**Manual de Manutenção da Aplicação AR**

Este guia descreve passo‐a‐passo o processo de incluir novos conteúdos (cards, modelos 3D, imagens de marcação) e publicar uma nova versão da aplicação Unity + Lightship ARDK.

**1. Pré‐requisitos**

* Ter o repositório Git da app clonado e atualizado na máquina.
* Unity 6000.034f1 (versão especificada pelo Lightship ARDK).
* Acesso à pasta StreamingAssets e ao asset catalog.json.
* Unity aberto no projeto correto.

**2. Editando o JSON de Cadastros (catalog.json)**

1. Abra o catalog.json com o seu editor de texto ou o json\_editor.py.
2. A estrutura geral é:

jsonc

CopyEdit

{

"sections": [

{

"title": "Secção X",

"cards": [

{

"id": "casa001",

"scanImageName": "marker\_casa",

"modelPrefabName": "HousePrefab1",

"latitude": 38.72,

"longitude": -9.14,

"price": 350000,

"type": "MORADIA",

"size": "3 quartos",

"area": "120",

"image": "house1",

"detailsLeft": "…",

"detailsRight": "…",

"description": "…"

},

… outros cards …

]

},

… outras secções …

]

}

1. **Para adicionar um novo card**:
   * Abaixo de "cards": [ … ], crie um novo objeto com todos os campos acima.
   * **scanImageName** deve igualar o **Name** que usará no **XR Reference Image Library**.
   * **modelPrefabName** deve coincidir com a **chave** registada no **ModelRepository** (veja seção 4).
   * Salve o ficheiro.
2. Copie o JSON atualizado para Assets/StreamingAssets/catalog.json (substituindo o anterior).

**3. Importando o Modelo 3D no Unity**

1. No Unity, importe o asset 3D (FBX/GLTF/etc.) para a pasta Assets/Models/.
2. Configure suas **Import Settings** (scale, colliders, materiais).
3. Arraste o prefab finalizado para **Assets/Prefabs/Models/**.

**4. Registrar o Prefab no ModelRepository**

O ModelRepository expõe um **Dictionary<string,GameObject>** que mapeia nomes → prefabs:

csharp

CopyEdit

public class ModelRepository : MonoBehaviour

{

[Serializable] struct Entry { public string key; public GameObject prefab; }

[SerializeField] private Entry[] entries;

private Dictionary<string,GameObject> map;

void Awake() {

map = entries.ToDictionary(e=>e.key, e=>e.prefab);

}

public GameObject GetPrefab(string key) => map.TryGetValue(key,out var p)? p : null;

}

1. Selecione o GameObject **ModelRepository** na cena **Scan**.
2. No Inspector, em **Entries**, clique em “+” para adicionar uma linha:
   * **Key** = o mesmo modelPrefabName que colocou no JSON.
   * **Prefab** = arraste o prefab que acabou de importar.
3. Salve a cena.

**5. Importando a Imagem‐Marcador no XR Reference Image Library**

1. Arraste o ficheiro de imagem (.png, .jpg) para Assets/Images/Markers/.
2. Selecione-o, no **Inspector** defina:
   * **Texture Type** = Default
   * **Max Size** ≥ 1024
   * **Compression** = None
3. Abra seu **XR Reference Image Library** (Assets/**ReferenceImageLibrary.asset**).
4. Clique em **Add Image**, selecione o asset importado.
5. Em cada entry:
   * **Name** = igual a scanImageName do JSON.
   * **Specify Size** = ✓, defina a largura real em metros (ex: 0.15).
6. Salve; Unity irá reconstruir a DB de ARCore.

**6. Atualizando Sprites e Outros Assets**

* Se adicionou novos tipos (type), coloque os ícones em Assets/Resources/PinSprites/<TYPE>.png.
* Para novos botões ou imagens de fundo (nav bar, filtros), importe para Assets/UI/Sprites/ e atualize o UIPageNavigator ou CatalogFilterController conforme necessário (via Inspector).

**7. Testes Locais**

1. **No Editor**:
   * Execute em **Play**, abra o **Catalog**, veja se o novo card aparece e carregue nos filtros.
   * Na cena **Scan**, teste marcar o novo scanImageName simulado (via XR Simulation) e confirme o novo prefab apareça.
2. **No Android**:
   * Construa o APK e instale.
   * Teste real‐life: marque a imagem impressa, verifique o spawn e o painel de edição.

**8. Commit & Versionamento**

1. No Git, faça:

bash

CopyEdit

git add Assets/StreamingAssets/catalog.json

git add Assets/Models/... # novos modelos

git add Assets/Prefabs/... # novos prefabs

git add Assets/Images/Markers/... # novas imagens

git commit -m "feat: adicionar card <ID> com modelo e marcador"

git push

1. Gere uma nova **Release** no GitHub com o número de versão (ex: v1.2.0).

**9. Build e Publicação**

1. No Unity, vá **File → Build Settings**.
2. Para Android/iOS, selecione a plataforma e clique em **Build**.
3. Submeta o APK ou IPA nas respetivas stores.