UNIVERSIDADE FEDERAL DE SANTA CATARINA - CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

Desenvolvimento de um Modelo de Geração Automática de *Wireframes* no App Inventor a partir de Arquivos de Exportação do Penpot

Elizeu Santos Madeira

Florianópolis - SC
2023/2

Universidade Federal de Santa Catarina
Departamento de Informática e Estatística

Desenvolvimento de um Modelo de Geração Automática de *Wireframes* no App Inventor a partir de Arquivos de Exportação do Penpot

Trabalho de Conclusão de Curso de Graduação em Sistemas de Informação, do Departamento de Informática e Estatística, do Centro Tecnológico da Universidade Federal de Santa Catarina, requisito parcial à obtenção do título de Bacharel em Sistemas de Informação.

Autor: Elizeu Santos Madeira

Florianópolis - SC
2023/2

Elizeu Santos Madeira

Desenvolvimento de um Modelo de Geração Automática de Wireframes no App Inventor a partir de Arquivos de Exportação do Penpot

Trabalho de Conclusão de Curso de Graduação em Sistemas de Informação, do Departamento de Informática e Estatística, do Centro Tecnológico da Universidade Federal de Santa Catarina, requisito parcial à obtenção do título de Bacharel em Sistemas de Informação.

Prof.ª Dr.ª rer. nat. Christiane Gresse von Wangenheim, PMP
Orientadora
Universidade Federal de Santa Catarina

Prof. Dr. Jean Carlo R. Hauck
Banca
Universidade Federal de Santa Catarina

Ramon Mayor Martins
Banca
Universidade Federal de Santa Catarina

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIAÇÕES

CSS - *Cascading Style Sheet*

JSON - *JavaScript Object Notation*

SVG - *Scalable Vector Graphics*

UI - *User Interface*

TCC - Trabalho de Conclusão de Curso

XML - Extensible Markup Language

HTTP - Hypertext Transfer Protocol

ACM - *Association of Computing Machinery*

IEEE - *Institute of Electrical and Electronics Engineers*

Capes - Coordenação de Aperfeiçoamento de Pessoal de Nível Superior

NI - Não Informada

SUS - *System Usability Scale*

HTML - HyperText Markup Language

PDF - *Portable Document Format*

JPEG - *Joint Photographic Experts Group*

PNG - *Portable Network Graphics*

RESUMO

Com a modernização dos processos de desenvolvimento de interface de sistemas vieram também ferramentas cada vez mais complexas e poderosas para auxiliar a produtividade no desenvolvimento de software. Hoje no *design* de interface são utilizadas tipicamente ferramentas de *design* gráfico. Um exemplo é o Penpot, que permite seus usuários desenvolverem *design* de interfaces de forma gratuita. Os *designs* de interfaces criados podem ser exportados usando formatos não-proprietários e livres. Mesmo já fornecendo código de CSS, o Penpot ainda não permite diretamente converter o *design* criado em código de aplicativo móvel. Para facilitar esta conversão pode-se criar uma ferramenta que automaticamente cria o código de um aplicativo p.ex. na ferramenta App Inventor como base para o desenvolvimento de um *app* funcional. Assim, o objetivo do presente trabalho é levantar o estado da arte em relação a este tipo de serviço, e desenvolver uma ferramenta para gerar automaticamente o código de um projeto App Inventor a partir de um *design* exportado do Penpot. Espera-se que estes resultados facilitem e deixem o processo de desenvolvimento de aplicativos móveis mais eficientes, podendo-se aplicar esta solução tanto no ensino de computação quanto também por usuários finais criando apps com App Inventor.

Palavras chave: App Inventor, *Wireframe*, *Design* de Interface de Usuário, Penpot.

1. INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

Conforme a sociedade avança e os meios digitais são inseridos no dia-a-dia dos cidadãos de forma gradual e permanente, é necessário preparar a sociedade para lidar com as novas profissões (ou mesmo com a modernização das profissões já existentes) já na vida estudantil (Grover; Pea, 2013). Por este motivo, a necessidade de introduzir o ensino de computação já na educação básica é importante (Lye; Koh; 2014). Neste contexto, uma alternativa para ensinar a computação é por meio do desenvolvimento de aplicativos móveis usando App Inventor (Patton *et al.*, 2019). O App Inventor (AppInventor, 2023), um ambiente baseado em blocos, proporciona a simplicidade no desenvolvimento de aplicações, incluindo o *design* de interface, até um aplicativo funcional para a plataforma Android.

Tipicamente o processo de desenvolvimento de um aplicativo inicia-se com uma *sketch,* que é um esboço feito geralmente com papel e caneta para uma expressão mais conceitual das funcionalidades e da interface do futuro aplicativo (Schlatter *et al*., 2013). A partir do *sketch* é gerado um *wireframe* geralmente usando uma ferramenta de *design* gráfica, como p.ex. Figma (Figma, 2023), aprimorando e complementando o processo de *design* de interface com o *design* visual. Após a etapa de desenvolvimento da interface, a próxima etapa é desenvolver o código do aplicativo. Isto pode ser feito, p.ex. usando o App Inventor.

Uma das alternativas gratuitas para o desenvolvimento de interfaces de aplicações móveis é a ferramenta online de *design* gráfico Penpot (Penpot, 2023). A ferramenta Penpot é *código aberto* e gratuito, fornecendo uma riqueza de recursos e facilidade de organização e operação dos recursos para desenvolver de forma facilitada e eficiente o desenvolvimento de interfaces de aplicações móveis. Ele também permite exportar seus projetos usando formatos de domínio público (JSON e SVG).

No entanto, estas ferramentas, mesmo gerando até código de HTML/CSS, não oferecem a possibilidade de gerar aplicativos funcionais, apenas interfaces. Assim, para criar um app a partir do *design* criado necessita-se a

implementação manual em ambientes de programação, como p.ex. utilizando o App Inventor, uma plataforma web que permite criar aplicativos para sistema operacional Android desde a interface com o usuário até a regra de negócio (MIT, 2023), criando assim um app funcional. Neste processo, para agilizar o desenvolvimento pode-se usar o projeto exportado do Penpot convertido para o formato de importação de projetos do App Inventor (arquivo ".aia") possibilitando assim a importação do *design* criado no Penpot diretamente no App Inventor.

1.2. OBJETIVOS

Objetivo geral

O objetivo geral deste trabalho é desenvolver uma ferramenta para converter um projeto de *design* de interface exportado do Penpot em um arquivo ".aia" de importação do App Inventor.

Objetivos Específicos

O1. Analisar a fundamentação teórica sobre desenvolvimento de apps com App Inventor e design de interfaces de usuário com Penpot.

O2. Analisar o estado da arte em relação a ferramentas para conversão automática de projetos Penpot para App Inventor.

O3. Desenvolver e testar uma ferramenta de suporte que automaticamente converta um projeto Penpot em um arquivo ".aia" do App Inventor.

O4. Avaliar a qualidade da ferramenta desenvolvida.

Premissas e restrições

O trabalho é realizado de acordo com o regulamento vigente do Departamento de Informática e Estatística (INE – UFSC) em relação aos Trabalhos de Conclusão de Curso. O modelo proposto tem como foco a conversão de projetos Penpot para arquivos ".aia" do App Inventor, não abordando arquivos de outros tipos ou ferramentas. A conversão de elementos do projeto Penpot foca-se somente em elementos de UI visíveis do App Inventor. O sistema a ser desenvolvido considera as versões do Penpot e App Inventor correntes durante a execução desse tema.

1.3. METODOLOGIA DE PESQUISA

A metodologia de pesquisa utilizada neste trabalho é dividida nas seguintes etapas.

Etapa 1 – Fundamentação teórica

Estudando, analisando e sintetizando os conceitos principais e a teoria referente aos temas a serem abordados neste trabalho é apresentado a fundamentação teórica utilizando a metodologia de revisão narrativa (Cordeiro *et al*., 2007). Nesta etapa são realizadas as seguintes atividades:

A1.1 – Análise teórica sobre desenvolvimento de apps com App Inventor

A1.2 - Análise teórica sobre *design* de interface de usuário com Penpot

Etapa 2 – Estado da arte

Nesta etapa é realizado levantamento do estado da arte, utilizando a abordagem do mapeamento sistemático seguindo o procedimento proposto por Petersen *et al*. (2008), em relação a ferramentas existentes que permitem a conversão de projeto de UI para App Inventor. Esta etapa é dividida nas seguintes atividades:

A2.1 – Definição e Execução da busca e seleção de ferramentas relevantes

A2.2 – Extração e análise de informações relevantes

Etapa 3 – Desenvolvimento

Nesta etapa é desenvolvida uma ferramenta para conversão de projeto Penpot para um arquivo “.aia” do App Inventor, seguindo um processo de desenvolvimento de software iterativo (Larman; Basili, 2003). Esta etapa é dividida nas seguintes atividades:

A3.1 – Análise de requisitos

A3.2 – Modelagem do sistema

A3.3 – Implementação e teste do sistema

Etapa 4 – Avaliação

Nesta etapa é avaliada a qualidade da ferramenta desenvolvida em termos de completude e acurácia da interface gerada no arquivo “.aia” realizando uma

avaliação com 3 projetos de design de interface. Esta etapa é dividida nas seguintes atividades:

A4.1 – Definição da avaliação e criação do projetos de design a serem avaliados

A4.2 – Execução da conversão

A4.3 – Análise dos resultados

1.4. ESTRUTURA DO DOCUMENTO

No capítulo 2 deste documento são apresentados os conceitos que serão abordados ao longo deste trabalho. No capítulo 3 é feito uma revisão do estado da arte da última iteração da academia e indústria na geração de arquivos de importação do App Inventor usando o serviço Penpot. No capítulo 4 é apresentado a solução desde a análise de requisitos até detalhes da implementação da ferramenta. No capítulo 5 é feita uma avaliação da ferramenta desenvolvida. No capítulo 6 é apresentado a conclusão deste TCC.

2. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados os conceitos relevantes pertinentes a este trabalho iniciando pelo App Inventor, sua história, suas funções, vantagens e desvantagens no seu uso, suas ferramentas e formato de importação de projetos. Em seguida será apresentado o sistema Penpot e seus conceitos seguido de uma análise dos formatos de exportação da ferramenta.

2.1 APP INVENTOR

App Inventor (MIT, 2023) é um ambiente web de programação baseado em blocos que possibilita o desenvolvimento de aplicativos funcionais para Android.

A parte visual dos aplicativos é desenvolvida utilizando diversos recursos, desde componentes visuais até a seleção de recursos do dispositivo dispostos em um menu lateral (Figura 1.2) separados por seções (Figura 1.1). Para utilizar algum recurso, arrasta-se o componente desejado para a parte central da tela e disponibilizá-la na posição escolhida (quando se aplica).

Em termos de componentes visuais, o App Inventor fornece diversos recursos conforme indicado na Figura 1.2. Uma lista completa é apresentada no Apêndice A.

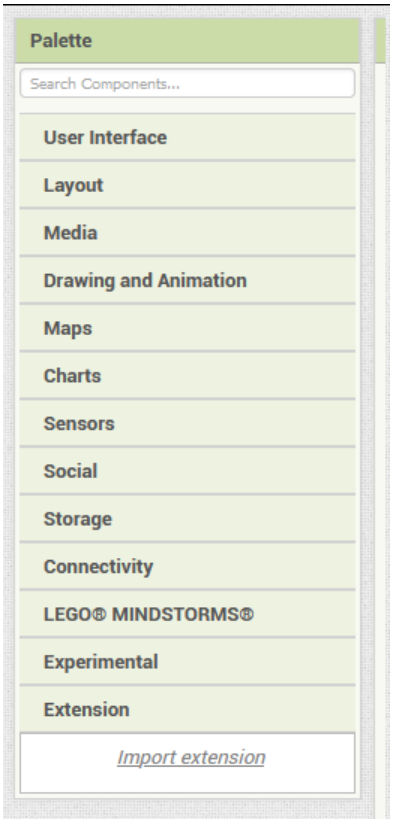


Figura 1.1 - Seções do Designer

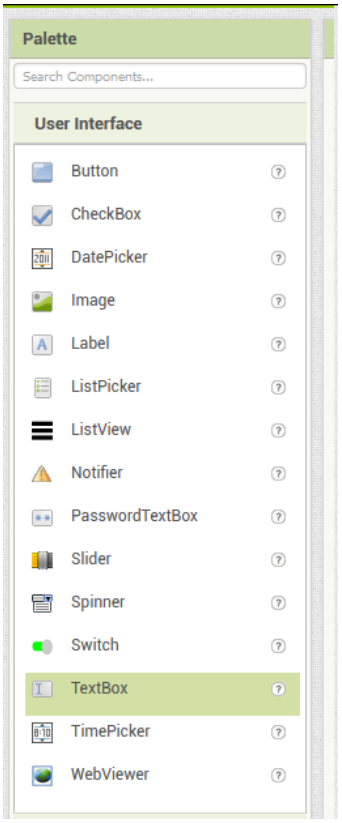


Figura 1.2. Componentes de interface de usuário

Para projetar o layout destes elementos o App Inventor também fornece a possibilidade de alinhamento destes elementos visuais tanto de forma horizontal quanto vertical (Figura 1.3).

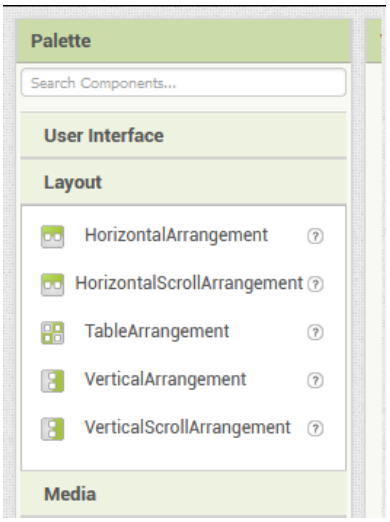


Figura 1.3 - Opções de layout de componentes App inventor

A pré-visualização da interface do *app* desenvolvido é exibida na parte

central da tela (Figura 1.4). A atualização da visualização dos elementos ocorre no momento em que suas características visuais são alteradas.

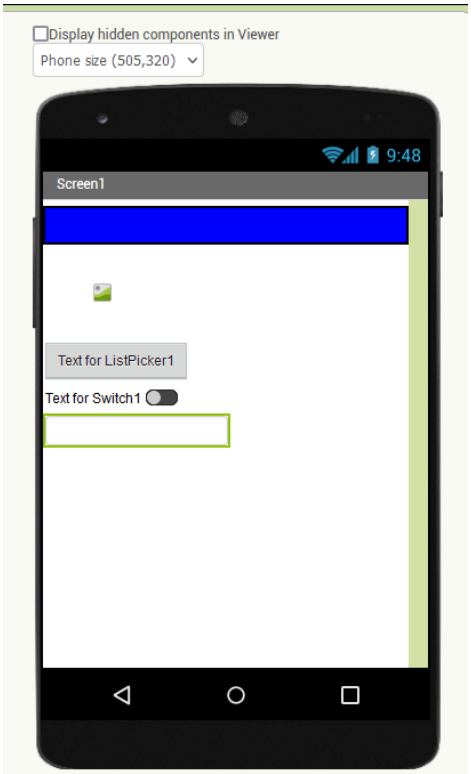


Figura 1.4 - Tela da aplicação em desenvolvimento

O desenvolvimento das funcionalidades das aplicações usando App Inventor é feito usando blocos visuais que representam as diferentes estruturas de dados e conceitos de programação (Figura 1.5).

Figura 1.5. Blocos de programação no *app* Inventor

2.1.1 Formatos de exportação/importação de projetos no App Inventor

O App Inventor disponibiliza opções de exportação e importação dos projetos. Os arquivos gerados para exportação são dispostos em uma variedade de pastas e arquivos de configuração e empacotados em uma única pasta compactada com o formato “.aia”. A extensão “.aia” é apenas um arquivo compactado “.zip” renomeado.

Os arquivos de exportação do formato “.aia” têm a seguinte estrutura (Figura 2.1):

- Uma pasta “*assets*” em que as imagens e elementos externos ao App Inventor que são adicionados são mantidos dentro do arquivo de exportação;
- Uma pequena estrutura de pastas que iniciam a partir de “src” em que estão os arquivos “.bky” e “.scm”:
    - Os arquivos “.bky” possui um XML internamente que contém a parte lógica é mapeada para ser recriada posteriormente em uma importação;
    - Os arquivos “.scm” possuem um JSON internamente em que os componentes visuais são mapeados para serem recriados posteriormente em uma importação.
- Uma pasta “youngandroidproject” que possui um arquivo “project.properties” com informações gerais sobre os macrocomponentes do App Inventor, como a pasta principal em que os arquivos “.bky” e “.scm” são armazenados, incluindo o tema geral do aplicativo, nome das telas adicionadas, entre outras.

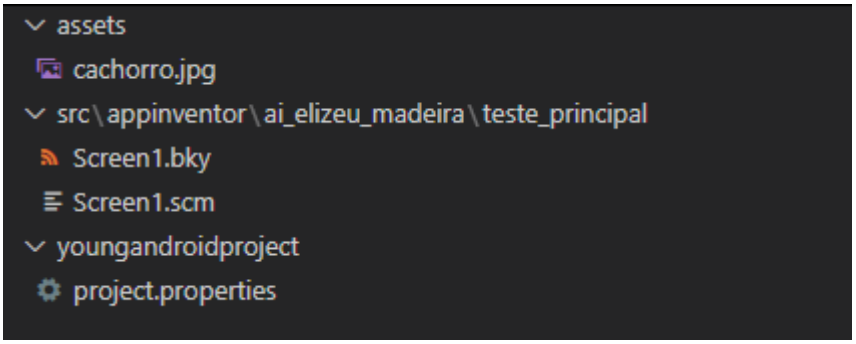


Figura 2.1 - Exemplo de uma estrutura de um arquivo “.aia” que possui apenas uma tela e uma imagem inserida

Arquivos “.bky”

Arquivos “.bky” armazenam uma estrutura XML (Figura 2.2) que representa a parte lógica da aplicação que são desenvolvidos usando blocos visuais dentro do App Inventor (Figura 2.3). Para cada tela do projeto, um arquivo “.bky” é adicionado. Cada declaração global de variável ou componente de repetição, por exemplo, representa uma *tag* XML. Dentro de uma tag que representa um bloco lógico, todas as condições e caminhos tomados pelas estruturas lógicas do aplicativo.

Figura 2.2 - Representação parcial da estrutura XML dos componentes lógicos exibindo a variável global "número" e do componente “if” com o teste realizado e os procedimentos para cada condição - representação arquivo “.bky”

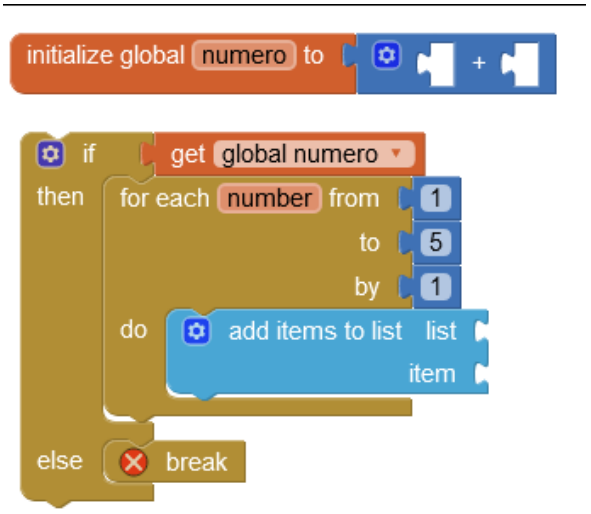


Figura 2.3 - Representação da variável global "número" e do componente “if” com o teste realizado e os procedimentos para cada condição - representação blocos visuais App Inventor

Arquivos “.scm”

Arquivos “.scm” armazenam a disposição dos componentes visuais adicionados ao projeto. Para cada tela do projeto, um arquivo “.scm” é adicionado. Dentro do arquivo é exibida uma estrutura json circundada por uma notação própria. O JSON adicionado contém informações básicas gerais sobre a tela adicionada acompanhado da propriedade “$Componentes” que armazena uma lista dos componentes visuais adicionados (Figura 2.4).

Figura 2.4 - Representação parcial do arquivo ".scm" exibindo informações gerais básicas sobre a tela "Screen1" acompanhado de alguns componentes: Barra superior horizontal Azul; Imagem; e uma lista.

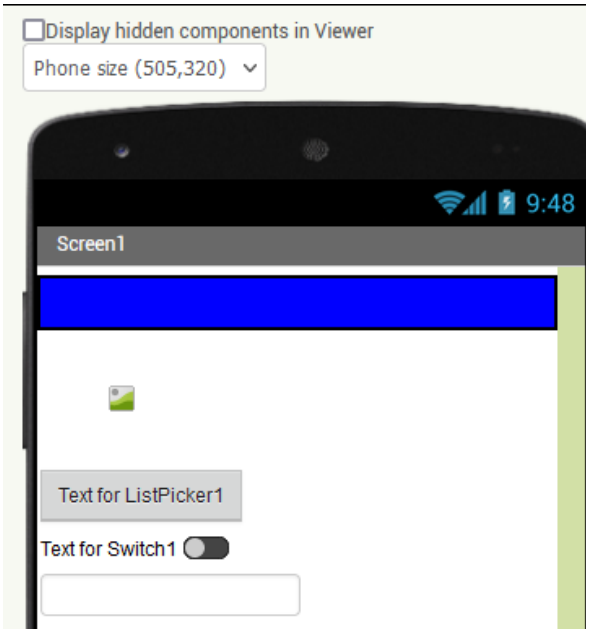


Figura 2.5 - Representação visual recriada pelo arquivo de importação ".scm" da imagem ao lado

2.2 PENPOT

Penpot (Penpot, 2023) é uma plataforma gratuita de *design* e prototipagem de sistemas para ser usada através de um navegador via protocolo HTTP. Penpot possui foco no uso de tecnologias abertas e padrões da indústria ao prover recursos para sua comunidade de forma fácil e acessível e viabilizando o *design* colaborativo (Figura 2.6). Dentre suas funcionalidades, o Penpot possibilita visualizar parte de seus *designs* no formato CSS e SVG, a importação de novas fontes e o reuso de componentes. Além disso, possui canais digitais para troca de conhecimento com a comunidade por meio de fóruns, vídeos explicativos e curadorias de conteúdos disponibilizados tanto pela comunidade quanto pela equipe do Penpot.

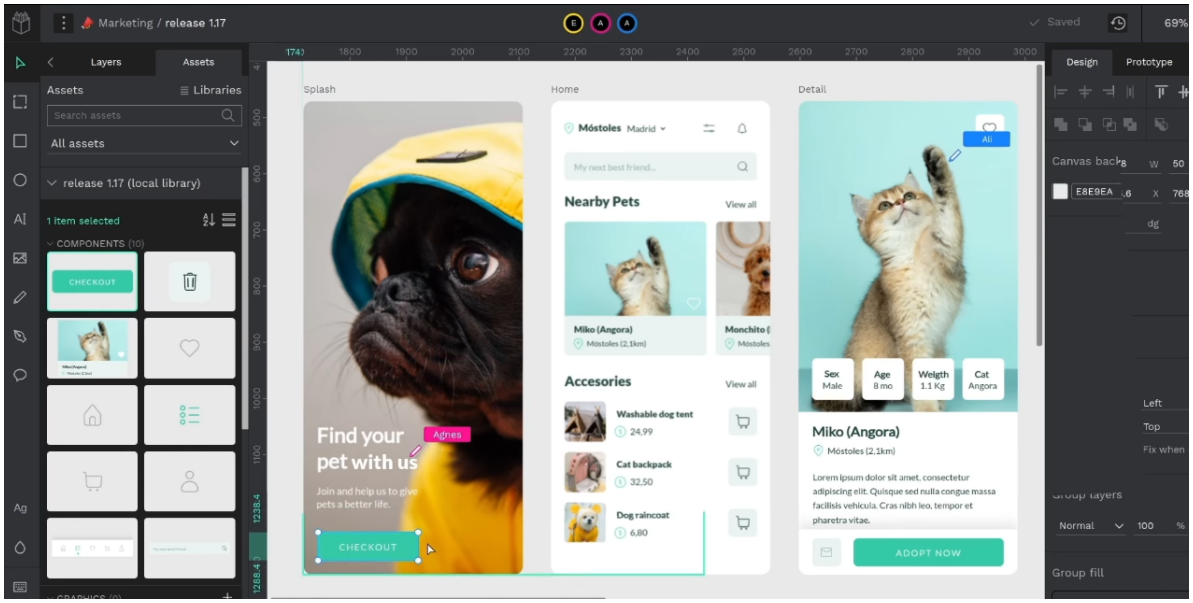


Figura 2.6 - Desenvolvimento de interfaces para app mobile usando a ferramenta Penpot

Penpot permite fazer o design de diversos elementos de interface de usuário a partir de formas básicas como retângulos e círculos, inserção de imagens e textos, curvas livres e comentários. Estes podem ser agrupados e renomeados. As formas geométricas que formam um componente podem ser agrupadas de forma hierárquica de forma a facilitar a visualização. Esta representação hierárquica se reproduz na exportação dos componentes. Os componentes são comumente organizados em *boards* que representam as diferentes telas dos sistemas a serem criados (Figura 2.7) e em páginas que contém seus próprios *layers*.

