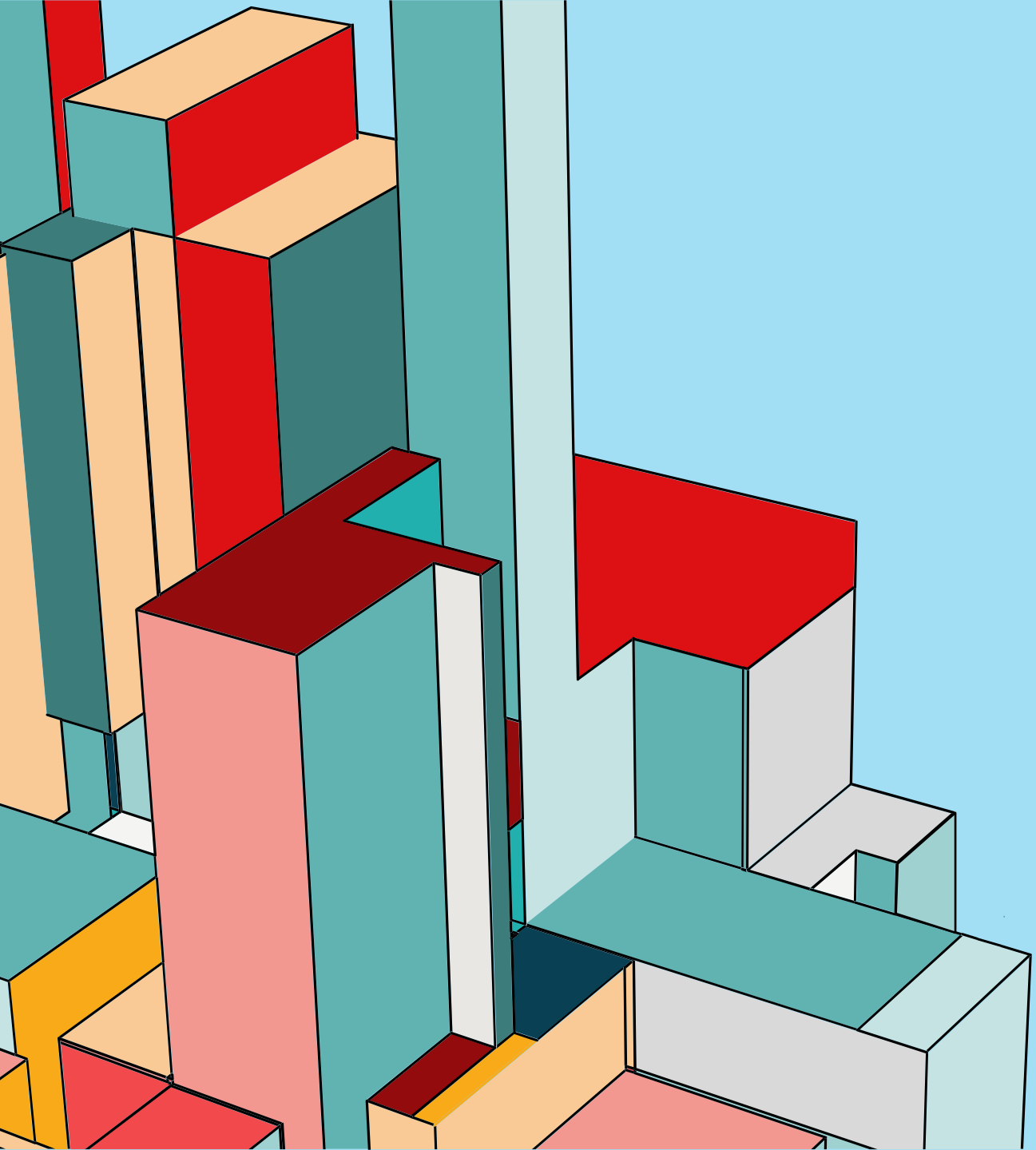


UNIVAS
UNIVERSIDADE DO VALE DO SAPUCAÍ

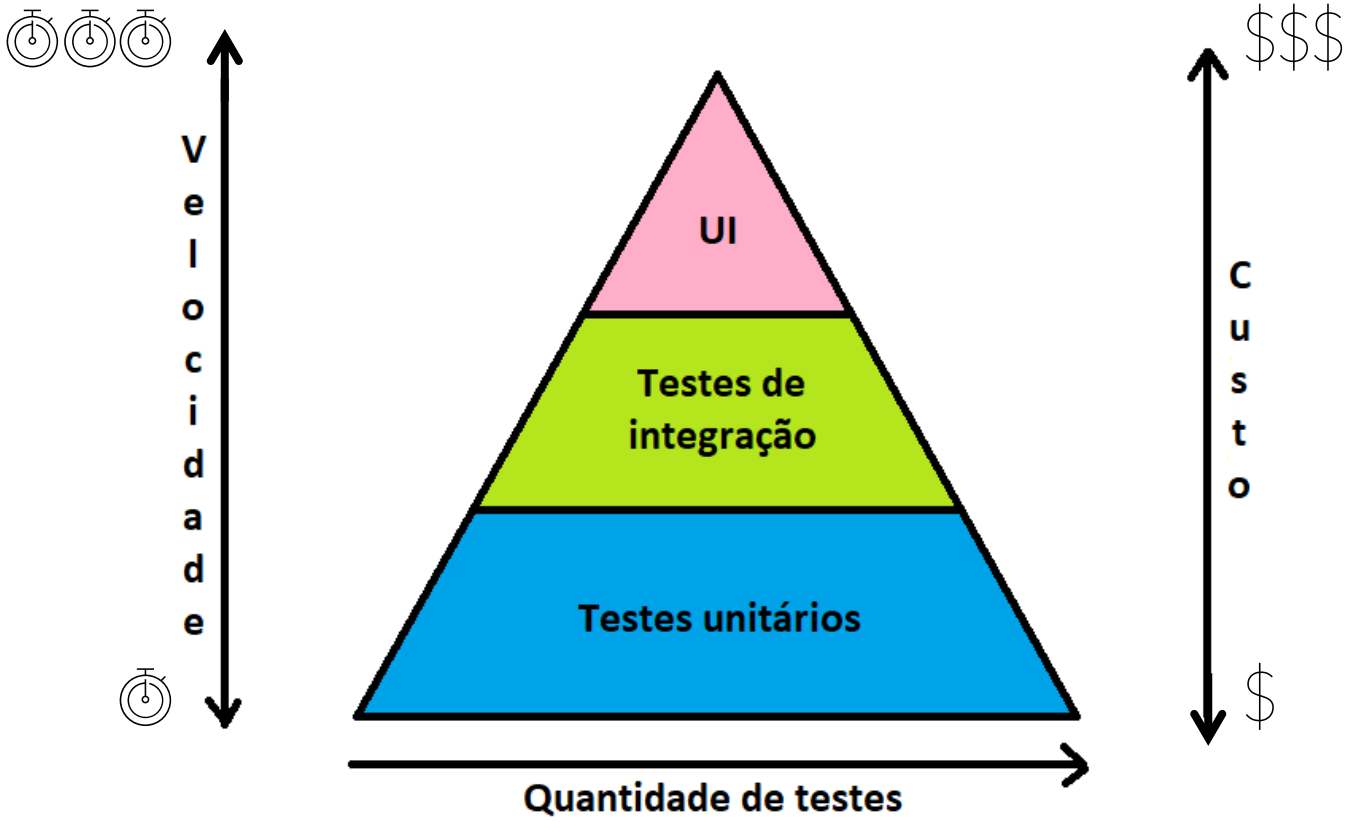
TESTE DE SOFTWARE

TESTES E2E – END TO END

Prof. Flávio Belizário da Silva Mota
Universidade do Vale do Sapucaí – UNIVAS
Sistemas de Informação

