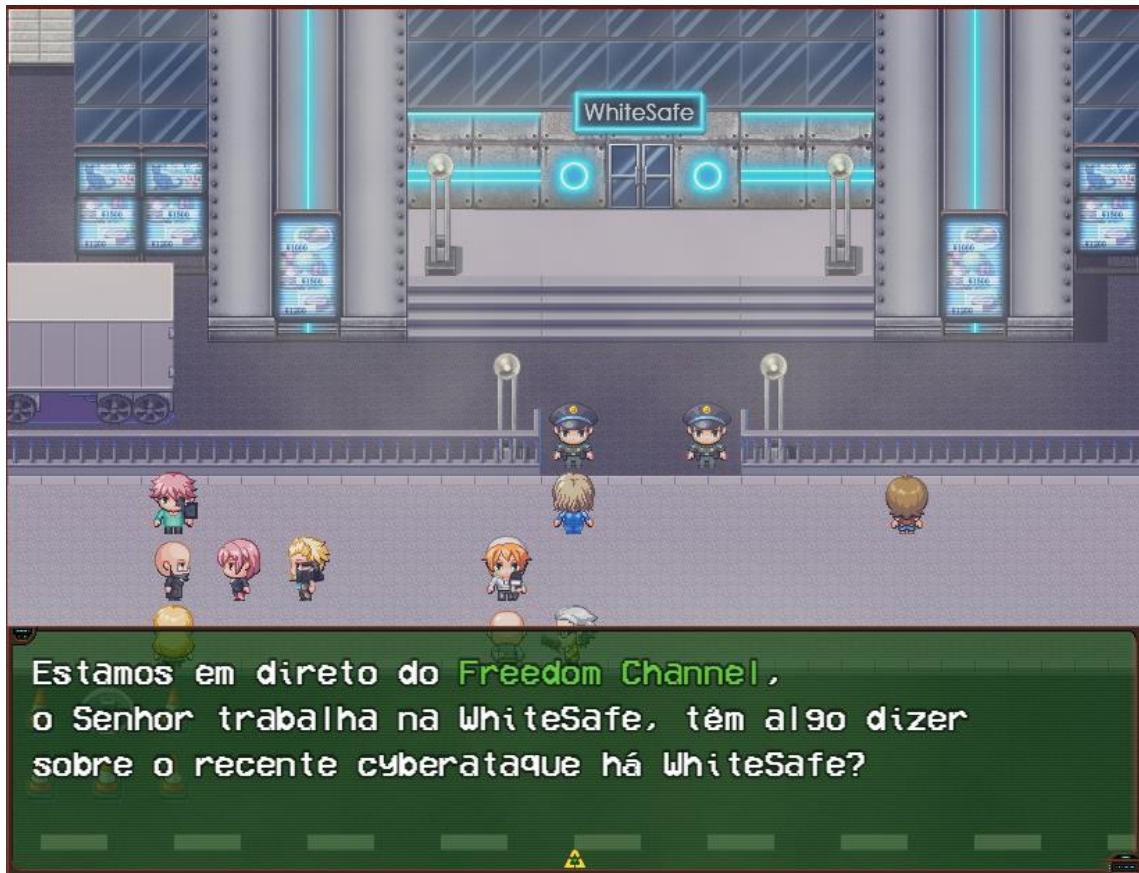


Figura 1 - Cutscene do Jogo



Figura 2 - Cutscene do Jogo

#### 4.1.2. Parallax Mapping

Uma das funcionalidades que implementamos no prólogo foi o uso de parallax mapping, isso é como um segundo layer de mapa que se movimenta independentemente do utilizador e dá a ideia de movimento, utilizamos por exemplo no metrô para dar a ideia de movimento dentro de um túnel.



Figura 3 - Cutscene do Jogo

#### 4.1.3. Minijogos

Foram criados vários mini jogos para o utilizador explorar neste capítulo como os mini jogos do casino (dados, slot e black jack) que serão explicados em detalhe mais à frente. Outro mini jogo criado que foi utilizado para progressar é o hacking que iremos utilizar mais a frente na história, este mini jogo foi baseado no hacking do system schock 2 em que existem 3 tipos de terminais, os que detetão o player instantaneamente e os que o player poderá interagir esses tem a chance de ter um vírus e afetar a vida do utilizador. O objetivo é formar uma linha de 3 computadores válidos. É possível dar reload a combinação 3 vezes caso o player fique preso e não consiga formar uma ligação válida entre os PCs.



Figura 4 - Hacking System Shock 2

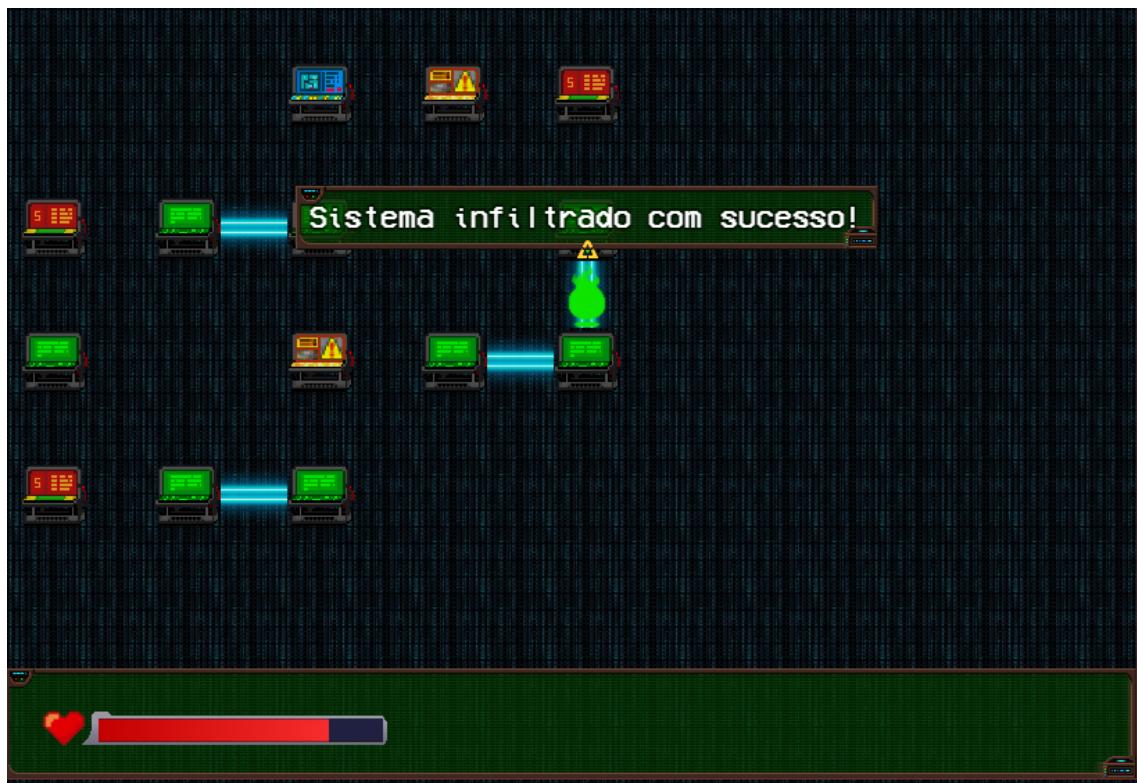


Figura 5 - Minigame de Hacking

Para realizar o mini jogo usou-se uma combinação de javascript e eventos. Porém este mini jogo ainda pode vir a mudar para o tornar mais parecido ao mini jogo que tínhamos realizado ao começo em godot antes de ir para MV.

#### 4.1.4. Qualidade e interação

Por natureza os jogadores exploram as possibilidades e as fraquezas de cada jogo e “caçam” por bugs, por isso tentamos prever várias possíveis ações que o utilizador pode tentar para “partir” o jogo e assim dar uma experiência melhor.



Figura 6 - Cidade durante dia

#### 4.1.5. Multimédia

Com os conhecimentos ganhos na cadeira de multimédia nós tentamos deixar o jogo o mais interessante possível com edições feitas no photoshop, variados sons editados no audacity e até animações realizadas no Da vinci resolve.



Figura 7 - Email usado durante jogo

#### 4.1.6. Uso de Plugins externos

Devido a isto ser um projeto final de engenharia nós não pretendemos usar muitos plugins externos pois aí tirava o trabalho de programação, sem contar com problemas de licenças e bugs. Porém também não estamos a reinventar a roda e tal como em outros softwares o programador usa bibliotecas para lhe facilitar a vida também aqui usamos plugins para melhorar o jogo, como no caso de plugins gráficos, de sons e plugins que melhoraram a qualidade do engine.



Figura 8 - Cidade Lighting Noite

## 4.2. Plugin de geração de mapas

Este plugin foi desenvolvido pelo Afonso Vitório.

O processo de criação de mapas é bastante demorado, para o desenvolvedor, não sendo sempre possível criar mapas novos e muitas vezes é repetido o mesmo mapa ao longo do jogo, criando um ambiente bastante monótono para o jogador. De forma a evitar estes problemas, decidimos desenvolver algoritmos para a geração de mapas. Especialmente úteis para os níveis de “dungeons” ou zonas onde o player defronta inimigos, que possuem quase sempre elementos semelhantes.

O plugin que desenvolvemos neste sprint baseia-se no uso de autómatos celulares, e temos como inspiração o jogo da vida elaborado pelo matemático John Conway. [86] Podem ser vistos mais detalhes no anexo na página 89

Na imagem abaixo podemos ver os parâmetros deste plugin que o seu utilizador pode especificar dentro do motor de jogo. Em Map IDs o utilizador define os mapas onde tenciona que o plugin execute, em alive e dead limit o utilizador, define os limites que abordamos no parágrafo anterior. Em “Number of Iterations”, o utilizador decide por quantas iterações o autómato celular vai executar. Em “Randomness” o utilizador decide qual a probabilidade de uma célula começar “viva” ou “morta”, sendo 0 nenhuma célula “viva” e 100, todas as células “vivas”, antes de começarem as iterações do

autómato celular. Nos restantes parâmetros, o utilizador pode definir qual o tile correspondente às células “vivas” e “mortas”, pode também definir um segundo tile que será distribuído aleatoriamente entre as células “vivas” e o tile que irá ser usado para a borda do mapa.

Parameters	
Name	Value
Map IDs	28
Alive Limit	2
Dead Limit	1
Number of Iterations	5
Randomness	35
Alive Cell Tile ID	2816
Dead Cell Tile ID	2096
Second Alive Cell Tile ID	2912
Border Tile ID	2096

Figura 9 - Plugin Parameters

Para além disto, o plugin também possui documentação e textos de ajuda que podem ser visualizados pelo utilizador dentro do engine.

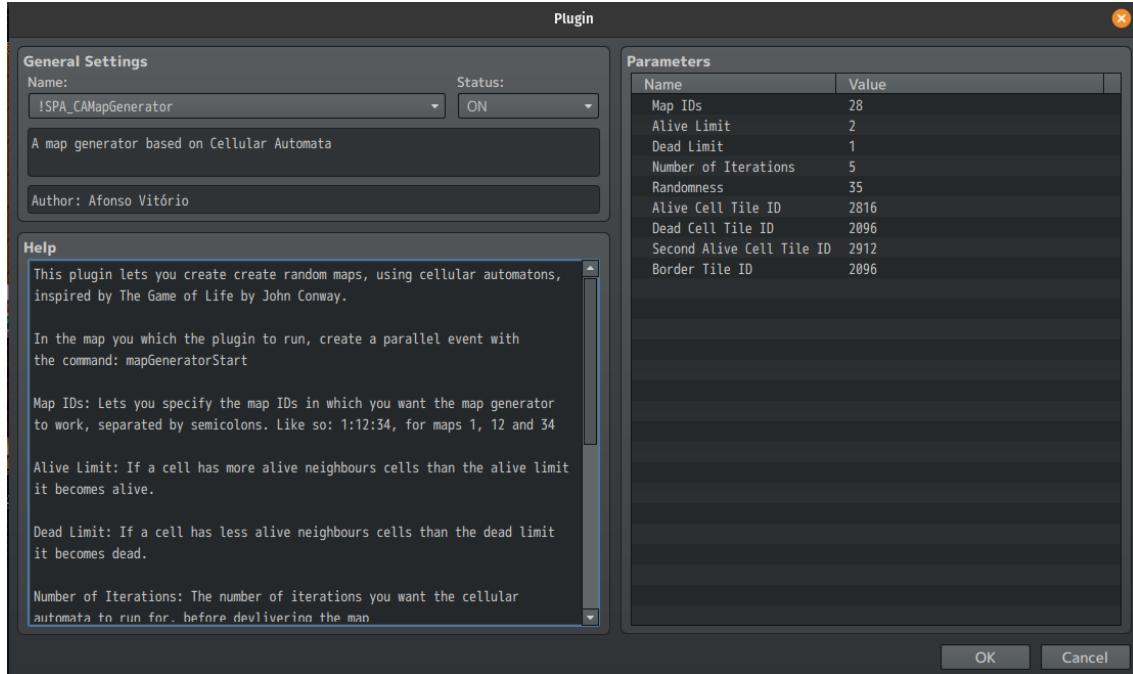


Figura 10 - Informação do Plugin In Editor

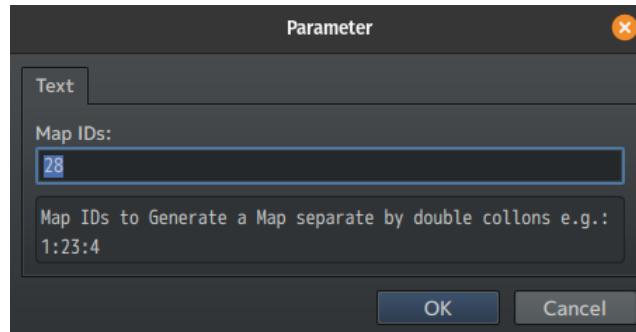


Figura 11 - ID do mapa

Uma questão interessante sobre a qual ponderamos foi o facto de, caso uma célula possuísse uma célula vizinha que estivesse fora do mapa, esta tanto pode ser considerada “viva” ou “morta” pelos autómatos celulares. Como o objetivo é tornar o mapa o mais interessante possível consideramos as células fora do mapa como “vivas” para efeitos do algoritmo, mas deixamos que fosse o utilizador do plugin a decidir quais os tiles que iriam estar no limite do mapa.

Outra questão sobre a qual ponderamos foi o facto de ser possível não existir um caminho percorrível entre o jogador e os eventos. O plugin com os parâmetros certos, possui praticamente sempre uma área percorrível, no entanto caso a intenção seja uma área não percorrível estas também se podem gerar, alterando os parâmetros do plugin.

O plugin foi feito para que qualquer que seja a dimensão do mapa especificado pelo utilizador do engine, este gera sempre um mapa que utiliza todo o espaço. Todos os eventos e personagens que tenham sido colocados nesse mapa, irão ser repositionados, de forma aleatória para cima de tiles que representam células “vivas”, ou seja nunca temos os eventos e personagens em zonas onde não é suposto.

Como já referimos anteriormente, para utilizar o plugin basta especificar nos parâmetros os IDs no engine dos mapas em que queremos este algoritmo a funcionar e para além disto, o utilizador tem também de colocar um evento a rodar em paralelo nesse mapa, que envie um comando específico ao plugin.

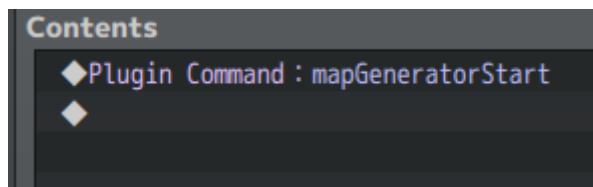


Figura 12 - Invocar Plugin no jogo

Os resultados obtidos pela utilização deste plugin são bastante interessantes, especialmente pelo facto de ser possível a alteração de bastantes dos parâmetros

através do engine. Podemos por exemplo criar vários mapas com áreas percorríveis e não percorríveis, ou somente percorríveis de vários tipos diferentes, as possibilidades são imensas.

A colocação aleatória dos personagens pelo plugin também se mostrou bastante interessante, e torna o mapa gerado altamente estimulante. Especialmente quando nos personagens utilizamos o plugin, também por nós desenvolvido de uma simples IA através de máquinas de estados.

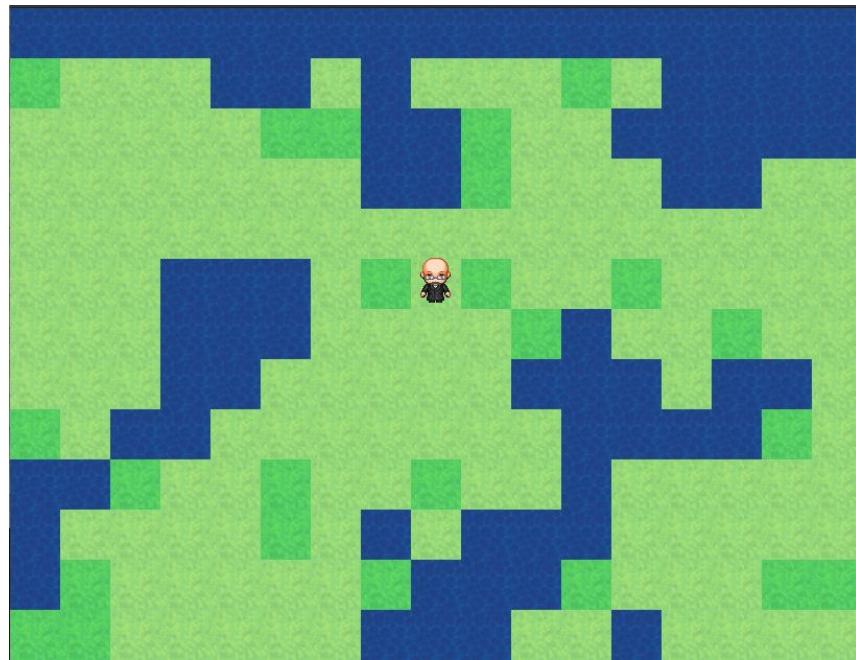


Figura 13 - Geração de Mapa aleatório



Figura 14 – Geração de Mapa aleatório com diferentes tiles



Figura 15 - Geração de Mapa aleatório com NPC

Futuramente,encionamos abordar outros algoritmos mais complexos para a geração aleatória de mapas, de forma a conseguir mapas ainda mais interessantes e de forma aleatória.

