



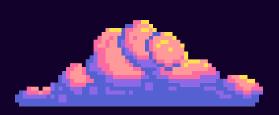


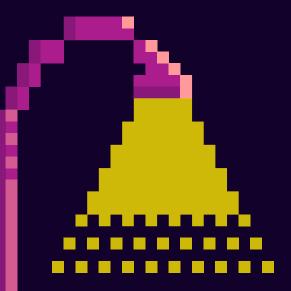
INTRODUÇÃO

O nosso jogo se chama Infernal, e a ideia é ele ser um RPG-2D, com multíplas classes e 6 fases diferentes.

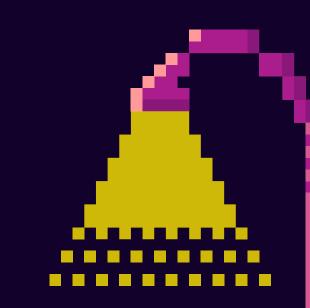
O jogo está sendo produzindo na linguagem de programação python, utilizando a biblioteca Pygame, utilizando assets gratuitos para os sprites e texturas.MySql como banco de dados e Tkinter como interface



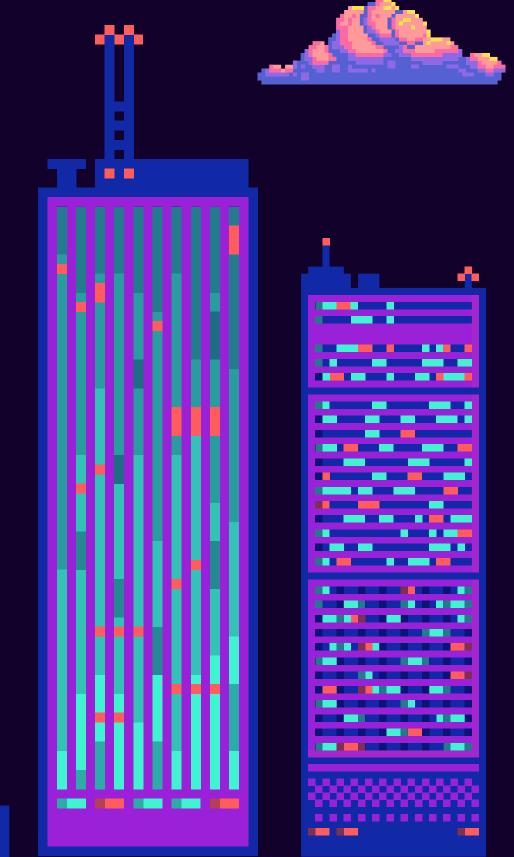


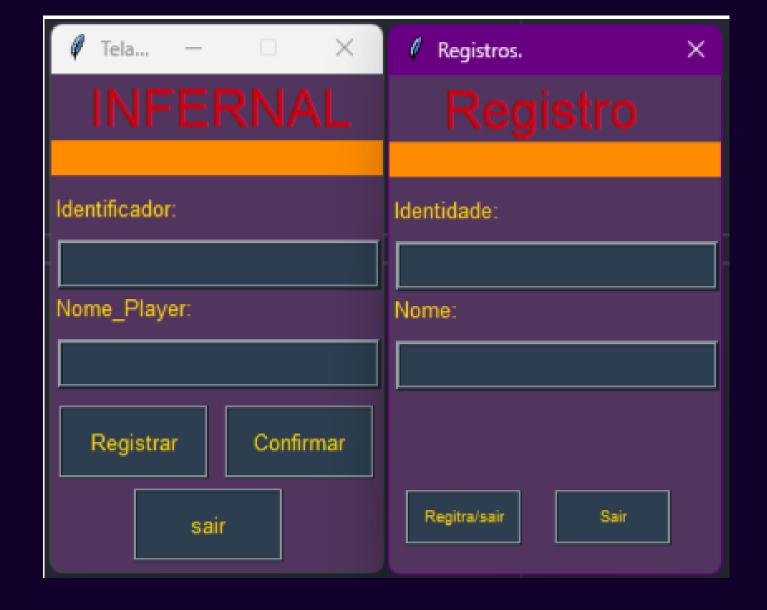


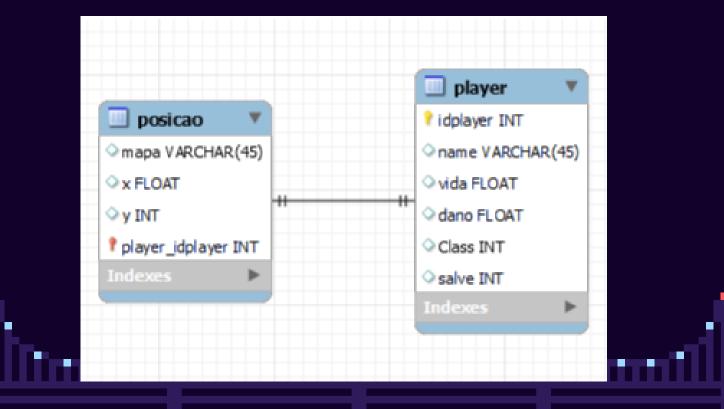
Um jovem aventureiro encontra um Navio misterioso enquanto pescava. Nele ele encontra uma sala trancada com 3 cadeados e um mapa com um bilhete: "Busque as 3 chaves nos lugares marcados e você poderá entrar na sala do tesouro.". E então o jovem decide ir em uma expedição em busca da riqueza.