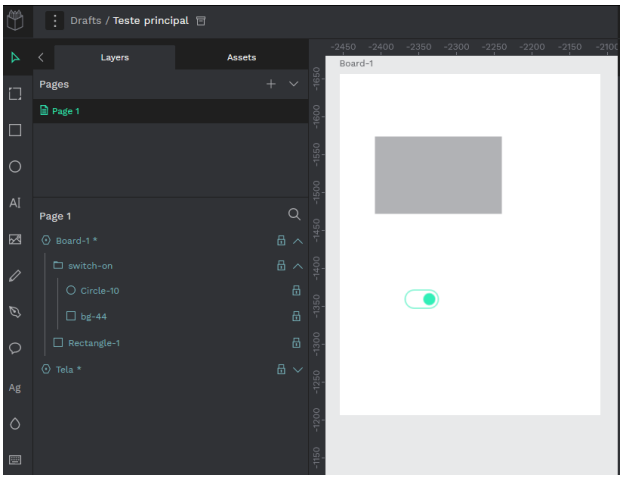


Figura 2.7 - Formas básicas inseridas no projeto

É possível também gerir os recursos importados para dentro do projeto como

bibliotecas de terceiros e uma biblioteca local (Figura 2.8). Ambos os tipos de bibliotecas possuem o mesmo sistema de organização e é explicado em mais detalhes no capítulo 2.2.1.

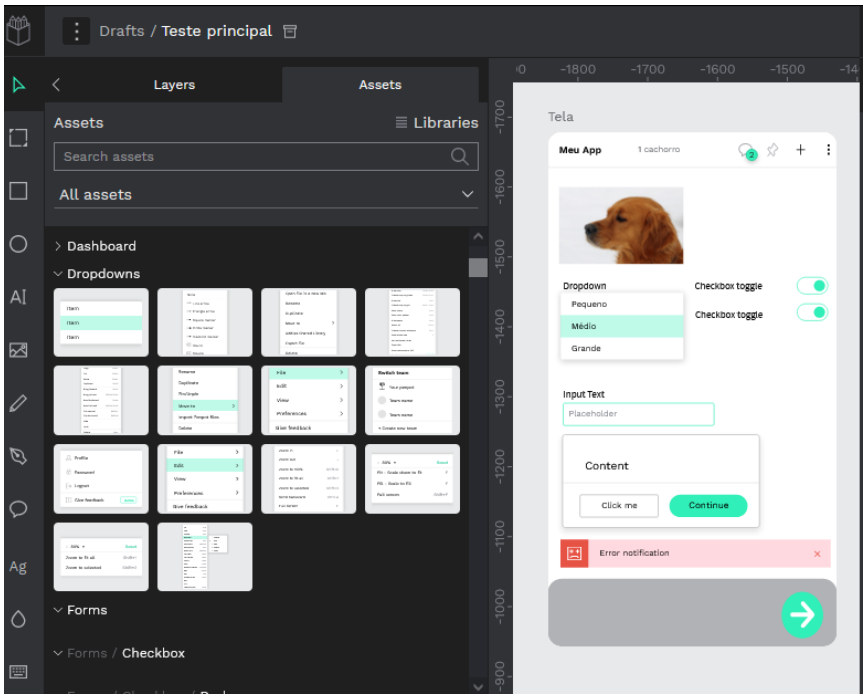


Figura 2.8 - Gestão de recursos do projeto

Cada um dos recursos possuem uma organização interna de forma a facilitar a classificação. Essa organização é de forma hierárquica como p.ex o componente de botão (Figura 2.9) que é organizado de acordo com a sua característica e sub-categorizado por sua aparência. A notação utilizada segue o critério da equipe de desenvolvimento ou do autor da biblioteca.

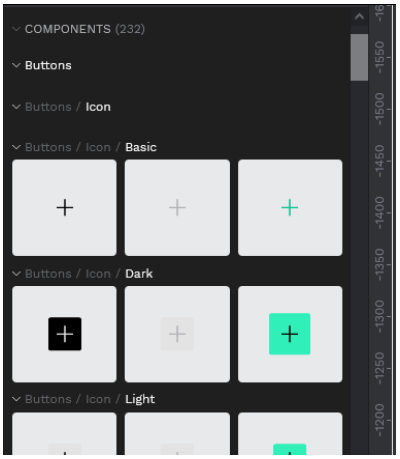


Figura 2.9 - Organização de um componente de Botão.

Os recursos são separados entre uma única biblioteca local e bibliotecas importadas. Ambos os tipos de bibliotecas possuem o mesmo sistema de organização.

É possível alterar as propriedades das formas criadas para adequar quaisquer *designs* como, por exemplo, a espessura e cor das bordas das formas geométricas, formatação do texto inserido, posicionamento, sombreamento, entre outros (Figura 2.10).

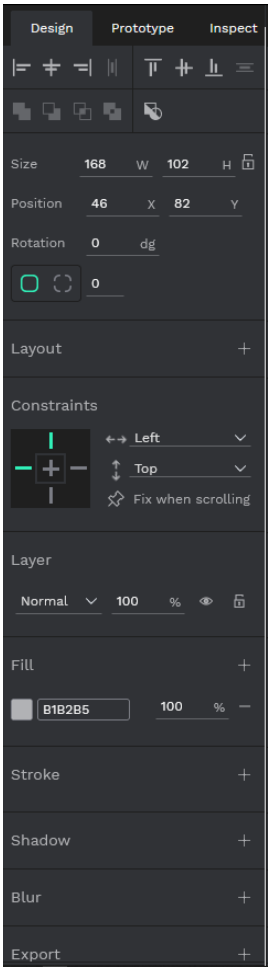


Figura 2.10 - Formatação do elemento

Também é gerado automaticamente o código CSS equivalente de todos os componentes criados para para representar o elemento em arquivos de exportação e CSS para adequação em projetos de sistemas *Web* (Figura 2.11).

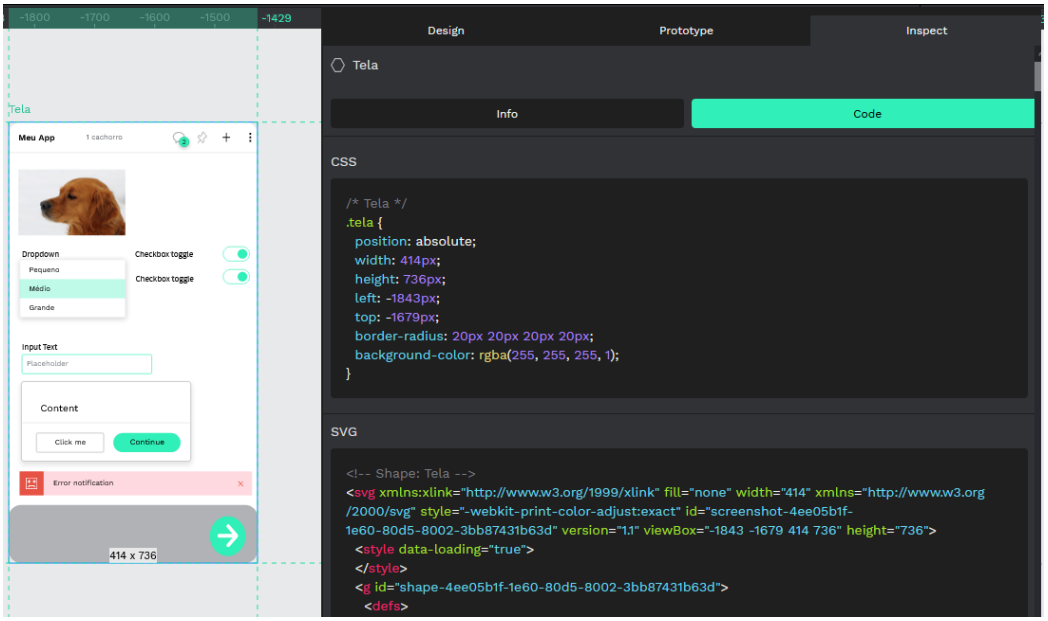


Figura 2.11 - Inspeção do elemento

2.2.1 Bibliotecas de componentes personalizados

Penpot permite a criação de bibliotecas de componentes personalizados. Os componentes devem ser criados a partir das formas geométricas disponíveis na barra de ferramentas ou a partir de outros componentes. Bibliotecas são formas e configurações pré-definidas que auxiliam na padronização visual do projeto. As bibliotecas podem ser importadas de terceiros e/ou entre projetos da mesma equipe. É possível criar componentes visuais como botões, *labels*, tabelas, etc, bem como agrupar fontes especiais, coleção de imagens e listagem de cores usadas em um projeto.

Diversas bibliotecas podem ser importadas em um mesmo projeto. Uma biblioteca que foi importada dentro do projeto será automaticamente atualizada conforme a fonte original da biblioteca for atualizada fora do projeto. Cada um dos componentes podem ser organizados de forma hierárquica de forma a facilitar a classificação como p.ex o componente de botão na Figura 2.12.

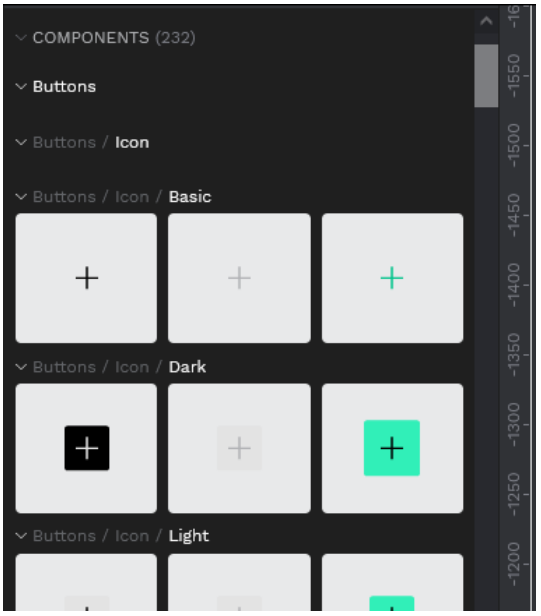


Figura 2.12 - Organização de um componente de Botão.

O Penpot disponibiliza a opção de exportar uma biblioteca local para uso em outros projetos, independente do formato escolhido para exportação (“.zip” ou “.penpot”).

Na biblioteca podem ser inseridos:

- Componentes inteiros (representação gráfica dos elementos usando formas geométricas do próprio Penpot) como botões, modais, caixas de textos, entre outros;
- Imagens nos formatos “.gif”, “.png”, “.svg”, “.webp”, “.jpg”, “.jpeg”, “.jfif”, “.pjpeg” e “.pjp” que podem representar ícones, imagens vetoriais diversas ou quaisquer imagens que podem ser reutilizadas (Figura 2.13);
- Cores pré-definidas que fazem parte da paleta do projeto;
- Tipografias personalizadas das fontes do projeto.

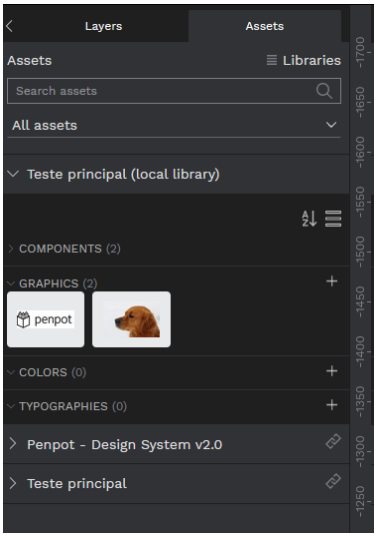


Figura 2.13 - Painel de recursos

Os designs de produtos são feitos através de formas básicas, recursos (imagens e fonte importadas para o projeto) e componentes (combinação de recursos e formas básicas que podem ser reutilizados) bibliotecas (Figura 2.14).

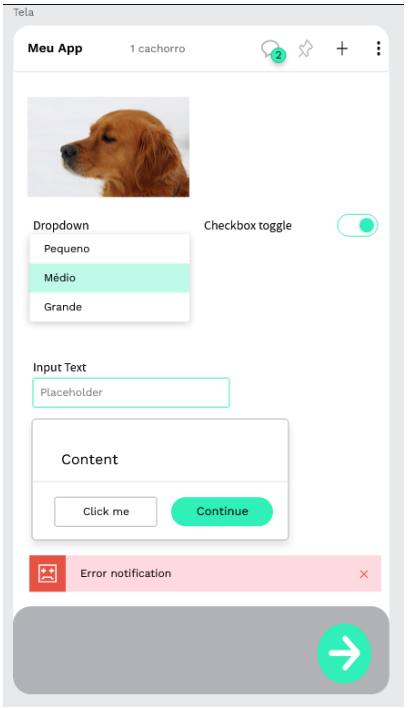


Figura 2.14 - Exemplo de tela de um sistema mobile usando os recursos de bibliotecas, formas básicas e textos simples

2.2.2 Formatos de exportação/importação Penpot

Penpot disponibiliza duas formas de exportar seus projetos, diretamente em um único “.svg” ou em formato de arquivo. Existem duas formas de exportar um

projeto Penpot no formato de arquivo: com extensão “.penpot” ou “*standard file*” que retorna um arquivo no formato “.zip”. Este arquivo “.zip” contém um manifesto e uma estrutura de pastas com arquivos “.svg” com o conteúdo do projeto. Apenas arquivos “.json” e “.svg” são exportados pelo modelo *standard file*. Todas as imagens utilizadas são criptografadas com o algoritmo *base64*.

O arquivo “.penpot” é um arquivo binário e pode ser importado somente no próprio Penpot. É mais rápido para gerar o arquivo do projeto e mais rápido para fazer a importação posteriormente. A extensão “.penpot” pode ser lida apenas pelo próprio Penpot.

Ao utilizar bibliotecas dentro do projeto, o Penpot exibe opções sobre como as bibliotecas serão tratadas dentro do projeto a ser exportado, independente do formato escolhido: exportar as bibliotecas de forma separada; incluir os arquivos das bibliotecas compartilhadas na biblioteca do arquivo; e tratar os arquivos das bibliotecas como objetos básicos (Figura 2.15). Para o formato “.penpot”, as bibliotecas e componentes respeitam as mesmas regras escolhidas para o formato “.zip”, no entanto, não é possível visualizar a organização interna do arquivo exportado por se tratar de um arquivo binário.

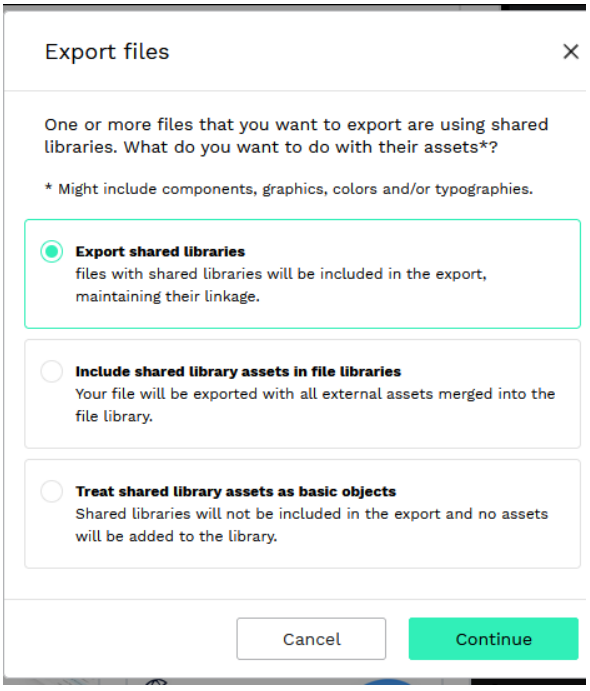


Figura 2.15 - Modal de exportação de projeto com bibliotecas

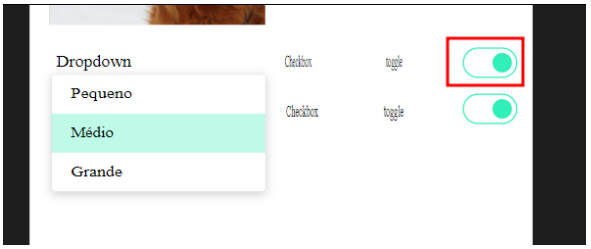


Figura 2.16 - Exemplo de componente de terceiro (destacado no retângulo vermelho à direita) importado no projeto

Caso o componente tenha sido alterado no projeto principal, suas propriedades originais ainda podem ser acessadas dentro da pasta do componente que é exportado separadamente usando disponível no arquivo de exportação e referenciado no manifesto do arquivo exportado conforme a Figura 2.15. Esta possibilidade não se aplica à opção “Tratar os arquivos das bibliotecas como objetos básicos” pois nenhuma biblioteca é exportada juntamente com o projeto.

```
1161    <symbol id="407436e0-efb1-11eb-836e-355af247e295" viewBox="0 0 44 24" penpot:path="Forms / Switch">
1162      <title>switch-on</title>
1163      <g id="shape-407436e0-efb1-11eb-836e-355af247e295" rx="0" ry="0">
1164        <penpot:shape penpot:name="switch-on" penpot:hidden="false" penpot:type="group"
1165          penpot:transform="matrix(1.000000, 0.000000, -0.000000, 1.000000, 0.000000, 0.000000)"
1166          penpot:transform-inverse="matrix(1.000000, -0.000000, 0.000000, 1.000000, 0.000000, 0.000000)"
1167          penpot:flip-x="false" penpot:flip-y="false" penpot:rotation="0" penpot:center-x="22"
1168          penpot:center-y="12"></penpot:shape>
1169        <defs></defs>
```

Figura 2.17 - Propriedades do componente “Switch ligado” no SVG da biblioteca original do componente exportado usando a opção “Incluir os arquivos das bibliotecas compartilhadas na biblioteca do arquivo”

Exportar as bibliotecas de forma separada

Para o formato “*standard file* (SVG + JSON)” e utilizando a opção “Exportar as bibliotecas de forma separada” as bibliotecas serão exportadas juntamente com o projeto principal. As bibliotecas são organizadas dentro de uma pasta separada, com todos os componentes que fazem parte da mesma, mesmo os que não são utilizados dentro do projeto original (Figura 2.18). Dessa forma, ao importar o projeto, as bibliotecas de terceiros utilizadas também são importadas no projeto.

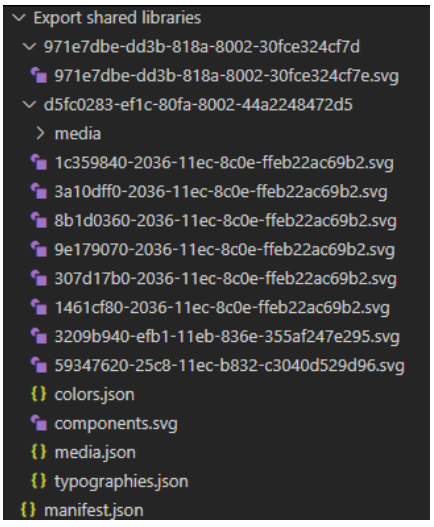


Figura 2.18- Hierarquia de um projeto Penpot com uma tela exportado usando a opção “Exportar as bibliotecas de forma separada”

Um componente criado no Penpot é representado por uma pequena hierarquia de nós e atributos dentro do XML exportado (Figura 2.19).

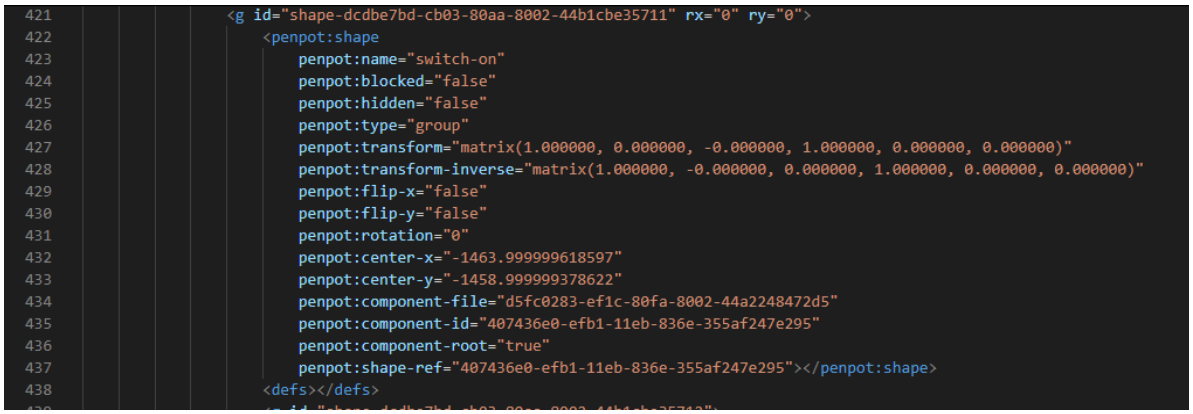


Figura 2.19 - Representação do componente “Switch ligado” no SVG da tela principal de um projeto Penpot exportado usando a opção “Exportar as bibliotecas de forma separada”

Dentre os atributos do componente exportado, os seguintes itens são relevantes pois servem de base para identificação e posicionamento na etapa de desenvolvimento da ferramenta, de acordo com a Figura 2.19:

- Linha 423: Nome do componente
- Linha 425: Indicativo se o componente está sendo exibido na página
- Linha 432: Posicionamento do centro do elemento no eixo X
- Linha 433: Posicionamento do centro do elemento no eixo Y
- Linha 434: Id do arquivo da biblioteca a qual o componente pertence
- Linha 435: Id do componente dentro da biblioteca a qual pertence

Incluir os arquivos das bibliotecas compartilhadas na biblioteca do arquivo

Utilizando a opção “Incluir os arquivos das bibliotecas compartilhadas na biblioteca do arquivo” o Penpot incorpora os arquivos e componentes das bibliotecas de terceiros utilizados como recursos da biblioteca particular ao exportar o projeto. Dessa forma, as bibliotecas utilizadas não serão exportadas e os componentes das bibliotecas de terceiros utilizadas no projeto não estarão discriminadas como tal. Apenas os componentes utilizados são exportados. O Penpot cria um arquivo chamado “components.svg” que contém os arquivos usados no projeto principal como uma biblioteca independente (Figura 2.20).

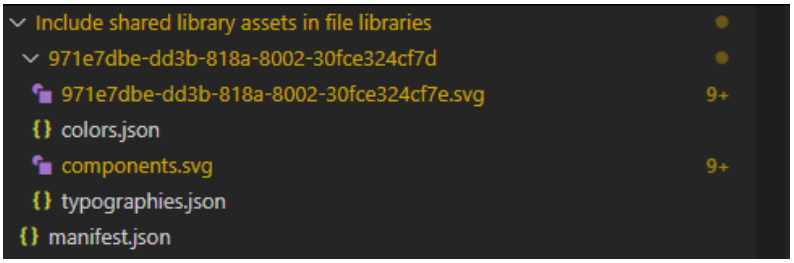


Figura 2.20 - Hierarquia de um projeto Penpot com uma tela exportado usando a opção “Incluir os arquivos das bibliotecas compartilhadas na biblioteca do arquivo”

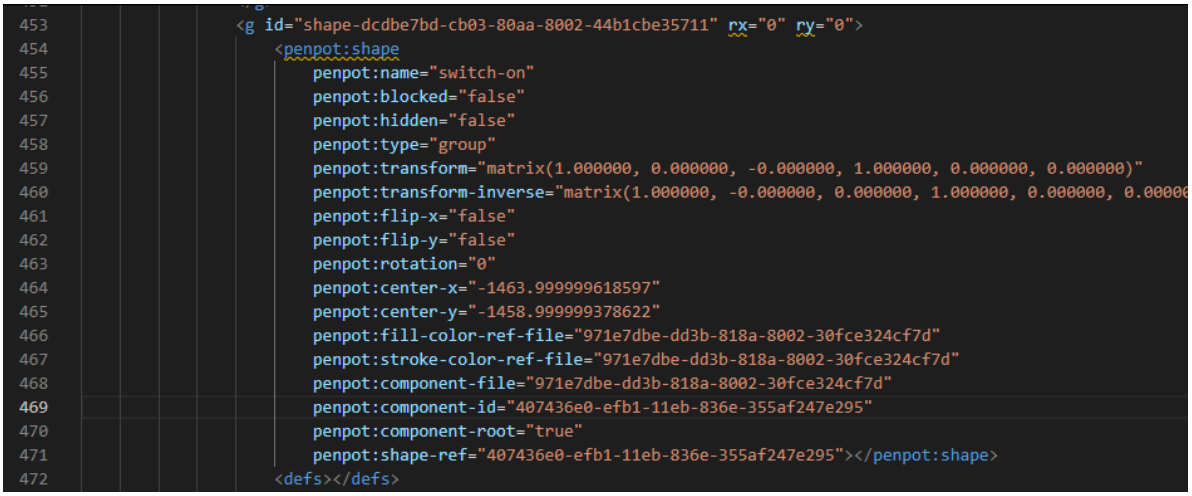


Figura 2.21 - Representação do componente “Switch ligado” no SVG da tela principal de um projeto Penpot exportado usando a opção “Incluir os arquivos das bibliotecas compartilhadas na biblioteca do arquivo”

Dentre os atributos do componente exportado, os seguintes itens são relevantes pois servem de base para identificação e posicionamento na etapa de desenvolvimento da ferramenta, de acordo com a Figura 2.21:

- Linha 455: Nome do componente
- Linha 457: Indicativo se o componente está sendo exibido na página
- Linha 464: Posicionamento do centro do elemento no eixo X
- Linha 465: Posicionamento do centro do elemento no eixo Y
- Linha 468: Id do arquivo da biblioteca a qual o componente pertence. Neste caso está referenciando o próprio arquivo.
- Linha 469: Id do componente dentro do arquivo “componentes.svg”

Tratar os arquivos das bibliotecas como objetos básicos

Utilizando a opção “Tratar os arquivos das bibliotecas como objetos básicos”

o Penpot não exporta os componentes e arquivos das bibliotecas utilizadas juntamente com o projeto. Ao invés disso, o Penpot incorpora os componentes das bibliotecas utilizados como objetos criados manualmente diretamente no projeto original. Ou seja, estes recursos (*assets*) não estarão disponíveis para reuso ao importar o projeto. Este comportamento se aplica aos componentes das bibliotecas locais e de terceiros. É o formato que exporta a menor quantidade de arquivos (Figura 2.22).