### 4.3. Plugins de Animação dos Inimigos

Estes plugins foram desenvolvidos pelo Afonso Vitório

De maneira a dar vida aos inimigos, no segundo sprint deliberamos também desenvolver dois plugins que executam ações nos inimigos.

O funcionamento deste plugin é baseado numa máquina de estados, quando o jogador está a uma determinada distância, o inimigo segue o jogador, quando está na tile ao lado do jogador, executa o ataque e quando o jogador está não alcançável, o inimigo deambula pelo mapa.

Na imagem abaixo estamos a iniciar o plugin para animar o evento 1 para perseguir o jogador se este estiver a uma distância de 8 tiles e atacar caso a uma distância de 1 tile.

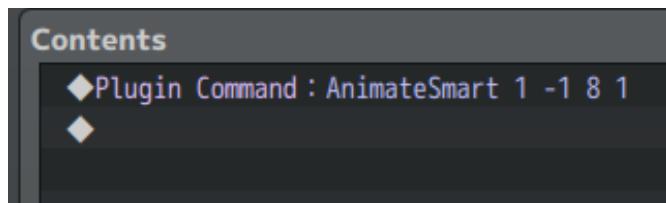


Figura 16 - Invocar Plugin no jogo

Foi também utilizado um gerador de números aleatórios que decide a probabilidade de um inimigo atacar o jogador e a probabilidade de o inimigo se aproximar ou afastar do jogador quando este ainda não foi detectado. Desta forma as ações do inimigo parecem mais orgânicas.

É importante referenciar em especial um problema que existe com esta abordagem para a animação dos inimigos, caso o jogador seja “detetável” pelo inimigo e tente perseguir o jogador, não possui um rota que possibilite chegar ao jogador.

Na figura abaixo podemos ver uma situação em que o inimigo está a detetar o jogador e tenta ir até este, no entanto o seu algoritmo não “sabe” que está bloqueado por uma série de objetos que o impedem de chegar ao jogador e leva bastante tempo até sair.



Figura 17 - Problema da abordagem da Máquina de Estados para animação dos inimigos

De forma a solucionar este problema, ao longo do terceiro sprint, foram feitas alterações nestes plugins de forma a implementar um algoritmo de pathfinding.

Inicialmente, havíamos decidido utilizar o algoritmo A\* para realizar o pathfinding. A escolha da utilização deste algoritmo deveu-se a ser um algoritmo que é rápido e consome poucos recursos, comparativamente a outros algoritmos como por exemplo o algoritmo de Dijkstra. O A\* realiza uma pesquisa somente nos nós que são promissores

de alcançar o objetivo e alcança sempre o objetivo, se tal for possível. Este algoritmo é também dos mais populares para este tipo de uso em desenvolvimento de jogos.

Apesar de ter sido feita pesquisa sobre algoritmos e de termos iniciado a implementação, descobrimos que o motor de jogo já possuía um mecanismo de pathfinding, que com algum trabalho conseguimos adaptar ao nosso plugin de animação dos inimigos e fazer com que este realize a ação pretendida.

Foi feita a alteração ao mecanismo de jogo de encontrar caminhos para um evento específico e esta alteração foi embutida no nosso plugin. Desta forma sempre que um inimigo está a “perseguir” o jogador e este se “esconde” atrás de um obstáculo o inimigo “sabe” como chegar até ao jogador. Evitamos a necessidade de implementar o algoritmo A\* neste motor de jogo e realizar trabalho redundante.

De notar que este plugin atualmente, com as devidas modificações, permite dar aos inimigos uma animação semelhante às animações de inimigos que se encontram no mercado para este género de jogo (2D de ação).

#### **4.4. Plugins para Active Combat System**

Estes plugins foram desenvolvidos pelo Fabian Nunes

Quando decidimos realizar este projeto nós pensamos em realizar um sistema de combate semelhante ao que existe em jogos como Zelda, Hyperlight Drifter e hotline miami. O sistema, de active combat, apesar de ser muito geral nos jogos atuais é um sistema com uma certa complexidade ainda mais no engine utilizado (MV), em outros engines como o Unity é muito mais fácil porém no maker é mais complexo pois ele vem com um sistema de combate já construído que é turn based combat no estilo D&D .



Figura 18 - Combate default do RPG maker



Figura 19 - Jogo usando ABS

O nosso objetivo é ter um combate em tempo real onde o utilizador possa disparar no mapa. Apesar de ser complexo existem alguns sistemas para o Maker chamados ABS (action battle system) como por exemplo o MOG Chrono Engine, Alpha ABS e o QUASI ABS, esses plugins são muito complexos e pretendem criar o seu próprio tipo de

combate, porém nós tínhamos as nossas próprias ideias logo decidimos criar o nosso próprio ABS.

Esse ABS era suposto ter ataques físicos tal como disparos no mapa, seria possível trocar de armas, apanhar munições, etc. Também seria possível os inimigos atacarem da mesma forma o inimigo, ao criarmos o nosso próprio ABS seria possível escalá-lo para futuras funcionalidades como IA dos inimigos, HUD do jogo.

O foco do plugin neste sprint foi criar o combate da perspectiva do jogador, para isso criamos o design de diferentes armas como também dos projéteis. Os projéteis envolviam vários conceitos complexos de desenho, então utilizamos uma biblioteca chamada Galv Projectiles (<https://forums.rpgmakerweb.com/index.php?threads/galvs-map-projectiles.68281/>) que permite criar projéteis, essa Biblioteca foi posteriormente modificada para permitir o disparo de múltiplos projéteis, tal como o disparo na diagonal e para alinhar os disparos baseados nas diferentes armas e diferentes posições do player.

O plugin criado para o combate serve para utilizar o teclado para mudar entre as armas, como também fazer os diferentes disparos, ou ataque físico e animação, como gerir os sons, e gerir as diferentes munições e fazer reload.



Figura 20 - ABS Ataque Melee



Figura 21 - ABS Pistola



Figura 22 - ANS Shotgun



Figura 23 - ABS Assault Rifle

Nós criamos por agora 4 tipos de armas: a crowbar que realiza ataques físicos (ataque ao que está à frente), uma pistola que dispara tiros simples, uma shotgun que dispara três tiros que forma um raio e uma assault rifle que dispara automaticamente enquanto se pressiona o ataque.

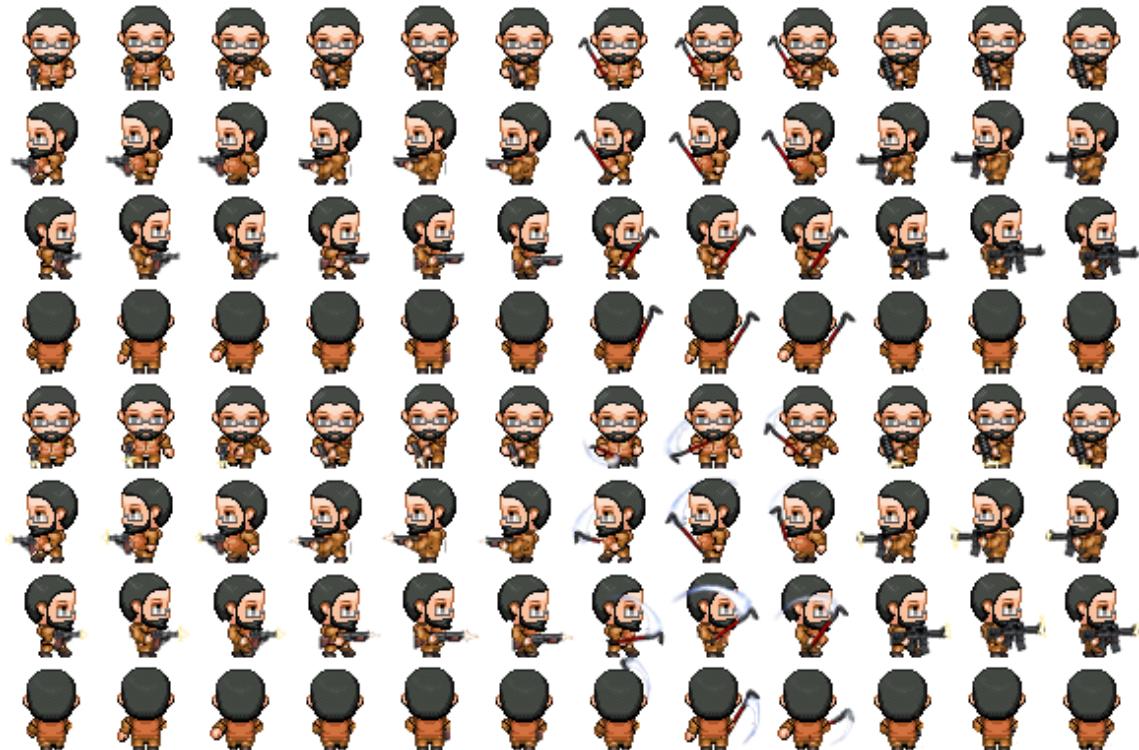


Figura 24 - Sprite Sheet Armas

Enquanto algumas foram fáceis de implementar como a pistola outras exigiram maiores conhecimentos das fraquezas do engine como a assault rifle (teve de se criar uma função separada que a cada disparo esperasse 60 frames para dar uma ideia sequencial e para não consumir demasiados recursos) e a crowbar (teve de se criar uma animação que ao ataque fosse realizada mas que fosse independente do resto do evento a acontecer para dar a ideia de movimento contínuo).

Em futuros sprints este plugin será melhorado para incluir o combate realizado pelos npc tal como tweaks para o futuro multiplayer. Também será documentado, explicando as dependências e utilizando Plugin Commands para facilitar o usuário comum do maker.

## 4.5. Plugins Gráficos HUD, Menus e Título

Estes plugins foram desenvolvidos pelo Fabian Nunes

O engine MV já traz uma interface gráfica criada para jogos do tipo RPG, essa interface é criada utilizando javascript em uma estrutura que se repete em toda a interface gráfica que é: Scene, Windows, Drawings.

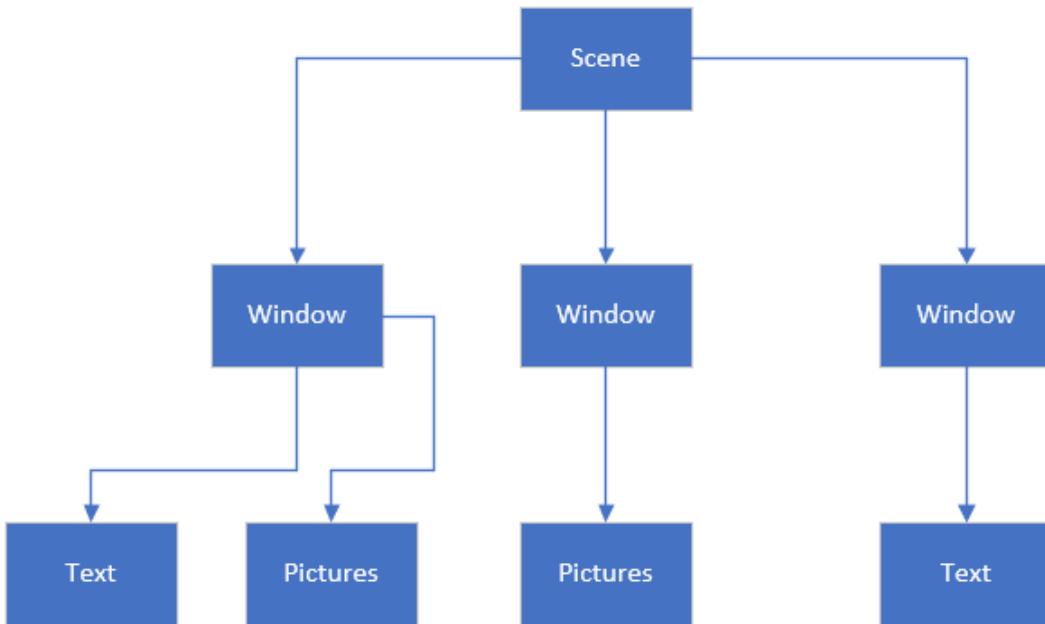


Figura 25 - Hierarquia de uma scene no maker

Uma scene é o ecrã atual do jogo, pode ser um menu, o mapa o título etc, essa scene pode conter várias windows, uma window refere-se as partes do ecrã, o default game map só tem uma grande window enquanto partes como o menu tem diversas, cada window é responsável por elementos diferentes do UI, sendo que cada uma poderá realizar internamente algoritmos ou desenhar elementos no ecrã como texto e imagens tal como desenhar inputs para o utilizador como menus e caixas de comandos. O maker implementa classes javascript chamadas Scene Base e Window Base tal como muitas outras, as Scenes já existentes herdam destas classes, dessa forma nós podemos construir tanto UI como partes totalmente novas no nosso jogo utilizando a herança destas classes e construir as nossas scenes e windows.

Um dos aspectos que nós criamos ter era o nosso próprio UI, jogos do estilo Action RPG têm o UI de alguma forma no ecrã do jogo e são mostrados em tempo real como também são simplistas, e visto que nós queremos implementar um sistema baseado em combate no mapa não faria sentido usar menus já implementados do maker.

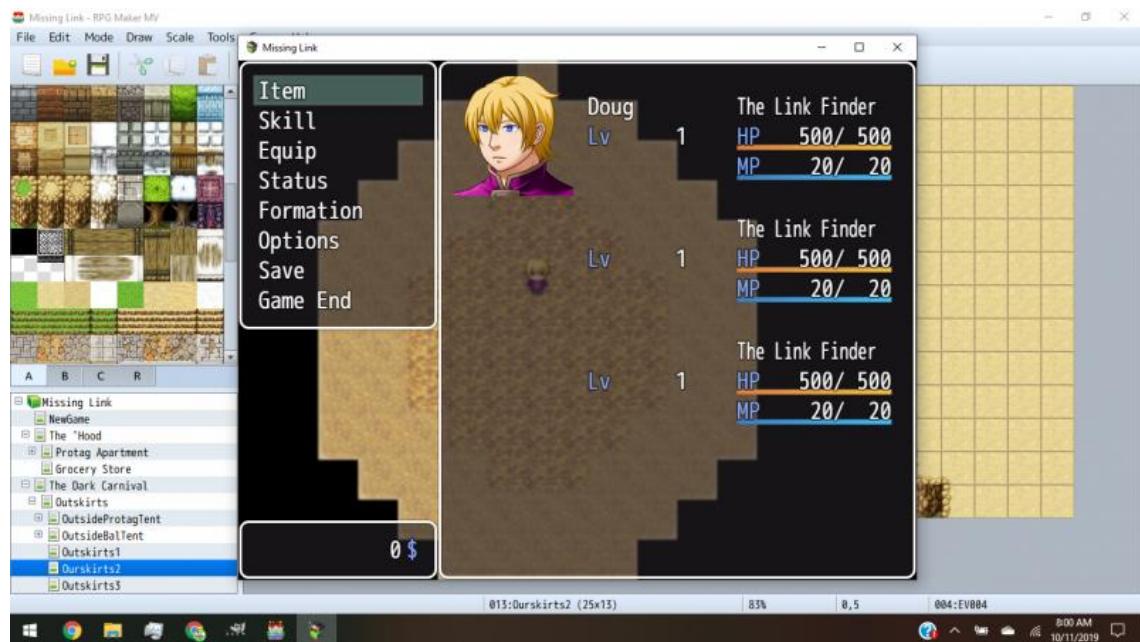


Figura 26 - Default menu no maker



Figura 27 - Hud do jogo Legend of Zelda

Para realizar isto criamos um plugin em javascript que desenhasse na Scene do jogo uma Window onde estivessem as informações relevantes ao utilizador como vida atual, munição e o dinheiro para comprar itens.

Para isso utilizamos funções que desenham texto e barras que serão atualizados a cada frame.



Figura 28 - Hud do Plugin

Com cada plugin feito o nosso conhecimento do funcionamento interno do engine veio aumentando permitindo nos modificar o jogo a outros níveis para não parecer outro jogo feito no maker. Para isso a nossa intenção foi mudar o UI do jogo todo, para isso este sprint focamos nos na mudança do Title Screen, Pause Menu tal como o HUD já falado.

O title screen do maker é simples, possuindo uma imagem como splash screen, o título simples e um menu vertical, nós utilizando Javascript criamos um título interativo que utiliza um vídeo como background, e um menu na horizontal animado, também criamos um script que permite utilizar diferentes Fonts no projeto tal como o título (o maker só permite por default uma Font chamada Gamefont no css). Um dos benefícios de criar este plugin do título foi a maior flexibilidade no título, pois permite editarmos os comandos da janela e editar, remover e adicionar os que pretendemos (tal como no futuro o multiplayer).



Figura 29 - Default Titlescreen Maker



Figura 30 - Titlescreen com o Plugin

Outro aspecto do jogo é o pause screen, por default o pause screen é o UI mostrado anteriormente que possui informações da personagem, itens, equipamento etc, visto que nós criamos o nosso próprio in screen HUD nós não precisávamos daquele então criamos o nosso próprio plugin que implementa um simples menu para opções e loading, tal como um background customizado, uma janela diferentes das restantes e uma música de fundo. Esses aspectos tiveram de ser feitos por javascript pois têm de alterar os default do maker.



