Manual de Manutenção da Aplicação RA

Estágio IPV – Vision4You  
Pedro Duarte de Almeida Alves Costa

1. **Introdução**

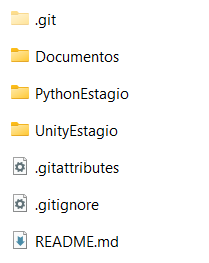
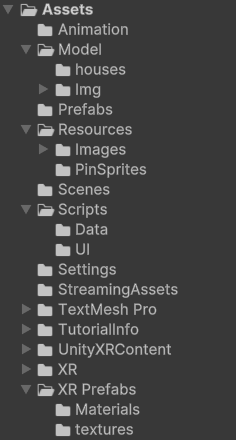
**Guia** para manutenção, atualização e publicação da aplicação de Realidade Aumentada (Unity + Lightship ARDK).

* **Objetivo:** Ensinar como adicionar Cards, modelos 3D e marcadores, além de publicar novas versões.
* **Público-alvo:** Desenvolvedores e equipa de manutenção.

1. **Pré-Requisitos do Ambiente**

| **Ferramenta** | **Versão Mínima** | **Ícone** |
| --- | --- | --- |
| Unity | 6000.034f1 |  |
| Python + an\_editor.py | 3.10 |  |

1. **Estrutura do Projeto**



Img 1 e 2- Estrutura do Ficheiro Git e da pasta Assets

1. **Inserção e Atualização de Conteúdos**

**Fluxo para catalog.json:**

Uma imagem com texto, diagrama, file, Tipo de letra

Descrição gerada automaticamente

Img 3 - Fluxo de edição do Json

**Abrir** Assets/StreamingAssets/catalog.json seguindo o template abaixo:

Para começar, abra a pasta **PythonEstagio** no seu compilador de código preferido (VisualStudio, VisualStudio Code, PyCharm, etc…)

Uma imagem com texto, Tipo de letra, captura de ecrã, design

Descrição gerada automaticamente

Figure 2 - Pasta PythonEstagio

Com a pasta aberta no compilador, introduza o seguinte comando no terminal:

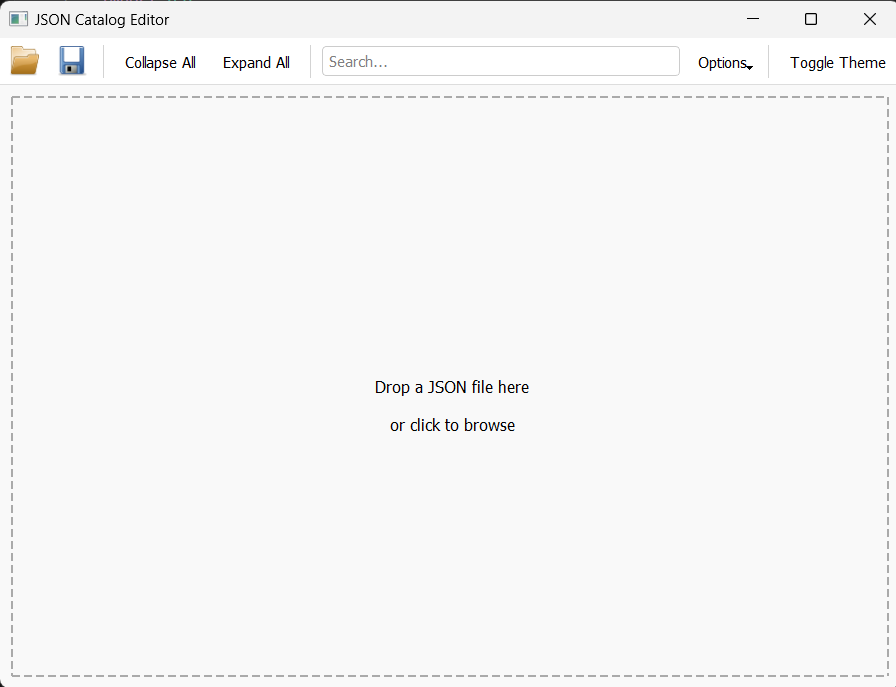
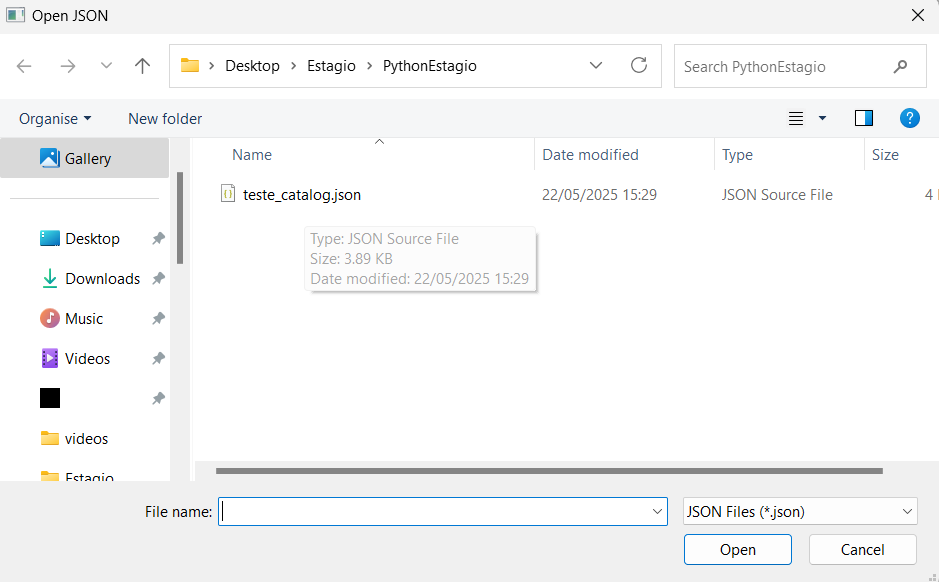
*- “python json\_editor.py”*