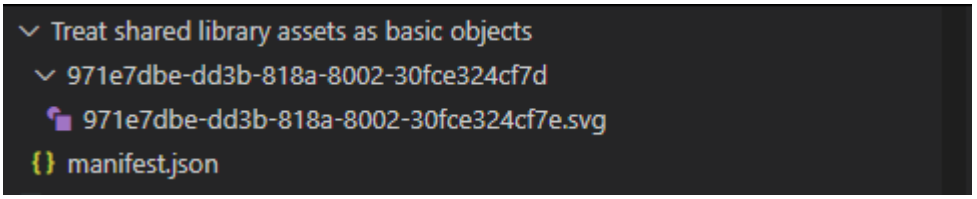


Figura 2.22 - Hierarquia de um projeto Penpot com uma tela exportado usando a opção “Tratar os arquivos das bibliotecas como objetos básico”

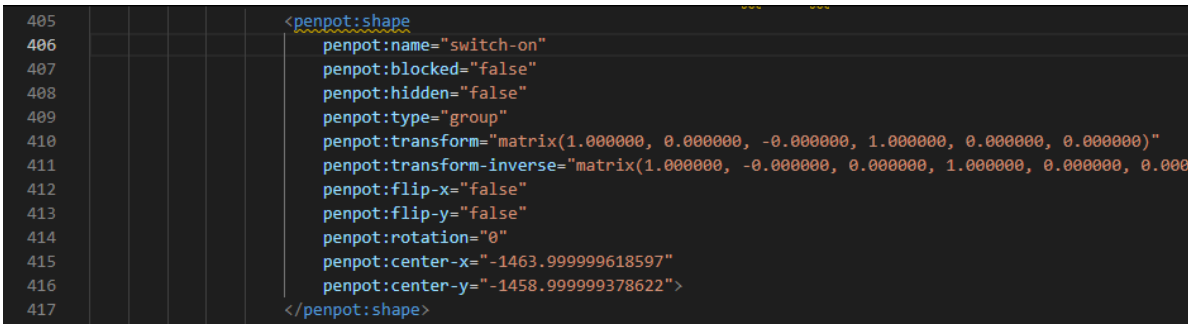


Figura 2.23 - Representação do componente “Switch ligado” no SVG da tela principal de um projeto Penpot exportado usando a opção “Tratar os arquivos das bibliotecas como objetos básico”

- Linha 406: Nome do componente
- Linha 408: Indicativo se o componente está sendo exibido na página
- Linha 415: Posicionamento do centro do elemento no eixo X
- Linha 416: Posicionamento do centro do elemento no eixo Y

Como nesta opção de exportação os componentes não são exportados, não é possível visualizar as propriedades originais do componente caso ele tenha sido modificado dentro do projeto original

3. ESTADO DA ARTE

A fim de levantar o estado da arte sobre a geração automática de *design* de interface de telas no App Inventor a partir de *designs* de interface criados no Penpot, são apresentados neste capítulo os resultados de um mapeamento sistemático da literatura com base no procedimento definido por Petersen *et al*. (2008).

3.1. DEFINIÇÃO DO PROTOCOLO DE REVISÃO

O objetivo da realização deste mapeamento sistemático é responder a seguinte pergunta de pesquisa: Quais abordagens existem para a **geração automática de *design* de interface de telas a partir de *design* de interface criado no Penpot?**

Para realização desse mapeamento, esta pesquisa foi refinada nas seguintes perguntas de análise:

PA1. Quais modelos/ferramentas existem para gerar interface do App Inventor a partir do Penpot?

PA2. Quais os dados de entrada/saída?

PA3. Quais elementos são detectados?

PA4. Qual tecnologia foi empregada para gerar estas interfaces?

PA5. Como foi medida a qualidade do resultado e quais resultados foram obtidos?

Fontes. Foram escolhidas para a realização da busca as principais bases de dados e bibliotecas digitais do campo da computação, incluindo ACM Digital Library, IEEE Xplore Digital Library, arXiv.org E-print Archive e Scopus com acesso via portal Capes. Para minimizar o risco de omissão de material relevante são realizadas também buscas no Google de forma complementar.

***String* de busca.** Com base na pergunta de pesquisa, para calibração da *string* de busca foram realizadas diversas buscas informais, com termos de buscas relevantes e seus sinônimos (Tabela 1). Usou-se também sinônimos para minimizar o risco de omissão de trabalhos relevantes.

Termo chave	**Sinônimos**	**Tradução**
design de interface de usuário		“user interface design”, UI design
App Inventor	code	“App Inventor”, code
Penpot	Figma, “ferramenta de design”	Penpot, Figma, “design tool”

Tabela 3.1 - Termos de busca e respectivos sinônimos

Após a realização das buscas informais, definiu-se uma *string* de busca específica, para aplicar nas bases de dados mencionadas de modo a encontrar todos os artigos relevantes previamente conhecidos, e um número satisfatório de artigos possivelmente relevantes adicionais:

```
("user interface design" OR “UI design”) AND ("App Inventor" OR “code”) AND (“Penpot” OR “Figma” OR "design tool")
```

Uma vez definida a *string* de busca, foi realizada sua adaptação para as diferentes bases de dados consideradas (Tabela 3.2).

Base de dados	***String* de Busca**
ACM Digital Library	[[All: "user interface"] OR [All: ui]] AND [All: design] AND [All: "app inventor"] AND [[All: penpot] OR [All: figma] OR [All: "design tool"]]
IEEE Xplore	("user interface" OR UI) AND design AND "App Inventor" AND (Penpot OR Figma OR "design tool")
Scopus	(TITLE-ABS-KEY ("user interface design") AND TITLE-ABS-KEY ("App Inventor" OR "code") AND TITLE-ABS-KEY ("design tool")) AND PUBYEAR > 2008 AND PUBYEAR > 2008
arxiv	order: -announced_date_first; size: 50; date_range: from 2009-01-01 ; include_cross_list: True; terms: AND all=design*; AND all=ui OR "user interface*"; AND all="App Inventor*"; AND all=Penpot OR Figma OR "design tool" OR "graphic design"
Google Scholar	("user interface" OR UI) AND design AND "App Inventor" AND (Penpot OR Figma OR "design tool")

Tabela 3.2 - Strings de buscas utilizadas nas diferentes bases de dados

Critérios de inclusão e exclusão. Para a seleção de artefatos relevantes são adotados os seguintes critérios de acordo com a pergunta de pesquisa:

- São considerados artigos científicos e artefatos que apresentem ferramentas

de geração de interfaces de usuário no App Inventor a partir de *designs* de interfaces em ferramentas de *design* gráfico.

- São incluídos apenas artefatos em inglês.
- São considerados apenas ferramentas que automaticamente geram o código das interfaces a partir de um *design* da interface.
- São considerados artefatos publicados em qualquer data.

Critérios de qualidade. Foram considerados apenas artefatos que apresentem informação substancial sobre a abordagem da geração automática.

3.2. EXECUÇÃO DA BUSCA

A busca dos artigos foi realizada em maio de 2023 pelo autor do presente trabalho e revisada pela orientadora. A busca inicial resultou em 6 artigos, dos quais foram selecionados artigos relevantes de acordo com os critérios de inclusão, exclusão e qualidade (Tabela 3.3).

Base de dados	**Quantidade de resultados da busca**	**Quantidade de resultados analisados**	**Quantidade de artigos potencialmente relevantes**	**Quantidade de artigos relevantes**
ACM Digital Library	6	6	1	1
IEEE Xplore	0	0	0	0
Scopus	2	2	1	1
arxiv	0	0	0	0
Google Scholar	82	82	1	1
Total (sem duplicados)				2

Tabela 3.3. Resultados da execução da busca

Por meio da leitura dos títulos e resumos de todos os trabalhos encontrados na busca inicial, foram determinados os artigos potencialmente relevantes ao tema de acordo com os critérios de inclusão e exclusão, sendo selecionados os artigos realmente relevantes por meio da sua leitura completa e aplicação dos critérios de qualidade.

Após a aplicação de todos os critérios, foram identificados apenas 2 trabalhos

relevantes publicados.

3.3 ANÁLISE DOS RESULTADOS

Para responder à questão de pesquisa, as informações relevantes às perguntas de análise foram extraídas do artigo relevante encontrado.

Os artigos selecionados foram lidos de forma completa e os dados foram extraídos pelo autor e revisados pela orientadora. No caso em que o artigo não apresenta nenhuma informação a ser extraída sobre um determinado dado, a falta desta informação é indicada como não informada (NI).

PA1. Quais modelos/ferramentas existem para gerar interface do App Inventor a partir do Penpot?

Não foi encontrado nenhum artigo implementando uma ferramenta ou apresentando um modelo para gerar interface do App Inventor a partir do Penpot.

A partir do refino dos termos de busca foram encontrados apenas alguns artigos em que um sistema de interface é convertido para um arquivo de importação de um outro sistema de desenvolvimento de aplicações. O primeiro deles, Pacheco *et al*. (2021), desenvolve um modelo de geração de arquivos de interface para desenvolvimento de sistemas Outsystems a partir da ferramenta “Sketch”. Embora os detalhes do desenvolvimento da ferramenta em si tivessem sido omitidos, as técnicas e as metodologias empregadas foram explicadas de forma satisfatória. O segundo artigo, Baulé (2020), utiliza *sketches* desenvolvidas usando papel e caneta como método de entrada para geração de *wireframes* no App Inventor. As *sketches* são convertidas em uma representação digital usando técnicas de *Deep learning* para posteriormente serem convertidas em um arquivo de importação do App Inventor. Os artigos podem ser encontrados de acordo com a referência mostrada na tabela 3.5.

Citação	Referência
(Pacheco *et al*., 2021)	Pacheco, J., Garbatov, S., & Goulao, M. Improving Collaboration Efficiency Between UX/UI Designers and Developers in a Low-Code Platform. In Proc. of ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion, IEEE, 2021.
(Baulé, 2020)	Baulé, Daniel de Souza. Desenvolvimento de um Modelo de Geração Automática de Wireframes no App Inventor a partir de Sketches usando Deep Learning. Trabalho e Conclusão de Curso (Bacharelado em Sistemas de Informação) - Universidade Federal de Santa Catarina, Florianópolis, 2020.

Tabela 3.5 - Artigos relevantes

PA2. Quais os dados de entrada/saída?

Observando os artigos que mencionam ou implementam alguma ferramenta de conversão de interface para código, foi possível observar alguma variação nos dados de entrada e saída conforme apresentado na Tabela 3.6.

Citação	Dados de entrada	Dado de saída
(Pacheco *et al*., 2021)	Arquivo Sketch ".sketch" que consiste de um arquivo compactado contendo uma estrutura de pastas e arquivos que representam a interface criada	Arquivo Outsystems ".oml"
(Baulé, 2020)	Fotos de *sketches* feito em papel	Arquivo ".aia" do App Inventor

Tabela 3.6 - Dados de entrada e saída das ferramentas desenvolvidas

Quanto aos dados de entrada, Pacheco *et al*. (2021) usa diretamente os dados exportados pela ferramenta *Sketch* no formato ".sketch" e exporta os elementos mais usados para arquivos ".oml", formato proprietário da plataforma Outsystems para criação de interfaces. Baulé (2020) é o único trabalho encontrado que trata de desenvolvimento de interfaces para App Inventor, porém usa como entrada fotos de *sketches* feitos com papel e caneta.

PA3. Quais elementos são convertidos?

Em ambas as pesquisas, apenas os elementos mais comumente utilizados foram considerados para converter de uma plataforma para outra (Tabela 3.7).

Citação	Quantidade de elementos considerados	Lista dos elementos considerados	Tipos de Software	Plataforma
(Pacheco *et al*., 2021)	20 elementos	NI	Apps	Outsystems
(Baulé, 2020)	9 elementos	Label, Button, Image, Textbox, ListPicker, Switch, CheckBox, Slider, Map	Apps	App Inventor

Tabela 3.7 - Lista de elementos mais comumente utilizados

Pacheco *et al*. (2021) escolheram os elementos consultando especialistas da Outsystems com o objetivo de identificar os elementos mais usados, porém no artigo não consta quais são. Baulé (2020) selecionou apenas 9 elementos que são utilizados em ao menos 5% de apps no App inventor.

PA4. Qual tecnologia foi empregada para gerar estas interfaces?

Pacheco *et al.* (2021) não explicam a tecnologia empregada. Baulé (2020) utilizou uma ferramenta própria desenvolvida em Python tanto para fazer as análises das imagens quanto para gerar o *script* de conversão para ".aia". A geração do protótipo em App Inventor é feita em duas etapas:

Etapa 1: Detecção dos componentes de interface usando um modelo de detecção de objetos (YOLOv3) usando *Deep Learning*. O resultado desta primeira etapa é uma listagem de componentes intermediária indicando o tipo de elemento, o nível de confiança e posições na imagem (coordenadas do ponto central, altura e largura). A listagem intermediária é salva apenas na memória em tempo de execução (Figura 3.1).

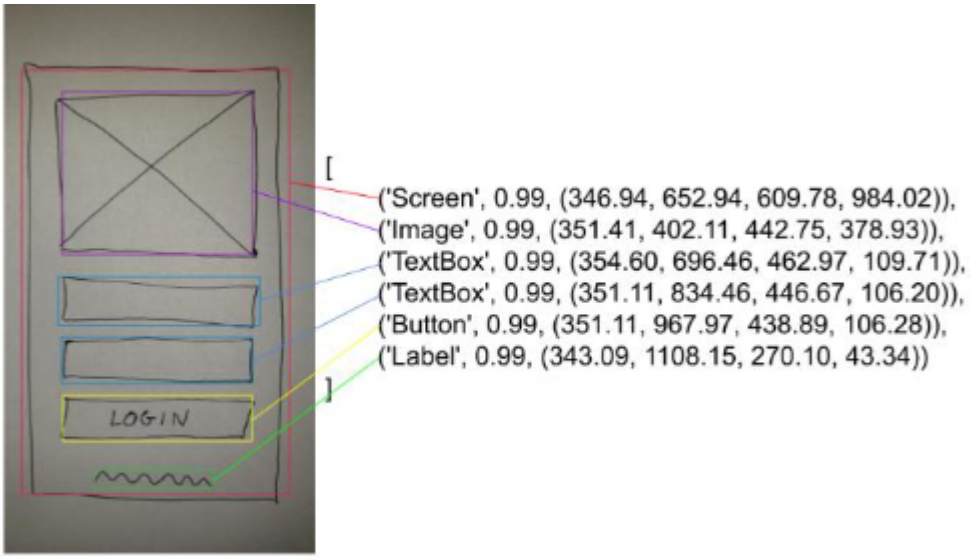


Figura 3.1 - Representação dos componentes detectados usando técnicas de *machine learning*

Etapa 2: A segunda etapa usa regras para criar e otimizar o protótipo de interface no App Inventor analisando a sobreposição de elementos e adequação dos elementos restantes a categorizações de posição. O passo final é converter a lista de componentes detectados para o sistema de posicionamento do App Inventor e exportá-los no formato “.scm”. Para que o App Inventor aceite o arquivo gerado é necessário encapsulá-lo em um arquivo “.aia” seguindo a mesma convenção vista no capítulo 2.1.1: Incluir um arquivo “.bky” juntamente com o arquivo “.scm” em uma estrutura de pastas que representa uma tela (cada tela possui uma pasta separada na mesma estrutura) sem configuração alguma, pois o sistema aborda apenas o aspecto visual do desenvolvimento e não seu funcionamento interno; Uma pasta “assets” com as imagens detectadas, embora nenhuma imagem seja exportada usando este sistema; e um arquivo de propriedades “project.properties” com informações básicas do app exportado.

Após a geração dos arquivos que compõem as telas e implementação das funcionalidades do sistema, o modelo exporta as propriedades gerais do app (nome do app, nome das telas, etc.) para que seja possível encapsular os arquivos gerados em uma extensão “.aia” do App Inventor.

PA5. Como foi medida a qualidade do resultado e quais resultados foram obtidos?

Pacheco *et al*. (2021) avaliou a qualidade com base em seis projetos reais usando uma métrica interna da Outsystems para calcular o total de componentes detectados pela ferramenta desenvolvida. O resultado foi de 99.6% na habilidade de identificar e instanciar um componente corretamente e 80% de componentes que a ferramenta consegue detectar. Após esta avaliação inicial os resultados com base nestes seis projetos foram encaminhados a cinco especialistas *front-end* da Outsystems. Os especialistas compararam componente a componente entre os projetos reais e a recriação gerada por meio da ferramenta. Embora nem todos os projetos se beneficiem da geração de interface, nos projetos em que é possível aplicar houve um acréscimo entre 150% a 400% no número de telas que podem ser criadas.

Baulé (2020) analisou a qualidade da sua abordagem por meio de um teste com 29 alunos e professores com formação em Ciência da Computação e/ou *Design* e 6 alunos e 1 professor representando o público-alvo no contexto da educação básica. Neste teste os participantes geraram o código para 10 imagens de *sketches* predefinidos e um *sketch* feito por eles mesmo (Figura 3.2 e 3.3). Foram avaliados 4 fatores: correspondência entre o resultado da ferramenta e *sketch*, tempo de processamento da ferramenta, usabilidade da ferramenta, vantagens e desvantagens da utilização da ferramenta no processo de desenvolvimento de um app. Referente a correspondência entre o resultado da ferramenta e *sketch*, a ferramenta obteve 100% de correspondência. O tempo médio de processamento da ferramenta foi 57,7 segundos para gerar um aplicativo com 3 telas, o que foi considerado satisfatório pelos participantes, tendo apenas 6,9% dos participantes consideraram o tempo de execução acima do esperado. A usabilidade da ferramenta obteve 100% de resultados positivos dos participantes e 96,6% avaliaram a ferramenta como fácil de usar. Além disso, foi adotado o SUS (*System Usability Scale*) com o qual obteve uma pontuação média de 92,16, indicando excelente grau de satisfação. As principais vantagens apontadas são a facilidade de uso, praticidade e potencial em economia de tempo no processo de desenvolvimento de um App. Nenhuma desvantagem foi apontada, apenas sugestões de melhorias como implementação de cores nos *previews*, melhoria nas instruções de uso e utilização de formatos variados e principalmente questões de posicionamento.

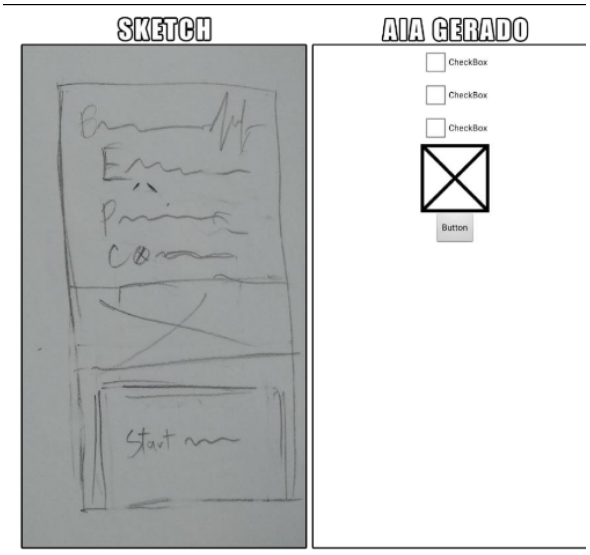


Figura 3.2 - Exemplo de aia gerado a partir de uma *sketch* feita a partir de papel e caneta

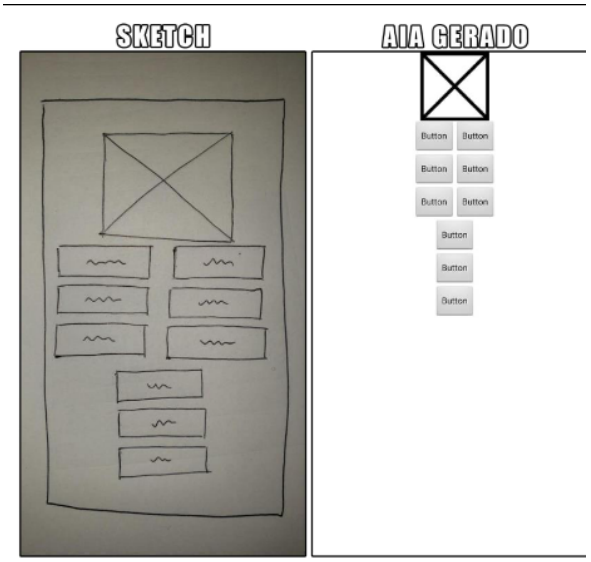


Figura 3.3 - Exemplo de aia gerado a partir de uma *sketch* feita a partir de papel e caneta

3.4 DISCUSSÃO

O resultado dessa revisão sistemática demonstra que ainda não existe nenhuma solução que gera um protótipo de App Inventor completo a partir do *design* de interface de usuário criado em uma ferramenta de *design* gráfico, como por exemplo o Penpot.

Muitos trabalhos e artigos que focam na geração de arquivos de importação para App Inventor têm foco no aspecto de desenvolvimento da lógica e não da interface. Os demais trabalhos encontrados que desenvolvem a conversão de uma plataforma para outra ou que possuem foco em geração de arquivos possuem foco em outros sistemas como por exemplo o caso de Pacheco *et al.* (2021) que exporta para Outsystems ou diretamente para arquivos HTML/CSS para ser renderizado normalmente por um *browser*.

Os poucos trabalhos encontrados mais relacionados com o foco da presente pesquisa mostram estratégias semelhantes de classificação de elementos para geração de um modelo intermediário. Ambos consideram apenas os elementos mais utilizados ao invés de uma abordagem mais completa, além disso, elementos visuais temporários como p.ex modais também são desconsiderado e/ou alinhado p.ex. a todos os elementos visuais citados por guias de estilo como p.ex. o *Material Design* 3 (Google, 2023). Estas ferramentas também somente abordam os aspectos visuais sem que a regra de negócio fosse considerada. Observa-se também que os

trabalhos possuem margem de acerto que são consideradas aceitáveis pelos seus usuários pois suas entradas são parte do desenvolvimento de aplicações reais e o tempo de conversão dessas ferramentas são considerados curto. Dessa forma, mesmo que o modelo gerado tenha que ser corrigido parcialmente ou mesmo rejeitado totalmente por seus usuários, desenvolver novamente a interface não interfere no prazo de entrega dos sistemas devido ao tempo de conversão dessas ferramentas.

As avaliações das ferramentas encontradas indicam que é possível com um grau de acurácia aceitável a geração dos *wireframes* com tempo de processamento aceitável e uma boa satisfação dos avaliadores.

Existem *plugins* implementados por terceiros para geração de *wireframe* em Android Studio a partir do Figma (Android Studio, 2023) utilizando um dos assistentes de importação do Android Studio e *plugins* comerciais que possibilitam a importação de projetos criados no Adobe XD para o Android Studio como, p.ex, o “Export Kit” (Export Kit). Porém ainda não existem *plugins* deste tipo para a ferramenta Penpot.

Ameaças à validade da revisão da literatura. Para o mapeamento sistemático algumas ameaças a validade foram levantadas e para cada ameaça foi aplicado uma estratégia para mitigar seus eventos minimizando seus impactos.

A omissão de estudos relevantes é um risco mitigado por meio da construção cuidadosa de uma *string* de busca, no entanto, estudos que não abrangem explicitamente o uso das ferramentas “Penpot” em conjunto com a ferramenta “App Inventor” não possuem relevância e mesmo abrangendo mais ferramentas com a inclusão de outros termos que possam estar vinculados a pesquisa nenhum estudo parece estar sendo feito nesse sentido. Além disso, a ferramenta “Penpot” foi lançada a pouco tempo sendo improvável que estudos envolvendo a ferramenta estejam em curso. Por isso, para reduzir o risco de omissão foram também incluídos sinônimos no *string* de busca. Também foi feita uma busca em diversos portais de artigos com relevância acadêmica para assegurar que o número máximo de resultados fossem obtidos.

