CHANGED 13 DIAS ATRÁS

0

Workshop 18 — TypeScript

> Nós vamos "tipar" o projeto existente e adicionar uma funcionalidade de inserção de workshop usando TypeScript, mantendo a arquitetura modular (`api` , `dom` , `main`).

> Também temos uma versão da API disponível em http://workshop.verum.tech/, para facilitar a implementação do frontend

---

Parte 1 — Adicionar TypeScript e testar a transpilação

1) Instalar TypeScript

```
npm i -D typescript @types/toastify-js
```

2) Criar `tsconfig.json`

Crie na raiz (ajuste se necessário):

```
{
  "compilerOptions": {
    "target": "ES2022",
    "module": "ESNext",
    "moduleResolution": "bundler",
    "lib": ["ES2022", "DOM", "DOM.Iterable"]
```

Workshop 18 — TypeScript
Parte 1 — Adicionar TypeSc...
Parte 2 — Adicionar inserçã...
Critérios de aceite
Dicas rápidas

Expand all
Back to top
Go to bottom

Nós vamos "tipar" o projeto existente e adicionar uma funcionalidade de inserção de workshop usando TypeScript, mantendo a arquitetura modular (`api` , `dom` , `main`).

Também temos uma versão da API disponível em http://workshop.verum.tech/, para facilitar a implementação do frontend

---

Parte 1 — Adicionar TypeScript e testar a transpilação

1) Instalar TypeScript

```
npm i -D typescript @types/toastify-js
```

2) Criar `tsconfig.json`

Crie na raiz (ajuste se necessário):

```
{
  "compilerOptions": {
    "target": "ES2022",
    "module": "ESNext",
    "moduleResolution": "bundler",
    "lib": ["ES2022", "DOM", "DOM.Iterable"]
```

Workshop 18 — TypeScript
Parte 1 — Adicionar TypeSc...
Parte 2 — Adicionar inserçã...
Critérios de aceite
Dicas rápidas

Expand all
Back to top
Go to bottom

Nós vamos "tipar" o projeto existente e adicionar uma funcionalidade de inserção de workshop usando TypeScript, mantendo a arquitetura modular (`api` , `dom` , `main`).

Também temos uma versão da API disponível em http://workshop.verum.tech/, para facilitar a implementação do frontend

---

Parte 1 — Adicionar TypeScript e testar a transpilação

1) Instalar TypeScript

```
npm i -D typescript @types/toastify-js
```

2) Criar `tsconfig.json`

Crie na raiz (ajuste se necessário):

```
{
  "compilerOptions": {
    "target": "ES2022",
    "module": "ESNext",
    "moduleResolution": "bundler",
    "lib": ["ES2022", "DOM", "DOM.Iterable"]
```

```
const WORKSHOPS_PATH = '/workshops'; // ajuste se for '/api/workshops'

export async function searchWorkshops(q: string): Promise<WorkshopList> {
  const { data } = await axios.get<WorkshopList>(
    `${API_BASE}${WORKSHOPS_PATH}`,
    { params: { q }, headers: getHeaders() }
  );
  return data;
}
```

Em `src/dom.ts`, **tipar** parâmetros:

```ts
import type { Workshop } from './api';

export function renderList(container: HTMLElement, items: Workshop[] = []): void {
  // ... (manter implementação)
}

export function updateMeta(el: HTMLElement, text = ''): void {
  el.textContent = text;
}

export function renderLoading(container: HTMLElement) {
  // ... (manter implementação)
}
```


> Nós vamos "tipar" o projeto existente e adicionar uma funcionalidade de inserção de workshop usando TypeScript, mantendo a arquitetura modular (`api`, `dom`, `main`).