Figura 31 - Default Main menu do maker

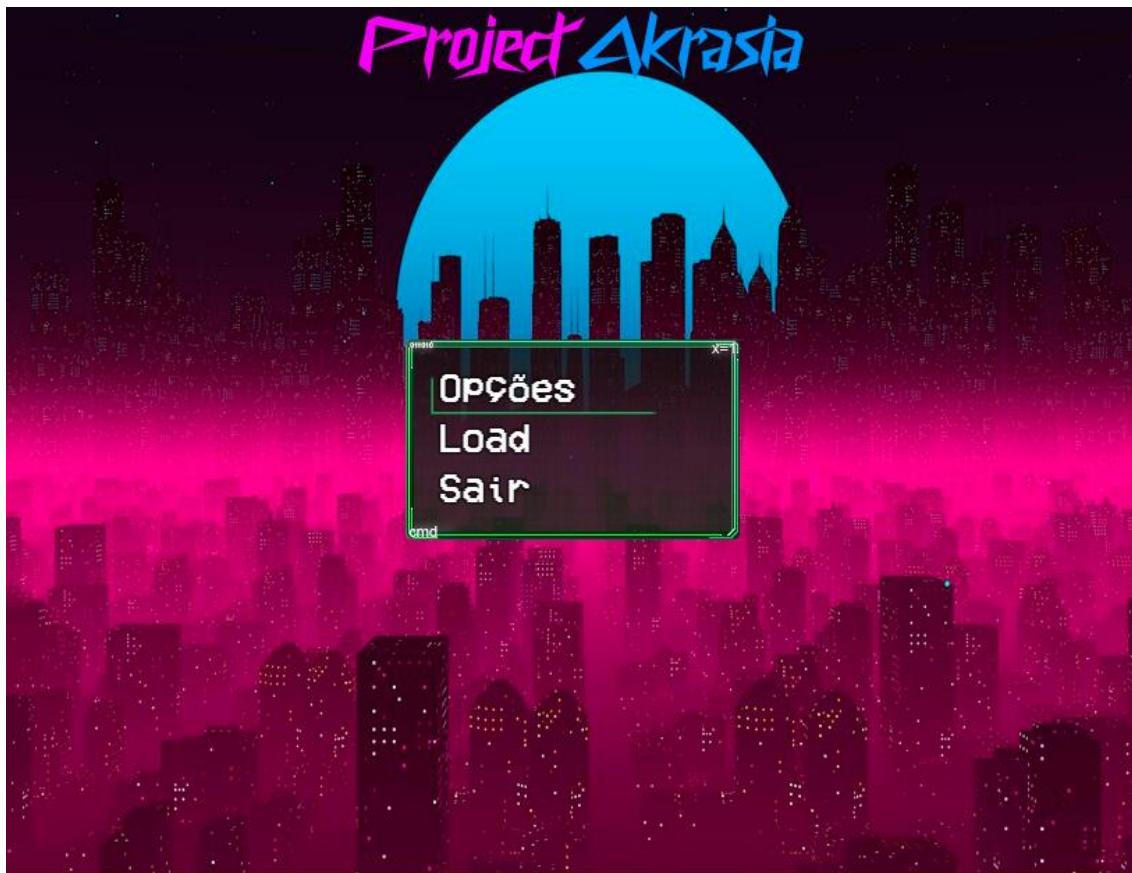


Figura 32 - Main Menu usando o Plugin

Estes plugins ainda irão sofrer alterações no futuro pois, serão criados UIs para outras situações e outras funcionalidades.

## 4.6. Plugin Utilitários

Estes plugins foram desenvolvidos pelo Fabian Nunes

Para realizar diversas funcionalidades extra no jogo foi preciso criar plugins que permitissem auxiliar essas funções, uma funcionalidade criada no segundo sprint foi a possibilidade de alterar as teclas utilizadas no jogo, isso é uma funcionalidade muito importante nos jogos atuais.

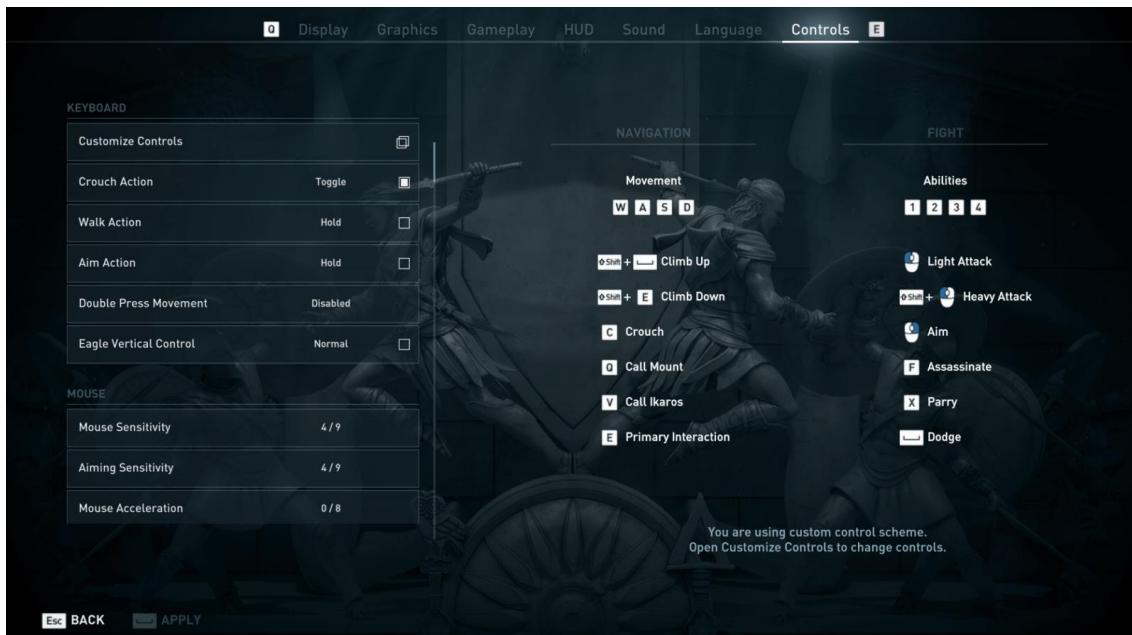


Figura 33 - Mudar comandos no jogo Assassin's Creed

Existem diversas maneiras e métodos para realizar isto e cada jogo implementa a sua, nós depois de testar várias possibilidades utilizando Javascript, nós chegamos a conclusão que seria o melhor método em ter dois layouts a escolha para o utilizador, isso devido ao Engine usar os keycodes do javascript e guardar essas informações em um array que é chamado ao início do jogo.

Nós criamos uma opção que tem os dois layouts a escolha no menu das opções para isso foi criada uma scene e windows que mostra a opção escolhida e uma imagem do layout.

Para guardar de forma persistente a escolha do utilizador foi criada uma classe utilizando node que irá escrever a configuração das teclas em um ficheiro Json que irá ler ao início do jogo para mapear as teclas, caso o utilizador apague esse ficheiro o jogo irá criar um novo com a configuração default das teclas.



Figura 34 - Mudar de comandos Com o plugin



Figura 35 - 2º Configuração de Teclado

Pretendemos criar um comando personalizado utilizando um arduino em um futuro sprint, logo a criação deste plugin ajudou-nos a adquirir os conhecimentos que precisamos para criar uma a possibilidade de usar um comando no jogo.

Ainda foi criado um saving system para auxiliar o utilizador durante o que já foi implementado caso ele saia do jogo.

O maker tem um sistema de saving próprio que nunca sofreu muitas alterações (a mais recente versão o MZ já traz auto save system porém o MV ainda não) dessa forma decidimos criar o nosso próprio saving system. Esse plugin ainda está em uma fase de maturação e vai ser modificado no próximo sprint.

O maker possui múltiplos slots para fazer save, guardando os dados em ficheiros json para carregar futuramente. No sprint 2 começamos a analisar o funcionamento desta parte do engine no core dele para conseguir realizar a nossa própria função de save, no próximo sprint iremos editar a janela de loading.



Figura 36 - Default Save System do Maker

Decidimos que não seria necessário tantos slots sendo que teríamos só 2, um de autosave que iria guardar o jogo sempre que o player mudasse de mapa (muito semelhante aos rpgs atuais) e outro slot para guardar o jogo manualmente em zonas destinadas para isso.

Logo criamos um script que permitisse isso, também construímos uma window que irá mostrar se o jogo foi gravado com sucesso e após isso desaparece do ecrã.



Figura 37 - AutoSave em um jogo

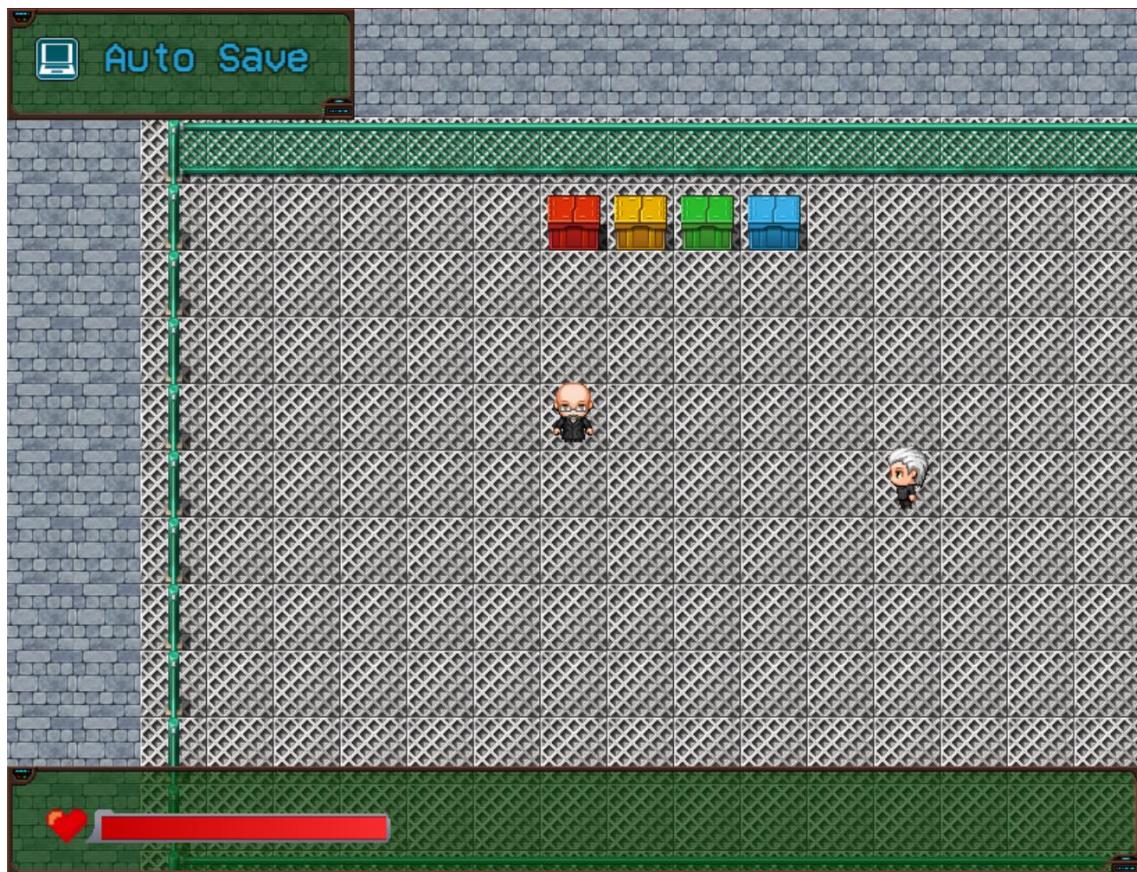


Figura 38 - Auto Save utilizando o Plugin

## 4.7. Plugins Minijogos

Estes plugins foram desenvolvidos pelo Fabian Nunes.

No sprint 1 nós criamos um mapa do jogo que era um casino onde o jogador poderia jogar vários mini jogos onde poderia ganhar dinheiro. Na primeira interação esses mini jogos eram simples e foram criados usando os eventos do jogo ou Plugins externos. No sprint 2 decidimos de forma a ficar mais diferente e interessante e não ficarmos dependentes de código externo criar os nossos próprios mini jogos. Criamos então um plugin para uma slot machine, jogo dos dados e uma espécie de blackjack.

Criar minijogos gráficos é algo das coisas mais complexas no maker pois como foi explicado antes é totalmente baseado em código (parecido pygames e libgdx), normalmente isto é suportado pelo engine (no caso do unity e godot), desta forma tentamos realizar mini jogos simples mas atraentes para o utilizador, usando uma combinação de imagens e sons que dessem vida a estes mini jogos.

O mais complexos e primeiro a ser feito foi o jogo da slot machine, nós tínhamos visualizado e inicialmente utilizado um Script que estava incluído no engine porém devido ao maker ser de desenvolvedores japoneses o código também o era, sendo assim realizamos o nosso baseado em expectativas hard coded e mais simples.



Figura 40 - Slot Machine Plugin do Maker

Conseguimos detetar após realizar um estudo como as slot machines funcionavam e o algoritmo que elas usavam para chegar ao resultado, a slot machines implementada pelo maker não seguia em si as slots machines reais mas mais as slot machines virtuais que tem 5 colunas e 3 linhas (porém pelo que percebemos do código eles só analisavam valores consecutivos e não como nas slots machines virtuais que podem ganhar em diagonais). Nós seguimos uma estrutura mais realista, as slots encontradas em casinos em que são 3 colunas com uma linha e apostar-se um valor fixo com os prêmios já definidos.



Figura 41 - SlotMachine Plugin Criado por nós



Figura 42 - SlotMachine em caso de perda

Nós utilizamos o frame da slot deles e editamos a imagem e criamos a nossa própria slot, em si o jogo é uma scene com diversas windows como viemos a falar até agora, a window superior que tem o texto e imagens que irão mudar dependendo se o jogador ganhou ou não, a window do slot onde terá as imagens que em si são um array que têm uma expectativa associada tal como um valor a ganhar se saírem, a window onde irá mostrar o dinheiro atual e aposta e uma window de comandos para aposta, sair da scene e começar o jogo.

Em si ao fazer spin as imagens vão fazer um comando move onde irão aparecer e desaparecer pelos quadrados amarelos, para dar a ilusão de movimento e que temos mais de esses elementos no ecrã, o que irá aparecer o ecrã é o símbolo respetivo ao elemento do array que para cada slot.

O prêmio e as expectativas foram baseadas em valores reais que analisamos.

Outro jogo que tínhamos era o jogo do dado, este tinha sido realizado exclusivamente usando o event system do maker, em que tínhamos uma imagem de cada fase do dado e as ações que realizamos era com as funcionalidades de move image, loops e randoms, porém apercebemo-nos que não era possível. Desta forma adicionamos outros elementos ao minijogo como

texto indicativo, background etc e decidimos converter o jogo de eventos para javascript.

O objetivo do jogo é o player adivinhar qual face do dado vai sair das seis, também fizemos uma versão em que joga com um npc em que os dois lançam os dados e o que conseguir a maior soma de pontos ganha.



Figura 43 - Plugin de Jogo de Dados

Diferente dos outros jogos este foi realizado utilizando a scene do jogo sendo que foram só criadas windows, isto porque estavam a existir conflitos e bugs ao criar uma scene com as funcionalidades das imagens porém atingimos o mesmo resultado e por um lado até é melhor, pois o resto do jogo está a ocorrer ao mesmo tempo enquanto se criarmos uma scene é como se fosse um jogo novo que está a correr por cima do base. Este jogo possui 3 funções que serão chamadas. Uma é o show dice que irá mostrar os dados (mostra o número de dados desejados e a posição), o throw dice para lançar os dados (usando o script call move do maker para dar a ilusão de lançamento) e o remove dice que os vais remover do ecrã, o valor do dado será extraído da imagem que sair, guardamos o nome de cada fase num array (ex nome dice\_1), então quando obtém o valor final basta extrair o valor do nome da imagem e guardar numa variável.

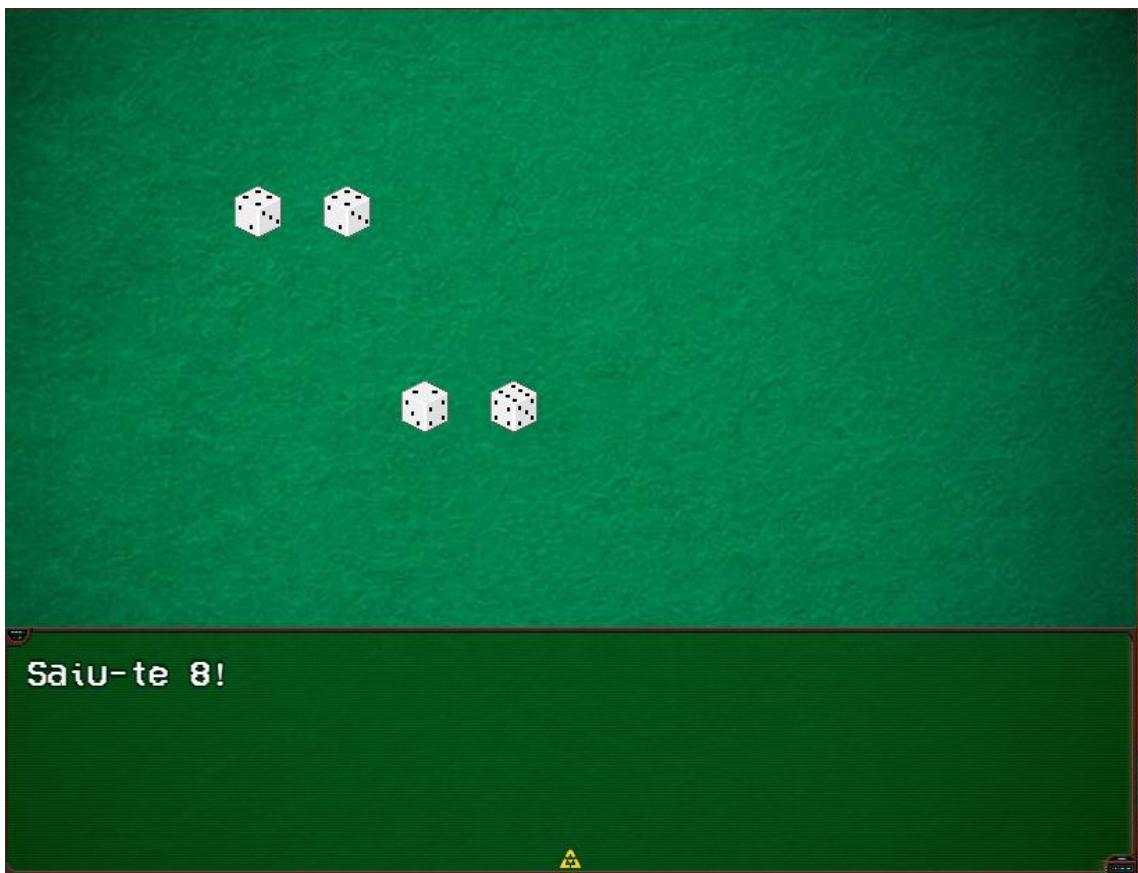


Figura 44 - Plugin de Jogo de dados contra Npc

O jogo contra um npc é semelhante pois iremos usar os mesmos métodos duas vezes e guardar o resultado para comparar no final.