O fato de que somente um pesquisador, o autor deste trabalho, realizou esta revisão sistemática, pode comprometer a isenção da aplicação dos critérios de inclusão e exclusão. Porém para minimizar estes riscos, seguiu-se exatamente o

protocolo da revisão definido, e os resultados da busca, seleção e análise foram discutidos com a orientadora até um consenso ser obtido.

4. SOLUÇÃO: Penpot2AIA

Este capítulo apresenta o modelo desenvolvido para a geração automática de *wireframes* no App Inventor a partir de arquivos de exportação do Penpot.

4.1 ANÁLISE DE REQUISITOS

O objetivo deste trabalho é o desenvolvimento de um modelo de geração automática de *wireframes* em App Inventor a partir de arquivos de exportação da ferramenta Penpot. O modelo é dividido em duas etapas de acordo com a Figura 4.1.

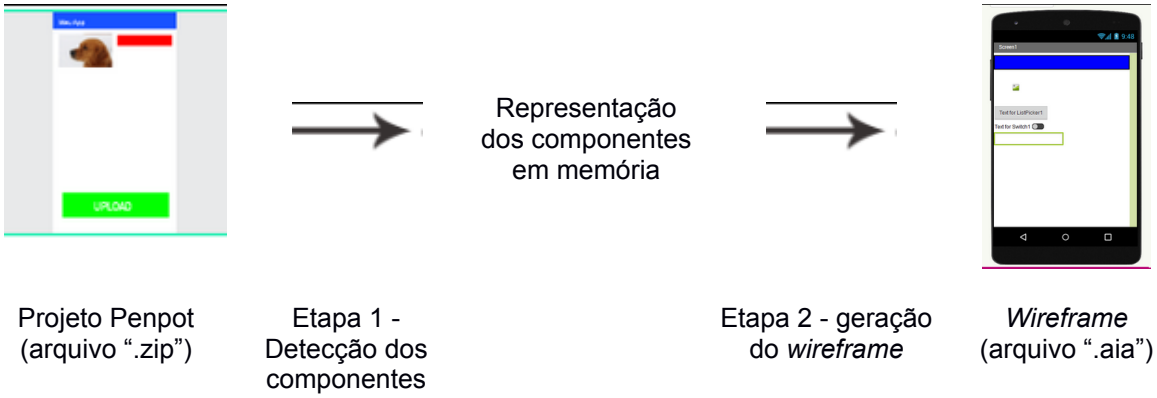


Figura 4.1- Etapas de geração de *wireframes* a partir de arquivos de exportação Penpot

Assume-se que foi criado o projeto de *design* de interfaces de usuário do app usando a ferramenta Penpot. Assume-se que para a criação deste projeto de *design* de UI foi usada uma biblioteca pré-definida de componentes alinhados ao Material *Design* 3 (Google, 2023), conforme apresentado no Anexo A. Ao final da criação do projeto de *design* de UI no Penpot, assume-se que o projeto é exportado no Penpot no formato ".zip" (SVG + JSON). A exportação desse projeto é a entrada do sistema da geração do *wireframe* no App Inventor.

A primeira etapa é a detecção dos elementos presentes no projeto Penpot. Nesta etapa a ferramenta identifica as telas e para cada tela os elementos presentes no projeto, sua posição na tela, atributos básicos como cores, tamanho, etc. Essas informações são armazenadas em memória.

Na segunda etapa a ferramenta gera de forma automática os *wireframes* na App Inventor a partir dos elementos detectados no arquivo de exportação do Penpot

no primeiro passo. Para isso, a ferramenta cria a hierarquia de arquivos que formam a representação do sistema em arquivos “.aia”.

A saída do processo de conversão é o arquivo “.aia”, que então pode ser utilizado pelo usuário importando no App Inventor para continuar o processo de desenvolvimento de aplicativo móvel com base no projeto de *design* feito anteriormente no Penpot.

Requisitos funcionais		
RF1. Fazer upload dos arquivos “.zip” do Penpot	**Entrada**: projeto exportado no Penpot no formato “.zip” (svg + json)	**Saída**: arquivo “.aia”
RF2. Detectar elementos de UI a partir de um projeto Penpot enviado para a ferramenta	**Entrada**: arquivo “.svg” contido do “.zip” (svg + json)	**Saída**: definição dos elementos de design de UI armazenados em memória
RF3. Gerar automaticamente o wireframe a partir dos elementos detectados	**Entrada**: definição dos elementos de design de ui armazenados em memória	**Saída**: arquivo “.scm”
RF4. Detectar automaticamente imagens inseridas no projeto Penpot enviado para a ferramenta	**Entrada**: imagens codificadas contidas na tag de imagem que faz parte do elemento detectado	**Saída**: arquivo “.png” ou “.jpg”
Requisitos não funcionais		
Performance: o processo não deve levar mais do que 5 min		
Integração: com a ferramenta Penpot		
Plataforma: o sistema deve ser acessível a partir de um endereço *web* através de um navegador de internet.		
Restrições: necessário navegador web compatível com a tecnologia *stream* tais como: *Chrome* 89; *Edge* 89; *Firefox* 102; *Opera* 76.		
Idioma: todas as telas devem estar disponíveis no Português do Brasil e Inglês		
Apenas imagens nos formatos “.png” e “.jpg” serão identificadas nos projetos enviados para conversão		

4.2 ETAPA 1: DETECÇÃO DE COMPONENTES DE INTERFACE

A criação do sistema web responsável por converter o projeto Penpot para “.aia” inicia-se pela análise dos formatos de exportação do Penpot (Figura 4.2). O

Penpot disponibiliza duas formas de exportar os projetos criados: diretamente em imagem ou um único arquivo que agrega todas as informações e arquivos necessários para recriar o projeto em outra conta. Exportando diretamente em imagem é possível escolher entre quatro formatos: SVG, PDF, JPEG ou PNG. Escolhendo o formato de exportação por imagem os componentes são exportados separadamente. O Penpot permite que seja exportado mais de um componente e/ou um mesmo componente em mais de um formato, nestes casos um arquivo “.zip” será exportado para que todos os componentes sejam disponibilizados ao mesmo tempo. Exportando diretamente em arquivos de exportação do Penpot, o sistema disponibiliza duas opções: “.penpot” e “.zip ". O formato “.penpot” é um arquivo binário que pode ser lido somente pelo próprio Penpot. A opção “.zip” disponibiliza 3 opções de como os arquivos serão exportados internamente dentro do arquivo “.zip”. Uma análise mais detalhada de cada opção foi descrita no capítulo 2.2.2.

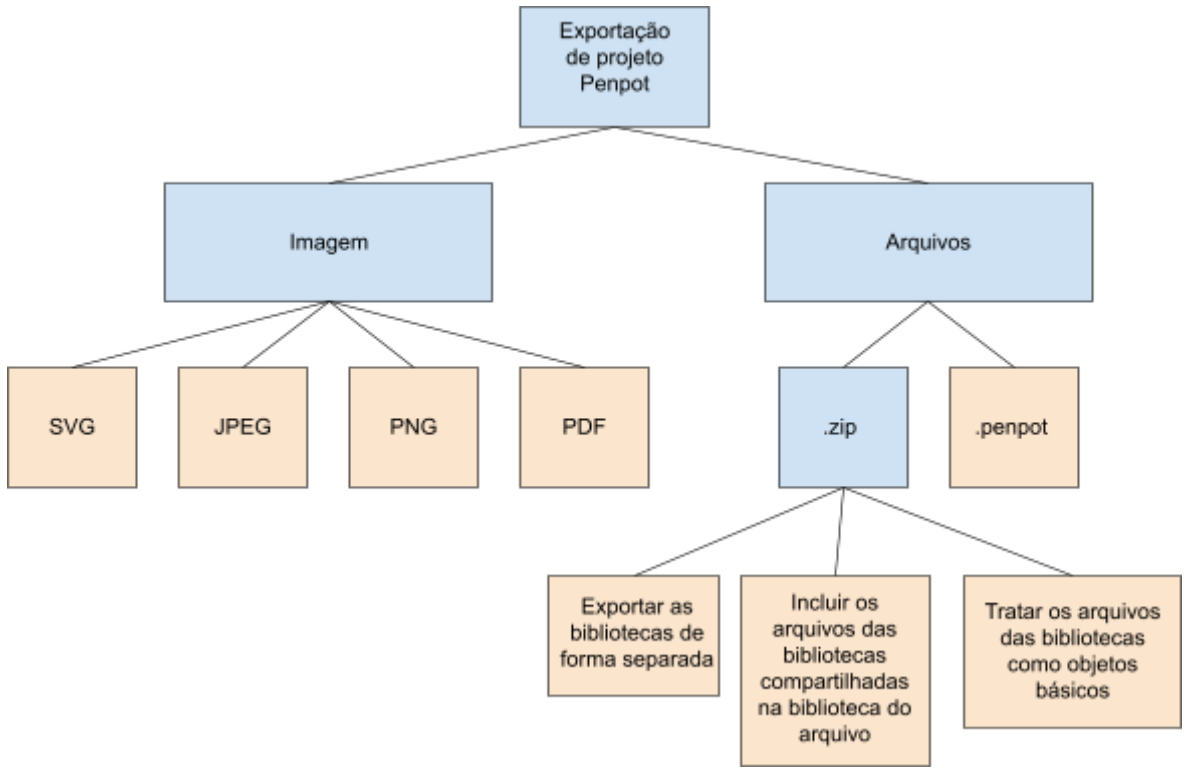


Figura 4.2 - Formatos de exportação Penpot

Para fazer a detecção dos elementos foi escolhida a opção “.zip” nas opções de exportação por arquivos. Ao exportar por este formato o nome do elemento junto de suas propriedades (dimensões, posição, cores, texto interno, etc) são exportadas

em um diretório interno do “.zip” no formato “.svg” separados por página.

Para restringir o número de componentes, foi criada uma biblioteca de Penpot que contém somente os componentes aceitos pelo sistema (Tabela 4.1). A detecção dos componentes contempla somente os componentes do projeto criados a partir dessa biblioteca. A biblioteca é disponibilizada pelo Penpot2AIA. Esse conjunto de componentes foi selecionado levando em consideração os elementos de interface mais tipicamente utilizados em aplicativos App Inventor.

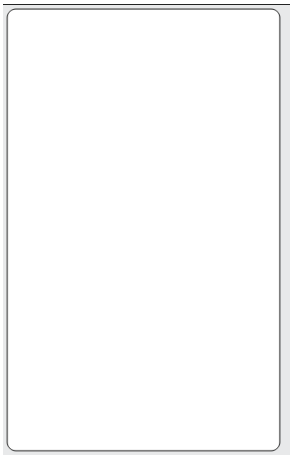
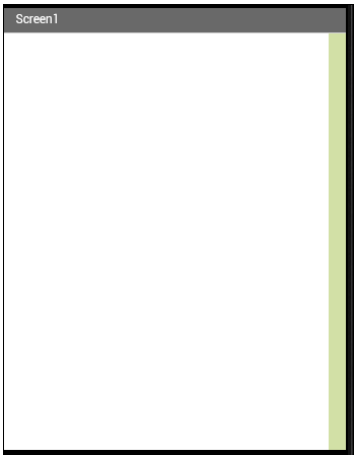
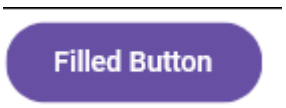
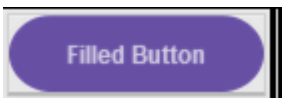
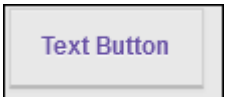
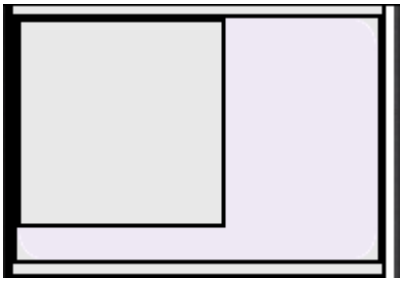
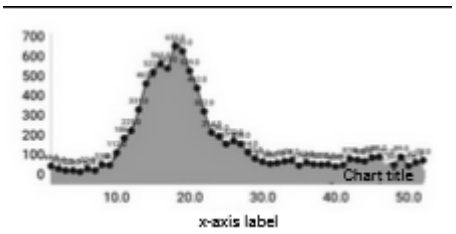
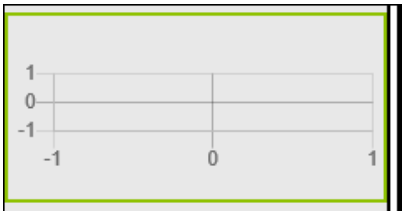
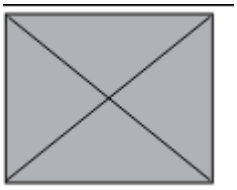
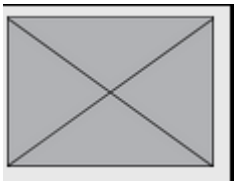
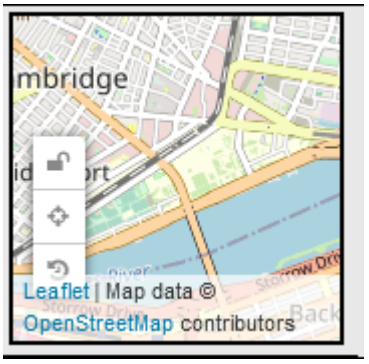
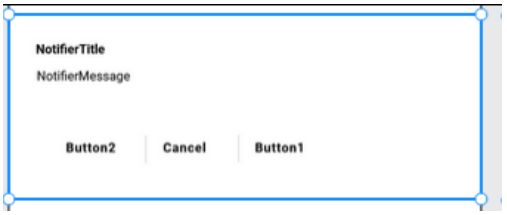
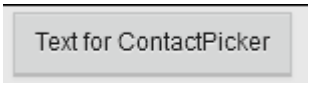
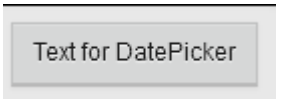
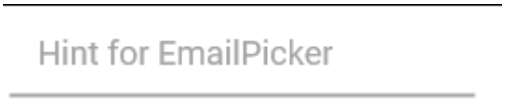
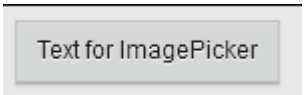
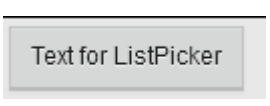
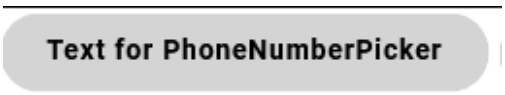
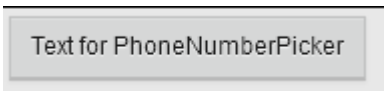
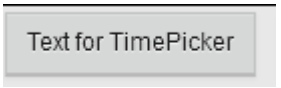
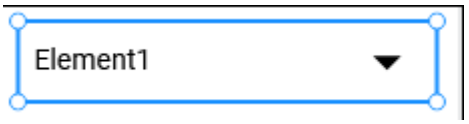
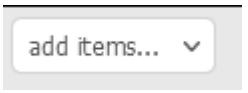
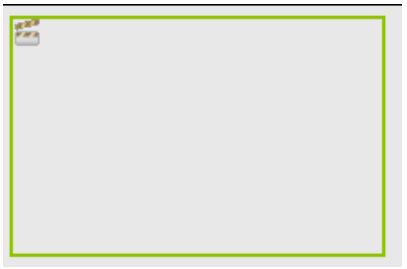
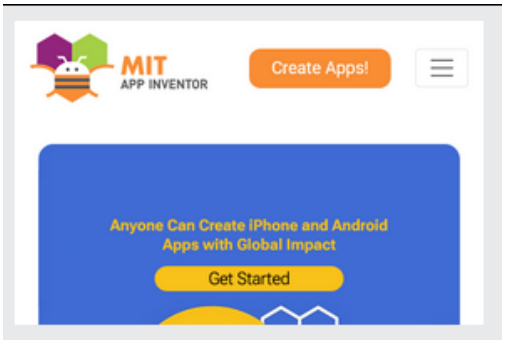
Componente	**Representação visual no Penpot**	**Representação visual no App Inventor**
Tela		
App Bar		
Button Filled		
Button no background		
Card		

Chart		
Image		
Image		
Notification		
Contact Picker		
Date Picker		
Email Picker		
Image Picker		
List Picker		

Phone Number Picker		
Time Picker		
Spinner		
Password Textbox		
Text Box - hint		
Top app bar - kebab		
Videoplayer		
Webviewer		
Switch		
Slider		

Checkbox		

Tabela 4.1 - Lista de componentes contemplados pelo sistema Penpot2AIA

Para iniciar a detecção dos elementos é necessário enviar o arquivo “.zip” de exportação Penpot no sistema Penpot2AIA. O arquivo enviado possui um arquivo “manifest.json” e um diretório contendo um “.svg” para cada página exportada com seus respectivos componentes descritos internamente. O “manifest.json” possui tanto o nome da pasta exportada quanto o nome dos arquivos das páginas e acessar essas informações é o procedimento do sistema para detectar os elementos.

O arquivo da página do projeto Penpot é um arquivo “.svg” mas pode ser lido como um arquivo “.xml”. Cada componente é representado por um nodo contendo as informações necessárias para criar a representação em memória de componentes. A identificação do componente é feita usando a propriedade “penpot:name”. As propriedades necessárias para incluir o componente na representação em memória são: “largura”, “altura”, “ponto central x”, “ponto central y”, “posição x” e “posição y”. Para alguns componentes a propriedade “texto” e “cor” também são necessárias.

Cada componente possui uma estrutura xml interna (Figura 4.3) e as informações podem estar em qualquer nível da representação. Os atributos que podem conter essas informações são: “@penpot:svg-viewbox-width” e “@width” para largura; "@penpot:svg-viewbox-height" e "@height" para altura; "@x" para a posição x; "@y" para a posição y; "@penpot:center-x" para o ponto central x; "@penpot:center-y" para o ponto central y; "@penpot:content" para a cor interna do componente; “#text” para o texto interno do componente. Para alguns componentes a largura e altura são restritos a valores pré-definidos e não precisam ser detectados pelo sistema. Há também alguns casos em que a largura e altura ou dimensões não são informadas pelo “.svg”. Nesses casos é possível determinar as propriedades faltantes a partir das demais.

```
<g id="shape-b991647b-a558-8080-8002-e6db72e26ac4" rx="0" ry="0">
    <penpot:shape penpot:name="Chart - area" penpot:type="group"
        penpot:transform="matrix(1.000000, 0.000000, 0.000000, 1.000000, 0.000000, 0.000000)"
        penpot:transform-inverse="matrix(1.000000, 0.000000, 0.000000, 1.000000, 0.000000, 0.000000)"
        penpot:rotation="0" penpot:center-x="323" penpot:center-y="83.5"
        penpot:fill-color-ref-file="c65f3168-697c-808c-8002-d61a11b1d881"
        penpot:stroke-color-ref-file="c65f3168-697c-808c-8002-d61a11b1d881"
        penpot:component-file="c65f3168-697c-808c-8002-d61a11b1d881"
        penpot:component-id="b991647b-a558-8080-8002-e6daa72287c2" penpot:component-root="true"
        penpot:shape-ref="b991647b-a558-8080-8002-e6daa72287c2">
        <penpot:layout-item penpot:layout-item-h-sizing="fix" penpot:layout-item-v-sizing="fix">
        </penpot:layout-item>
    </penpot:shape>
    <defs></defs>
    <g id="shape-b991647b-a558-8080-8002-e6db72e26ac5" rx="0" ry="0">
        <penpot:shape penpot:name="Chart - area" penpot:type="group"
            penpot:transform="matrix(1.000000, 0.000000, 0.000000, 1.000000, 0.000000, 0.000000)"
            penpot:transform-inverse="matrix(1.000000, 0.000000, 0.000000, 1.000000, 0.000000, 0.000000)"
            penpot:rotation="0" penpot:center-x="323" penpot:center-y="83.5"
            penpot:fill-color-ref-file="c65f3168-697c-808c-8002-d61a11b1d881"
            penpot:stroke-color-ref-file="c65f3168-697c-808c-8002-d61a11b1d881"
            penpot:component-file="c65f3168-697c-808c-8002-d61a11b1d881"
            penpot:component-id="0bb03d6e-f3b9-80aa-8002-1b18ff907f68"
            penpot:shape-ref="0bb03d6e-f3b9-80aa-8002-1b18ff907f68"></penpot:shape>
        <defs></defs>
        <g id="shape-b991647b-a558-8080-8002-e6db72e26ac6">
            <penpot:shape penpot:name="areachartBW.jpg" penpot:type="image"
                penpot:transform="matrix(1.000000, 0.000000, 0.000000, 1.000000, 0.000000, 0.000000)"
                penpot:transform-inverse="matrix(1.000000, 0.000000, 0.000000, 1.000000, 0.000000, 0.000000)"
                penpot:proportion="2.074235807860262" penpot:proportion-lock="true" penpot:rotation="0"
                penpot:center-x="319.609865470852" penpot:center-y="79.10964912280701" penpot:rx="0" penpot:ry="0
                penpot:fill-color-ref-file="c65f3168-697c-808c-8002-d61a11b1d881"
                penpot:stroke-color-ref-file="c65f3168-697c-808c-8002-d61a11b1d881"
                penpot:component-file="c65f3168-697c-808c-8002-d61a11b1d881"
                penpot:shape-ref="0bb03d6e-f3b9-80aa-8002-1b18ff907f69">
                <penpot:layout-item penpot:layout-item-h-sizing="fix" penpot:layout-item-v-sizing="fix">
                </penpot:layout-item>
            </penpot:shape>
            <defs>
                <pattern patternUnits="userSpaceOnUse" x="239" y="38" height="86.33026315789473"
                    width="169.28071748878924" data-loading="false" id="fill-0-rumext-id-5">
                    <g>
                        <g>
                            <rect x="0" y="0" width="169.28071748878924" height="86.33026315789473" fill="none">
                            </rect>
                            <image
```

Figura 4.3 - Estrutura XML de um componente

A saída da detecção de componentes UI a partir de arquivos de exportação ".zip" do Penpot é uma representação em memória do nome do projeto e das telas com seus respectivos componentes (Figura 4.4).

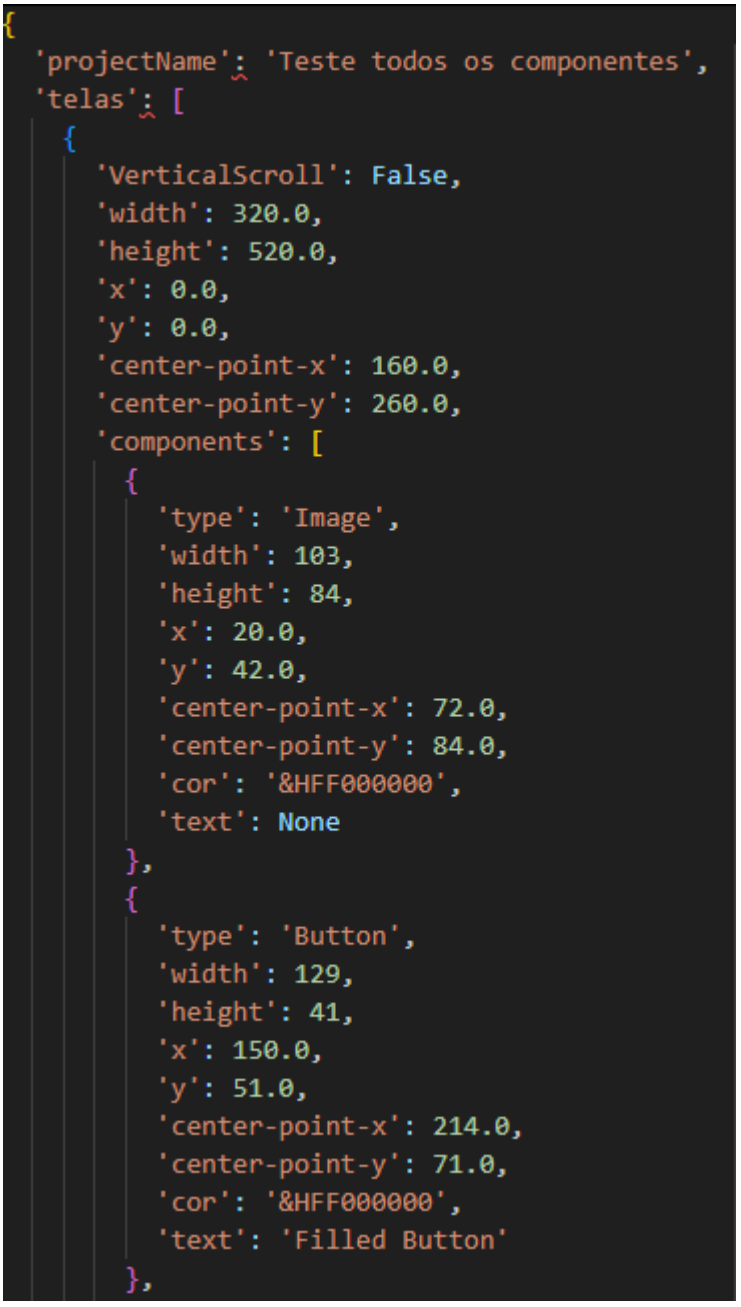


Figura 4.4 - Representação dos componentes em memória

Os pontos centrais dos elementos (propriedades “center-point-x” e “center-point-y”) são usados para determinar quais componentes fazem parte de qual tela. Caso um componente fique “para fora” da tela ele ainda pode ser considerado caso seu ponto central fique dentro do perímetro de alguma tela. Componentes que não estejam no perímetro de nenhuma tela são ignorados. Componentes que sobrepõem outros componentes também são ignorados.