TELO DE LOGIN E BANCO DE DADOS

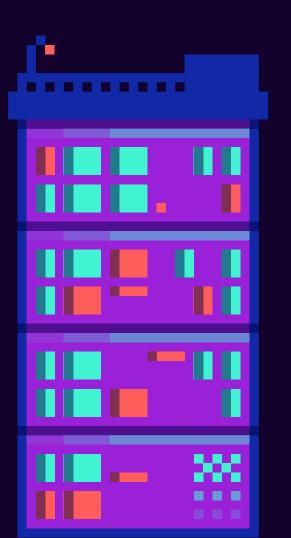


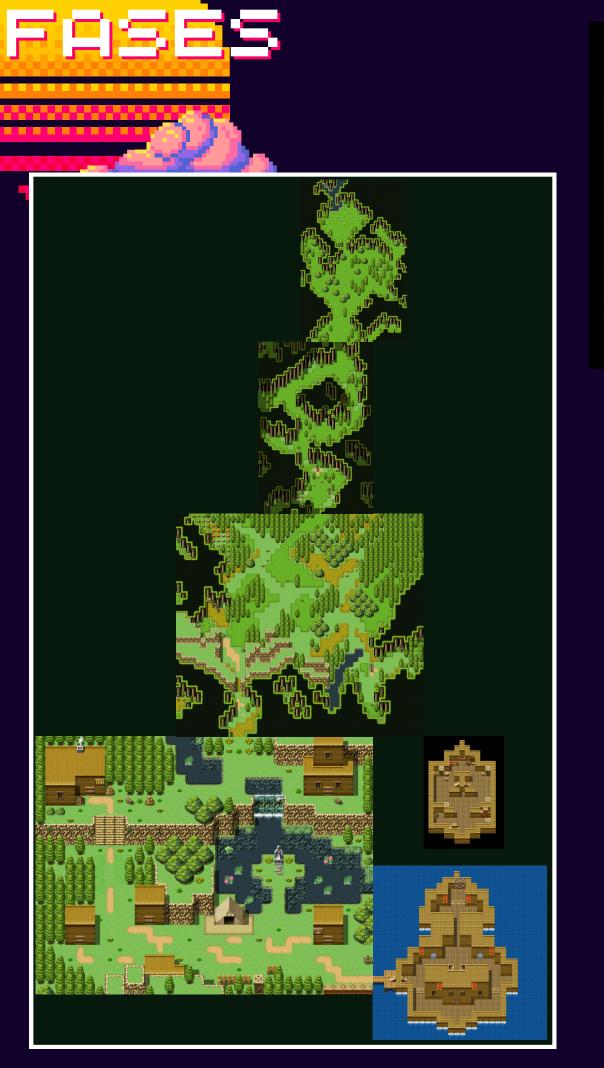




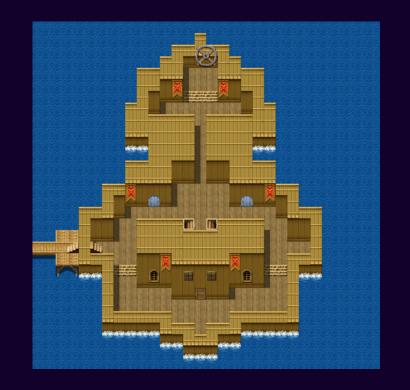




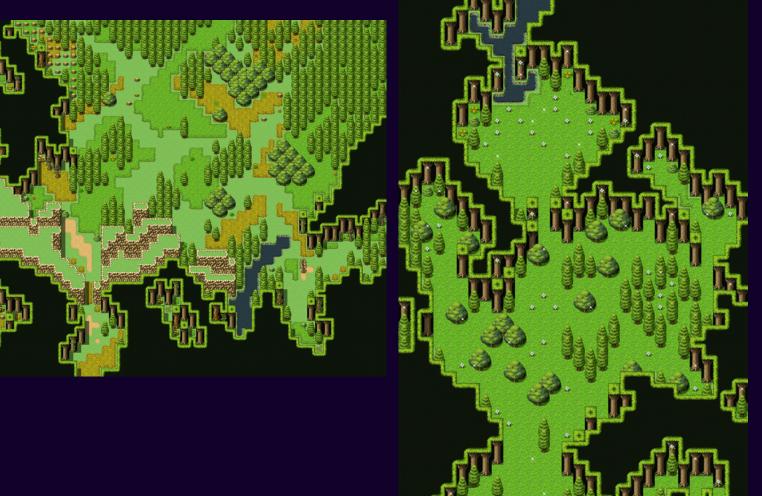












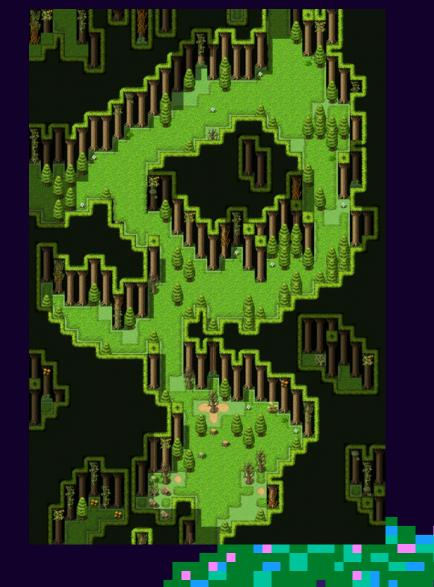


DIAGRAMA Banco_de_Dados Menu app: pygame.Surface conn login. ourser sprite_atual: int id_t * __init__(name: str) sprites: dict nome_t - tabela: pygame.Surface image: pygame.Surface + __enter__() +__init__(self, imagem) rect: pygame.Rect. + __exit__(exo_type, exo_value, traceback) nome + pegar_sprite(self, frame, largura, altura, escala) -> pygame.Surface jogador: diot. + __init__() lid: lint + remover_jogador(idplayer: int) # arruma(var. str) -> str - name: str. + visualizar_jogador(idplayer: int) registran() + visualizar_jogadores() - olasse: str + logian() - dano: int + upadate_player(dicionario: dict) + configura() - hpc int + adicionar_jogador(idplayer: int, ..., y: int = 0) - ultimohit: int - cooldownhit int velocity: pygame.math.Vector2 Cena - esquerda: bool Jogo ultimo_check: int app image: pygame.Surface **SCHOOL** ultima_acap: str - rect: pygame.Rect eloek atacando: intifundo musica_tema_ __init__(self, groups, image, posicao=(0, 0) bolas: list - textura_solo running. + update(self) retangulo_atualizado: boolsprites entidade __init__(self, app, groups.... x=0.y=0) cens bloods * animar(self, acap) inimigos. + input(self) bola_de_fogo- ataque(self). + update() player sprite_atual: int t move(self) + draw() mapas sprites; dist thecar_tela(self). + close() - image: pygame.Surface mapa_atual * checar_colisces(self, direcac) camera + hit(self) rect: pygame.Rect + update(self) - velocity: pygame.math.Vector2 Camera orlar_mapa(nome). - tempo_gerat: int game resetar_mapa(nome) seguirinimigo: int. width esquerds: bool height - ultimo_check: int + gen_textura_sola() - ultima_acac: str - update(i) - retangulo_atualizado: bool - camera atacando: int __init__(game, width, height). + __init__(self, groups, image, posicao=(0, 0), parametros=(j) + apply(entidade) -> pygame.Rect + animar(self, acao) + draw(surface, group) + move(self) + update(jogador) + checar_collsoes(self, direcap, dx, dy) + update(self)