Por último foi realizado um jogo com semelhanças ao black jack mas muito mais simples. No sprint 1 fizemos o jogo à base de mensagens e variáveis com eventos porém nós queríamos que o jogo fosse semelhante aos anteriores, por isso usamos como nos jogos anteriores uma scene nova e Windows.

O jogo é mais simples que o blackjack tradicional, o utilizador faz uma aposta e será tirada uma carta aleatória, após isso será perguntado se a próxima carta será maior, menor ou igual. Se o jogador acertar ganha o dobro da aposta.



Figura 45 - Jogo de BlackJack

Para realizar este plugin usamos windows com comandos e as funções de draw picture para adicionar o texto e as imagens.



Figura 46 - Plugin de Jogo de Blackjack



Figura 47 - Plugin de Jogo Blackjack

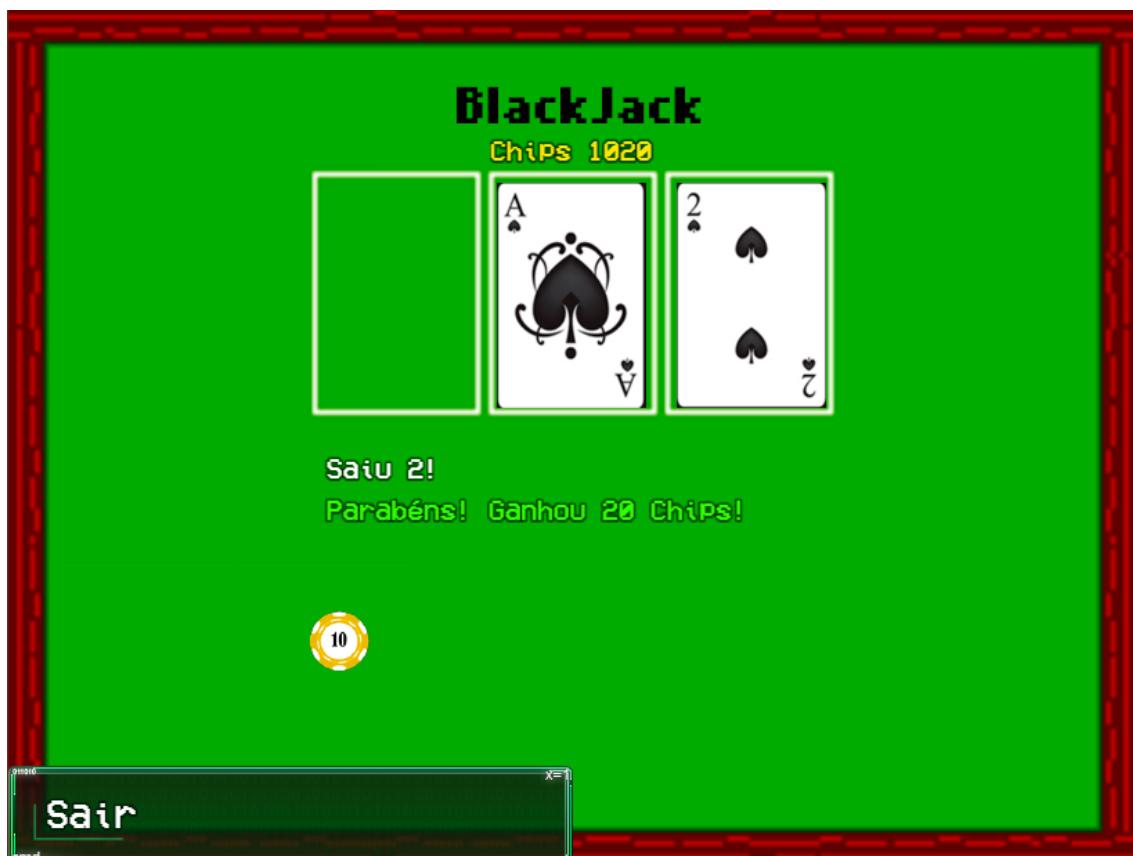


Figura 48 - Plugin de jogo de blackjack

As Scenes e Windows funcionam em uma hierarquia, em que possui 4 funções obrigatórias, o Initialize para herdar os bases, o Create para criar os componentes, o Start para inicializar os componentes e o update para atualizar os componentes a cada frame. Opcionalmente, caso o utilizador não mate a scene (usando o pop) e queira eliminar subcomponentes utiliza a função terminate.

O ciclo de vida dos componentes é o seguinte:

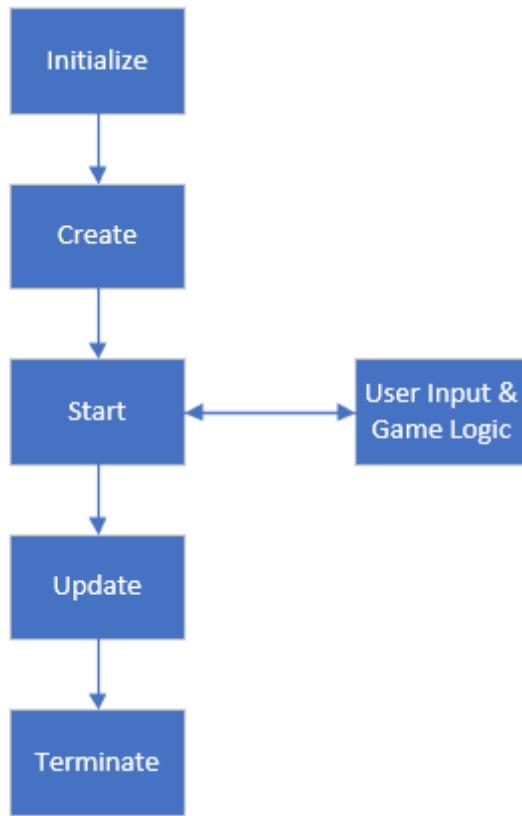


Figura 49 - Ciclo de Vida de um componente

## 4.8. Plugins Multiplayer

Estes plugins foram desenvolvidos pelo Hugo Henriques e pelo Fabian Nunes.

Com o intuito de adicionar alguma interação entre os jogadores e aumentar o dinamismo do projeto decidimos criar o modo multiplayer. Este modo tem como objetivo colocar dois jogadores num mapa para combaterem entre si, os jogadores irão aparecer em lados opostos do campo de batalha e terão que procurar nas caixas espalhadas pelo mapa as armas para lutarem (posteriormente pensamos em adicionar coletes que diminuem o dano, munição própria para cada arma, entre outras funcionalidades), quem conseguir derrotar primeiro o adversário ganha a partida e ganha também 1 ponto para o ranking online de jogadores.

Este modo foi inicializado no sprint 2 pelo Hugo Henriques criando a comunicação entre dois jogadores em localhost, utilizando websockets criadas

na tecnologia Ratchet PHP que será abordada mais a frente. No sprint 3 o Fabian Nunes dedicou-se a realização do servidor onde estariam guardadas as informações de cada utilizador como diversas funcionalidades necessárias “in game” para este modo funcionar.

O modo de multiplayer é um grande desafio, pois esta área de game devolpment está praticamente inexplorada no RPG Maker devido a existirem engines muito mais aptos desta tarefa, porém utilizando os conhecimentos adquiridos no curso nós iremos conseguir criar um sistema para um multiplayer simples.

É de notar que já existem dois plugins que são o Alpha Net e o Online Core, porém um deles foi abandonada e outro exige uma configuração específica não permitindo utilização de diversos plugins e singelplayer sendo que é focado para MMORPG. O objetivo do nosso multiplayer é ser Match Based e ser compatível projetos existentes e com a possibilidade de utilizar plugins externos criados por nós ou outros developers. Sem falar que o maker será compatível com qualquer tipo de backend ou base de dados, enquanto plugins como o Online Core utilizavam tecnologias fixas.

### **4.8.1. REST API**

Para conseguirmos obter dados e enviar para uma base de dados seria necessário ter um servidor que possuísse uma API Rest para fazer pedidos entre o cliente e o servidor, para isso existiam inúmeras tecnologias possíveis, sendo a primeira escolha o Slim PHP, por ter sido uma tecnologia falada numa cadeira anterior e por ser PHP (tal como o ratchet) mas devido a vários problemas na sua implementação e por ter uma documentação muito fraca decidimos mudar de tecnologia e foi escolhida a framework Spring no Java, esta escolha deveu-se a já termos trabalhado com Java Rest no curso e por causa de Spring ter uma documentação excelente.

Para ser mais fácil de entender o código e para modificar no futuro foi seguida a convenção MVC do Spring:

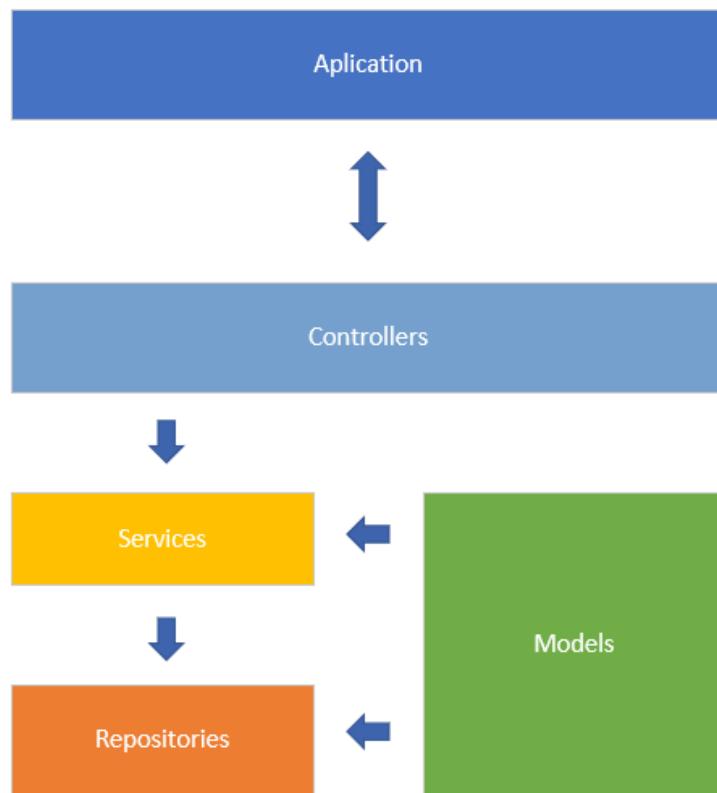


Figura 50 - MVC Spring

Deste modo iremos ter as classe controller que irão receber request e enviar respostas para o cliente (aplicação), os controllers irão chamar services que irão tratar da lógica com a base de dados e com os models (estruturas de dados) os serviços irão receber os dados de querys a base de dados que serão feitas nos repositories (cada tabela terá uma interface repositorie). Desta forma existe uma divisão de onde serão feitas as coisas.

Foram criadas diferentes rotas para as diferentes ações:

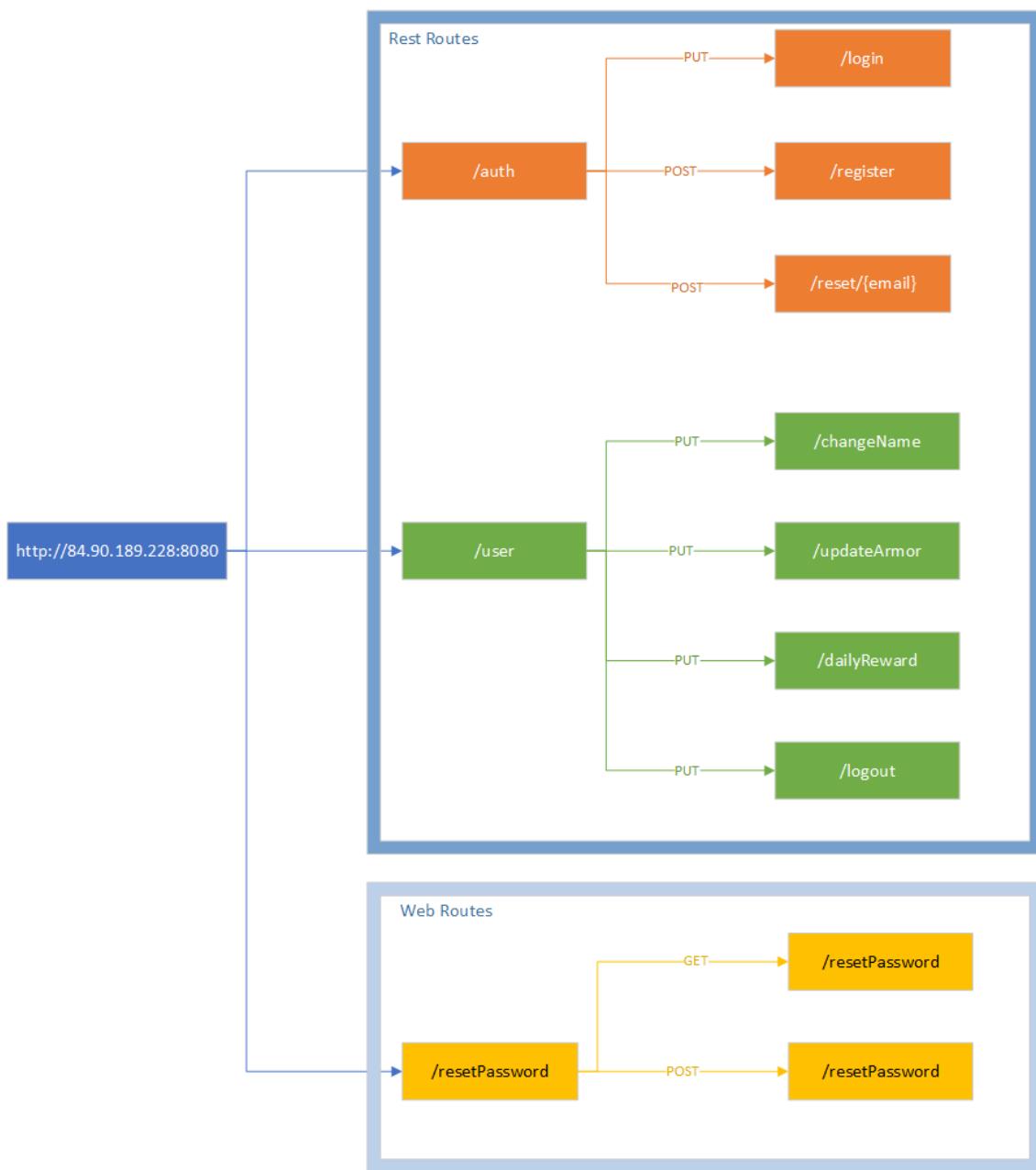


Figura 51 - Endpoints

A rota /auth é relativa a ações de autenticação como login, registo e o pedido de reset da password.

A rota /user é relativa a ações que são feitas durante o jogo como mudar de username, melhorar a armadura, receber o daily reward ou fazer logout do jogo.

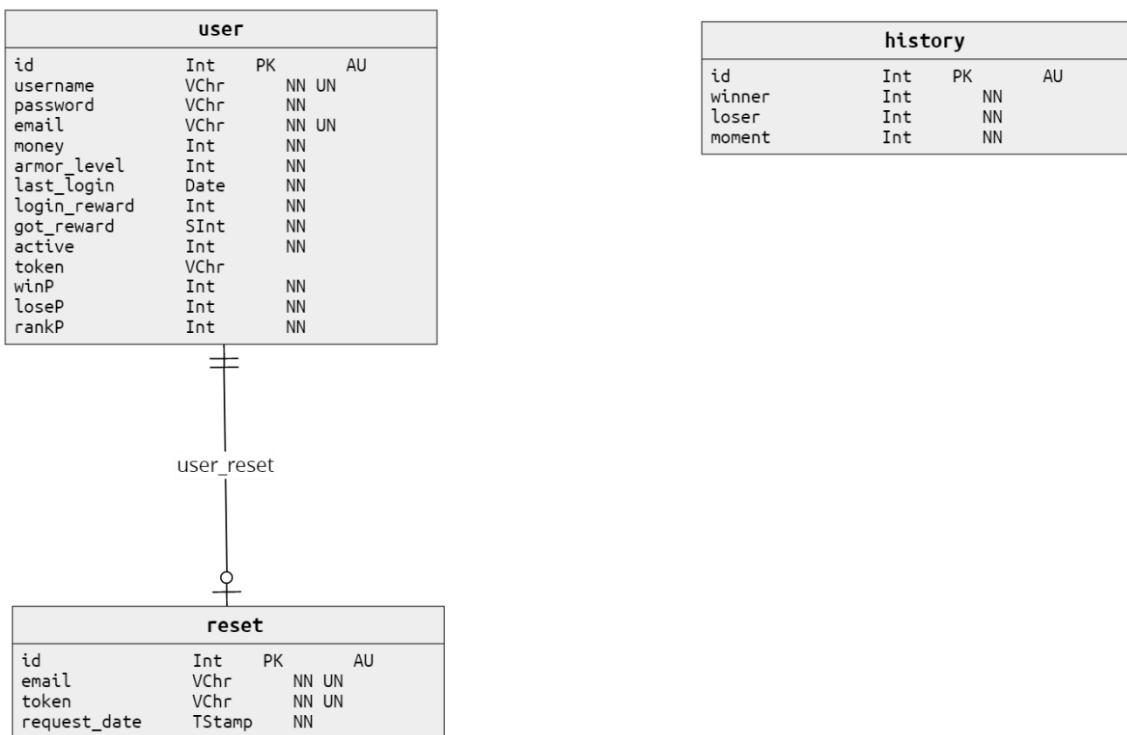
Para fazer reset da password foi necessário criar uma web API pois o reset da pass é feito a partir de uma página HTML que está no servidor web, sendo que neste caso não temos Rest mas web normal apesar de os pedidos serem a mesma Http.