4.3 ETAPA 2: GERAÇÃO DE CÓDIGOS DE WIREFRAMES

Os componentes são identificados pelo App Inventor em um arquivo ".aia" por meio de um arquivo contendo uma estrutura em JSON. Um estudo preliminar sobre a estrutura que compõem um arquivo ".aia" foi apresentado no capítulo 2.2.1. Para cada componente identificado pela etapa de detecção, o sistema possui uma versão compatível com o JSON contido dentro do arquivo de importação do App Inventor. Alguns componentes da biblioteca do Penpot são convertidos para mais de um componente no App Inventor (Figura 2.5), como p.ex. o componente "Notification" que possui três botões e dois textos (Figura 4.5).

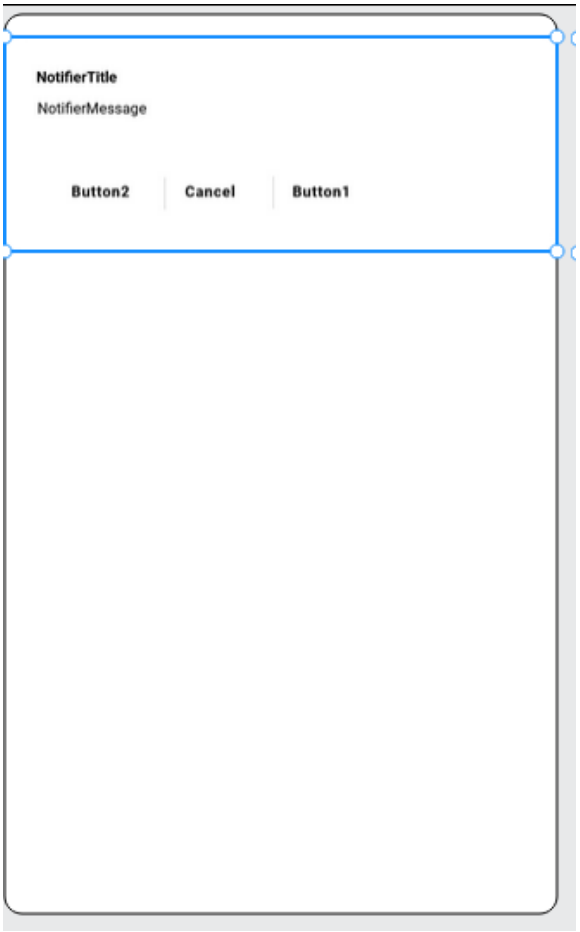


Figura 4.5 - Componente "*Notification*" no Penpot

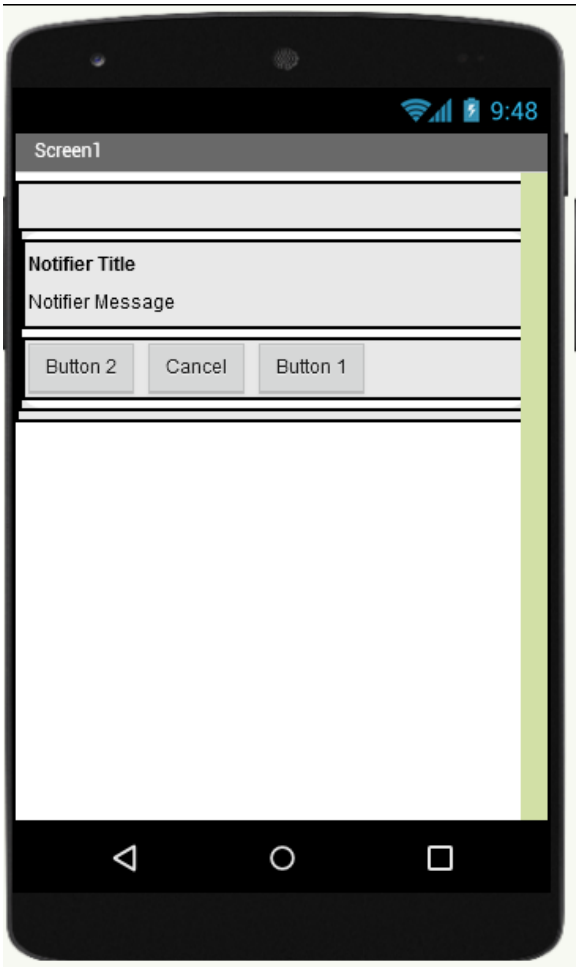


Figura 4.6 - Componente "*Notification*" no App Inventor gerado pelo Penpot2AIA

Apesar dos componentes possuírem uma representação exata de seu posicionamento, o App Inventor utiliza um sistema próprio de *layout* para posicionar

os elementos na tela. Os elementos são organizados de forma hierárquica utilizando-se de uma série de arranjos e componentes (Figura 4.7). O App Inventor utiliza os seguintes arranjos:

- Arranjo Horizontal;
- Arranjo Horizontal com *Scroll*;
- Arranjo em Tabela;
- Arranjo Vertical;
- Arranjo Vertical com *Scroll.*

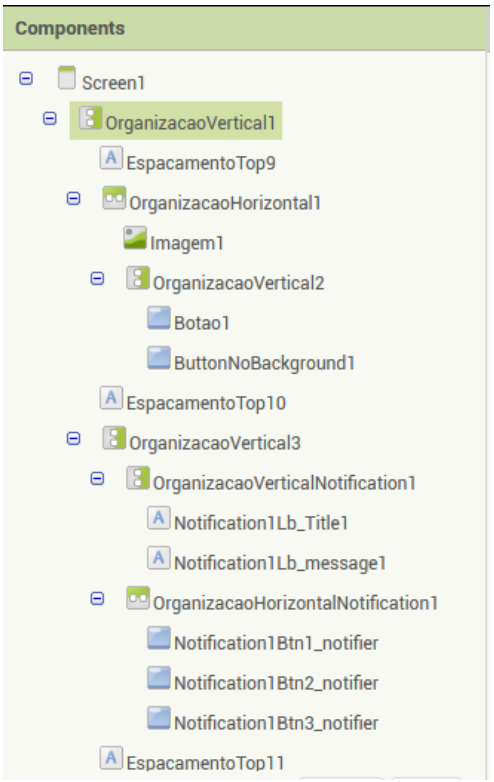


Figura 4.7 - Organização de componentes no App Inventor

Para este sistema foi utilizado somente o Arranjo Vertical (*VerticalArrengement*) e Arranjo Horizontal (*HorizontalArrengement*). Para criar a estrutura de arranjos do App Inventor que mais se aproxima do mapeamento identificado, foi utilizado um algoritmo recursivo desenvolvido por Baulé (2020) demonstrado na Figura 4.8.

```
funcao alinhar(componentes, orientacao)
    // cria uma lista para representar os componentes do alinhamento atual
    alinhamentoAtual = lista()
    // ordenar os componentes da lista de acordo com a orientação do alinhamento atual
    componentes = ordenarLista(componentes, orientacao)
    // enquanto a alista de componentes não estiver vazia
    para_cada componenteAtual em componentes faca
        componentes.remove(componenteAtual)
        // verifica se existem na lista componnetes alinhados com o atual no sentido contrário
        componentesAAnalisar = lista()
        novosComponentes = lista()
        componentesAAnalisar.adiciona(componenteAtual)
        //Para cada componente alinhado novo, verifique também se existem outros componentes alinhados com este
        para_cada componenteAnalisado em componentesAAnalisar faca
            componentesAAnalisar.remove(componenteAnalisado)
            para_cada componente em componentes faca
                // Se os componente estiver alinhado e não tiver sido verificado, adiciona e verifique
                se componenteAnalisado.estaAlinhado(componente, orientacao) entao
                    se componente nao em novosComponentes entao
                        componentesAAnalisar.adiciona(componente)
                        novosComponentes.adiciona(componente)
                    fim_se
                fim_se
            fim_para
        fim_para
        // se não estiver alinhado com outros componentes, adicione na lista, senão alinhe estes primeiro
        se novosComponentes.vazia() entao
            alinhamentoAtual.adiciona(componenteAtual)
        senao
            componentes = componentes - novosComponentes
            novosComponentes.adiciona(componenteAtual)
            alinhamentoAtual.adiciona(alinhar(novosComponentes, nao orientacao))
        fim_se
    fim_para

    retorna tupla(orientacao, alinhamentoAtual)
fim_funcao
```

Figura 4.8 - Pseudo-código de alinhamento de componentes no App Inventor

Foi desenvolvido ainda um algoritmo para inserir uma série de *labels* para atuarem como espaçadores entre componentes de forma a simular a propriedade *top* de cada componente. Estes espaçadores foram inseridos apenas no primeiro Arranjo Vertical gerado pelo sistema (Figura 4.9). Os demais arquivos que fazem parte da estrutura do “.aia” (Figura 2.1), como os arquivos “.bky” e o arquivo “project.properties”, também foram gerados utilizando o algoritmo desenvolvido por Baulé (2020).

```
funcao AdicionaEspacamento(listaDecomponentes):
    componenteRaiz = listaDecomponentes[0]
    espacadores = lista()
    indice = 0
    posicaoTopo = 0
    continuaProximo = Falso

    para_cada componente em componenteRaiz.componentes faca
        // caso tenha inserido um espacador na iteracao anterior entao adiciona um índice e pula a iteracao
        // atual caso contrário o algoritmo avaliará o mesmo caso novamente e entrar em loop infinito
        se continuaProximo entao
            continuaProximo = Falso
            continua
        fim_se

        // caso o componente atual seja um arranjo, captura o valor do top do componente mais acima
        // e a posição do componente mais abaixo somado a altura do componente mais abaixo (capturado em um passo anterior)
        se componente.tipo = "arranjoHorizontal" ou componente.tipo = "arranjoVertical" entao
            menorPosicaoTopo = componente.menorPosicaoTopo
            maiorPosicaoFundo = componente.maiorPosicaoFundo
        senao
            // caso não seja um arranjo, captura a posição top do componente e a posição top somado à sua altura (capturado em
            // um passo anterior)
            menorPosicaoTopo = componente.posicaoTopo
            maiorPosicaoFundo = componente.Fundo
        fim_se

        // estancia o espaçador, definido pela diferença da posição Y "atual" da análise e a posição top do
        // componente/arranjo a ser avaliado
        espacador = arredonda(valorAbsoluto(posicaoTopo - menorPosicaoTopo))

        // caso um espaçador tenha sido definido, estancia um label que servirá de espaçador
        se espacador > 0 entao
            espacadorcomponente = novo Label
            espacadorcomponente.nome = "EspacamentoposicaoTopo"
            espacadorcomponente.altura = maiorEntre(espacador, 1)
            espacadorcomponente.largura = 1
            espacadorcomponente.texto = " "
            componenteRaiz.componentes.insereNoIndice(indice, espacadorcomponente)

            // como na linha 40 foi adicionado um novo componente a lista que esta sendo analisada é necessário
            // ignorar a análise do próximo componente, caso contrário, o algoritmo analisará o mesmo componente novamente
            continuaProximo = Verdadeiro
        fim_se

        // atualiza a posição da análise
        posicaoTopo = maiorPosicaoFundo
    fim_para
fim_funcao
```

Figura 4.9 - Pseudo-código de adição de espaçador entre componentes

Os arquivos gerados são organizados em um arquivo “.zip” seguindo a estrutura interna de um arquivo “.aia” para ser importado diretamente pelo App Inventor (Figura 4.10).

```
funcao salvarAia(pastaProjeto, projeto)
    // gera o arquivo ".aia" vazio
    salveNaPasta = concatena(pastaProjeto, projeto.NomeApp, ".aia")

    // compacta o arquivo ".aia", transformando-a em uma pasta
    // em seguida "entra" na pasta compactada
    dentro compactaArquivo(salveNaPasta) como meuZip faca
        // salva todos os estaticos (fontes, imagens, etc..) na pasta "static/assets"
        para_cada estatico em projeto.estaticos faca
            meuZip.escreva(concatena("static/assets/", estatico), estatico)
        fim_para

        // cria um arquivo de texto chamado "project.properties" e salva as propriedades dentro dele
        meuZip.escreva("youngandroidproject/project.properties", projeto.gerarPropriedades())

        // escreve as telas do sistema
        para_cada tela em projeto.telas faca
            // gera o arquivo JSON compatível com o App Inventor
            meuZip.escreva(concatena("src/appinventor/penpot2aia/", projeto.NomeApp, "/", tela.nome, ".scm"), tela.gerarAIAJson())
            // escreve um arquivo ".bky" vazio
            meuZip.escreva(concatena("src/appinventor/penpot2aia/", projeto.NomeApp, "/", tela.nome, ".bky"), "")
        fim_para
    fim_dentro

    // retona o endereço do arquivo ".aia"
    retorna salveNaPasta
fim_funcao
```

Figura 4.10 - Pseudo-código para gerar o arquivo “.aia” desenvolvido por Baulé (2020)

4.4 FERRAMENTA WEB

A partir do modelo criado foi desenvolvida uma ferramenta web, Penpot2AIA. A ferramenta possui:

- Uma tela inicial onde o serviço de conversão pode ser acessado;
- Uma tela com informações gerais do projeto;
- Uma tela com termos de serviço e política de privacidade.

Todas as telas e informações expostas ao usuário são disponibilizadas nos idiomas inglês e português. A troca de idioma pode ser acessada no cabeçalho da ferramenta.

A ferramenta permite que o usuário adicione múltiplos arquivos de exportação “.zip” de um projeto Penpot desenvolvido com a biblioteca “Penpot2AIA” e exportado usando a opção “Incluir os arquivos das bibliotecas compartilhadas na biblioteca do arquivo” ou “Tratar os arquivos das bibliotecas como objetos básicos”.

4.4.1 Arquitetura Penpot2AIA

A ferramenta Penpot2AIA não possui integração com nenhuma ferramenta externa. Os componentes da ferramenta Penpot2AIA são:

- *Container Web*: Módulo de apresentação e módulo de detecção;
- *Browser*: Navegador do usuário onde a interface gráfica é exibida, onde os

arquivos de importação são enviados e o arquivo de exportação é retornado ao usuário.

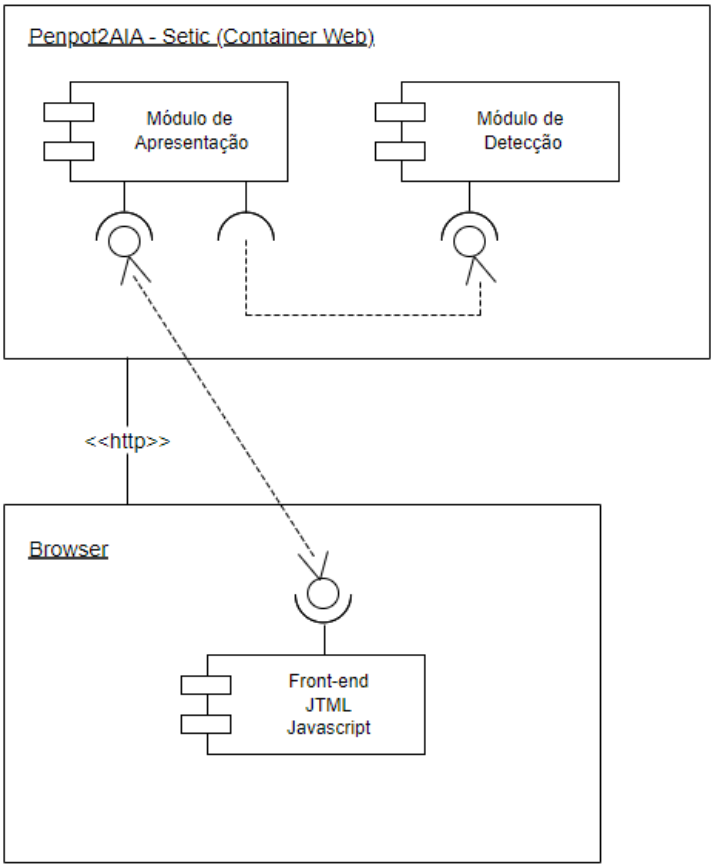


Figura 4.11 - Diagrama de módulos do Penpot2AIA

O módulo *Container Web* foi desenvolvido em Python, em conjunto do *framework* Flask, e é responsável pela apresentação das páginas, função de detecção e *download* do arquivo de exportação. A camada de *front-end* foi desenvolvida utilizando HTML5, CSS3 e Javascript. Foi usado ainda a biblioteca de javascript jQuery para auxiliar nas features de modal e carrossel (presentes nos tutoriais) e o *framework* de CSS Bootstrap para a implementação da *feature* de responsividade.

A ferramenta está disponível online: http://apps.computacaonaescola.ufsc.br/penpot2aia/. O código-fonte está disponível em: https://codigos.ufsc.br/gqs/penpot2aia.

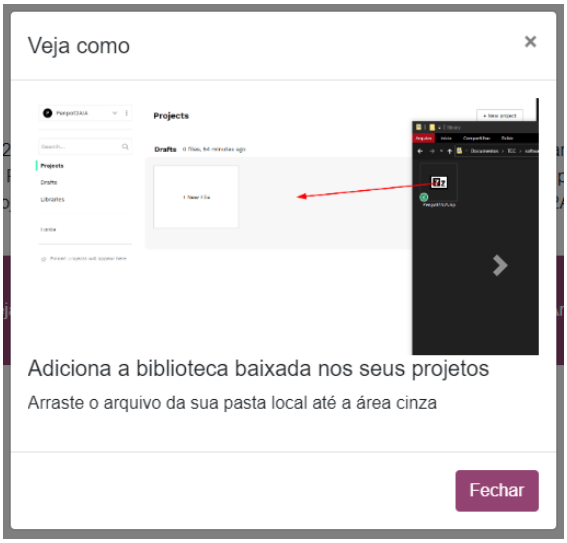
A ferramenta executa as etapas 1 e 2 descritas nos capítulos 4.2 e 4.3 para gerar um único arquivo de importação de App Inventor (arquivo “.aia”)

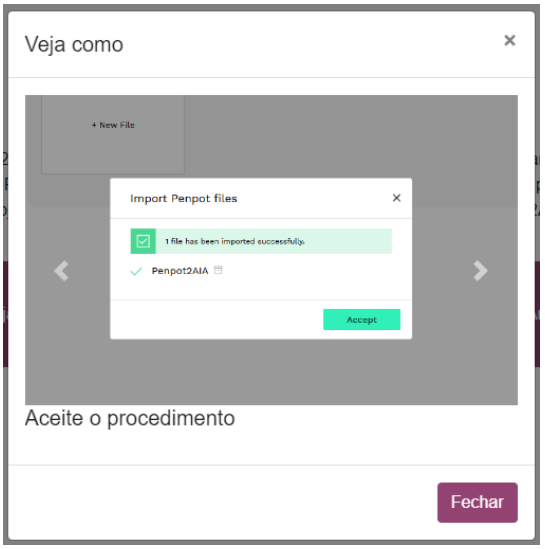
correspondente. O arquivo gerado pelo Penpot2AIA pode ser importado manualmente pelo usuário para a continuidade do desenvolvimento do aplicativo (Figura 4.1).

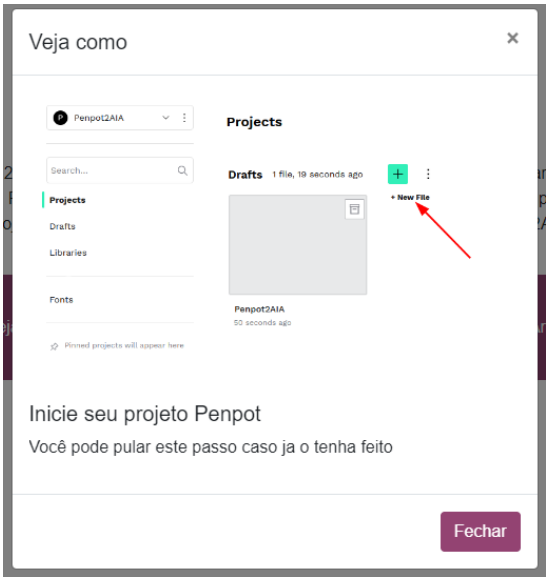
Ao acessar a ferramenta é apresentada uma série de passos que representam o *workflow* ao usuário (Figura 4.12).

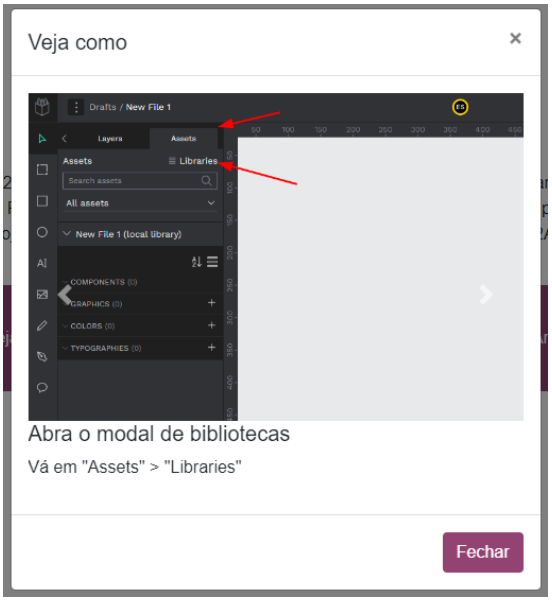
Figura 4.12 - Tela da ferramenta Penpot2AIA

Executando passo 1 é possível obter a biblioteca Penpot com os componentes aceitos pela ferramenta. O passo 2 mostra para o usuário como adicionar a biblioteca baixada no projeto Penpot por meio de um modal (Figura 4.13).









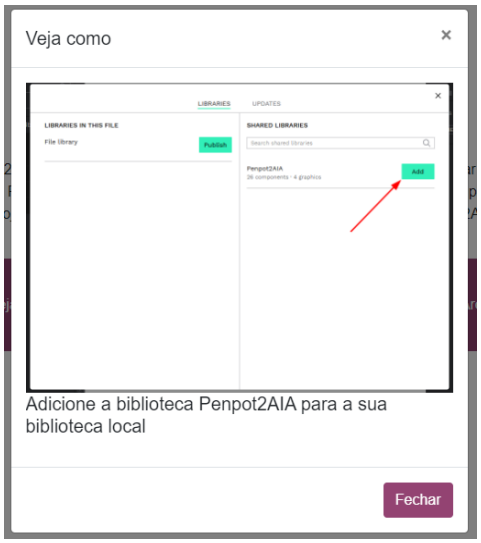


Figura 4.13 - Tutorial sobre a inclusão da biblioteca no projeto Penpot

O Passo 3 explica para o usuário como exportar corretamente o projeto também utilizando uma série de imagens em um modal (Figura 4.14).

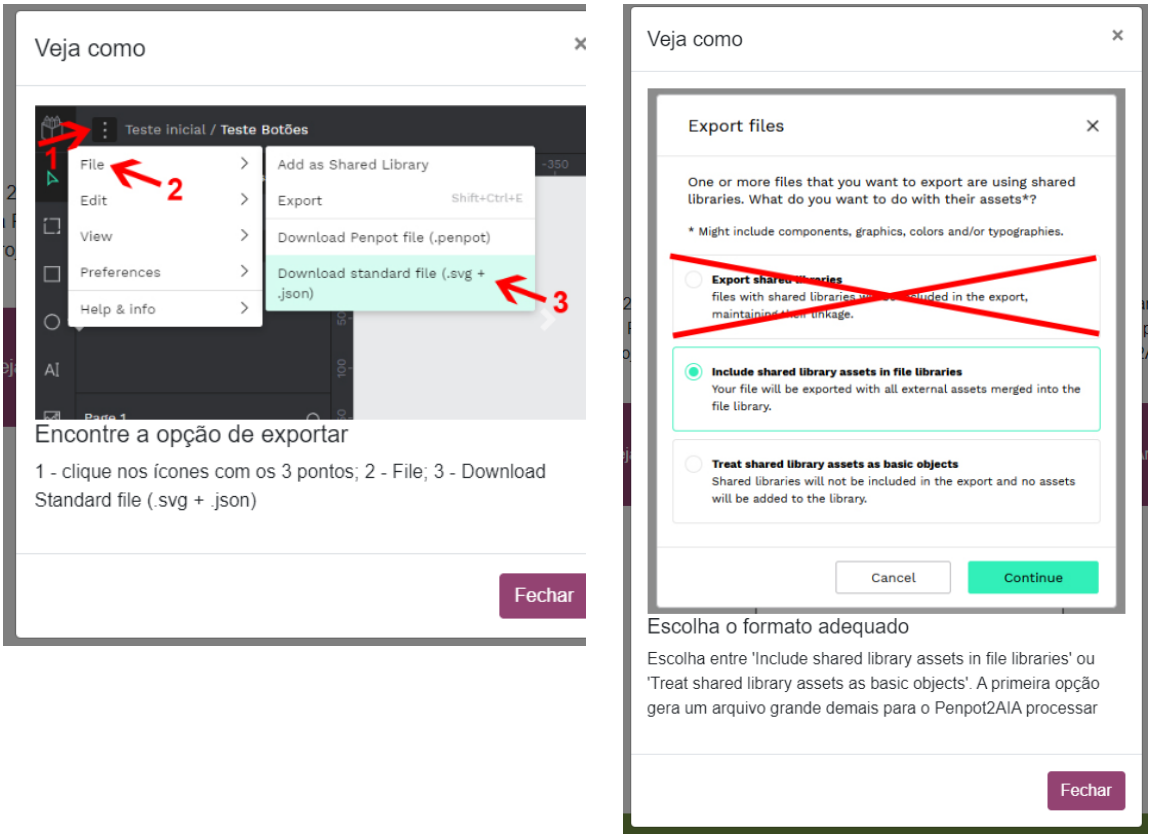


Figura 4.14 - Tutorial sobre como exportar projeto Penpot de acordo com o esperado pelo Penpot2AIA

No passo 4 é enviado o arquivo de exportação do Penpot para o processamento. Após o envio do projeto, a ferramenta disponibiliza o arquivo “.aia” correspondente para *download* utilizando *stream*, não sendo necessário mais nenhuma ação do usuário em caso de sucesso. O passo 5 exibe um passo a passo mostrando ao usuário como importar o arquivo “.aia” disponibilizado pela ferramenta no App Inventor.