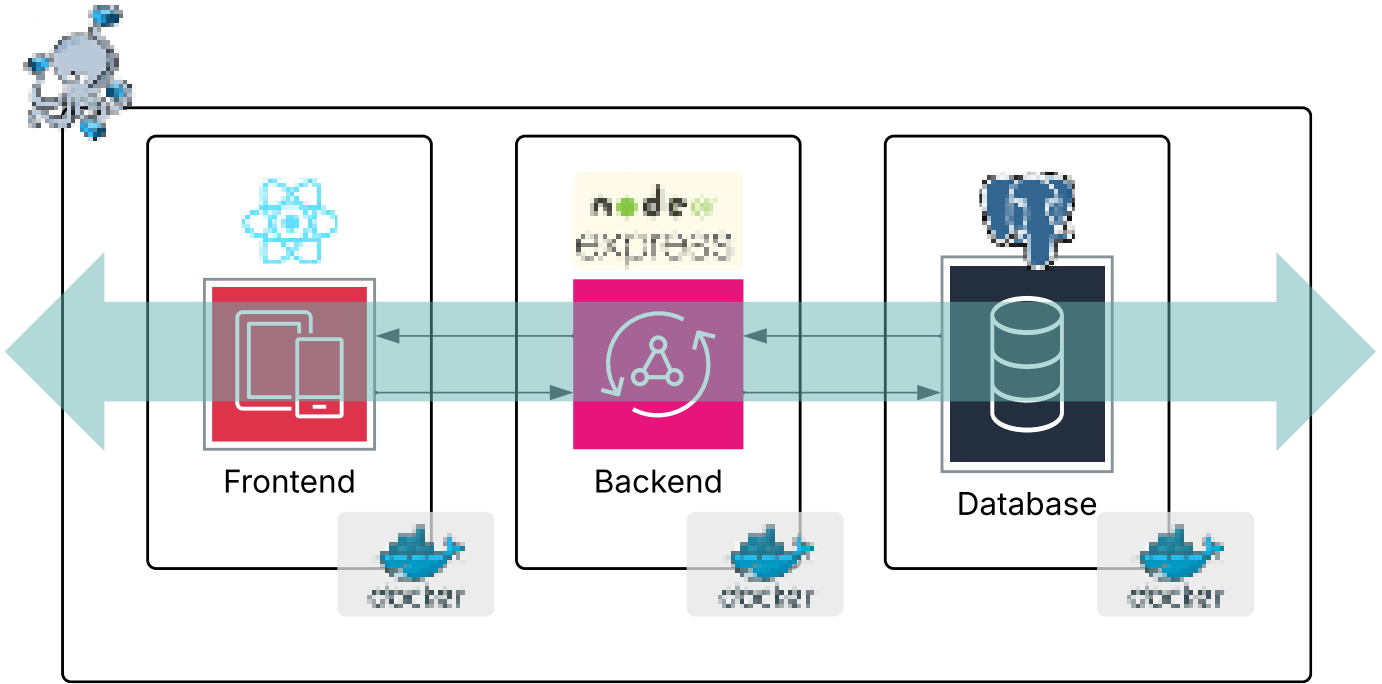
TESTES E2E

- Testes E2E validam o fluxo integral de uso do sistema, garantindo que todos os componentes envolvidos funcionem de forma alinhada.
- Validam o sistema exatamente como o usuário final o utiliza, reproduzindo fluxos completos, do início ao fim.
- Cada nível da pirâmide de testes é complementar: o E2E não substitui os anteriores.



TESTES E2E

- Percorrem todas as camadas do sistema de ponta a ponta.
- Focam em comportamentos emergentes, não em funções individuais.
- Envolvem ambiente realista: UI, API, banco, serviços externos.
- O objetivo é testar o sistema operando como um todo, não partes separadas.



PRINCIPAIS CARACTERÍSTICAS

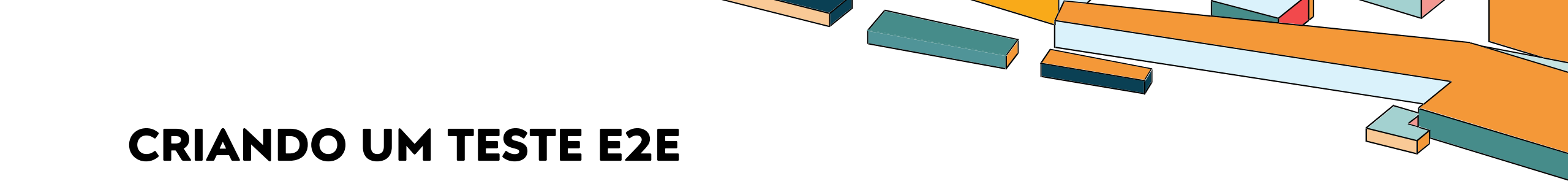
- Scripts simulam ações reais: clicar, digitar, navegar, esperar carregamentos.
- Avaliam a experiência final, não a lógica interna do código.
- Validam fluxos que refletem o comportamento humano dentro da aplicação.
- Garantem que o sistema reage de forma intuitiva e funcional no uso real.