A rota /user têm um aspeto importante pois ela requer um header authorization que será uma token gerada ao fazer login e será guardada na base de dados e pelo cliente, sempre que quiser fazer um pedido desta rota terá de enviar esse token, isto é um método de segurança implementado semelhante ao JWT. Existem métodos universais muito mais seguros como OAuth2 porém esses métodos são muito pesados e “overkill” para um projeto desta natureza, ainda mais quando um jogo deve ser o mais rápido possível.

## 4.8.2. Base de Dados

Os dados estão a ser guardados numa base de dados MySQL, atualmente possui 3 tabelas, a tabela user que possui os dados do utilizador como id, username, email, password etc. A tabela reset onde estarão os pedidos de reset de password, e a tabela history onde estará o histórico de cada batalha multijogador.

A base de dados possui eventos que funcionam como triggers em Oracle estes eventos vão de x em x tempo realizar alterações na base de dados, existe um evento que vai de 12 em 12 horas ver se existe algum pedido na tabela reset que tenha mais de 1 dia se tiver será apagado. Na tabela user existe um evento que de 1 em 1 hora vai ver a última ação de cada user e se essa foi a mais de uma hora será alterado o valor active para offline (de 1 para 0) e será eliminado o token de sessão. Para ter uma ideia melhor apresentamos aqui o modelo conceptual da base de dados gerada no Onda.



**Figura 52 - Modelo Conceptual**

### 4.8.3. Autenticação

Dentro do jogo irá haver um menu onde o jogador poderá realizar as diferentes ações antes que entre no multiplayer.



Figura 53 - Menu de entrada de multiplayer

#### 4.8.3.1. Registo

Para fazer o registo o utilizador vai fazer um pedido POST para o url /auth/register, onde irá mandar no corpo da mensagem o username, email e password em formato JSON, o username terá de ser único e ter pelo menos 8 caracteres, a password terá de ter pelo menos 8 caracteres, 1 letra maiúscula, 1 letra minúscula e 1 dígito e por último o email terá de ser único.

A palavra passe será encriptada utilizando o método BCrypt.

Caso o registo ocorra com sucesso será enviado um email para o email desejado.

As respostas que o servidor vai mandar ao registo serão:

**Tabela 5 - Resultado do Registo**

Código	Resultado
201	Registo com sucesso
401	Caso um campo esteja vazio, ou o campo seja inválido.
409	Caso já exista esse valor na base de dados

#### **4.8.3.2. Login**

Para realizar login é realizado um PUT request para /auth/login, o request levará no body o email e a password. Se os dados estiverem corretos será mandado para o cliente as informações do user sendo atualizados alguns dados importantes que serão:

O estado (ative) este estado reflete se o utilizador está offline, online na lobby, a espera de match making ou se estiver a jogar contra outro jogador.

**Tabela 6 - Estado do user**

Código	Significado
0	Offline
1	Online na lobby
2	Online a espera de partida
3	Online e a jogar contra outro jogador

Será também atualizado o campo de último login para o dia atual.

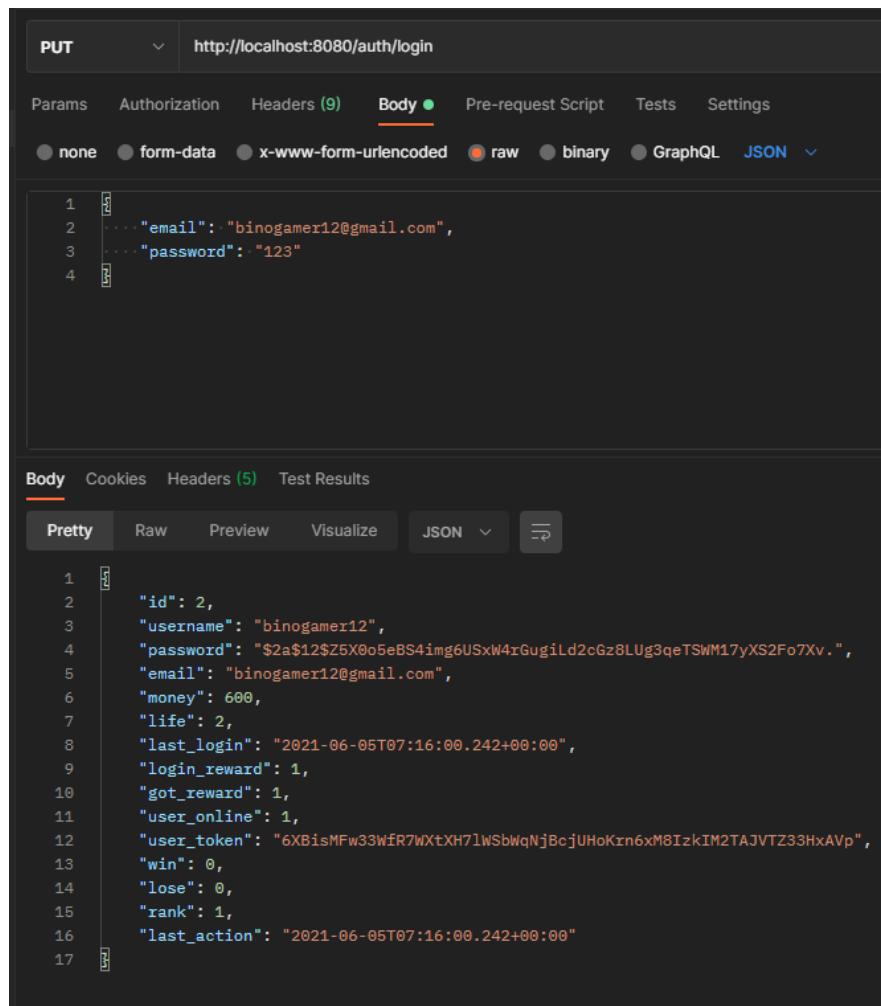


Figura 54 - Demonstração no postman

#### 4.8.3.3. Reset

Caso o utilizador se esqueça da sua palavra passe poderá pedir um reset de pass, para isso basta introduzir o seu email. Após isso será gerada uma token e uma timestamp que será guardada na base de dados, esse pedido terá a validade de um dia, caso o utilizador deixe passar o pedido será apagado da base de dados, após fazer o pedido o utilizador irá receber um mail com um link para a página de reset de password. Após mudar a password será eliminado o pedido da base de dados e a password será atualizada.

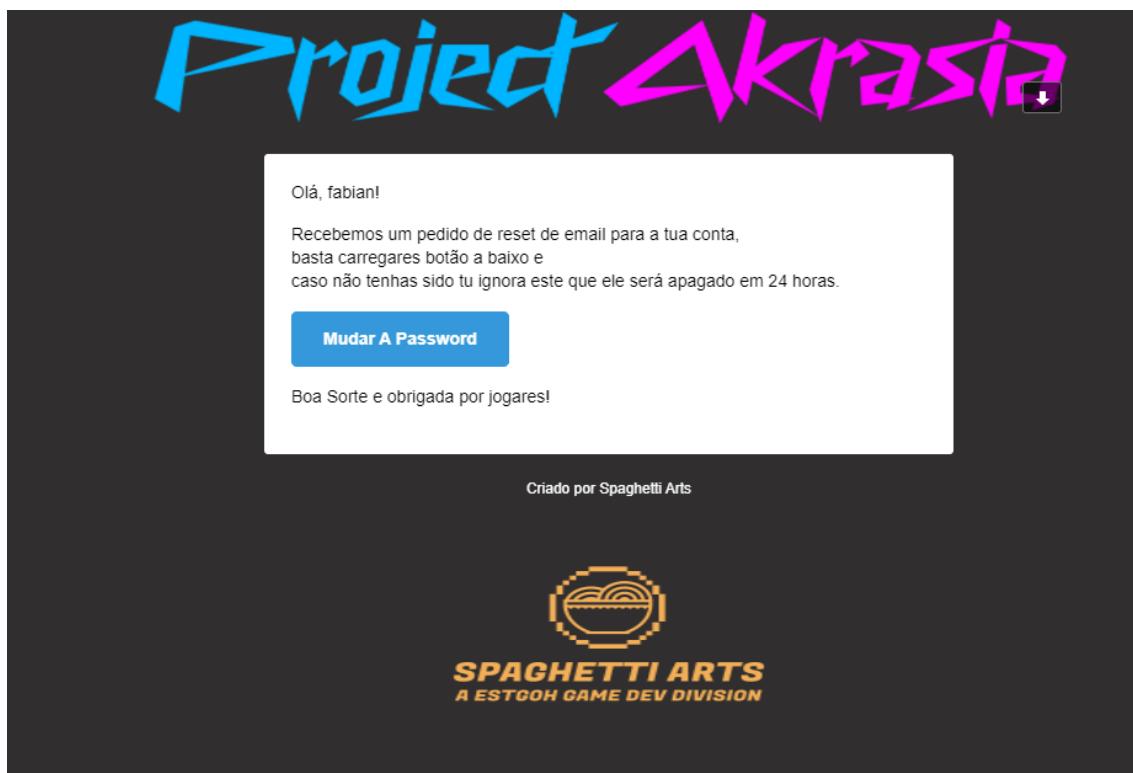


Figura 55 - Email de Reset

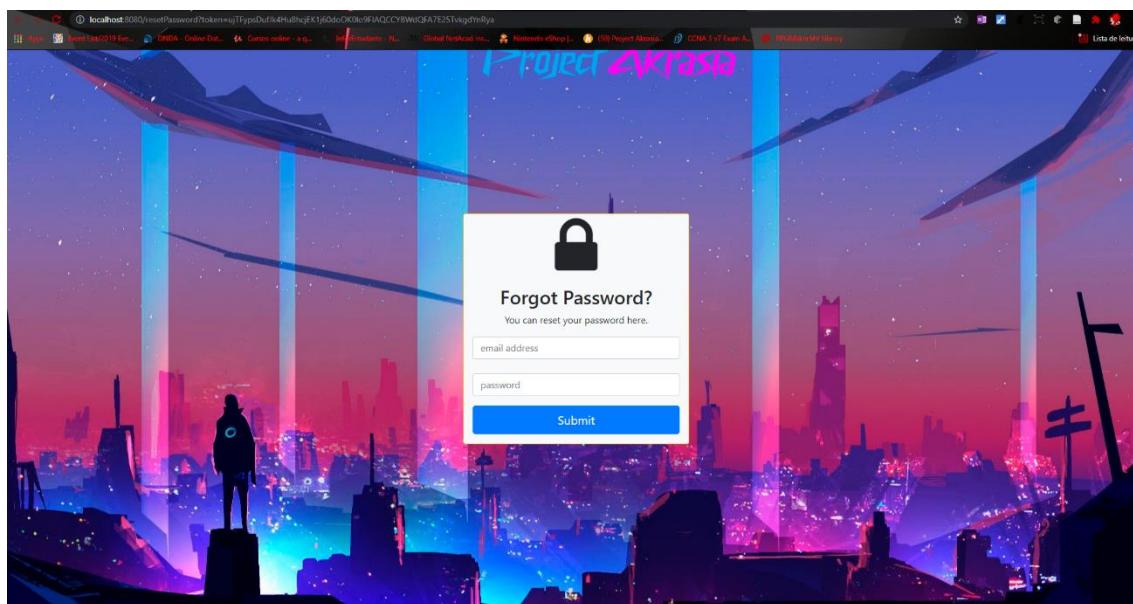


Figura 56 - Página de Reset de password

#### 4.8.3.4. Front end

Para a realização da autenticação no jogo foi preciso criar um plugin pois o maker não traz elementos de UI para escrita, por causa disso era impossível usar os métodos de sprints anteriores que recorriam a herança de partes da UI e modificavam se a gosto. Como o jogo é basicamente uma janela em localhost é possível utilizar HTML e CSS só que têm de ser feito recorrendo ao Javascript e a uma técnica chamada inner HTML que permite utilizar HTML dentro de ficheiros Javascript para modificar a DOM (a ideia do plugin é muito semelhante ao que as Frameworks de JS fazem), desta forma criou-se um formulário dentro de uma Scene do Maker utilizando as regras aprendidas anteriormente e acedeu-se aos campos por javascript.

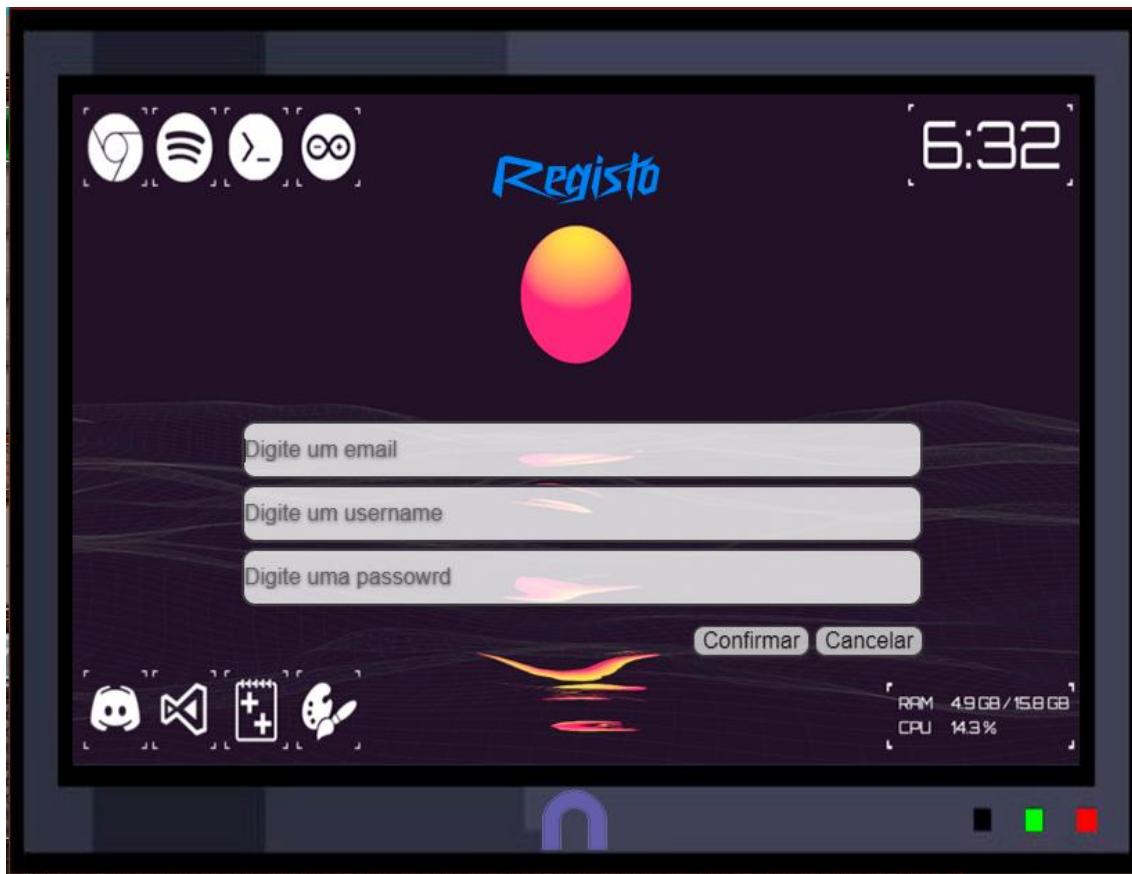


Figura 57 - Formulário dentro do jogo

Para comunicar com o servidor recorremos ao AJAX (Asynchronous JavaScript And XML), esta técnica permite fazer pedidos HTTP assíncronos a um servidor de forma assíncrona. Dependendo do código enviado de resposta, o maker irá responder de forma diferente ao jogador.

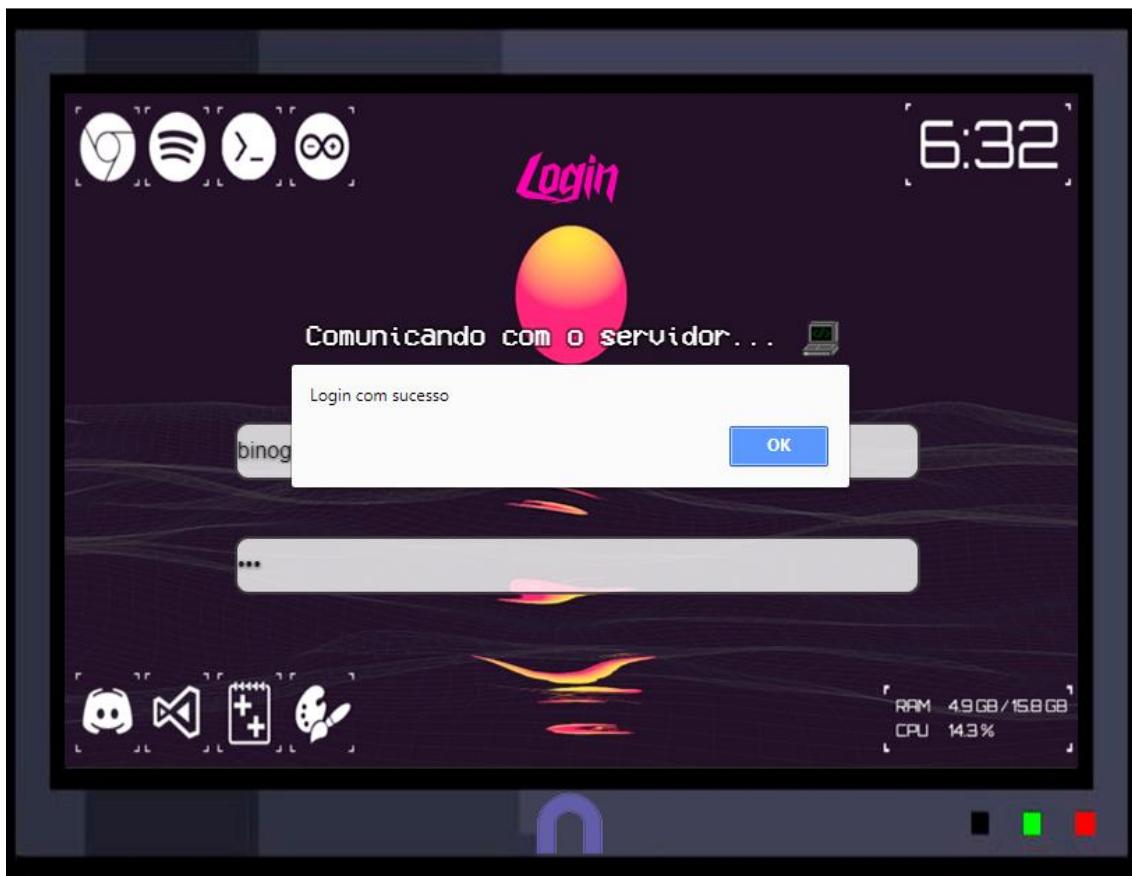


Figura 58 - Confirmação do servidor

#### 4.8.4. Lobby

Após fazer login o utilizador será transportado para uma lobby onde irá receber o daily reward, poderá atualizar a sua armadura, ver a sua informação ou entrar em uma partida PVP contra outro jogador.