5. AVALIAÇÃO DA FERRAMENTA

Neste capítulo é apresentada uma avaliação da ferramenta desenvolvida comparando telas criadas em projetos Penpot com as telas de *wireframes* App Inventor criadas automaticamente pela ferramenta.

5.1 DEFINIÇÃO DA AVALIAÇÃO

Com o objetivo de avaliar a qualidade da conversão do projeto Penpot para o *wireframe* App Inventor é realizado um estudo de caso, testando a conversão com 3 projetos diferentes de Penpot, que foram inspirados em aplicativos reais desenvolvidos usando o App Inventor, conforme apresentado nas Tabela 5.1- 5.3.

Xô Dengue

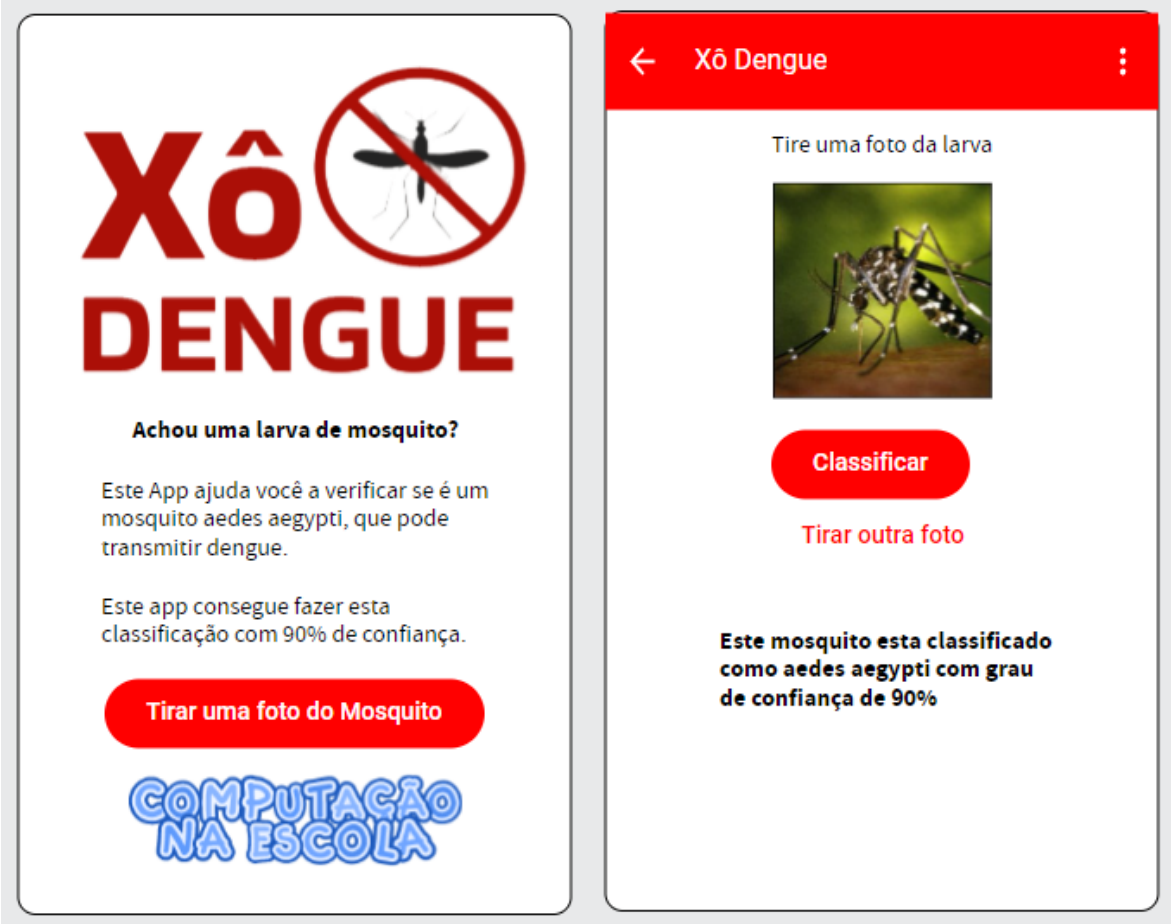


Tabela 5.1 - Design visual do projeto “Xô Dengue” criado com o auxílio do da biblioteca Penpot2AIA

Yú


Tabela 5.2 - Design visual do projeto “Yú” criado com o auxílio da biblioteca Penpot2AIA

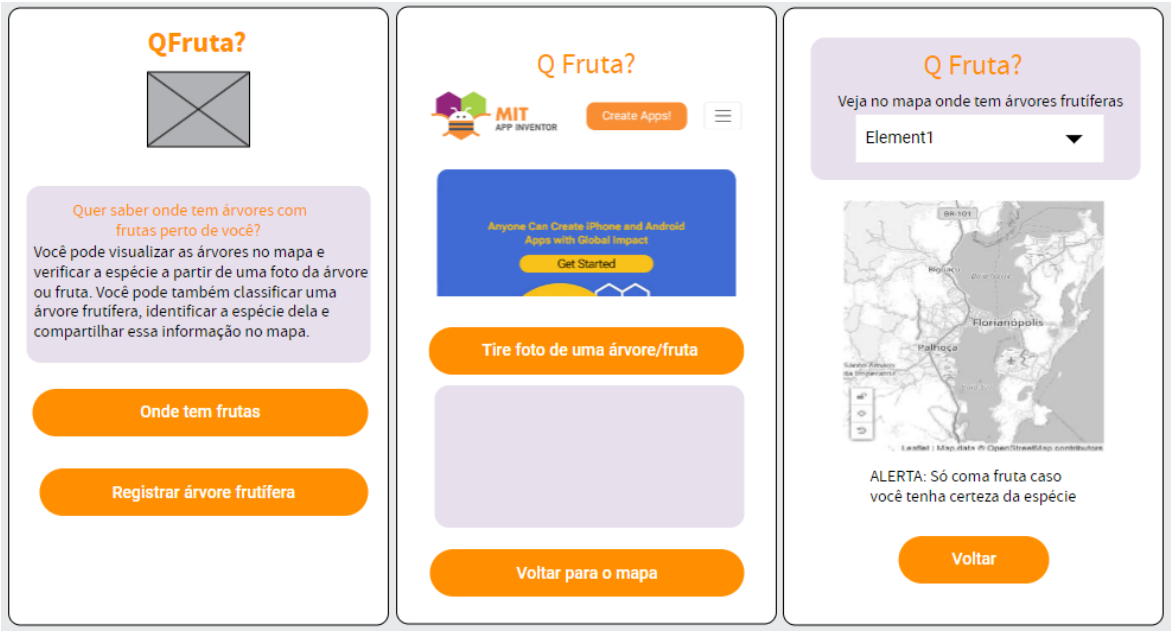
QFruta?


Tabela 5.3 - Design visual do projeto “Yú” criado com o auxílio do da biblioteca Penpot2AIA

5.2 EXECUÇÃO E ANÁLISE DA AVALIAÇÃO

Utilizando a ferramenta desenvolvida foram convertidos os projetos Penpot dos 3 apps para *wireframes* App Inventor. Os resultados são apresentados na Tabela 5.4- 5.6.

Xô Dengue	
	

Tabela 5.4 - Projeto “Xô Dengue” que foi desenvolvido usando a biblioteca Penpot2AIA e importado no App Inventor

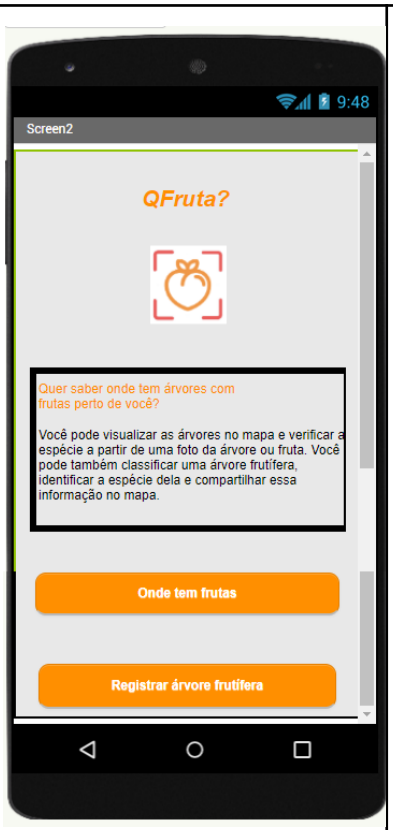
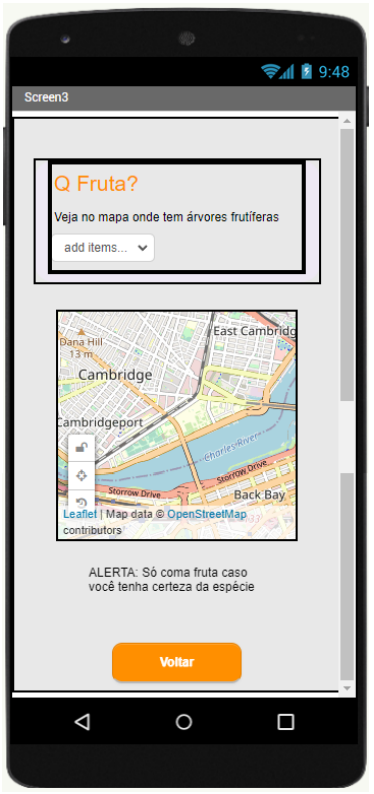
QFruta?		
		

Tabela 5.5 - Projeto “QFruta” que foi desenvolvido usando a biblioteca Penpot2AIA e importado no App Inventor

Yú


Tabela 5.6 - Projeto “Yú” que foi desenvolvido usando a biblioteca Penpot2AIA e importado no App Inventor

A ferramenta foi executada utilizando um servidor local e o tempo de execução para cada um dos projetos é apresentado na Tabela 5.7. Não houve erros na exportação do projeto e a instalação dos aplicativos exportados pelo App Inventor no dispositivo móvel ocorreu normalmente.

Projeto Penpot	Tempo de execução
Xô Dengue	340ms
Yú	358ms
QFruta? (tela 3)	350ms

Tabela 5.7 - Tempo de execução médio de conversão para cada projeto

Todos os tempos de execução < 360ms ficaram abaixo do tempo máximo especificado nos requisitos não funcionais.

5.3 RESULTADO

Todos os projetos Penpot foram exportados para o arquivo “.aia” e importado no App Inventor. A partir dos projetos importados no App Inventor foram gerados os arquivos “.apk” que foram instalados em um *smartphone* modelo Motorola Moto X4 com Android versão 9. Um evento de transição de tela foi inserido em um botão de cada tela.

Cada tela projetada foi comparada com o resultado final. Nenhuma configuração, como p. ex url de *webviewer*, posição inicial do Mapa, etc, foi acrescentada.

Projeto 1: App XôDengue

A Tabela 5.8 apresenta a comparação de cada tela do interface criado no Penpot e a versão do aplicativo gerada pelo App Inventor em formato de .apk.

Tela	Projeto Penpot	Apk gerado pelo App Inventor
1		

2		

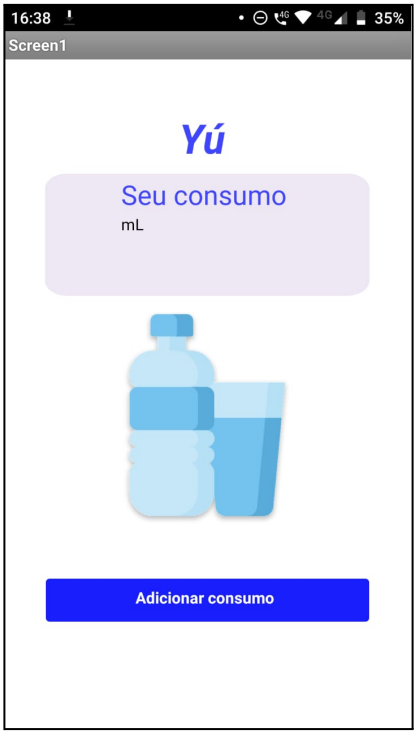
Tabela 5.8 - Comparativos entre projeto Penpot e aplicativo Android do App “Xô Dengue”

Na tela 1, o único ponto divergente do projeto idealizado é a falta de linha em branco. Isto ocorre pois o Penpot2AIA tem suporte a quebra de linha mas não identifica linhas em branco.

Na tela 2, o texto informando a classificação aparece com a largura automática. Isto ocorreu pois textos formatados com itálico, negrito ou com fonte maior que 14 *pixels* ocupam mais espaço do que a caixa que os circunda. Para remediar esta questão, foi inserida uma largura automática para textos com estas características.

Projeto 2: Yú

A Tabela 5.9 apresenta a comparação de cada tela do app “Yú” desenvolvido no Penpot e a versão do aplicativo gerada pelo App Inventor em formato de “.apk”.

Tela	Projeto Penpot	Apk gerado pelo App Inventor
1		
2		

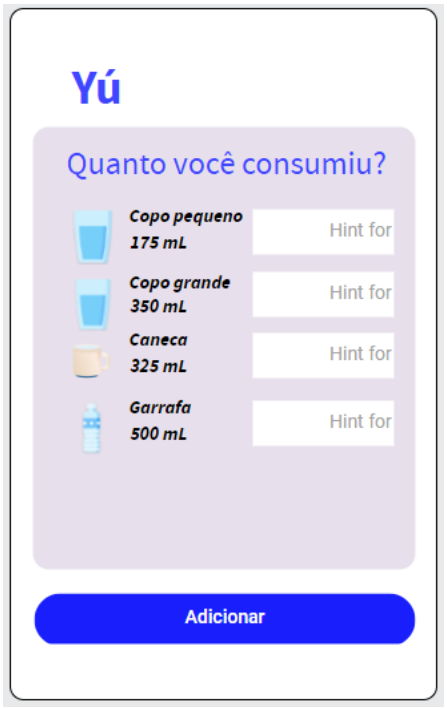
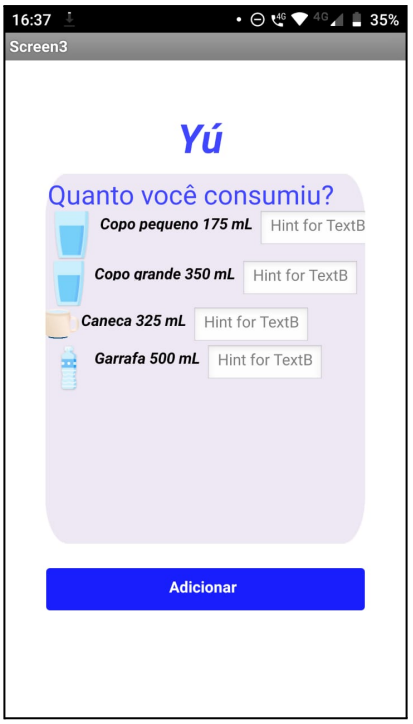
3		

Tabela 5.9 - Comparativos entre projeto Penpot e aplicativo Android do App “Yú”

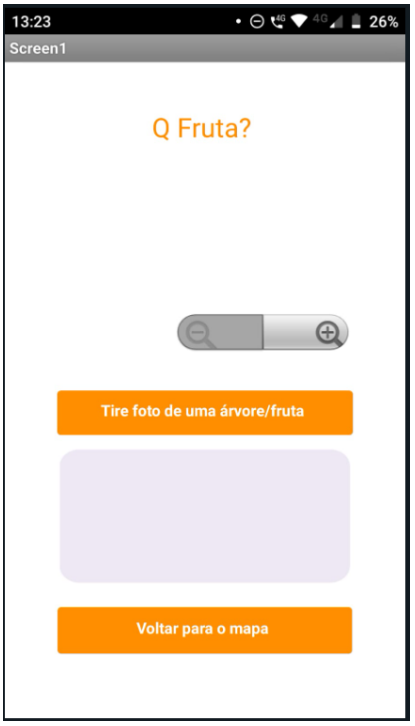
Na tela 1 o projeto apresenta um certo nível de fidelidade, somente variando no posicionamento de alguns *labels*.

A tela 2 apresenta a maior distorção entre os projetos. O Sistema identificou o título “Seu histórico” e os *sliders* como sendo partes do mesmo arranjo. Isto ocorreu pois a largura do título se assemelha a largura dos *sliders*. Ta tela 3 isto não ocorreu pois o título possui uma largura maior e portanto foi considerado como parte de um arranjo separado dos demais componentes.

Na tela 3 as telas se assemelham ao que foi idealizado durante o projeto. Os dois *labels* “Caneca 325mL” e “Garrafa 500mL” possuem uma largura menor do que os labels “Copo pequeno 175mL” e “Copo grande 350mL” trazendo suas respectivas caixas de textos fora do alinhamento. Caso a largura dos *labels* tivesse sido configurada com o mesmo valor esse comportamento não ocorreria.

Projeto 3: QFruta?

A Tabela 5.10 apresenta a comparação de cada tela do *app* “QFruta?” desenvolvido no Penpot e a versão do aplicativo gerada pelo App Inventor em formato de “.apk”.

Tela	Projeto Penpot	Apk gerado pelo App Inventor
1		
2		

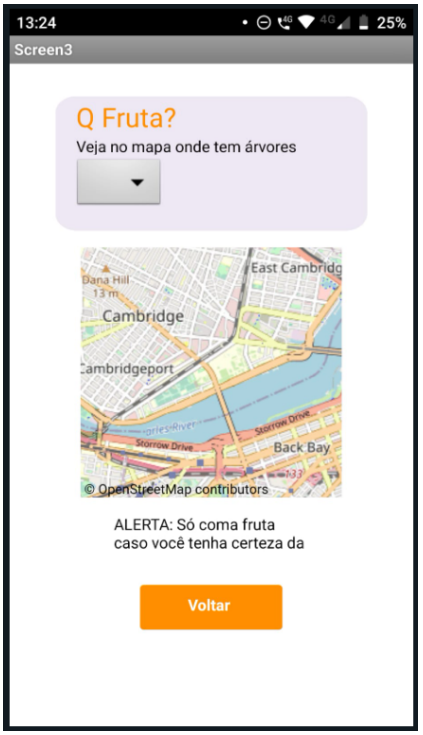
3		

Tabela 5.10 - Comparativos entre projeto Penpot e aplicativo Android do App “QFruta?”

As 3 telas não apresentam distorções, embora o problema do componente “*Card*” não centralizar os elementos corretamente tenha ficado evidente nesse aplicativo.

5.4 DISCUSSÃO

De forma geral observa-se que a ferramenta Penpot2AIA consegue converter projetos criados no Penpot com uma fidelidade aceitável para *wireframes* no App Inventor. Essa conversão foi feita com auxílio da biblioteca criada no Penpot e possível para uma variedade considerável dos principais elementos de design de interface tipicamente utilizados em telas de aplicativos móveis. Os elementos da biblioteca criada no Penpot estão também em conformidade com o guia de estilo Material Design 3 (Google, 2023), auxiliando na qualidade do design visual dos projetos sendo criados. Não houve erros na exportação dos projetos e as instalações ocorreram normalmente. Observa-se também que o tempo da execução da conversão (< 360ms) está bem abaixo do tempo especificado.

Porém, sendo uma pesquisa inicial observou-se também durante a avaliação algumas questões conforme listado na Tabela 5.11, incluindo também sugestões de melhorias futuras.

O sistema poderia comportar linhas em branco em textos.
O elemento “*Card*” não centraliza os elementos corretamente.
Os espaçamentos são inseridos apenas no primeiro arranjo vertical de cada tela. Seria bom se pudesse ser inserido em todos os arranjos da tela, inclusive nos arranjos horizontais para posicionar o elemento lateralmente para produzir telas mais fiéis ao idealizado.
Implementar mais componentes do *Material Design.* No momento, apenas os elementos mais usados e fáceis de serem transportados para o App Inventor são considerados.
Melhorar o arrendondamento dos botões no aplicativo final. Isso pode ser resolvido usando-se uma imagem de fundo, em vez de usar a propriedade de cor de fundo em cunjunto com a propriedade de formato do botão.
Alguns componentes na biblioteca Penpot2AIA são imagens, como p. ex. “App Bar” e “Card”, e não permitem configuração.

Tabela 5.11 - Oportunidades de melhoria da ferramenta Penpot2AIA

Ameaças à validade. Como qualquer trabalho empírico, a presente avaliação preliminar também apresenta ameaças à validade dos resultados apresentados. Uma das ameaças é o tamanho pequeno de somente três projetos de design de interface com poucas telas. E como a ferramenta Penpot2AIA requer o uso da biblioteca específica de componentes no Penpot, estes projetos foram criados de forma artificial especificamente para a avaliação. Desta forma devem ser realizadas novas avaliações no futuro utilizando projetos reais e mais completos para assegurar uma avaliação mais abrangente.

6. CONCLUSÃO

Neste trabalho o objetivo principal foi desenvolver um modelo de geração automática de *wireframes* no App Inventor a partir de arquivos de exportação do Penpot. Para atingir este objetivo foi realizada inicialmente uma análise da fundamentação teórica sobre desenvolvimento de *apps* com App Inventor e design de interface de usuários com Penpot (O1). Em seguida foi realizado o levantamento do estado da arte em relação a ferramentas existentes de ferramentas de conversão de projetos UI no Penpot para App Inventor (O2). Esta etapa constatou a ausência desse tipo de ferramenta para o App Inventor como também além da ausência de artigos envolvendo Penpot. Foram também analisados artigos e *softwares* que tem por objetivo conversão de formatos de arquivos de exportação para importação entre sistemas. Como principal resultado do trabalho foi desenvolvida uma ferramenta para converter automaticamente projetos de design de interface de *apps* criados com Penpot para *wireframes* no App Inventor (O3). Em uma avaliação preliminar a ferramenta demonstrou um tempo de execução bem abaixo do especificado e uma fidelidade aceitável do *wireframe* com o projeto original no Penpot (O4). O resultado da pesquisa é disponibilizado como ferramenta web e pode ser acessado online: http://apps.computacaonaescola.ufsc.br/penpot2aia/.

Assim foi criada uma contribuição tecnológica inovadora facilitando o uso da ferramenta Penpot no processo de desenvolvimento de aplicativos com App Inventor. Espera-se que essa ferramenta também contribua positivamente ao ensino de design de interface como parte do ensino de computação, facilitando e agilizando o processo de desenvolvimento.

Para trabalhos futuros, sugere-se a evolução da ferramenta como a detecção de elementos sem o uso de uma biblioteca auxiliar para abordar os componentes de diferentes criadores de interface no Penpot. Além do refino da geração do *wireframe* em si para que a interface gerada ao final seja mais fiel ao idealizado na etapa de desenvolvimento da interface no Penpot.

REFERÊNCIAS

AI2 SideBar, 2023. Disponível em: https://ullisroboterseite.de/android-AI2-SideBar-en.html. Acessado em: junho de 2023

Android Studio. CONVERTER designs em código no Android Studio, 2023. Disponível em: Https://developer.android.com/jetpack/compose/tooling/relay/convert-designs-android-studio?hl=pt-br. Acesso em: 26 jun. 2023

AppInventor, 2023. Disponível em: http://www.appinventor.org/ Acesso em: Dex 2023

BAULÉ, Daniel de Souza. Desenvolvimento de um Modelo de Geração Automática de Wireframes no App Inventor a partir de Sketches usando Deep Learning. Trabalho e Conclusão de Curso (Bacharelado em Sistemas de Informação) - Universidade Federal de Santa Catarina, Florianópolis, 2020.