> Também temos uma versão da API disponível em http://workshop.verum.tech/, para facilitar a implementação do frontend

---

Parte 1 — Adicionar TypeScript e testar a transpilação

1) Instalar TypeScript

```bash
npm i -D typescript @types/toastify-js
```

2) Criar `tsconfig.json`

Crie na raiz (ajuste se necessário):

```ts
{
  "compilerOptions": {
    "target": "ES2022",
    "module": "ESNext",
    "moduleResolution": "bundler",
    "lib": ["ES2022", "DOM", "DOM.Iterable"]
  }
}
```

```ts
  toastify({ text, gravity: "top", position: "right", backgroundColor: bg, duration: 2400
}

async function runSearch(term: string) {
  // ... (manter implementação)
}
```

5) Scripts de checagem

No `package.json` adicione:

```ts
"scripts": {
  "dev": "vite",
  "build": "vite build",
  "preview": "vite preview",
  "check": "tsc --pretty --noEmit"
}
```

6) Rodar e validar

```bash
npm run check       # typecheck (sem emitir arquivos)
npm run dev         # abra a URL, confirme a listagem
git add .
git commit -m "chore(ts): adiciona TypeScript e tipa módulos base"
```


Workshop 18 — TypeScript
Parte 1 — Adicionar TypeSc...
Parte 2 — Adicionar inserçã...
Critérios de aceite
Dicas rápidas

Expand all
Back to top
Go to bottom

Nós vamos "tipar" o projeto existente e adicionar uma funcionalidade de inserção de workshop usando TypeScript, mantendo a arquitetura modular (`api`, `dom`, `main`).

Também temos uma versão da API disponível em http://workshop.verum.tech/, para facilitar a implementação do frontend

---

Parte 1 — Adicionar TypeScript e testar a transpilação

1) Instalar TypeScript

```bash
npm i -D typescript @types/toastify-js
```

2) Criar `tsconfig.json`

Crie na raiz (ajuste se necessário):

```ts
{
  "compilerOptions": {
    "target": "ES2022",
    "module": "ESNext",
    "moduleResolution": "bundler",
```


```ts
    <input id="title" name="title" class="rounded-xl bg-white/5 ring-1 ring-white/10 px-4
           placeholder="Título" required />
    <input id="capacity" name="capacity" type="number" min="1" class="rounded-xl bg-white/
```

```
          placeholder="Capacidade" />
<textarea id="description" name="description" type="text" class="rounded-xl bg-white/5 rir
    <label class="flex items-center gap-2">
      <input id="isOnline" name="isOnline" type="checkbox" class="accent-sky-500">
      <span>Online</span>
    </label>
    <input id="location" name="location" class="rounded-xl bg-white/5 ring-1 ring-white/10
          placeholder="Local (se presencial)"/>

    <input id="startAt" name="startAt" type="datetime-local" class="rounded-xl bg-white/5
    <input id="endAt"   name="endAt"   type="datetime-local" class="rounded-xl bg-white/5

    <button type="submit"
      class="md:col-span-2 mt-2 rounded-xl px-5 py-2 font-medium bg-emerald-600 hover:bg-e
      Inserir
    </button>
  </form>
</section>
```

> **Dica UX:** se "Online" estiver marcado, deixe `location` opcional/disabled no submit.

---

2) API: tipos e assinatura do POST

> Nós vamos "tipar" o projeto existente e adicionar uma funcionalidade de inserção de workshop usando TypeScript, mantendo a arquitetura modular (`api` , `dom` , `main`).

> Também temos uma versão da API disponível em http://workshop.verum.tech/, para facilitar a implementação do frontend

---

Parte 1 — Adicionar TypeScript e testar a transpilação

1) Instalar TypeScript

```
npm i -D typescript @types/toastify-js
```

2) Criar `tsconfig.json`

Crie na raiz (ajuste se necessário):

```
{
  "compilerOptions": {
    "target": "ES2022",
    "module": "ESNext",
    "moduleResolution": "bundler",
    "lib": ["ES2022", "DOM", "DOM.Iterable"]
```

> **Then:** retorna `Workshop` criado (com `id` do backend)