Este comando inicia o **Editor de JSON**, onde podemos editar livremente o **catalog.json** e quando guardadas as alterações, o **Unity** é automaticamente atualizado.

Uma imagem com texto, Tipo de letra, captura de ecrã, design

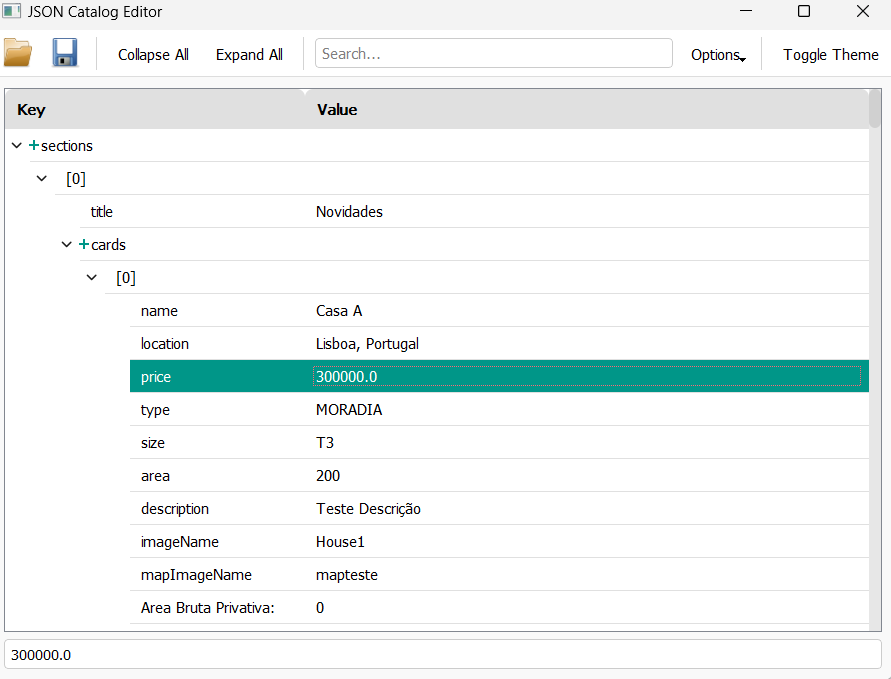
Descrição gerada automaticamente

Img 3 - Código para iniciar o Editor de JSON

Com o **Editor** aberto podemos selecionar o **JSON** presente na pasta **StreamingAssets**, ou simplesmente arrastar para o “*Drag and Drop*” inicial.

Img 4 e 6 - Drag and Drop e seleção do JSON

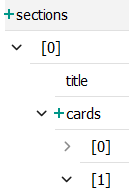
Com o **Editor** aberto podemos livremente adicionar, editar e eliminar, tanto secções, como **Cards** individuais.

Uma imagem com texto, captura de ecrã, número, software

Descrição gerada automaticamenteO **UI** do editor é simples, basta selecionar o elemento a editar e a barra no inferior do **Editor** é atualizada com o valor, que pode ser facilmente editado na mesma barra.