Export Kit. CONVERT XD to Android Studio XML and Java. Disponível em: https://exportkit.com/plugin/environments/xml/convert-xd-to-android-studio. Acesso em: 26 jun. 2023.

FIGMA. 2023. Disponível em: https://www.figma.com/. Acesso em: 26 jun. 2023.

GOOGLE. Material Design. Disponível em: <https://m3.material.io/>. Acesso em: 05 dez. 2023.

GROVER, S.; PEA, R. D. Computational thinking in K–12: A review of the state of the field. Educational Researcher, 42 (1), 38-43, 2013.

LARMAN, C.; BASILI, V. R. Iterative and Incremental Development: A Brief History. Computer, 36(6), pp. 47-56, 2003.

LYE, S. Y.; KOH, J. H. L. Review on teaching and learning of computational thinking through programming: What is next for K-12?. Computers in Human Behavior, 41(1), pp. 51–61, 2014.

MIT. App Inventor, 2023. Disponível em: http://appinventor.mit.edu Acessado em: abril 2023

Pacheco, J., Garbatov, S., & Goulao, M. Improving Collaboration Efficiency Between UX/UI Designers and Developers in a Low-Code Platform. In Proc. of ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion, IEEE, 2021.

PATTON, E.; TISSENBAUM, M.; Harunani, F. (2019). MIT App Inventor: Objectives, Design, and Development. Disponível em: <https://link.springer.com/content/pdf/10.1007%2F978-981-13-6528-7_3.pdf>. Acesso em: março 2023.

PENPOT. 2023. Disponível em: https://penpot.app/. Acesso em: junho de 2023

PETERSEN, K.; FELDT, R.; MUJTABA, S.; MATTSSON, M. Systematic Mapping Studies in Software Engineering. In: Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering, Bari, Italy, 2008, p. 68-77.

SCHLATTER, Tania & LEVINSON, Deborah. Visual Usability: Principles and Practices for Designing Digital Applications. 2013.

APÊNDICE A

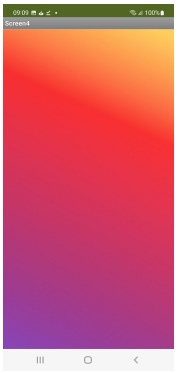
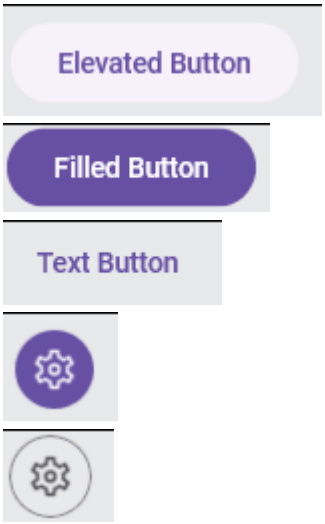
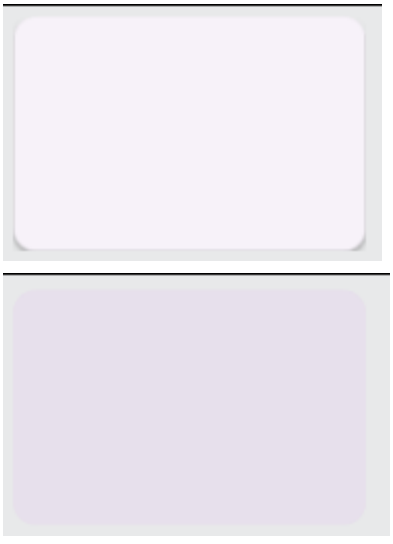
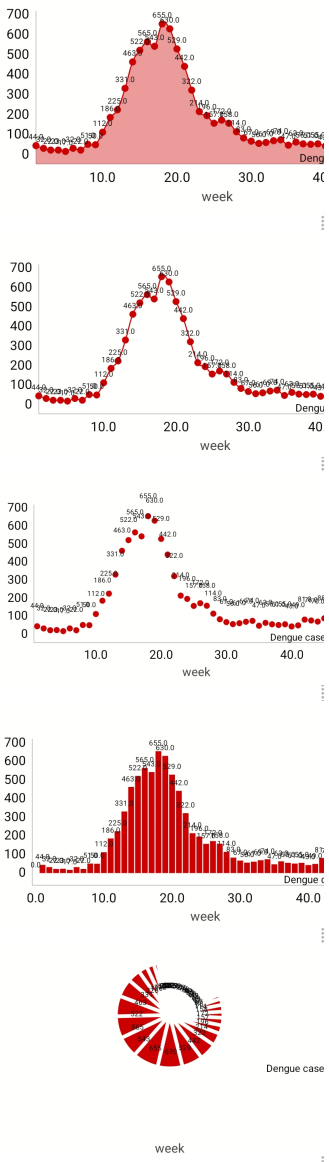
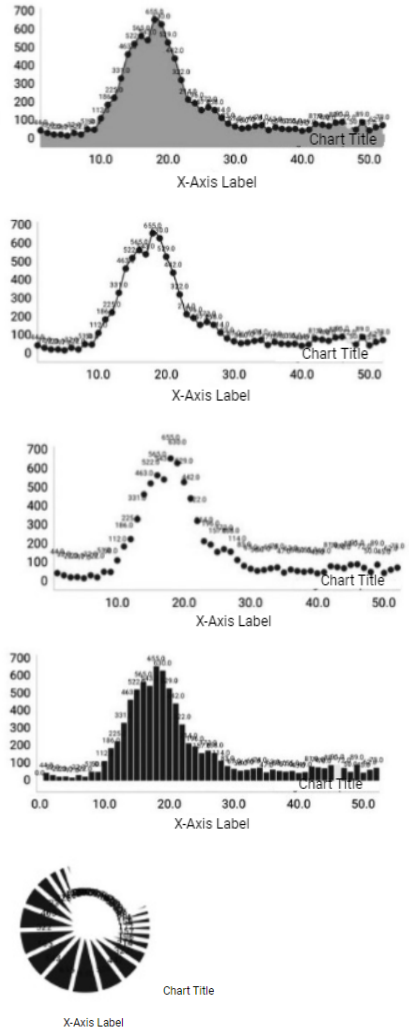
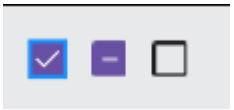
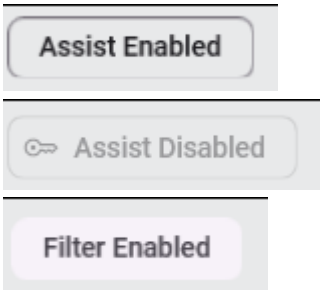
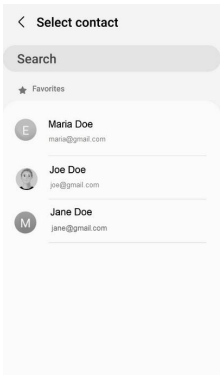
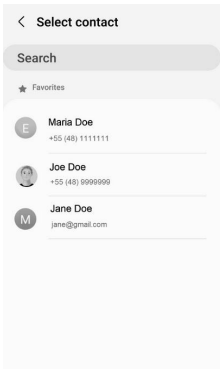
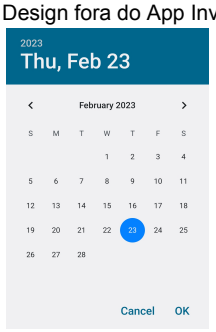
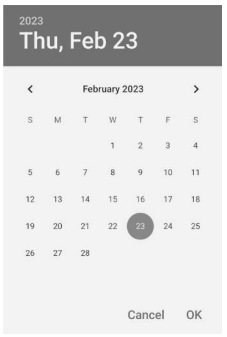
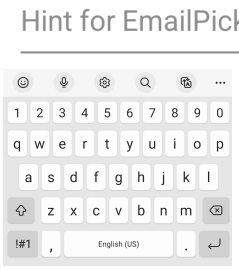
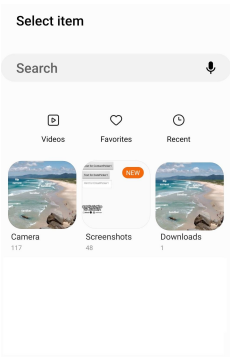
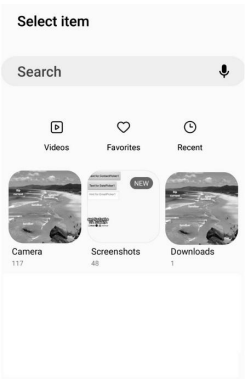
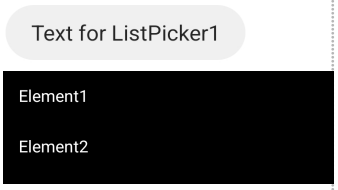
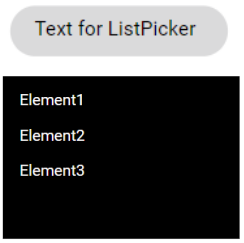
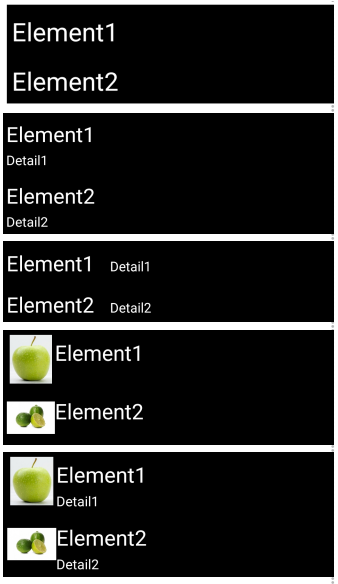
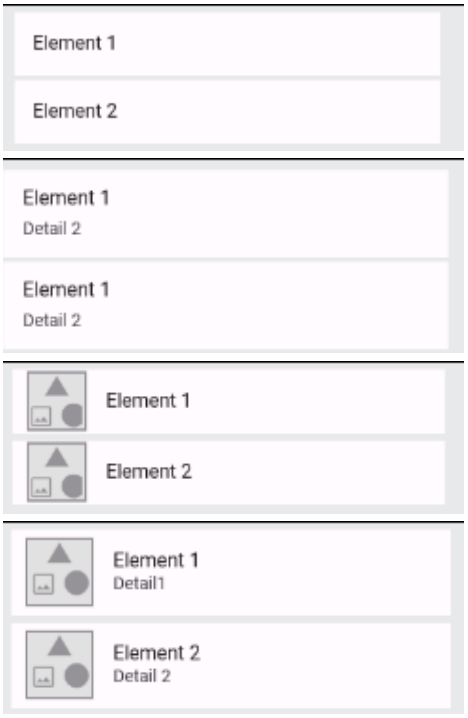
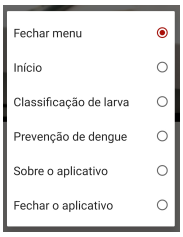
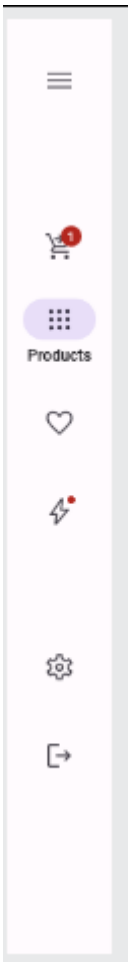
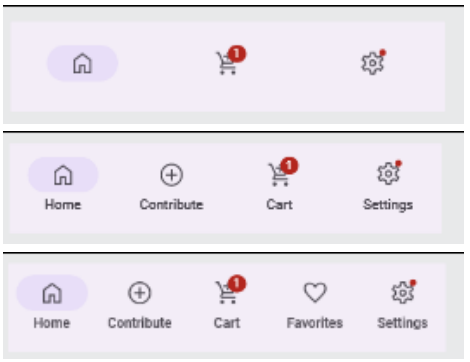
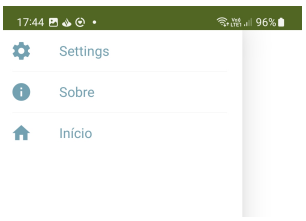
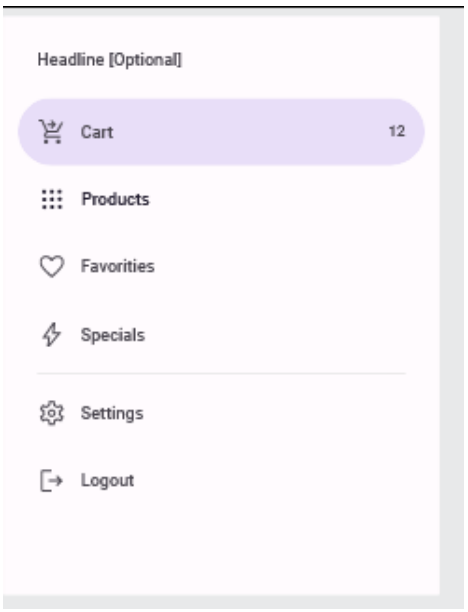
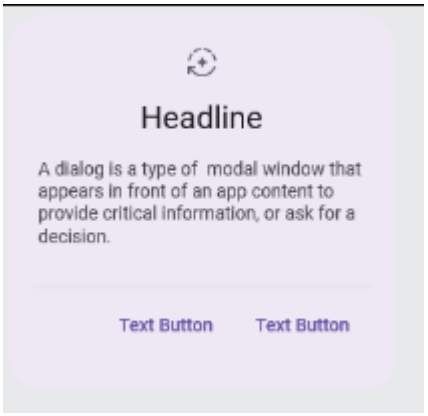
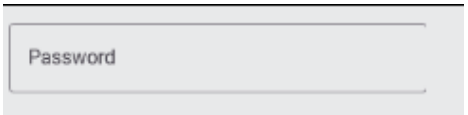
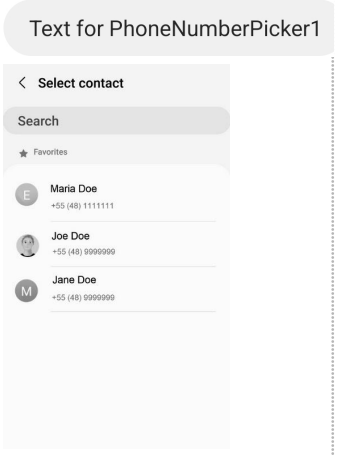
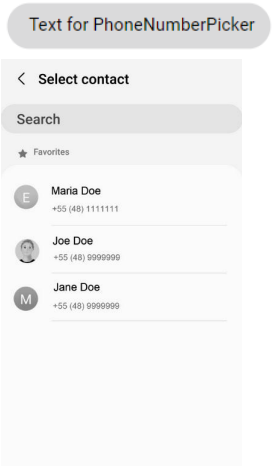
Elemento de UI visível no App Inventor	**Descrição**	**Exemplo**	***Design Kit* UI Penpot / Componente *Material Design* 3 (design adaptado para o Material *Design* v3 ou *design* kit Penpot)**
BackgroundImage	Uma imagem que aparece no fundo da tela		--
App Inventor Button	Um componente com a habilidade de detectar cliques. No App Inventor: padrão, redondo e retangular. No Penpot: Elevado: preenchido; Apenas texto; Ícone preenchido; Ícone contornado.	Text for Button1 Text for Button2 Text for Button3 (pode também ser uma imagem, texto, etc)	
Cards			

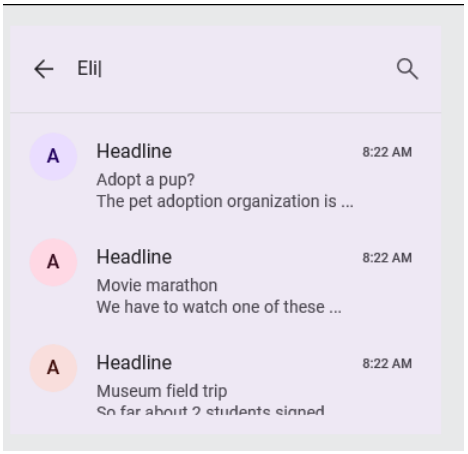
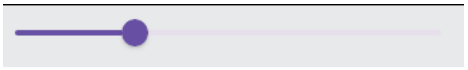
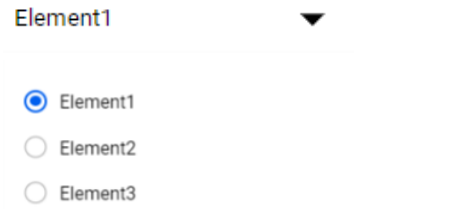
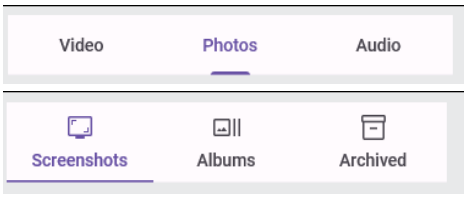
Chart	Componente que desenha um gráfico de linha, scatter, barra e pizza;		
CheckBox	Um componente que lança um evento quando o clicado	Text for CheckBox1 Text for CheckBox1	
Chips			
ContactPicker	Um botão que, quando clicado, exibe uma lista de contatos para escolha.	Text for ContactPicker1 *Design* fora do App Inventor:	Text for ContactPicker

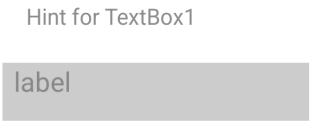
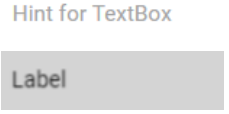
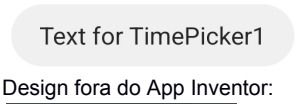
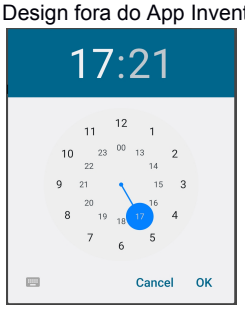
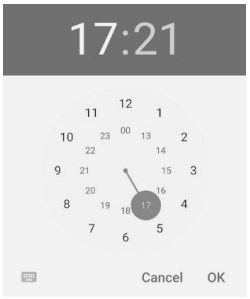
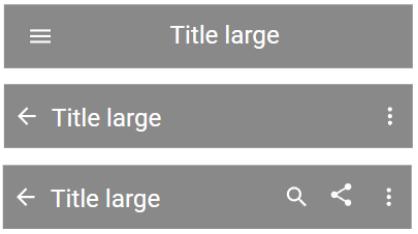
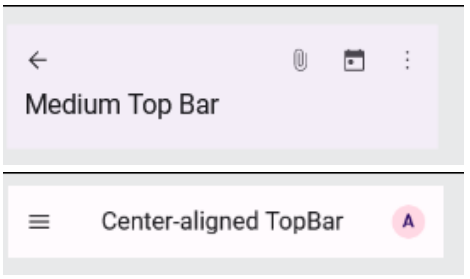
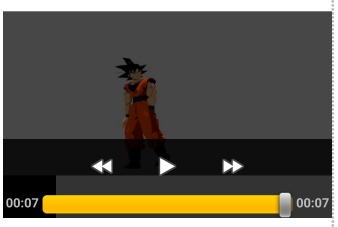
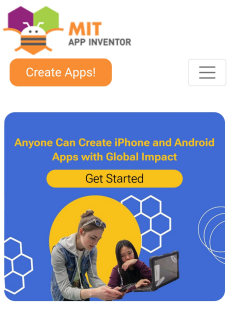
			
DatePicker	Um componente que, quando clicado, exibe uma caixa de diálogo pop-up que permite ao usuário selecionar a data.	Text for DatePicker1 Design fora do App Inventor: 	Text for DatePicker 
EmailPicker	Um tipo de caixa de texto na qual o usuário pode inserir o nome ou endereço de email de um contato e seu telefone será exibido em um menu contendo escolhas para autocompletar o texto de entrada.		
Image	Um componente para exibir uma imagem		
ImagePicker	Um botão de propósito específico. Quando o usuário clica em um seletor de imagem, a galeria de imagem do dispositivo é exibida a o usuário pode	Text for ImagePicker1 *Design* fora do App Inventor	Text for ImagePicker

	selecionar uma imagem..		
Label	Um componente para exibir um pedaço de texto, que é especificado na propriedade do elemento.	Text for Label1	Sem necessidade de componente
ListPicker	Um botão que quando clicado, exibe uma lista de textos para o usuário escolher.		
ListView	Um componente para exibir uma lista de elementos de texto e imagem		

Map	Um container bi-dimensional que carrega pedaços de um mapa ao fundo e permite múltiplos elementos marcadores para identificar pontos no mapa.		
Menu	Feito com Spinner		
Navigation bar	Um componente que exibe opções para alternar entre as diversas telas do sistema.		

Navigation drawer	Apenas com extensão		
Notifier/Dialog	Um componente que exibe diálogos de alerta, mensagens e alertas temporários.		
PasswordTextBox	Uma caixa de texto para inserir senhas.		
PhoneNumberPicker	Um botão que, quando clicado, exibe uma lista de contatos com seus números de telefone para escolher.		

Progress Indicator	Apenas com extensão		
Radio button	Feito com checkboxes		
Search			Search bar 
			Search view 
Slider	Uma barra de progresso que adiciona um botão de arrastar.		
Spinner->Menu	Um componente que exibe um pop-up com uma lista de elementos.	Element1 	
Switch	Um componente que lança um evento quando o usuário clica nele	Text for Switch1 	
Tabs	Um componente para exibir e ocultar áreas diferentes na mesma tela		

TextBox	Uma caixa de texto para o usuário preencher	 Este elemento não possui borda	
TimePicker	Um botão que, quando clicado, exibe uma caixa de diálogo popup que permite o usuário selecionar as horas	 Design fora do App Inventor: 	 
--	--	--	
Top app bar			
VideoPlayer	Um componente multimídia capaz de exibir vídeos.		
WebViewer	Um componente para visualização de páginas web.	Exibe um website 	--

APÊNDICE B

Desenvolvimento de um Modelo de Geração Automática de *Wireframes* no App Inventor a partir de Arquivos de Exportação do Penpot

Elizeu Santos Madeira, Christiane Gresse von Wangenheim, Jean C. R. Hauck, Ramon Mayor Martins

Dep.de Informática e Estatística - Universidade Federal de Santa Catarina (UFSC) - Florianópolis - SC - Brasil

elizeu.nadeira@grad.ufsc.br, c.wangenheim@ufsc.br, jean.hauck@ufsc.br, ramon.mayor@ifsc.edu.br

Abstract. *The present study aimed to facilitate the creation of interface designs for the development of functional apps. Today, as part of systems development, several interface prototypes are created using tools such as Penpot. However, Penpot still does not allow the creation of apps itself, only their interfaces, but it does allow to export projects in non-proprietary and free formats. To facilitate the conversion of interface designs to a functional application, a tool - Penpot2AIA- has been developed that automatically creates the code for an application in the App Inventor tool. We expect that in this way the tool facilitates the app develpment process.*
Resumo. O presente estudo tem por objetivo facilitar a criação de designs de interface para o desenvolvimento de apps funcionais. Hoje, como parte do desenvolvimento de sistemas, diversos protótipos de interfaces são criadas usando ferramentas como o Penpot, Contudo, Penpot ainda não permite a criação dos apps em si, apenas suas interfaces, mas permite exportar seus projetos em formatos não proprietários e livres. Para facilitar a conversão de designs de interface para aplicativos funcionais, uma ferramenta – Penpot2AIA - foi desenvolvida uma ferramenta que automaticamente cria os códigos para uma aplicação no App Inventor. Espera-se desta forma facilitar o processo de desenvolvimentos de apps.

1 Introdução

Conforme a sociedade avança e os meios digitais são inseridos no dia-a-dia dos cidadãos de forma gradual e permanente, é necessário preparar a sociedade para lidar com as novas profissões (ou mesmo com a modernização das profissões já existentes) já na vida estudantil (Grover; Pea, 2013). Por este motivo, a necessidade de introduzir o ensino de computação já na educação básica é importante (Lye; Koh; 2014). Neste contexto, uma alternativa para ensinar a computação é por meio do desenvolvimento de aplicativos móveis usando App Inventor (Patton *et al.*, 2019). O App Inventor (AppInventor, 2023), um ambiente baseado em blocos, proporciona a simplicidade no desenvolvimento de aplicações, incluindo o *design* de interface, até um aplicativo funcional para a plataforma Android.

Tipicamente o processo de desenvolvimento de um aplicativo inicia-se com uma *sketch,* que é um esboço feito geralmente com papel e caneta para uma expressão mais conceitual das funcionalidades e da interface do futuro aplicativo (Schlatter *et al*., 2013). A partir do *sketch* é gerado um *wireframe* geralmente usando uma ferramenta de *design* gráfica, como p.ex. Figma (Figma, 2023), aprimorando e complementando o processo de *design* de interface com o *design* visual. Após a etapa de desenvolvimento da interface, a próxima etapa é desenvolver o código do aplicativo. Isto pode ser feito, p.ex. usando o App Inventor.

Uma das alternativas gratuitas para o desenvolvimento de interfaces de aplicações móveis é a ferramenta online de *design* gráfico Penpot (Penpot, 2023). A ferramenta Penpot é *código aberto* e gratuito, fornecendo uma riqueza de recursos e facilidade de organização e operação dos recursos para desenvolver de forma facilitada e eficiente o desenvolvimento de interfaces de aplicações móveis. Ele também permite exportar seus projetos usando formatos de domínio público (JSON e SVG).