---

3) `main.ts` : capturar o submit e montar o payload

No topo de `src/main.ts` , selecione o form e campos:

```
// Seletores da parte de inserção
const addForm = document.getElementById('add-form') as HTMLFormElement;
const titleEl = document.getElementById('title') as HTMLInputElement;
const descriptionEl = document.getElementById('description') as HTMLInputElement;
const capacityEl = document.getElementById('capacity') as HTMLInputElement;
const isOnlineEl = document.getElementById('isOnline') as HTMLInputElement;
const locationEl = document.getElementById('location') as HTMLInputElement;
const startAtEl = document.getElementById('startAt') as HTMLInputElement;
const endAtEl = document.getElementById('endAt') as HTMLInputElement;
```

Implemente a função de parseNewWorkshopFromForm:

```
import type { NewWorkshop, Workshop } from './api';
import { createWorkshop } from './api';

// Converte 'YYYY-MM-DDTHH:mm' local para ISO (UTC ou local → decida e mantenha consistent
function toIsoFromLocal(input: string): string {
  if (!input) throw new Error('Data/hora inválida.');
```

Workshop 18 — TypeScript
Parte 1 — Adicionar TypeSc...
Parte 2 — Adicionar inserçã...
Critérios de aceite
Dicas rápidas

Expand all
Back to top
Go to bottom

Nós vamos "tipar" o projeto existente e adicionar uma funcionalidade de inserção de workshop usando TypeScript, mantendo a arquitetura modular (`api` , `dom` , `main`).

Também temos uma versão da API disponível em http://workshop.verum.tech/, para facilitar a implementação do frontend

Parte 1 — Adicionar TypeScript e testar a transpilação

1) Instalar TypeScript

```
npm i -D typescript @types/toastify-js
```

2) Criar `tsconfig.json`

Crie na raiz (ajuste se necessário):

```
{
```

```
  "compilerOptions": {
    "target": "ES2022",
    "module": "ESNext",
    "moduleResolution": "bundler",
    "lib": ["ES2022", "DOM", "DOM.Iterable"]
}

addForm.addEventListener('submit', async (e) => {
  e.preventDefault();
  try {
    const payload = parseNewWorkshopFromForm(); // TODO implementar
    // TODO: chamar createWorkshop(payload) e aguardar resposta
    // TODO: limpar formulário / feedback positivo
    // TODO: (opcional) chamar novamente a busca atual para "ver" o novo item
    toast('Workshop inserido com sucesso!', 'ok');
  } catch (err) {
    console.error(err);
    const msg = err instanceof Error ? err.message : 'Erro ao inserir';
    toast(msg, 'err');
  }
});
```

> **Importante:** mantenha o tratamento de exceções no `main.ts` para centralizar a UX (toasts/meta) e deixar o `api.ts` focado em I/O.

---

4) Rodar e verificar

```
npm run check   # typecheck: não deve haver erros após implementar os TODOs
npm run dev     # testar fluxo de inserção
```


> Nós vamos "tipar" o projeto existente e adicionar uma funcionalidade de inserção de workshop usando TypeScript, mantendo a arquitetura modular (`api` , `dom` , `main`).

> Também temos uma versão da API disponível em http://workshop.verum.tech/, para facilitar a implementação do frontend

---

Parte 1 — Adicionar TypeScript e testar a transpilação

1) Instalar TypeScript

```
npm i -D typescript @types/toastify-js
```

- **Parte 2**
  - Novo formulário de inserção disponível na UI.
  - Função `createWorkshop(payload: NewWorkshop)` criada (assinatura + implementação).
  - Validação mínima no `parseNewWorkshopFromForm` (datas, título, local se presencial, capacidade).
  - Ao inserir: toast de sucesso; em falha: toast com mensagem clara.
  - (Opcional) A lista é atualizada para incluir o novo item.

Dicas rápidas

- **Datas**: `datetime-local` não inclui timezone. Decida se converterá para `toISOString()` (UTC) ou manterá como local (ISO sem Z). **Alinhe** com o formato esperado pela API.
- **Tipos ajuda**: use `as HTMLInputElement` nos seletores. Em coleções, prefira `NodeListOf<HTMLElement>` .
- **Erros legíveis**: lance `new Error("mensagem clara")` no parser do form para o `catch` exibir no toast.
- **Sem libs extras**: evite trazer zod/yup aqui; foque em TypeScript + validação manual.