ESCOLA SUPERIOR DE MEDIA ARTES E DESIGN
POLITÉCNICO DO PORTO

ALP

FICHA DE EXERCÍCIOS
ATIVIDADE LETIVA

Programação Web I

UNIDADE CURRICULAR

Ficha 03 - Components

FICHA

Abra o VSC e crie projetos Vue para cada exercício

1) Este exercício explora a passagem de dados do pai para filho através de props
   a) Crie um componente **StudentCard.vue** que recebe os seguintes props:
      - **name** (string, obrigatório)
      - **studentNumber** (number, obrigatório)
      - **course** (string, com valor default "TSIW")
      - **year** (number, com valor default 2)
   b) O componente deve exibir todas estas informações de forma estruturada
   c) Adicione validação aos props
      a. **year** deve estar entre 1 e 3
      b. **studentNumber** deve ser um número positivo
   d) No App.vue, crie um array com dados de 3 alunos diferentes
   e) Renderize um StudentCard para cada aluno, passando os dados via props

2) Vamos explorar a reatividade das props e introduzir um evento básico
   a) Crie um componente **ProductCard.vue** que recebe:
      - **product** (object, obrigatório) com propriedades: name, price, image (URL) e inStock
   b) O componente deve exibir:
      - Imagem do produto
      - Nome do produto
      - Preço formatado (€)
      - Badge "Em Stock" ou "Esgotado" consoante o valor de inStock
      - Botão "Adicionar ao Carrinho" (apenas Se inStock for true)
   c) Crie um componente **ProductList.vue** que:
      - Mantém um array de produtos em **data()**
      - Renderiza múltiplos ProductCard

- Passa cada produto via props

d) Quando o botão "Adicionar ao Carrinho" é clicado, o ProductCard deve emitir um evento **add-to-cart** com o nome do produto

e) O ProductList deve capturar este evento e exibir um alerta com "Produto X adicionado ao carrinho!"

f) No **App.vue**, utilize apenas o componente ProductList

Desafio: Adicione um botão no ProductList que altera o stock de um produto aleatório e observe a reatividade nos cards

3) Este exercício foca na emissão de eventos personalizados do filho para o pai

a) Crie um componente **RatingStars.vue** que:

- Exibe 5 SVGs de estrelas
- Permite ao utilizador clicar numa estrela para dar uma classificação de 1 a 5
- Emite um evento **rating-changed** como valor da classificação escolhida

b) Crie um componente **MovieReview.vue** que:

- Tem dados sobre um filme (título e descrição)
- Exibe esses dados
- Utiliza o componente RatingStars
- Captura o evento **rating-changed** e guarda a classificação em **data()**
- Exibe a classificação atual abaixo das estrelas (exemplo: "Classificação: 4/5")
- Tem um botão "Marcar como Visto" que emite o evento **mark-as-watched**
- Exibe um indicador visual quando o filme foi visto

c) No **App.vue**, utilize o componente MovieReview

Desafio: Adicione múltiplos MovieReview

4) Vamos combinar props e events para criar uma comunicação completa entre componentes

a) Crie um componente **TaskItem.vue** que recebe:

- **task** (object) com: id, description, completed

b) O TaskItem deve:

- Exibir a descrição da tarefa
- Exibir checkbox que reflete o estado **completed**
- Quando a checkbox é alterada, emitir evento **toggle-task** com o id da tarefa
- Exibir um botão "Remover" que emite o evento **delete-task** com o id
- Aplicar estilo de "riscado" se a tarefa estiver completa

c) Crie um componente **TaskInput.vue** que:

- Tem um input de texto e um botão "Adicionar"
- Quando o botão é clicado, emite evento **add-task** com a descrição da nova tarefa
- Limpa o input após emitir o evento
- Não permite adicionar tarefas vazias

d) Crie um componente **TaskManager.vue** que:

- Mantém um array de tarefas em **data()**
- Utiliza TaskInput e captura o evento **add-task** para adicionar tarefas
- Renderiza múltiplos TaskItem
- Captura eventos **toggle-task** e **delete-task** atualizando o array
- Exibe contador de tarefas totais e completas

e) No **App.vue**, utilize apenas o TaskManager

Desafio: Adicione persistência com localStorage

5) Este exercício explora a utilização de slots para criar componentes flexíveis e reutilizáveis

a) Crie um componente **FAQItem.vue** que:

- Representa uma única pergunta/resposta
- Tem três named slots:
    - **title:** a pergunta
    - **subtitle:** 1 ou 2 frases de resumo da resposta
    - **body:** a resposta detalhada
- Mantém um estado local **isOpen** (boolean, default false)
- Ao clicar no título, mostra/esconde o body consoante **isOpen**
- Exibe um indicador visual que muda com o estado

b) Crie um componente **FAQSection.vue** que:

- Recebe um prop **faqs** (array de objetos)
- Cada objeto deve ter: **id**, **title**, **subtitle**, **body**
- Exibe um título principal "Perguntas Frequentes"
- Renderiza um FAQItem para cada FAQ do array
- Preenche os slots para permitir passar o conteúdo de cada FAQ

c) No **App.vue:**

- Crie um array **faqs** com 4-5 objetos
- Utilizar o componente FAQSection passando o array via props

Desafio: Adicionar um botão "Fechar Todos" no FAQSection que esconde todas as FAQs; Implementar funcionalidade onde apenas 1 FAQ pode estar visível de cada vez (ao abrir uma, as outras fecham automaticamente)