Img 7 e 8 - Edição de valores

As secções e os **Cards** têm ambos um ícone de adicionar ( **+** ), que rapidamente adiciona uma nova secção (com 3 **Card** base) ou um **Card** base a uma secção.  
Para eliminar, basta clicar com o botão direito do rato e uma opção para eliminar é aberta.



Img 9 e 10 - Adicionar e Remover elementos

Tente respeitar a estrutura já existente para não criar problemas de conflitos para o **Unity**. Cada elemento do **JSON** é utilizado na app, por isso é essencial preencher todos os campos, principalmente os pertencentes aos **Cards**.

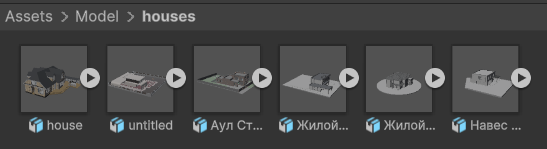
Os elementos **imageName**, **scanImageName** e **modelPrefabName** são para fazer a relação entre imagens e modelos com o **Card**. Para isso é necessário indicar o nome da imagem / modelo e, possivelmente, realizar uma pequena configuração no **Unity**. Este tema vai ser abordado mais à frente.

Quando estiver satisfeito com as modificações realizadas, pode guardar as alterações, tento sempre em consideração manter o nome **catalog.json** e a localização na pasta **StreamingAssets** (é aconselhada criação de um backup para a **catalog.json** inicial, de forma a não perder dados desnecessariamente).

**Importar Modelos 3D**

Passos no Inspector para **FBX**:

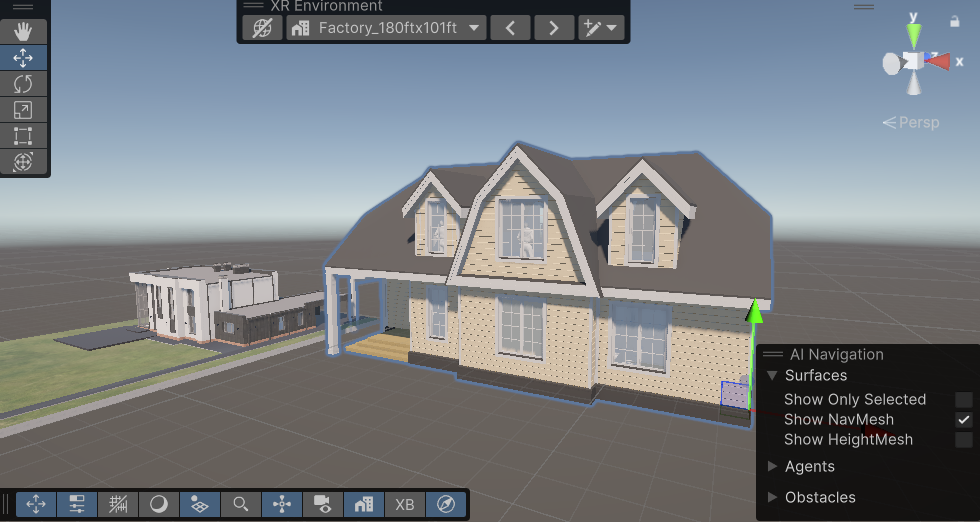
Colar o modelo 3D, em ficheiro **FBX** na pasta **Assets/Model/houses/**.



Img 11 - Pasta Assets/Model/houses/

Ajustar **Scale**, **Mesh Collider** e **Materials**. A forma mais fácil de manter uma escala constante entre todos os modelos é arrastar um **Prefab** já existente da pasta **Assets/XRPrefabs/** e usar o mesmo como referência para a escala.

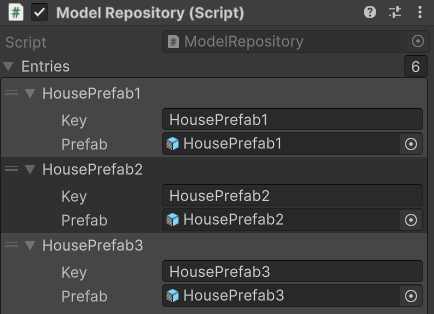
Mantenha a posição em (0,0,0).



Img 12 – Redimensionamento do modelo com base no Prefab

Quando ficar satisfeito com o modelo, arraste-o para a pasta **Assets/XRPrefabs/**, para criar o **Prefab**.

