Roadmap – Front-End

Este documento apresenta uma análise detalhada do que já foi implementado no front-end do jogo Baronet, o que ainda precisa ser adicionado e melhorado.

1. Tela de Criação de Personagem

Já implementado:

- Arquivo `tela_criacao_personagem.py` com base visual.
- Imagens de classes e armas em `Imagens/classes/`.
- Sons para feedback de interação.

A adicionar:

- Animação de seleção de classe (foco nas confirmadas).
- Escolha de tipo (Combat ou Outer Non-Combat).
- Exibição de atributos iniciais por classe.
- Escolha de arma inicial vinculada à classe.
- Integração com backend via API para salvar dados.

A corrigir/melhorar:

- Renomear `Tela_de_criação_do_perssonagem.py` (erro ortográfico).
- Melhorar feedback visual (seleções, botões pressionados).
- Usar fonte personalizada existente para identidade visual.

2. Sistema de Combate Visual (Battle UI)

Já implementado:

- Scripts parciais em `rpg.py`, `dano.py`, `classe do player.py`.
- Sons de ação em `efeito_sonoro/`.

- Imagens de armas e sprites básicas.

A adicionar:

- HUD com vida, habilidades, ícones dos aliados e contador de combo.
- Controles para ataque, dash, pulo (simples e duplo), esquiva, habilidade.
- Sprite com animações de movimentação lateral.
- Efeitos visuais para ataque e impacto.

A corrigir/melhorar:

- Centralizar controle de colisão e hitbox.
- Garantir escalabilidade visual.
- Isolar HUD como componente reutilizável.

3. Troca e Gestão de Equipe

Já implementado:

- Representações visuais de classes em `Imagens/classes/`.
- Estrutura de script modular com `lobby.py`.

A adicionar:

- Tela de formação de equipe (1 ativo + 3 reservas).
- Interface para ativar especiais dos reservas.
- Exibir sinergias entre membros da equipe.

A corrigir/melhorar:

- Agrupar tudo relacionado à equipe em um só módulo ou componente.
- Adicionar feedback visual para troca ou uso de habilidades.

4. Inventário e Interface de Itens

Já implementado:

- Scripts parciais em `armas.py` e `JOGADOR E ARMAS.py`.
- Imagens de armas em `Imagens/armas/`.

A adicionar:

- Inventário geral e rápido com sistema de drag-and-drop.
- Sistema de hotbar (teclas rápidas 1–4).
- Exibição de raridade com cores nos slots.

A corrigir/melhorar:

- Separar lógica de armas da interface visual.
- Adicionar ícones representativos para cada tipo de item.
- Padronizar nomes de arquivos: minúsculos, sem espaços.

5. Progressão e Evolução

Já implementado:

- Banco local `banco_de_dados.db`.
- Scripts com lógica de XP e dano.

A adicionar:

- Barra de XP visível na HUD.
- Tela de evolução com desbloqueio de habilidades por faixa.
- Feedback visual ao subir de nível.

A corrigir/melhorar:

- Conectar a progressão com backend (nível, faixa, habilidades).
- Criar interface para escolha de habilidades ao subir de faixa.

6. Sistema de Save/Load

Já implementado:

- Arquivo `usuario.json` com dados básicos do jogador.

A adicionar:

- Tela com slots de save e botão "Continuar Jogo".
- Opção de salvar manualmente e integração com backend.

A corrigir/melhorar:

- Padronizar o local onde os dados são armazenados.
- Criar interface visual dos saves com nome, data, nível e classe.

7. Inimigos, Drop de XP e IA Visual

Já implementado:

- Arquivo `criando monstros.py` com estrutura de IA simples.
- Sons de ataque e impacto.
- Imagens base de inimigos.

A adicionar:

- Animações básicas de movimentação dos inimigos.
- Drop visual de XP com animação.
- Exibir tipo de inimigo (nome ou ícone).
- IA visual: perseguição, ataque e esquiva.

A corrigir/melhorar:

- Centralizar os tipos de inimigo para fácil modificação visual.
- Separar lógica de IA (backend) da representação visual (frontend).