No entanto, estas ferramentas, mesmo gerando até código de HTML/CSS, não oferecem a possibilidade de gerar aplicativos funcionais, apenas interfaces. Assim, para criar um app a partir do *design* criado necessita-se a implementação manual em ambientes de programação, como p.ex. utilizando o App Inventor, uma plataforma web que permite criar aplicativos para sistema operacional Android desde a interface com o usuário até implementação das funcionalidades (MIT, 2023), criando assim um app funcional. O objetivo deste artigo é agilizar o desenvolvimento criando uma ferramenta para converter o projeto exportado do Penpot convertendo-o para o formato de importação de projetos do App Inventor (arquivo ".aia") possibilitando assim a importação do *design* criado no Penpot diretamente no App Inventor.

2 Trabalhos relacionados

Por meio de uma revisão sistemática, foram encontrados apenas alguns artigos em que um sistema de interface é convertido para um arquivo de importação de um outro sistema de desenvolvimento de aplicações.

Pacheco et al. (2021), desenvolve um modelo de geração de arquivos de interface para desenvolvimento de sistemas Outsystems a partir da ferramenta "Sketch". Embora os detalhes do desenvolvimento da ferramenta em si tivessem sido omitidos, as técnicas e as metodologias empregadas foram explicadas de forma satisfatória. Quanto aos dados de entrada, Pacheco et al. (2021) usa diretamente os dados exportados pela ferramenta Sketch no formato ".sketch" e exporta os elementos mais usados para arquivos ".oml", formato proprietário da plataforma Outsystems para criação de interfaces. Para avaliar a qualidade do modelo gerado, foi empregada uma métrica interna da da empresa para a qual o modelo foi desenvolvido e o resultado de 99.6% na habilidade de identificar e instanciar um componente corretamente e 80% de componentes que a ferramenta consegue detectar. Além disso, os resultados com base em seis projetos foram encaminhados a cinco especialistas front-end da Outsystems. Os especialistas compararam componente a componente entre os projetos reais e a recriação gerada por meio da ferramenta. Embora nem todos os projetos se beneficiem da geração de interface, nos projetos em que é possível aplicar houve um acréscimo entre 150% a 400% no número de telas que podem ser criadas

Baulé (2020), utiliza *sketches* desenvolvidas usando papel e caneta como método de entrada para geração de wireframes no App Inventor. As sketches são convertidas em uma representação digital usando técnicas de Deep learning para posteriormente serem convertidas em um arquivo de importação do App Inventor. Este é o único trabalho encontrado que trata do desenvolvimento de interfaces para App Inventor. Para avaliar a qualidade da ferramenta desenvolvida por meio de um teste com 29 alunos e professores

com formação em Ciência da Computação e/ou Design e 6 alunos e 1 professor representando o público-alvo no contexto da educação básica. Neste teste os participantes geraram o código para 10 imagens de sketches predefinidos e um sketch feito por eles mesmo. Foram avaliados 4 fatores: correspondência entre o resultado da ferramenta e sketch, tempo de processamento da ferramenta, usabilidade da ferramenta, vantagens e desvantagens da utilização da ferramenta no processo de desenvolvimento de um app. Referente a correspondência entre o resultado da ferramenta e sketch, a ferramenta obteve 100% de correspondência. O tempo médio de processamento da ferramenta foi 57,7 segundos para gerar um aplicativo com 3 telas, o que foi considerado satisfatório pelos participantes, tendo apenas 6,9% dos participantes consideraram o tempo de execução acima do esperado. A usabilidade da ferramenta obteve 100% de resultados positivos dos participantes e 96,6% avaliaram a ferramenta como fácil de usar. Além disso, foi adotado o SUS (System Usability Scale) com o qual obteve uma pontuação média de 92,16, indicando excelente grau de satisfação. As principais vantagens apontadas são a facilidade de uso, praticidade e potencial em economia de tempo no processo de desenvolvimento de um App. Nenhuma desvantagem foi apontada, apenas sugestões de melhorias como implementação de cores nos previews, melhoria nas instruções de uso e utilização de formatos variados e principalmente questões de posicionamento.

3 Metodologia de pesquisa

O objetivo geral deste trabalho é desenvolver uma ferramenta para converter um projeto de design de interface exportado do Penpot em um arquivo “.aia” de importação do App Inventor.

A primeira etapa foi estudar, analisar e sintetizar os conceitos principais e a teoria referente aos temas a serem abordados neste trabalho é apresentado a fundamentação teórica utilizando a metodologia de revisão narrativa (Cordeiro et al., 2007).

A segunda etapa consistiu em desenvolver uma ferramenta para conversão de projeto Penpot para um arquivo “.aia” do App inventor, seguindo um processo de desenvolvimento de software iterativo (Larman; Basili, 2003). Esta etapa foi dividida nas seguintes atividades: Análise de requisito; Modelagem do sistema; e implementação e teste do sistema.

Durante a última etapa foi avaliada a qualidade da ferramenta desenvolvida em termos de completude e acurácia da interface gerada no arquivo “.aia” realizando uma avaliação com 3 projetos de design de interface. Esta etapa é dividida nas seguintes atividades: Definição da avaliação e criação do projetos de design a serem avaliados; Execução da conversão; Análise dos resultados.

4 Penpot2AIA

A ferramenta desenvolvida foi batizada de Penpot2AIA e foi concebida como um sistema *web*. A conversão dos arquivos de exportação de Penpot para “.aia” foi dividido em duas etapas de acordo com a Figura 1.

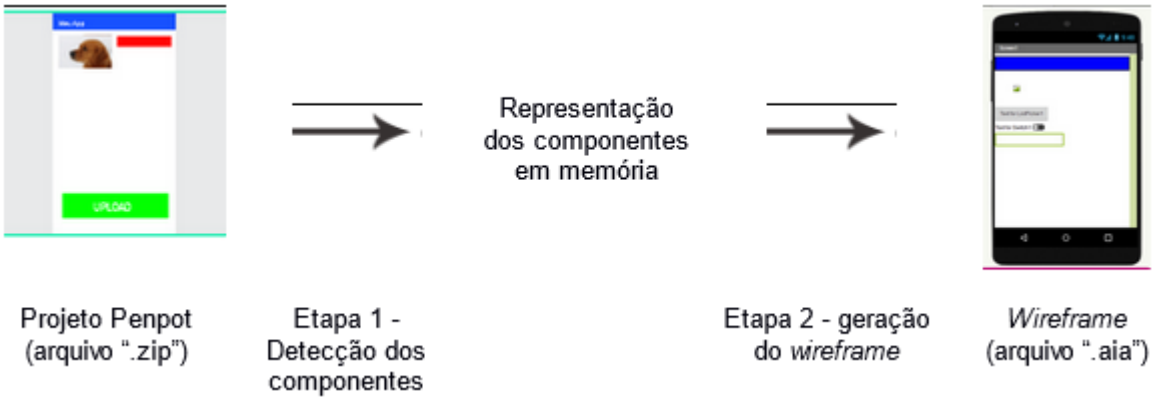


Figura 1. Etapas de geração de wireframes a partir de arquivos de exportação Penpot

Para que a entrada do sistema (arquivo de exportação Pentpot) seja válida, assume-se que foi usada uma biblioteca disponibilizada pelo Penpot2AIA, chamada "Penpot2AIA.zip", para confecção das telas e que o formato de exportação selcionado foi "JSON + SVG". A biblioteca "Penpot". A biblioteca "Penpot2AIA.zip" limita os componentes que podem ser usados na geração dos wireframes baseados nos componentes mais usados (Baulé, 2020).

A primeira etapa consiste na detecção dos elementos presentes no projeto Penpot. Nesta etapa a ferramenta identifica, com base em arquivos ".json" presentes no próprio arquivo de entrada (Figura 2).

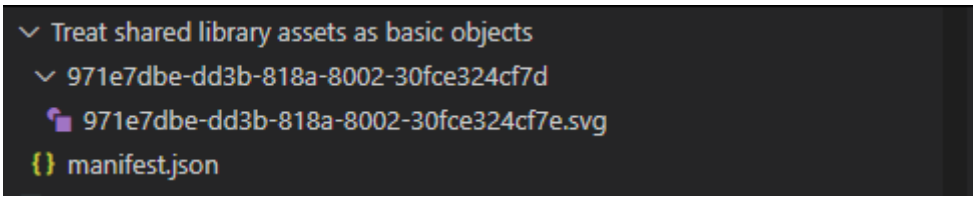


Figura 2. Exemplo de hierarquia de um projeto Penpot com uma tela exportado

Para cada tela os elementos presentes no projeto, sua posição na tela, atributos básicos como cores, tamanho, etc, usando por base os arquivos SVG armazenados dentro do arquivo ".zip" enviado à ferramenta conforme a Figura 3.

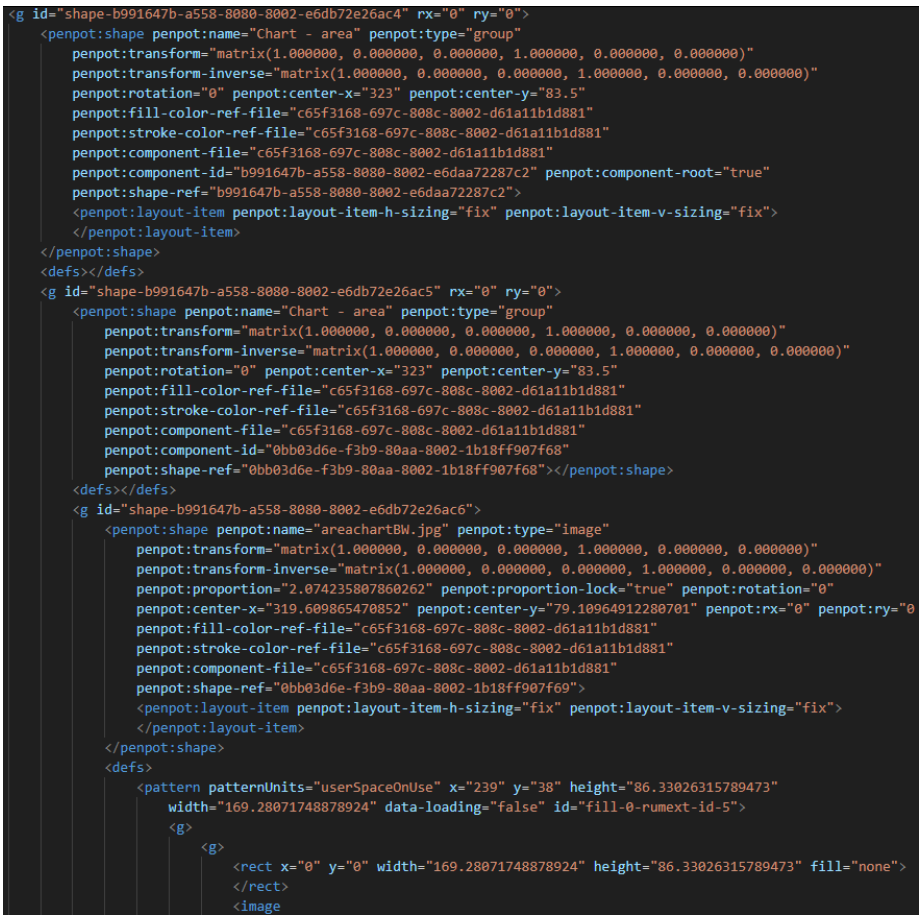


Figura 3. Estrutura XML presente no SVG de um componente

O resultado da primeira etapa é uma representação em memória, no formato JSON onde é sumarizada o nome do projeto, um *array* com todas as telas, e para cada tela, uma listagem dos componentes juntamente com suas propriedades conforme a Figura 4.

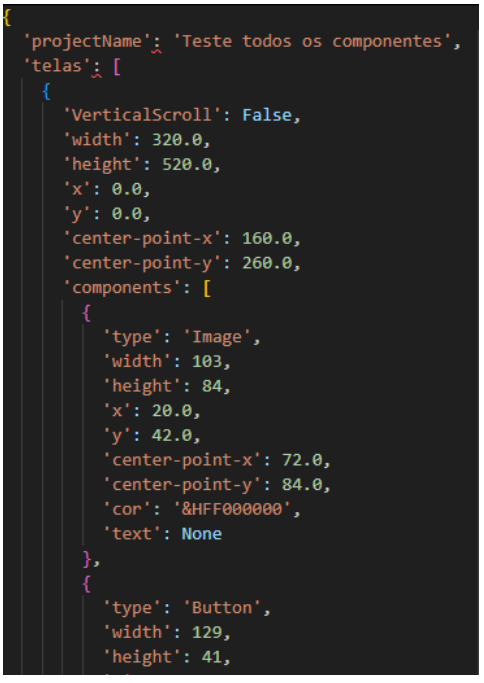


Figura 4. Representação dos componentes em memória

A segunda etapa utiliza a representação dos componentes em memória descrita na figura 4 e os converte na estrutura de arquivos que compoe o arquivo “.aia” funcional. Para cada tela é criado um arquivo “.scm” que possui os componentes de cada tela, porém, a representação de posicionamento dos componentes na tela usada pelo App Inventor é diferente da representação desenvolvida. Para criar uma representação equivalente entre os dois sistemas foi desenvolvido dois algoritmos: um algoritmo de conversão de posicionamento de componente para arranjos verticais e horizontais (sistema adotado pelo App Inventor); e um sistema de espaçadores para “afastar” os elementos verticalmente de forma a simular a propriedade “top”. As imagens usadas foram adicionadas no projeto “.aia” gerado para ser reutilizado no App Inventor.

O Penpot2AIA possui uma tela inicial, onde é disponibilizado a biblioteca “Penpot2AIA.zip” necessária para desenvolver as interfaces no Penpot, o botão para adicionar os arquivos “.zip” gerados e tutoriais ensinando o uso da ferramenta. A ferramenta possui versões nos idiomas Português e Inglês. Outras informações como “Políticas de uso”, “Termos de serviço” e “Sobre” são exibidas em páginas secundárias (Figura 5).

Figura 5. Tela da ferramenta Penpot2AIA

5 Avaliação da Ferramenta

Com o objetivo de avaliar a qualidade da conversão do projeto Penpot para o wireframe App Inventor é realizado um estudo de caso, testando a conversão com 3 projetos diferentes de Penpot, que foram inspirados em aplicativos reais desenvolvidos usando o App Inventor.

Todos os projetos Penpot foram exportados para o arquivo “.aia” e importado no App Inventor. A partir dos projetos importados no App Inventor foram gerados os arquivos “.apk”

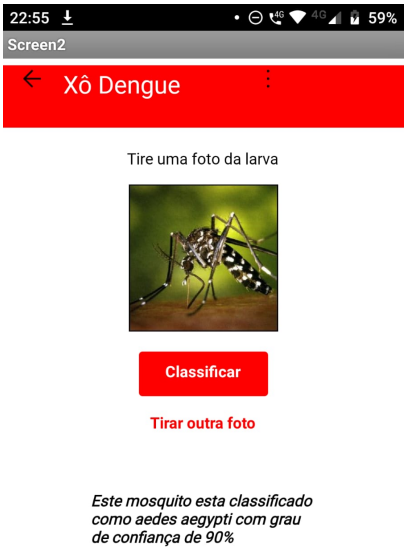
que foram instalados em um smartphone modelo Motorola Moto X4 com Android versão 9. Um evento de transição de tela foi inserido em um botão de cada tela.

Cada tela projetada foi comparada com o resultado final. Nenhuma configuração, como p. ex url de *webviewer*, posição inicial do Mapa, etc, foi acrescentada.

O primeiro projeto avaliado mostra um nível de fidelidade considerado bastante satisfatório exibindo apenas uma falta de linha em branco em uma frase com múltiplas linhas. O projeto Penpot e seu sua versão ".apk" instalado em um smartphone real pode ser visualizado na Tabela 1.

Tabela 1. Comparativos entre projeto Penpot e aplicativo Android do App "Xô Dengue"

Tela	Projeto Penpot	Apk gerado pelo App Inventor
1		

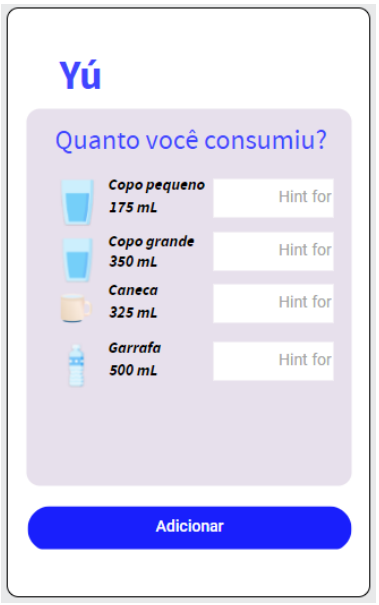
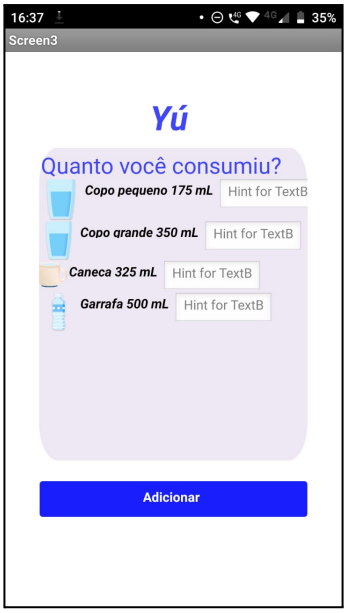
2		

O segundo projeto, na tela 2 apresenta a maior distorção entre os projetos. O Sistema identificou o título “Seu histórico” e os sliders como sendo partes do mesmo arranjo. Isto ocorreu pois a largura do título se assemelha a largura dos sliders. Ta tela 3 isto não ocorreu pois o título possui uma largura maior e portanto foi considerado como parte de um arranjo separado dos demais componentes.

Na tela 3 as telas se assemelham ao que foi idealizado durante o projeto. Os dois labels “Caneca 325mL” e “Garrafa 500mL” possuem uma largura menor do que os labels “Copo pequeno 175mL” e “Copo grande 350mL” trazendo suas respectivas caixas de textos fora do alinhamento. Caso a largura dos labels tivesse sido configurada com o mesmo valor esse comportamento não ocorreria. O projeto Penpot e seu sua versão “.apk” instalado em um smartphone real pode ser visualizado na Tabela 2:

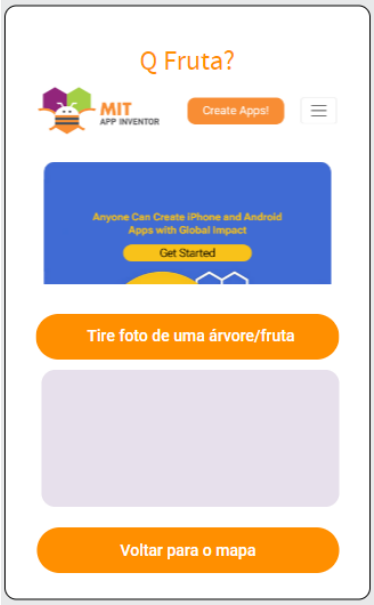
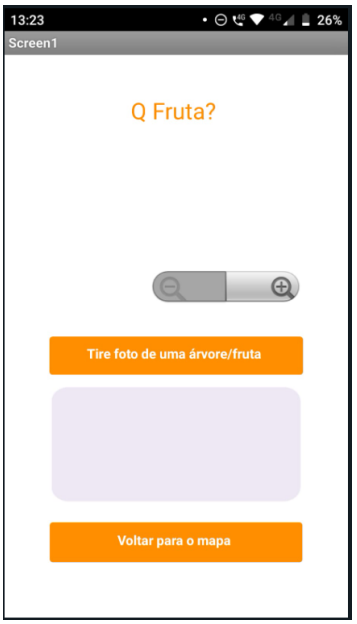
Tabela 2. Comparativos entre projeto Penpot e aplicativo Android do App “Yú”

Tela	Projeto Penpot	Apk gerado pelo App Inventor
1		
2		

3		

O terceiro projeto não apresentou muitas distorções entre o projeto desenvolvido no Penpot e eua equivalencia em formato de aplicativo no *smartphone* real conforme a Tabela 3:

Tabela 3. Comparativos entre projeto Penpot e aplicativo Android do App “QFruta?”

Tela	Projeto Penpot	Apk gerado pelo App Inventor
1		
2		

3		

6 Conclusão

Neste trabalho o objetivo principal foi desenvolver um modelo de geração automática de wireframes no App Inventor a partir de arquivos de exportação do Penpot. Para atingir este objetivo foi realizada inicialmente uma análise da fundamentação teórica sobre desenvolvimento de apps com App Inventor e design de interface de usuários com Penpot (O1). Em seguida foi realizado o levantamento do estado da arte em relação a ferramentas existentes de ferramentas de conversão de projetos UI no Penpot para App Inventor (O2). Esta etapa constatou a ausência desse tipo de ferramenta para o App Inventor como também além da ausência de artigos envolvendo Penpot. Foram também analisados artigos e softwares que tem por objetivo conversão de formatos de arquivos de exportação para importação entre sistemas. Como principal resultado do trabalho foi desenvolvida uma ferramenta para converter automaticamente projetos de design de interface de apps criados com Penpot para wireframes no App Inventor (O3). Em uma avaliação preliminar a ferramenta demonstrou um tempo de execução bem abaixo do especificado e uma fidelidade aceitável do wireframe com o projeto original no Penpot (O4). O resultado da pesquisa é disponibilizado como ferramenta web e pode ser acessado online: http://apps.computacaonaescola.ufsc.br/penpot2aia/.

Assim foi criada uma contribuição tecnológica inovadora facilitando o processo de desenvolvimento de aplicativos com App Inventor usando a ferramenta Penpot para o design de interfaces de usuário. Espera-se que essa ferramenta também contribua positivamente ao ensino de design de interface como parte do ensino de computação, facilitando e agilizando o processo de desenvolvimento.

Para trabalhos futuros, sugere-se a evolução da ferramenta como a detecção de elementos sem o uso de uma biblioteca auxiliar para abordar os componentes de diferentes criadores de interface no Penpot. Além do refino da geração do wireframe em si para que a interface gerada ao final seja mais fiel ao idealizado na etapa de desenvolvimento da interface no Penpot.

Referências

AI2 SideBar, 2023. Disponível em: https://ullisroboterseite.de/android-AI2-SideBar-en.html. Acesso em: junho de 2023.

Android Studio. CONVERTER designs em código no Android Studio, 2023. Disponível em: Https://developer.android.com/jetpack/compose/tooling/relay/convert-designs-android-studio?hl=pt-br. Acesso em: 26 jun. 2023.

AppInventor, 2023. Disponível em: http://www.appinventor.org/ Acesso em: Dex 2023

BAULÉ, Daniel de Souza. Desenvolvimento de um Modelo de Geração Automática de Wireframes no App Inventor a partir de Sketches usando Deep Learning. Trabalho e Conclusão de Curso (Bacharelado em Sistemas de Informação) - Universidade Federal de Santa Catarina, Florianópolis, 2020.

Export Kit. CONVERT XD to Android Studio XML and Java. Disponível em: https://exportkit.com/plugin/environments/xml/convert-xd-to-android-studio. Acesso em: 26 jun. 2023.

FIGMA. 2023. Disponível em: https://www.figma.com/. Acesso em: 26 jun. 2023.

GOOGLE. Material Design. Disponível em: <https://m3.material.io/>. Acesso em: 05 dez. 2023.

GROVER, S.; PEA, R. D. Computational thinking in K–12: A review of the state of the field. Educational Researcher, 42 (1), 38-43, 2013.

LARMAN, C.; BASILI, V. R. Iterative and Incremental Development: A Brief History. Computer, 36(6), pp. 47-56, 2003.

LYE, S. Y.; KOH, J. H. L. Review on teaching and learning of computational thinking through programming: What is next for K-12?. Computers in Human Behavior, 41(1), pp. 51–61, 2014.

MIT. App Inventor, 2023. Disponível em: http://appinventor.mit.edu Acessado em: abril 2023

Pacheco, J., Garbatov, S., & Goulao, M. Improving Collaboration Efficiency Between UX/UI Designers and Developers in a Low-Code Platform. In Proc. of ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion, IEEE, 2021.

PATTON, E.; TISSENBAUM, M.; Harunani, F. (2019). MIT App Inventor: Objectives, Design, and Development. Disponível em: <https://link.springer.com/content/pdf/10.1007%2F978-981-13-6528-7_3.pdf>. Acesso em: março 2023.

PENPOT. 2023. Disponível em: https://penpot.app/. Acesso em: junho de 2023

PETERSEN, K.; FELDT, R.; MUJTABA, S.; MATTSSON, M. Systematic Mapping Studies in Software Engineering. In: Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering, Bari, Italy, 2008, p. 68-77.

SCHLATTER, Tania & LEVINSON, Deborah. Visual Usability: Principles and Practices for Designing Digital Applications. 2013.

APÊNDICE C

Código-fonte da ferramenta Penpot2AIA:
https://codigos.ufsc.br/gqs/penpot2aia/