DESAFIOS DOS TESTES E2E

- Mais lentos do que testes unitários e de integração.
- Mais caros e complexos de manter.
- Dependem de vários componentes funcionando simultaneamente.
- Mais suscetíveis a falhas intermitentes (por ambiente, rede, tempo de carregamento).
- Exigem ambientes estáveis e bem configurados.

PLAYWRIGHT

- Ferramenta que ganhou bastante espaço por ter suporte a múltiplos navegadores (Chrome, Edge, Firefox, Safari).
- Tem suporte para cenários complexos e assíncronos.
- Fácil de escrever, manter e executar.
- Ideal para validar fluxos E2E reais no navegador.



CRIANDO UM TESTE E2E

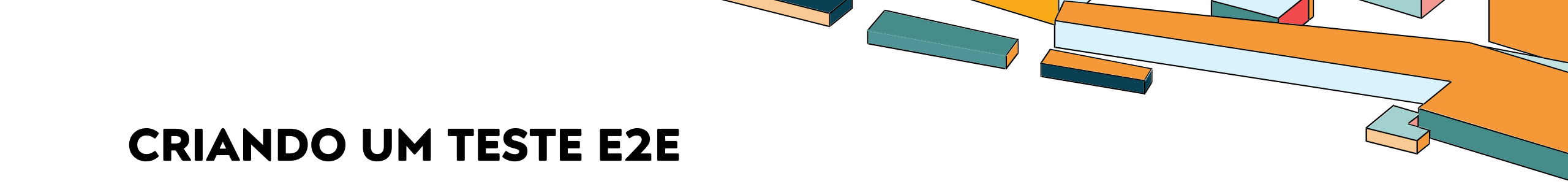
O objetivo da prática será testar a navegação entre as páginas do Task Manager.

Vamos utilizar o ambiente Docker para rodar a aplicação, porém o Playwright vai rodar localmente.

Sendo assim, vamos garantir que o nosso ambiente está rodando e tem dados:

```
docker-compose up -d
```

```
docker-compose exec backend npm run db:seed
```



CRIANDO UM TESTE E2E

No host (se estiver em ambiente Windows, utilize o Windows PowerShell), execute:

```
cd frontend
```

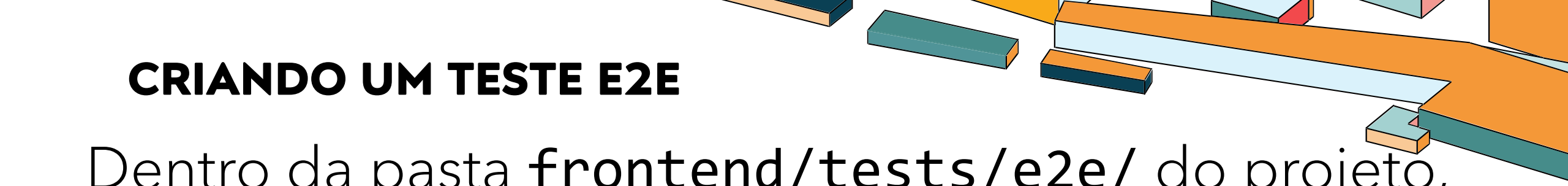
```
npm i -D @playwright/test
```

```
npx playwright install
```