##### 4.8.4.1. Daily Reward

O jogador irá receber sempre que fizer login pela primeira vez no dia receber uma recompensa (caso volte a fazer login no mesmo dia já não irá aparecer), essa recompensa será em forma de “dinheiro fictício”, caso faça logins todos os dias cada dia irá ganhar uma recompensa maior até chegar a uma sequência de 7 dias a partir do qual irá sempre receber o mesmo. Caso não faça login no próximo dia o utilizador voltará para o primeiro dia.



Figura 59 - Reward diário

#### 4.8.4.2. Atualizar Armadura

O jogador poderá aumentar a sua vida total para os combates PVP, o nível da armadura pode ir até ao valor 10, sendo cada nível mais caro, a vida resultante será calculada dentro do jogo baseado da seguinte forma:

$$\text{vida} = \text{armadura} * 100$$

A base de dados só irá guardar o nível de armadura.

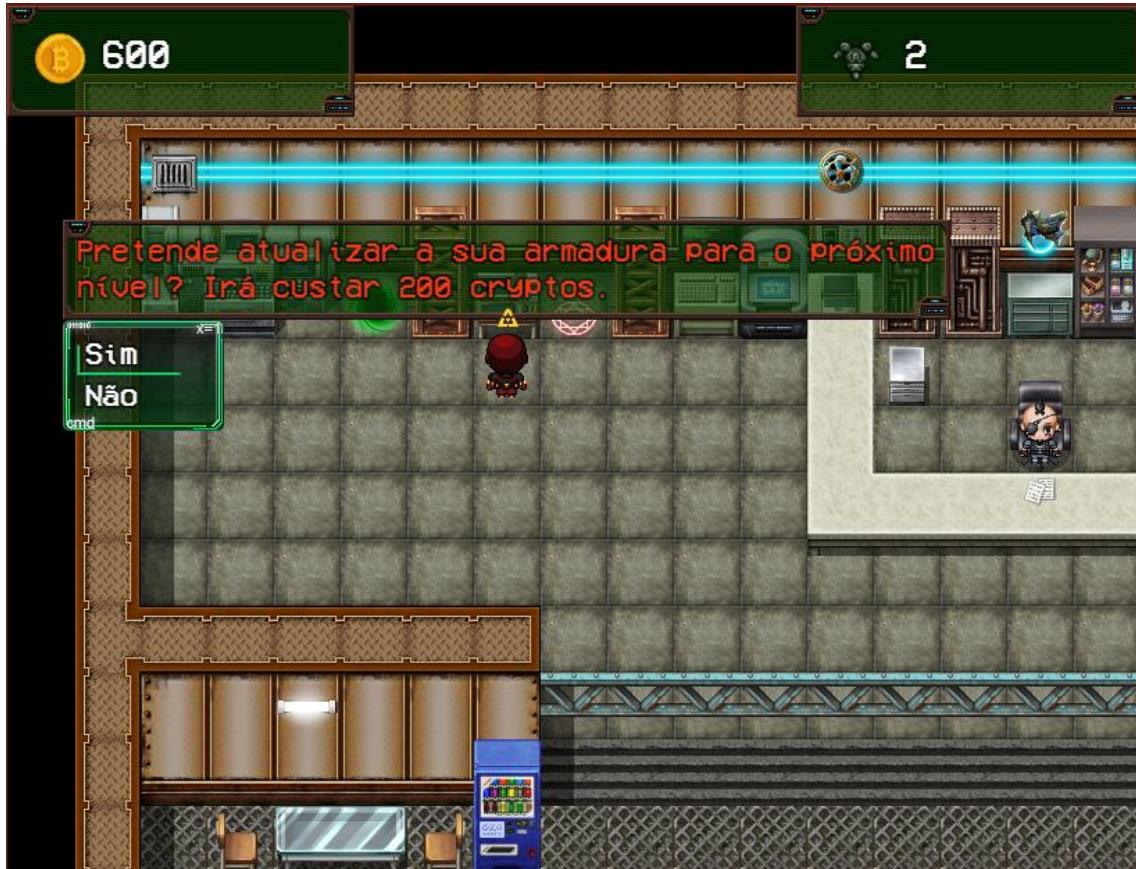


Figura 60 - Atualizar armadura

#### 4.8.4.3. Rank

Um dos aspectos que queríamos implementar no multiplayer era um sistema de ranking, quanto mais jogos o utilizador jogar e ganhar mais alto sobe no ranking.

De momento o PVP ainda está em desenvolvimento logo a ideia que temos é de existirem 3 ranks: Bronze, Prata e Ouro.

O rank que os utilizadores têm depende da quantidade de jogos que já realizou e quantos ganhou da seguinte forma:

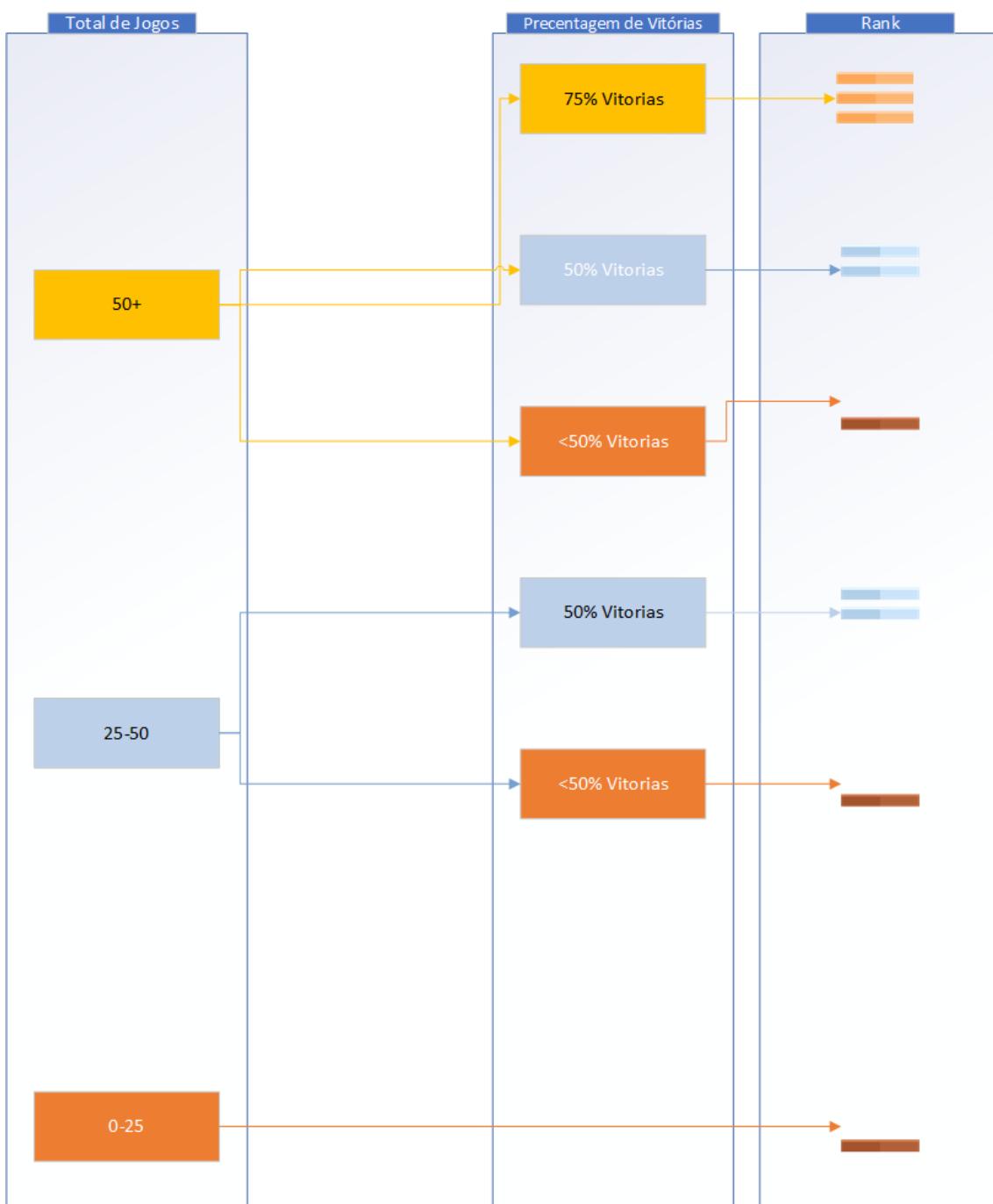


Figura 61 - Ranking System

Isto é um sistema de raking muito simples e serve mais como demonstração, normalmente um ranking system segue diversas regras e tem diversos patamares, mas aí já estaríamos a entrar em outro tipo de jogo e também não é o objetivo deste projeto.

O utilizador poderá visualizar o seu rank dentro do jogo na lobby onde terá a informação sobre a sua conta.



Figura 62 - Informação do utilizador

**IMPORTANTE:** Pela natureza do Rest não existe uma forma perfeita de saber se um utilizador ainda está a jogar ou não. Idealmente se o user fizer logout o sistema irá alterar o seu estado para offline e apagar o token de acesso gerado. Caso o utilizador termine o executável de outra forma foi realizado um método para conseguir saber se o user ainda está no jogo. Sempre que é feito um pedido HTTP ao servidor será guardada a timestamp atual. A base de dados terá um evento a correr de x em x tempo que irá percorrer os users e se algum tiver uma timestamp com mais de 1 hora o sistema assume inatividade e muda o estado para offline e apaga a token. Caso o utilizador ainda esteja a jogar e tente fazer um pedido depois disso o sistema irá notar e vai o mandar de novo para o login.

Para entender melhor a Rest API geramos o diagrama de classes do IntelliJ porém devido ao seu tamanho enviamos o link para que o leitor o possa abrir e fazer zoom, [Link do diagrama de classes](#).

## 4.8.5 Jogabilidade e Comunicação

Num jogo multiplayer necessitamos que ambos os jogadores estejam a visualizar o mesmo plano de jogo, ou seja o que acontece no lado do jogador 1 tem de acontecer no lado do jogador 2 e vice-versa.

### 4.8.5.1 Movimentação das personagens

Para comunicação das posições dos jogadores cada um necessita de estar constantemente a enviar a posição exata onde está (X,Y) e também a receber a localização do adversário, após receber essa localização é feita uma simulação do adversário que é posicionado de acordo com esta informação.

para além da posição necessitamos de saber a direção para a qual o jogador Se encontra para isso comunicamos constantemente essa direção (8 cima, 2 baixo, 4 esquerda , 6 direita, de acordo com o teclado numérico ) entre os jogadores.

Obtido assim a posição e a direção é percebido então a movimentação do adversário e reproduzido no npc simulado.

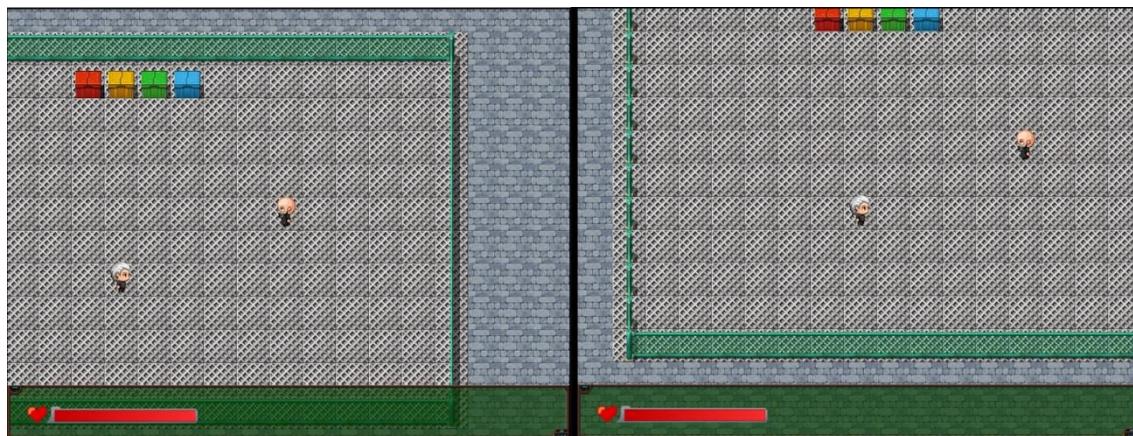


Figura 63 - Dois clientes a comunicarem

### 4.8.5.2 Sistema de disparo

Para a realização do combate foi utilizado o sistema criado anteriormente e adaptado para nova funcionalidade multiplayer.

No combate multiplayer é necessário que ambos os jogadores vejam os seus disparos e os disparos feitos pelo adversário, para que tal aconteça é necessário que estes sejam constantemente a ser comunicados através do websocket

Para realização desta funcionalidade, cada vez que é acionada a tecla de disparo é obtida essa informação do jogador e é enviado no conjunto dos dados que estão a ser

transmitidos através do websocket, o adversário por sua vez irá receber informação da ação do disparo e irá reproduzi-la do seu lado de acordo com os dados obtidos.

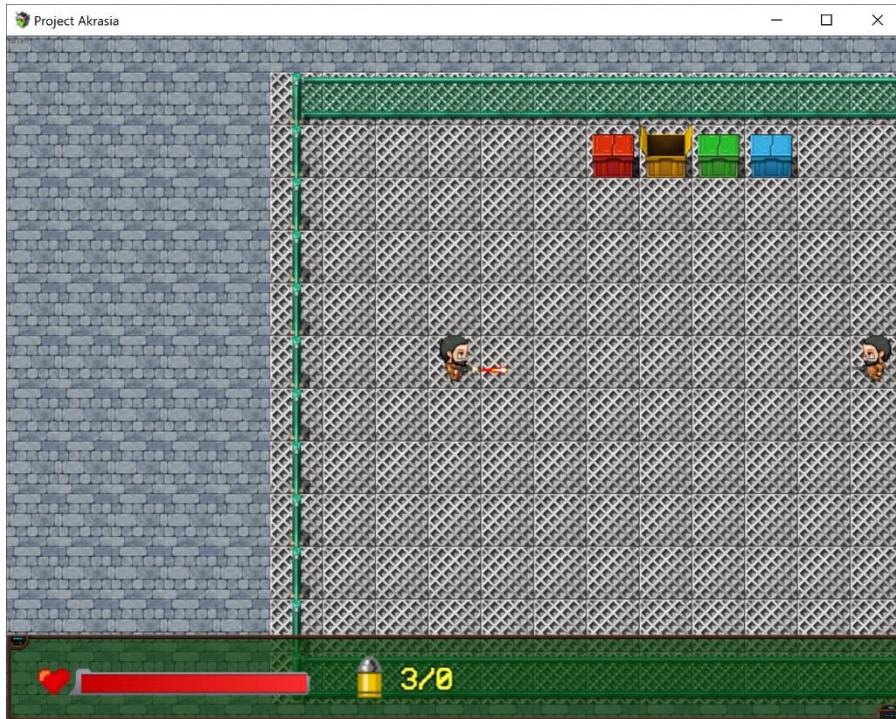


Figura 64 - Disparo entre jogadores

#### 4.8.5.3 Gestão da vida

Assim como os 2 sistemas anteriormente mostrados a vida de cada jogador tem de ser comunicada para que a mesma informação esteja correcta dos 2 lados. Esta é uma funcionalidade que se encontra em desenvolvimento mas que por sua vez funciona de forma similar ao que foi mostrado anteriormente. Pois é enviada a percentagem de vida que cada jogador e é recebido a vida do adversário em tempo real, assim sendo quando um jogador sofre dano o adversário observa que a sua vida desce.



Figura 65 - UI de Multiplayer

## 4.9. Plugin de Eventos Random

Este plugin foi desenvolvido pelo Hugo Henriques.