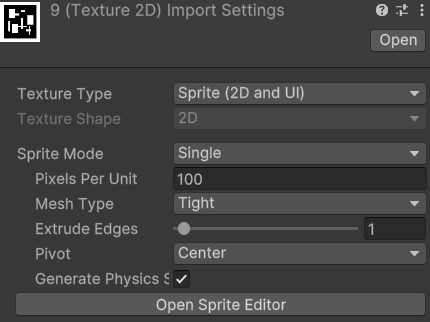
Para garantir a conecção com o JSON, tem de se registar o **Prefab** acabado de criar no **ModelRepository** (GameObject com o script **ModelRepository.cs** na cena ‘**Scan**’).  
O **JSON** só vai conseguir relacionar o **Card** ao **Prefab**, se o **Prefab** for arrastado para o **Inspector** e corretamente identificado (mesmo nome presente no **JSON**).



Img 13 – Inspector do GameObject com o script

**Importar Marcadores RA**

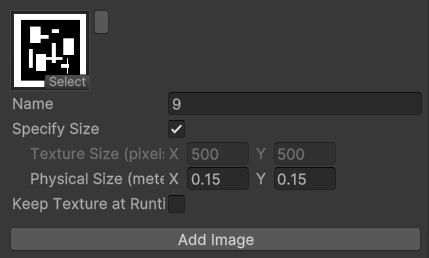
A forma de adicionar marcadores é bastante parecida com a de adicionar modelos.

****Começamos por colar as imagens na pasta **Assets/Model/Img/.**  
Com a imagem selecionada, noo Inspector mudamos o a *Texture Type* para **Sprite (2D and UI)**, o *Sprite Mode* para **Single** e damos **apply**.

Img 14 – Configuração do marcador

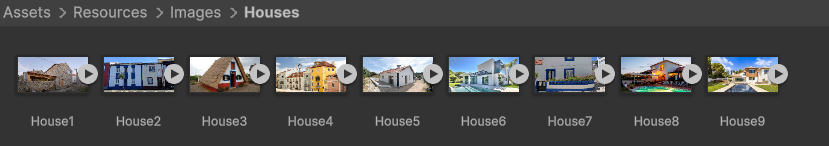
Com o marcador configurado, só é necessário adicionar o marcador à **ReferenceImageLibrary.asset**, presente na pasta **Assets/**.

Aqui é preciso arrastar o marcador que foi adicionado, dar-lhe um nome (igual ao **JSON**) e o tamanho na vida real.



Img 15 – Adição do marcador ao asset ReferenceImageLibrary

**Importar Imagens**

As **Imagens** não precisam de ser referenciadas em um **Game Object**, por isso basta adicionar as imagens na pasta **Assets/Resources/Images/Houses/** e ter a certeza de que o nome do ficheiro é o mesmo adicionado no **JSON**.

Img 16 – Pasta onde se adiciona as imagens

1. **Fluxo de Testes**

Agora vão ser apresentados os passos a tomar para testar as alterações feitas. Segue o fluxo de testes para confirmar as atualizações realizadas.

**5.1. Editor**Uma imagem com texto, diagrama, file, Tipo de letra

Descrição gerada automaticamente

Img 16 – Fluxo de Testes no Editor

**5.2. Dispositivo**

Uma imagem com texto, diagrama, file, esboço

Descrição gerada automaticamente

Img 17 – Fluxo de Testes no Dispositivo

1. **Build e Publicação**

Com as alterações concluídas e o fluxo de testes realizado, podemos “**Construir**” a App, exportando-a assim do **Unity,** pronta para ser instalada.

Agora basta confirmar que as configurações estão preparadas para exportar para android

**Edit – Project Settings – Player**:

* **Active Input Handling** : Old;

**Hierarchy Scan scene**:

* **EventSystem** : Input System UI – **OFF**, Standalone Input Module – **ON**;
* **Main Camera** : Tracked Pose – **OFF**, Tracked Pose (script) – **ON**;

**File – Build Profiles:**

* Garantir que a Plataforma é **Android**;
* Dar **Build**;

Em caso de erro na **Build**, consultar o fluxo de testes e o próximo capítulo. Se tudo correr bem, tem o **.apk** pronto para partilhar.

1. **Troubleshooting Comum**

| **Ícone** | **Erro** | **Causa** | **Solução** |
| --- | --- | --- | --- |
| 🚫 | JSON inválido | Sintaxe malformada | Validar com json\_editor.py |
| 🔍 | Marker não detectado | Nome divergente | Conferir Name no ReferenceImageLibrary |
| ❌ | Prefab não aparece | Key inválida no ModelRepo | Ajustar mapeamento scanImageName → key |
| ⚠️ | Erros de compilação em C# | Namespace em falta | Incluir using ...; no topo do script |