DIAGRAMA CASO DE USO

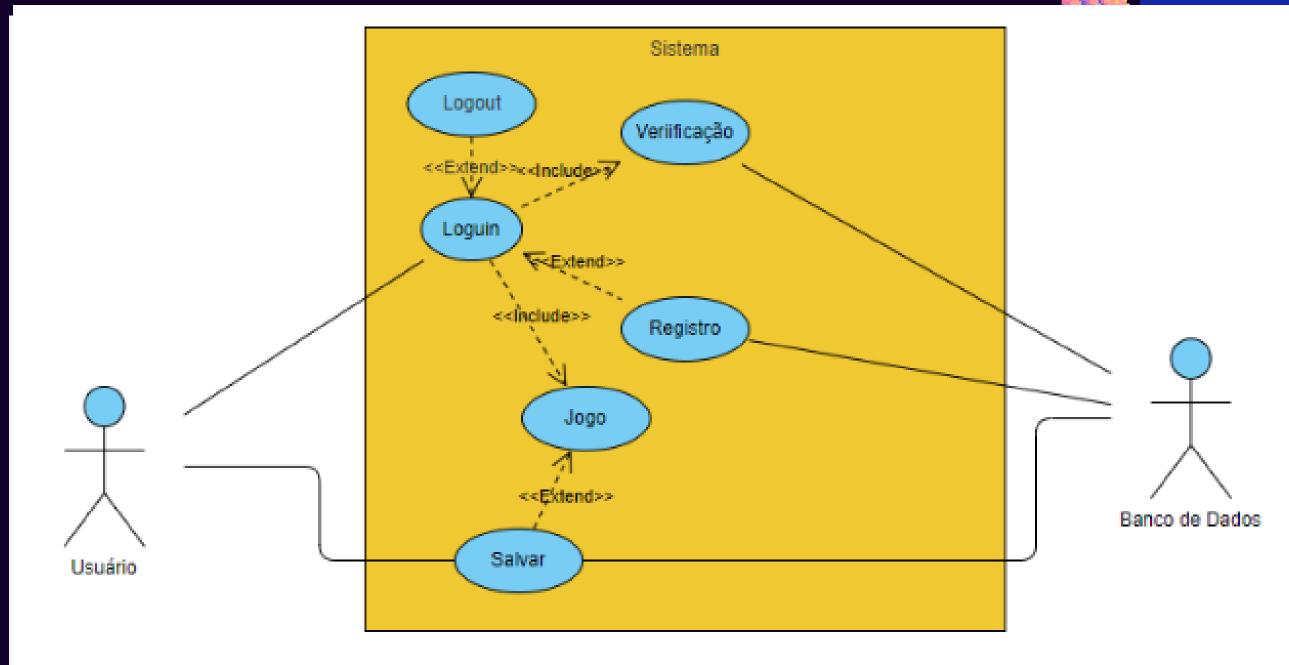
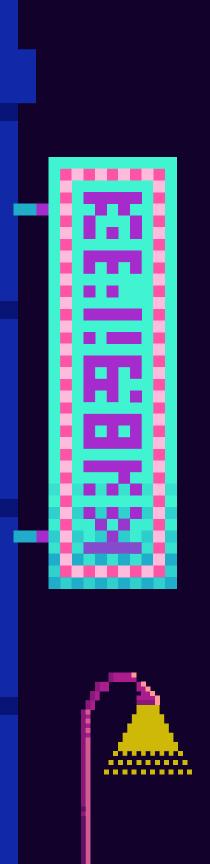


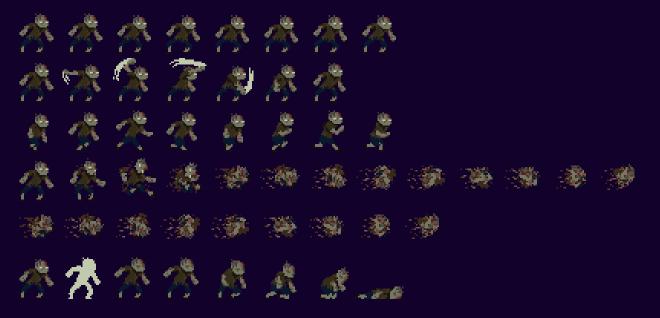
Figura 1. Caso de Uso - UC01.







INIMIGO











PALAVRAS DE FECHAMENTO

Como foi um jogo com uma equipe muito pequena o maximo que conseguimos deselvover foram as partes bases para criar outro mapas, com o tempo que nos ia desenvolvendo ia demorando para construir outro mapas pois era muitas coisas para integrar e ir mudando mas em visão geral ele continua muito versátil para a implementação de outras classes.