Para melhorar o sistema de combate entre os jogadores do multiplayer como referido anteriormente, decidimos implementar um algoritmo que espalhasse aleatoriamente e assertivamente os diversos eventos (objetos) pelo mapa (como por exemplo caixas com armas) e que mudasse os objetos de sítio cada vez que um jogo fosse iniciado. Este plugin está em desenvolvimento conjunto com o plugin do multiplayer e atualmente temos uma demo (em singleplayer) onde podemos ver o sucedido que as imagens a seguir representam:

Na imagem a seguir podemos ver 3 caixas de armas que foram espalhadas aleatoriamente pelo mapa:

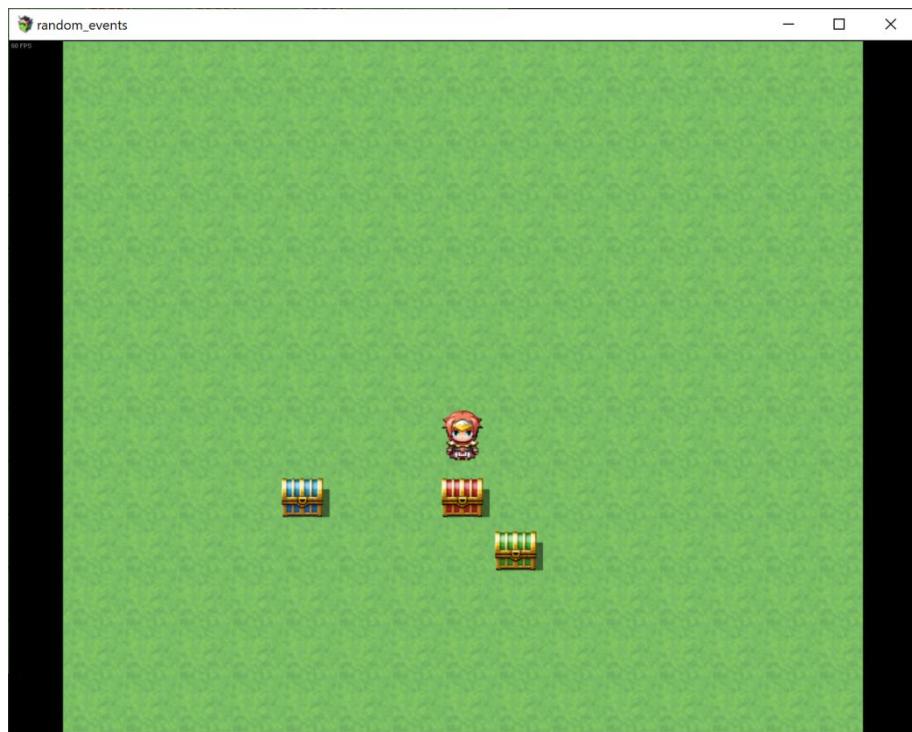
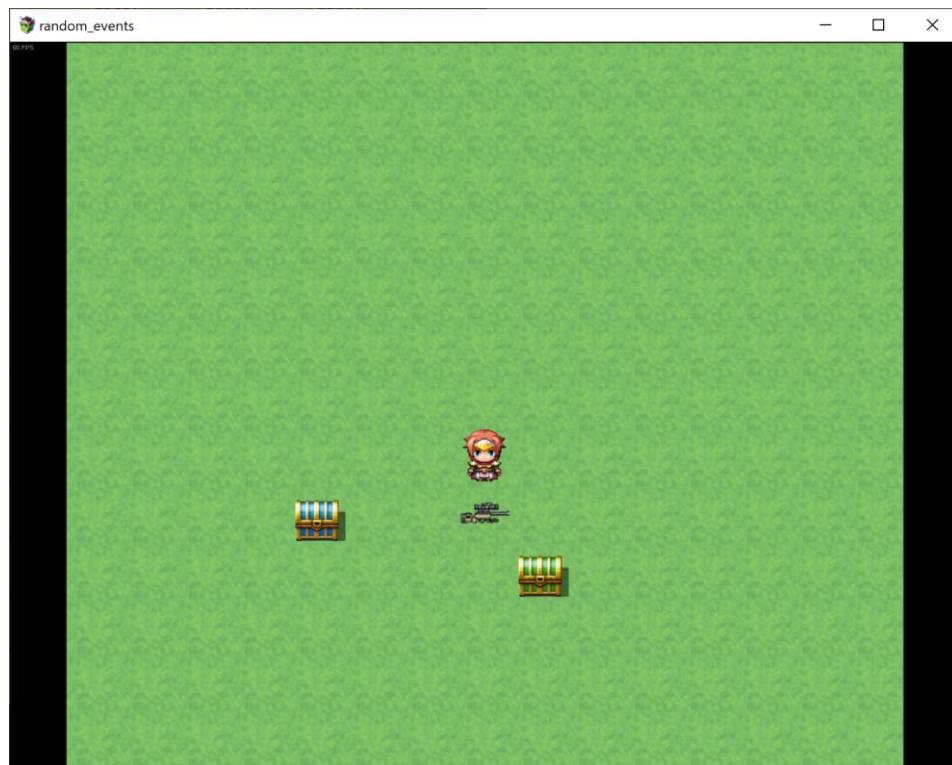


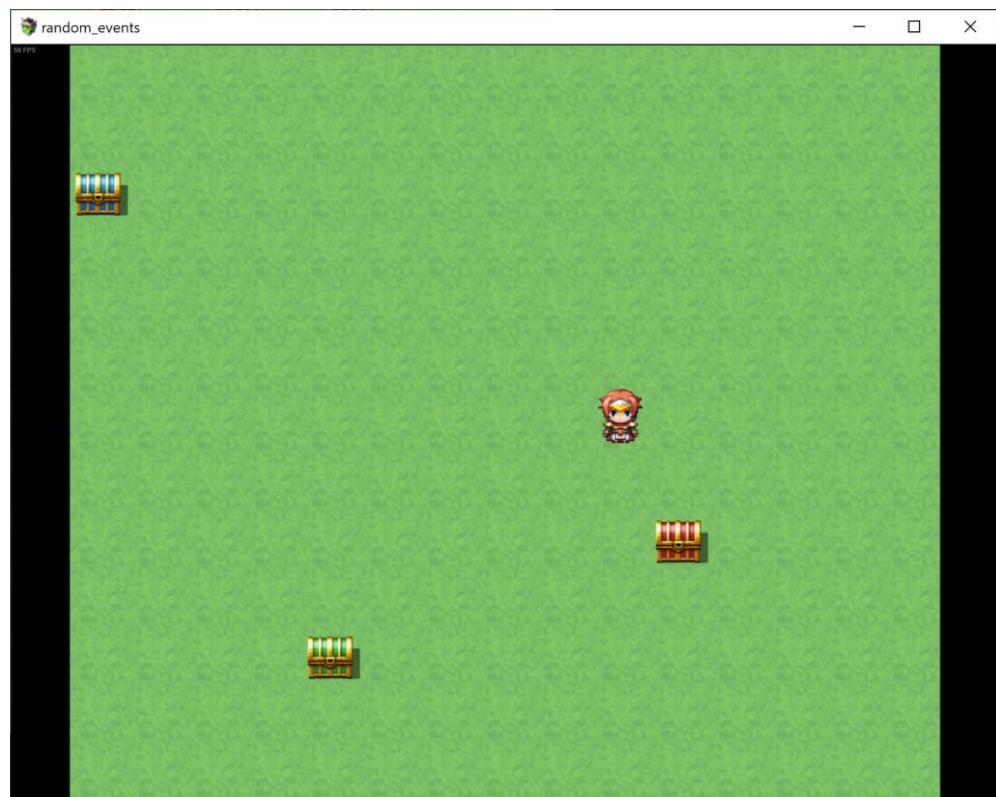
Figura 66 - Plugin de Eventos Randomizados

Na continuidade da imagem anterior podemos ver também que na caixa vermelha encontramos uma arma de precisão.



**Figura 67 – Plugin de Eventos Randomizados**

Agora ao iniciarmos um novo jogo verificamos então que as caixas mudaram de sítio.



**Figura 68 - Plugin de Eventos Randomizados**

E neste momento dentro da caixa vermelha temos uma pistola.

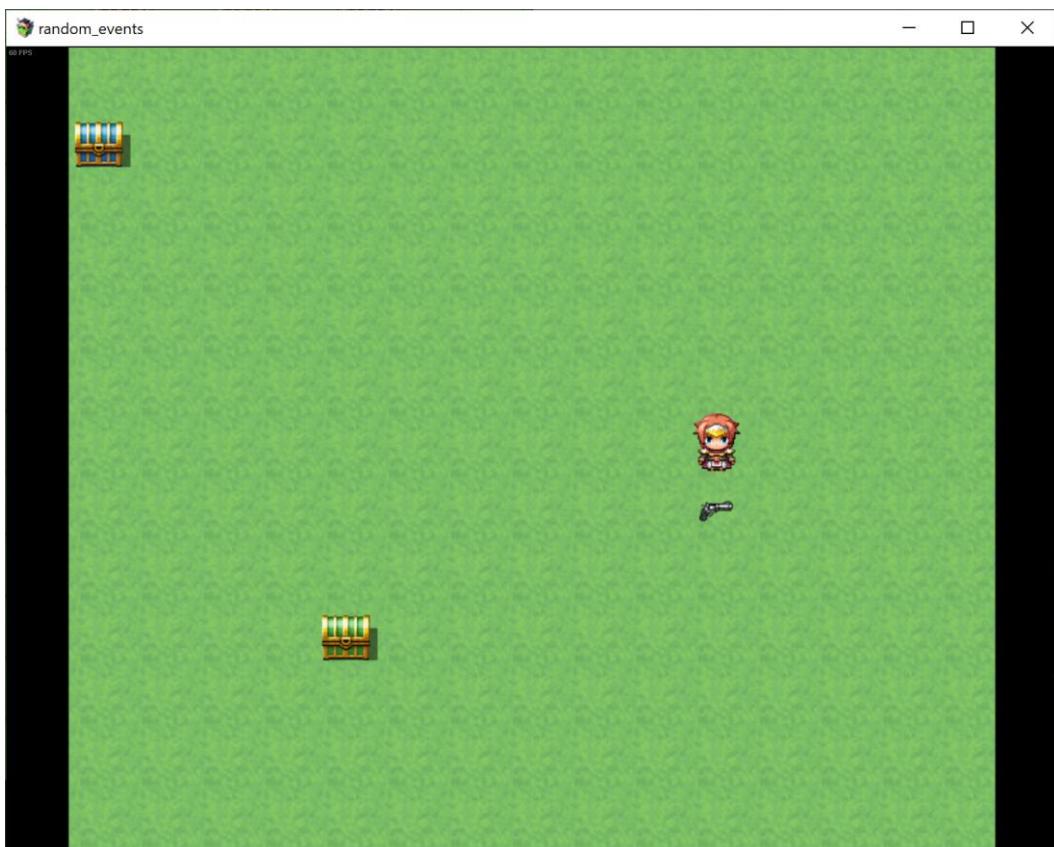


Figura 69 - Plugin de Eventos Randomizados

Através destas imagens podemos ver o funcionamento geral do plugin pois tanto as caixas como os objetos dentro delas mudam cada vez que um jogo inicia.

## 4.10. Servidor Ubuntu Server em Raspberry Pi

A implementação do servidor foi feito pelo Afonso Vitório.

De forma a hospedar os vários serviços do servidor da nossa aplicação, como WebSockets, Bases de Dados e API, decidimos criar o nosso próprio servidor.

Um servidor é um computador que fornece serviços de forma contínua a uma rede de computadores e como tal decidimos que também seria interessante criarmos o nosso próprio servidor, de forma a hospedar os serviços que precisamos no nosso jogo e demonstrar conhecimentos informáticos adquirimos.

Para escolha do hardware, optamos por um Raspberry Pi modelo 3B+, pois é um computador que já possuímos e o seu uso é muito pouco incomodativo para ter num apartamento partilhado.

Para escolha do sistema operativo optamos pelo Ubuntu Server 20.04 LTS, relativamente a outros sistemas operativos mais populares para este hardware como o Raspbian. A escolha deste sistema operativo deveu-se ao facto de ser um software arquitetado para servidores e não consumir recursos, com softwares que não iríamos utilizar nesta máquina, como interfaces gráficas, etc.

Esta versão Ubuntu Server é de arquitetura aarch64, comparativamente ao Raspbian, que é armhf, a vantagem do uso desta arquitetura 64 bits num processador ARM dá-nos algumas vantagens como uma maior velocidade de download/upload. Ao usar a versão LTS (Long-Term Support) do Ubuntu Server diminuímos também a chance de ter problemas ao longo do projeto com atualizações e configurações.

**Figura 70 - Especificações do Raspberry Pi 3B+**

Foram criadas contas para todos os membros do grupo e foram instalados todos os serviços necessários, como apache e mysql. Foi feito um encaminhamento de portos no nosso router de forma a que todos os membros possam aceder ao servidor e aos seus serviços através da internet.



**Figura 71 - Teste ao WebSocket através do IP Público do router**

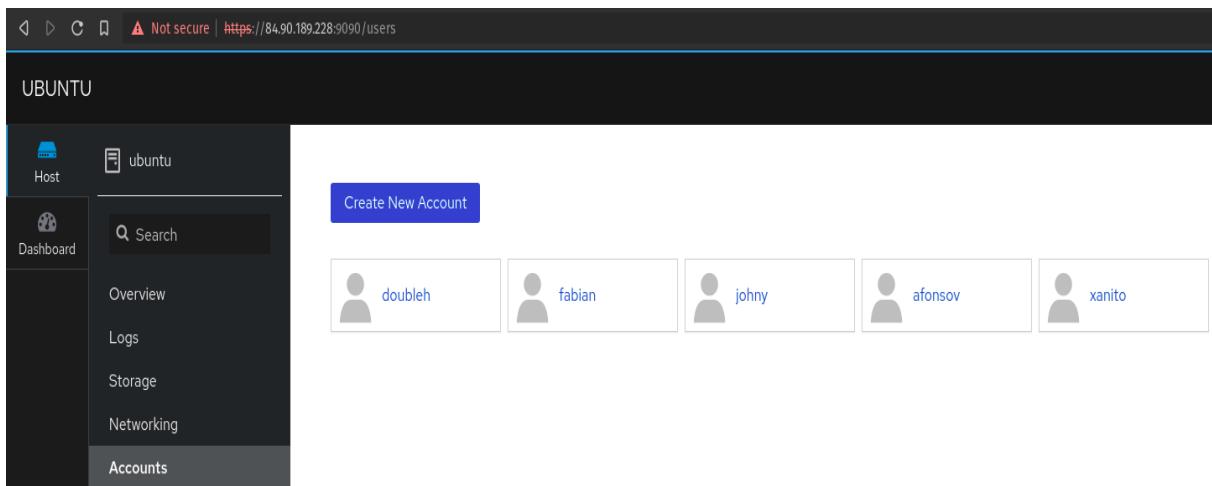


Figura 72 - Interface Cockpit no IP público do Servidor com os utilizadores

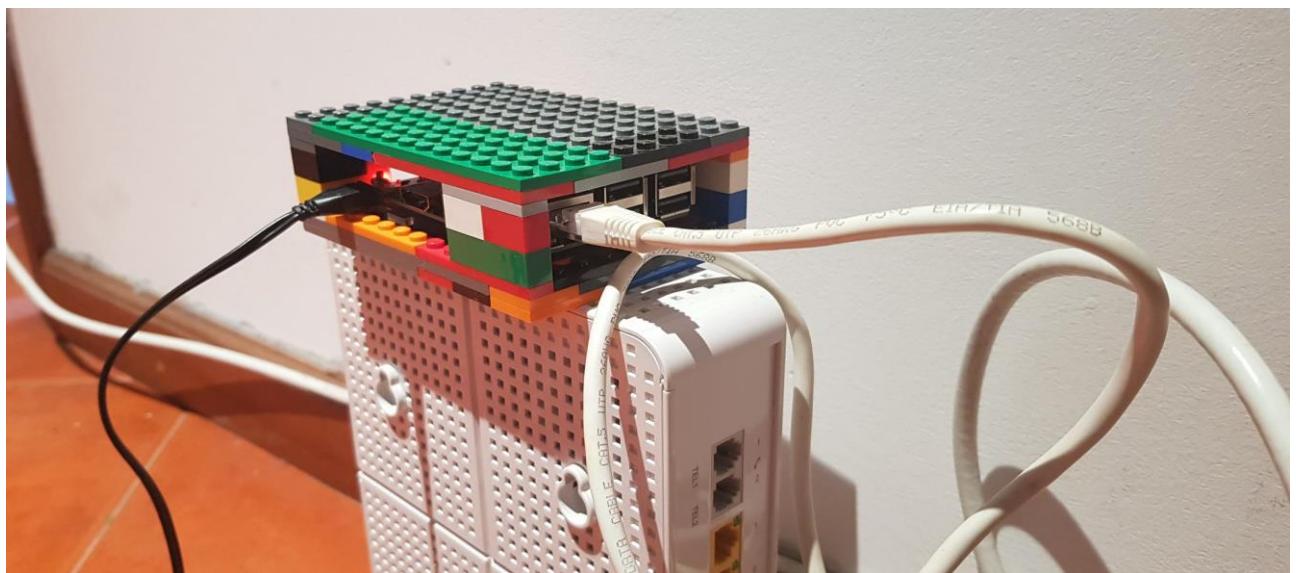


Figura 73 - Raspberry Pi no seu habitat

## 4.11. Instalação de Serviços no Servidor

A instalação de serviços no servidor foi feita pelo Hugo Henriques.

### 4.11.1 Instalação do Servidor HTTP

Dado a necessidade de possuir um servidor que atendesse às necessidades do plugin multiplayer online foi instalado o serviço Nginx, que é um servidor leve de HTTP, proxy reverso e proxy de e-mail IMAP/POP3.

O Nginx consome menos memória que o Apache, pois lida com requisições Web do tipo “event-based web server”, enquanto o Apache é baseado no “process-based server”, podendo trabalhar juntos.

### **4.11.2 Instalação do PHP**

De forma a tratar do conteúdo dinâmico do serviço por nós pretendido foi instalado a versão 7.4.3 do PHP. O PHP é uma linguagem interpretada livre, usada originalmente apenas para o desenvolvimento de aplicações presentes e atuantes no lado do servidor, capazes de gerar conteúdo dinâmico.

### **4.11.3 Instalação do Composer**

O Composer é um gestor de pacotes em nível de aplicação para a linguagem de programação PHP que fornece um formato padrão para gerir dependências de software PHP e bibliotecas necessárias.

Com a instalação desta ferramenta o servidor ficou então preparado para hospedar e sustentar o nosso serviço de comunicação necessário ao multiplayer.

## **4.12. Pesquisa e implementação da comunicação com websockets.**

Pesquisa e implementação feita pelo Hugo Henriques

Num jogo multiplayer necessitamos que haja uma constante comunicação de dados entre os jogadores para que o jogo aconteça, de forma a implementar esse requisito foi feita uma pesquisa e análise de vários sistemas de comunicação web e uma forma de o colocar operacional de modo a que um jogador possa interagir um com o outro de qualquer parte do planeta.