CRIANDO UM TESTE E2E

Dentro da pasta **frontend** do projeto, crie o arquivo **playwright.config.ts**

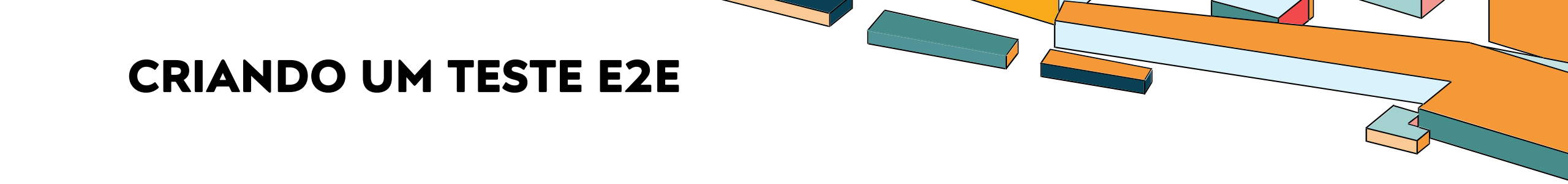
```
import { defineConfig } from '@playwright/test'
export default defineConfig({
  testDir: './tests/e2e',
  timeout: 30_000,
  retries: 0,
  reporter: [['list'], ['html', { open: 'never' }]],
  use: {
    baseURL: 'http://localhost:3000',
    headless: true,
    trace: 'on-first-retry',
    screenshot: 'only-on-failure',
    video: 'retain-on-failure'
  }
})
```



CRIANDO UM TESTE E2E

Dentro da pasta `frontend/tests/e2e/` do projeto, crie o arquivo `Navigation.e2e.spec.ts`

https://drive.google.com/file/d/1lMYKo_BdwHHrW_zXQqmUbpsiBuVJtUWM/view?usp=sharing



CRIANDO UM TESTE E2E

Para rodar o teste é possível executar:

Headless: roda todos os testes **sem abrir a janela do navegador**, utilizando o modo **headless** definido na configuração

```
npx playwright test
```

UI runner: abre a interface visual do Playwright para acompanhar a execução de cada teste

```
npx playwright test --ui
```

Relatório: **não executa testes**. Ele apenas abre o relatório da última execução.

```
npx playwright show-report
```

CRIANDO UM TESTE E2E

Vamos criar agora um teste para verificar a listagem de usuários. Dentro da pasta `frontend/tests/e2e/` do projeto, crie o arquivo `Users.e2e.spec.ts`

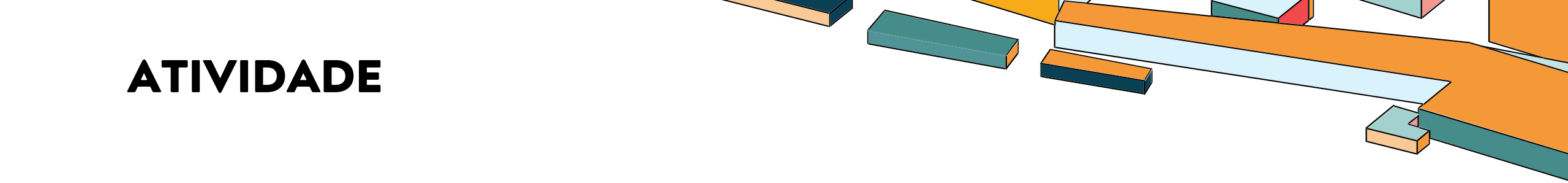
```
import { test, expect } from '@playwright/test'
test.describe('Usuários', () => {

  test('navega para Usuários e lista itens do backend', async ({ page }) => {
    await page.goto('/') // Dashboard
    await page.getByRole('link', { name: 'Usuários' }).click()
    // Título da seção
    await expect(page.getByRole('heading', { name: /Usuários/i })).toBeVisible()
    // Emails semeados (seed do backend)
    await expect(page.getByText(/john.doe@example.com/i)).toBeVisible()
    await expect(page.getByText(/jane.smith@example.com/i)).toBeVisible()
  });
});
```

CRIANDO UM TESTE E2E

Vamos criar agora um teste para cadastrar usuários. Dentro do mesmo arquivo `Users.e2e.spec.ts` vamos criar um novo teste

```
test('cria usuário e aparece na lista', async ({ page }) => {
    await page.goto('/users')
    await page.getByRole('button', { name: /Adicionar Usuário/i }).click()
    const uniqueEmail = `aluno.${Date.now()}@ex.com`
    await page.getByLabel('Nome:').fill('Aluno E2E')
    await page.getByLabel('Email:').fill(uniqueEmail)
    await page.getByRole('button', { name: /Criar/i }).click()
     // Aguarda recarga da lista
    await expect(page.getByText(uniqueEmail)).toBeVisible()
  });
```



ATIVIDADE

- Crie os testes para atualizar e excluir Usuários.
- Crie os testes de CRUD para Categorias.