Com a pesquisa foram encontrados os seguintes websockets:

- **PHP**
  - Ratchet
- **Node.js**
  - Socket.IO
  - WebSocket-Node
  - ws

- **Java**
  - Jetty
- **Ruby**
  - EventMachine
- **Python**
  - pywebsocket
- **Erlang**
  - Shirasu
- **C++**
  - libwebsockets
- **.NET**
  - SuperWebSocket

Com a realização de diversos testes a alguns dos websockets foi chegado a conclusão de que a tecnologia Ratched seria o que se adequava melhor as nossas necessidades.

O Ratchet é um websocket PHP, que consiste numa biblioteca (PHP) fracamente acoplada que fornece aos programadores ferramentas para criar aplicações bidirecionais em tempo real entre clientes e servidores através de WebSockets.

Após a escolha do método de comunicação foram instaladas as respetivas dependências, e foi criado um protocolo de comunicação para que haja ...

## 4.13. Requisitos Implementados

Tabela 7 - Requisitos

Requisito	Sprint	Afonso	Fabian	Hugo
História do prólogo	1	Principal	Principal	Secundário
Interações com ambiente e personagem	1	Principal	Principal	Principal
Hacking Minigame	1		Principal	
Design de Mapas do Prólogo	1	Secundário	Principal	Principal
Parallax Mapping	1	Secundário	Principal	Principal
Mecânica de Beber/Comer	1		Principal	Principal

Interação com veículos em movimento	1			Principal
HUD	1		Principal	
Sons Dinâmicos	1	Principal		
Combate Player	2		Principal	
Personalização de personagem e armamento	2			Principal
Remodelação do HUD	2		Principal	
Remodelação dos minijogos	2		Principal	
Criação de Title Screen Dinâmico e Menus	2		Principal	
Procedural Map Generator	2	Principal		
Animação dos inimigos (máquina de estados)	2	Principal		
Definição de controlos	2		Principal	
Início do desenvolvimento do multiplayer	2			Principal
Sistema de saving	2		Principal	
WebSockets Multiplayer	3	Secundário		Principal
Resolução de problemas de comunicação	3	Principal		Principal
Animação dos Inimigos (Pathfinding)	3	Principal		

Criação do servidor	3	Principal		
Instalação de Serviços no Servidor	3			Principal
REST API (backend) e Web (frontend e backend) para multiplayer em Spring	3		Principal	
Criação e gestão de base de dados MySQL	3		Principal	
Sistema de Reset de passwords por email (Criação de emails personalizados)	3		Principal	
Criação de Frontend no jogo para autenticação e reset de pass	3		Principal	
Criação de Lobby de Multiplayer com daily reward e upgrade system	3		Principal	
UI e HUD do multiplayer	3 e 4		Principal	
Comunicação Movimentação correta dos personagens no multiplayer	3			Principal
Sistema de Disparo e combate no multiplayer	3			Principal
Início do sistema de comunicação	3		Secundário	Principal

de Vida multiplayer				
Finalização do Sistema de comunicação de Vida	4(planeado)		Secundário	Principal
Continuação do Sistema de ataque no multiplayer com diversas armas	4 (Planeado)			Principal
Multiplayer completo	4 (Planeado)	Secundário	Principal	Principal
Sistema de ataque inimigo no singleplayer	4 (Planeado)	Principal		
Níveis da prova-conceito	4 (Planeado)		Principal	
Adaptação da animação de inimigos para um inimigo diferente (Boss)	4 (Planeado)	Principal		
Loja no Multiplayer	4 (Planeado)		Principal	
Criação de uma arcade com controlos (Opcional)	4 (Planeado)	Principal	Principal	Principal

De notar que os requisitos que colocamos para os próximos sprints, estão sujeitos a alterações, pois possivelmente haverá alguns que não se justifiquem e outros que se venham a mostrar mais importantes.

## 5. Plugins Externos

Como falamos previamente, game development não é diferente de outro software development, também aqui recorremos algumas vezes a bibliotecas externas para

auxiliar plugins que estamos a criar ou eventos. Durante o projeto nós tentamos utilizar o mínimo de códigos externos, porém também não faz sentido reinventar a roda e neste caso vários aspectos iriam levar sprints inteiros para desenvolver essas funcionalidades para conseguir realizar o que pretendemos.

Estamos atualmente a utilizar:

Tabela 8 - Plugins Externos

Nome	Autor	Funcionalidade
lavra Text Sound	lavra	Permite adicionar som as letras que vão aparecendo nas mensagens semelhante ao jogo undertale
YEP Core Engine	Yanfly	O core de funcionalidades para utilizar outros plugins deste autor
YEP Message Core	Yanfly	Altera como as mensagens do maker funciona e permite editar diversos aspectos delas
YEP Region Restrictions	Yanfly	Permite mudar dinamicamente dentro de scripts a passagem de uma tile
RS Message Align	biud436	Permite centrar as mensagens em pontos específicos da janela dinamicamente
HIME Custom Page Condition	Hime	Permite criar condições customizadas para eventos
Hime Self Variables	Hime	Cria self variables que funcionam como variáveis locais para cada evento
Kahs Core	Nilo K	O core de funcionalidades para

		utilizar outros plugins deste autor
Kahs Graphics	Nilo K	O core de gráfico para utilizar outros plugins gráficos deste autor
Kahs Advanced Lighting	Nilo K	O plugin implementa um lighting system dinamico utilizando a framework Pixi js
Orange Mapshot	Hudell	Permite construir tilesets no editor a partir de prints do ecrã (utilizado para development)
Galv Layer Graphics	Galv	Permite renderizar layers dinâmicos no ecrã (fog etc)
Galv Cam Control	Galv	Cria uma ou mais câmera dinâmicas para substituir a câmera estática do maker.
Galv Map Projectiles	Galv	Permite renderizar no mapa projéteis em tempo real via scripts
OcRam Audio EX	OcRam	Cria múltiplos bus de áudio que permite misturar sons para criar ambientes sonoros
Message Window Popup	Trioacotane	Permite criar janelas em formato pop up de mensagens

Vários destes plugins são utilitários e só introduzem funções que já vem por default em outros engines como Unity ou Godot, sendo que esta lista também não é estática pois irá estar em constante atualização e a cada sprint o nosso conhecimento do maker será maior e iremos remodelar certas partes realizadas em sprints anteriores, logo ainda não está definido se estes plugins se irão manter até ao final do projeto.

## 6. Conclusões

Neste capítulo vamos explicitar o que alcançamos, o que na nossa ótica correu bem neste sprint, o que não correu tão bem. Uma comparação deste sprint com o sprint anterior e o que pretendemos transportar para o próximo sprint.

### 6.1. Forças

Tal como no primeiro e no segundo sprint, também ficamos contentes com o resultado final. Ficamos contentes com os resultados que alcançamos tendo em conta a enorme carga de trabalho que existe noutras unidades curriculares.

No primeiro sprint o nosso foco foi principalmente aspectos não técnicos, da elaboração de um jogo, como o áudio, o movimento, o diálogo e interações. Tivemos uma grande carga de trabalho mas conseguimos ter o prólogo do jogo que pretendemos realizar, concluído. Contudo, apesar de serem tarefas criativas que nós apreciamos, não podemos demonstrar a nossa capacidade técnica.

No segundo sprint, o nosso foco foram tarefas mais técnicas da elaboração de um jogo e como tal mais relacionadas com a engenharia informática. Neste sprint exploramos e implementamos vários conceitos que permitem ter mecanismos e espaços para o combate no jogo. Consideramos que tivemos bons resultados.

Criamos vários plugins, que podem todos eles ser utilizados e melhorados por outros desenvolvedores neste motor. Dentro os quais um plugin de geração de mapas de forma aleatória através de autómatos celulares. Criamos também um plugin de animação de inimigos que utiliza uma máquina de estados. Criamos plugins de mini jogos. Criamos um plugin que permite alterar o mapeamento das teclas. Criamos um sistema de menus bastante interessante. Iniciamos o desenvolvimento do sistema de combate e iniciamos também o desenvolvimento do modo multijogador.

Ao longo deste terceiro sprint, continuamos o desenvolvimento da jogabilidade e do modo multijogador, tendo conseguido resultados bastante satisfatórios. Os inimigos conseguem encontrar um caminho para o jogador, já não ficam presos em obstáculos. É possível realizar registo e logins no multiplayer. Temos um sistema de recompensa de login diário. Temos um servidor nosso, que hospeda os serviços que necessitamos, já possuímos uma forma de comunicação entre clientes e servidor através de websockets.

Como tal, para nós, este sprint, tal como o anterior, tanto em volume de trabalho como no uso das nossas capacidades técnicas foi bastante positivo.

Na nossa opinião este projeto, está a abranger um enorme número de conceitos diferentes abordados no curso, necessários para o objetivo final, e isso é um aspeto a destacar. Já utilizamos conceitos como Programação e POO (Programação orientada a objetos), Sistemas Distribuídos, REST API (Application Programming Interface) e WebSockets. Sistemas Operativos e Gestão de Sistemas e Redes (Sistemas operativos de kernel Linux e os seus serviços). Redes de Computadores (Encaminhamento de portos, funcionamento do TCP/IP e vários serviços). Teoria da Computação (Máquinas

de estados e autómatos). Álgebra (cálculo vetorial das distâncias). Múltiplas linguagens abordadas no curso como Javascript, PHP, HTML, Java e SQL. Também a destacar os conhecimentos adquiridos a Tópicos Avançados de Base de Dados na leitura e pesquisa de artigos científicos que facilitaram a aprendizagem de novos conceitos. Em seguida, mostramos uma tabela para exemplificar este e outros temas abordados e as respectivas unidades curriculares.

**Tabela 9 - Matéria abordada**

Cadeira	Funcionalidade
Programação para a Internet	Programação do engine em Javascript como também os conceitos de HTML e CSS
Programação para a Internet 2	Programação das websockets em PHP
Programação 1 / Aplicada	Programação em Java do Servidor web
Base de Dados	Implementação de base de dados em MySQL
Tópicos Avançados de Base de Dados	Implementação de triggers e eventos dinâmicos na base de dados
Multimédia	Edição de imagem, vídeo e som utilizando software especializado
Engenharia de Software	Implementação de Práticas MVC
Sistemas Distribuídos	Implementação de REST API e Websockets
Sistemas Operativos / Tecnologias de Computadores	Funcionamento com o sistema operativo Linux
Teoria da Computação	Funcionamento e implementação de máquinas de estado
Gestão de Sistemas e Redes	Criação de servidores web em Linux
Álgebra/Física	Funcionamento e calculo de vetores e distância
Redes 1 / 2	Encaminhamento de portos e funcionamento TCP / IP
Interação com o utilizador	Acessibilidade e estrutura de jogo para facilitar a jogabilidade

Gestão de Projeto	Entendimento da metodologia Scrum e divisão de trabalho
-------------------	---

## 6.2. Limitações

No que toca às limitações, no que toca ao primeiro sprint, estivemos em dúvida se o uso deste motor nos iria limitar de alguma forma criativamente. No segundo sprint podemos concluir que o uso de Javascript nos dá praticamente toda a liberdade, que também existe outros motores.

Ainda relativamente ao primeiro sprint, o nosso foco agora está no produto viável mínimo, ou seja, em criar jogabilidade acima de tudo. Ao longo do primeiro e segundo sprint mantivemos reuniões diárias que foram bastante positivas, no entanto verificamos que estas não se justificavam neste terceiro sprint, pois via-mo-nos quase todos os dias presencialmente e o nosso trabalho tem vindo a ser cada vez mais independente.

Talvez gostássemos de ter apresentado ainda mais trabalho este sprint, mas infelizmente tal não é possível devido à enorme carga de trabalhos nas outras unidades curriculares.

Neste sprint também existiram uma série de dificuldades que requereram bastante tempo de desenvolvimento, nomeadamente problemas com as WebSockets, que funcionavam localmente, no entanto apresentaram um desafio para funcionar através de um servidor.

Quanto ao projeto em si, atualmente, não temos como objetivo apresentar no final um jogo concluído, até porque simplesmente não temos tempo suficiente para concluir um jogo na totalidade. Mas em ter uma prova de conceito daquilo que temos em mente e nesse percurso criar uma série de ferramentas, disponibilizadas gratuitamente e em código aberto, que podem ser utilizadas por outros desenvolvedores, nos seus jogos.

## 6.3. Trabalho Futuro

No próximo sprint iremos concluir e apresentar o modo multi-jogador e apresentar alguns níveis com inimigos no modo singleplayer. Para isso vamos continuar a desenvolver a jogabilidade e os plugins. Tencionamos também deixar à comunidade de desenvolvedores deste motor vários plugins interessantes.

## 7. Referências

### 7.1. Lista de Referências

1. Zackariasson, P., Styhre, A. and Wilson, T.L. (2006). Phronesis and Creativity: Knowledge Work in Video Game Development. *Creativity and Innovation Management*
2. Wikipedia Contributors (2019). Conway's Game of Life. [online] Wikipedia. Available at: [https://en.wikipedia.org/wiki/Conway%27s\\_Game\\_of\\_Life](https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life).
3. Murphy-Hill, E., Zimmermann, T. and Nagappan, N. (2014). Cowboys, ankle sprains, and keepers of quality: how is video game development different from software development? *Proceedings of the 36th International Conference on Software Engineering - ICSE 2014*.
4. Report: Gaming revenue to top \$159B in 2020. (2020). *Reuters*. [online] 12 May. Available at: <https://www.reuters.com/article/esports-business-gaming-revenues-idUSFLM8jkJMI>.
5. kinoar.github.io. (n.d.). *RPGMakerMV Library*. [online] Available at: <https://kinoar.github.io/rmmv-doc-web/index.html>.
6. GitHub. (2021). *rpgtkoolmv/corescript*. [online] Available at: <https://github.com/rpgtkoolmv/corescript>.
7. Moonglesoft. (n.d.). *Moonglesoft*. [online] Available at: <https://moonglesoft.wordpress.com/> [Accessed 10 May 2021].
8. RPG Maker Forums. (n.d.). *Alpha ABS (MV) [New build 1232]*. [online] Available at: <https://forums.rpgmakerweb.com/index.php?threads/alpha-abs-mv-new-build-1232.66713/> [Accessed 10 May 2021].

9. RPG Maker Forums. (n.d.). *Quasi ABS*. [online] Available at: <https://forums.rpgmakerweb.com/index.php?threads/quasi-abs.52049/> [Accessed 10 May 2021].
10. Harrigan, Kevin & Dixon, Michael & Fugelsang, Jonathan & Collins, Kc. (2010). Addictive gameplay: What casual game designers can learn from slot machine research. Future Play 2010: Research, Play, Share - International Academic Conference on the Future of Game Design and Technology. 10.1145/1920778.1920796.1
11. docs.spring.io. (n.d.). \*Spring Framework Documentation\*. [online] Available at: <https://docs.spring.io/spring-framework/docs/current/reference/html/>.
12. socketo.me. (n.d.). \*Ratchet - What is a WebSocket?\* [online] Available at: <http://socketo.me/docs/> [Accessed 6 Jun. 2021].
13. Wikipedia. (2020). \*bcrypt\*. [online] Available at: <https://en.wikipedia.org/wiki/Bcrypt>.
14. [www.mindrot.org](http://www.mindrot.org). (n.d.). *jBCrypt - strong password hashing for Java*. [online] Available at: <https://www.mindrot.org/projects/jBCrypt/> [Accessed 6 Jun. 2021].
15. www.thymeleaf.org. (n.d.). \*Thymeleaf\*. [online] Available at: <https://www.thymeleaf.org/>.
16. auth0.com (2019). *JWT.IO*. [online] Jwt.io. Available at: <https://jwt.io/>.
17. developer.mozilla.org. (n.d.). \*MDN Web Docs\*. [online] Available at: <https://developer.mozilla.org/pt-BR/> [Accessed 6 Jun. 2021].
18. Cipriano, M. (2021). *Nelderson/MV\_Online*. [online] GitHub. Available at: [https://github.com/Nelderson/MV\\_Online](https://github.com/Nelderson/MV_Online) [Accessed 6 Jun. 2021].

19. GitHub. (n.d.). KageDesu/Alpha-NET-Z. [online] Available at: <https://github.com/KageDesu/Alpha-NET-Z/wiki> [Accessed 6 Jun. 2021].

## 8. Anexos

### 8.1. Jogo da Vida e Autómato Celular

O “jogo da vida”, baseado em autómatos celulares, desenvolvido por John Conway, estabelece que há dois tipos de células, as “vivas” e as “mortas”. Seguindo um conjunto de regras e de várias iterações determinamos se uma célula permanece “viva” ou “morta”.

As regras do jogo da vida, consistem em caso uma célula “viva” tenha só uma célula vizinha “viva”, esta morre. Caso esta célula tenha 2 ou 3 células vizinhas “vivas”, esta célula vive para a próxima iteração. Caso uma célula “morta” tenha 3 vizinhas “vivas”, esta fica “viva” para a próxima iteração. Caso uma célula “viva” tenha 4 ou mais vizinhas “vivas” fica “morta” na próxima iteração.

Este conjunto de regras é bastante interessante para uma experiência científica, mas consideramos que para a geração de mapas que pretendemos não havia necessidade de todas estas regras.

O autómato celular que desenvolvemos consiste em caso uma célula “morta” tenha mais do que um certo número de células vizinhas vivas (limite de criação), esta fica viva. Por outro lado, caso uma célula “viva” tenha menos do que um certo número de células vizinhas “vivas”, esta fica “morta” (limite de destruição). De notar que estes limites são especificados pelo utilizador do plugin.

### 8.2. Diagramas

Serão aqui postos os links para os diagramas que foram ou não apresentados no relatório para permitir melhor visualização.

[Diagrama de Classes REST API.](#)

[Diagrama Conceptual da Base de dados.](#)

[Diagrama Físico da Base de Dados.](#)

[Diagrama MVC.](#)

[Diagrama dos endpoints.](#)