

Serverless Computing

Samuel Martins

Serverless Computing

- Possibilidade de execução de funções sem a necessidade de subir um servidor;
- Suporte a várias linguagens de programação (inclusive javascript);
- Cobrança baseada na execução e performance das funções (quando mais demorada a execução de uma função, maior o seu custo);
- Escalabilidade automática - provedor de cloud consegue escalar o uso da função na medida em que o seu uso cresce.

Serverless computing

Casos de uso

- Aplicações *serverless-side rendering*;
 - Artigo: [Building server-side rendering for React in AWS Lambda](#);
- Aplicações web e APIs em geral;
- Processamento de dados em tempo real;
- Back-end de IoT (internet das coisas).

Serverless computing

Plataformas

- AWS Lambda;
- Azure Functions;
- Google Cloud Functions;
- Netlify Functions;



AWS Lambda



Google Cloud Functions



Azure
Functions



Netlify Functions

Aplicações PWA

Samuel Martins

Aplicações PWA

Progressive Web Apps (PWA) são aplicações baseadas na Web que oferecem ao usuário funcionalidades como a possibilidade de utilização sem conexão (offline), notificação push e acesso a recursos nativos dos dispositivos móveis.



Aplicações PWA - o que torna uma aplicação web progressiva?

"Os Progressive Web Apps fornecem uma experiência instalável, semelhante a um aplicativo, em computadores e dispositivos móveis que são criados e entregues diretamente pela Web. Eles são aplicativos da web que são rápidos e confiáveis. E o mais importante, são aplicativos da web que funcionam em qualquer navegador. Se você está construindo um aplicativo da Web hoje, já está no caminho da criação de um aplicativo da Web progressivo."

Fonte: <https://developers.google.com/web/fundamentals/codelabs/your-first-pwapp?hl=pt-br>

Aplicações PWA

- Código JavaScript que executa em background;
- É associado a um domínio (scope);
- Gerencia as diversas páginas do escopo;
- Vive mesmo após o fechamento de uma página;
- Requer HTTPS para comunicação (Exceção localhost);
- Responde à eventos.

Aplicações PWA - Manifest

```
1 {  
2   "name": "Weather",  
3   "short_name": "Weather",  
4   "icons": [{  
5     "src": "/images/icons/icon-128x128.png",  
6     "sizes": "128x128",  
7     "type": "image/png"  
8   }, {  
9     "src": "/images/icons/icon-144x144.png",  
10    "sizes": "144x144",  
11    "type": "image/png"  
12  }],  
13  "start_url": "/index.html",  
14  "display": "standalone",  
15  "background_color": "#3E4EB8",  
16  "theme_color": "#2F3BA2"  
17 }
```

Aplicações PWA - Manifest

```
1 <!-- CODELAB: Add link rel manifest -->  
2 <link rel="manifest" href="/manifest.json">
```

Aplicações PWA - APIs

Service Worker API

Atua como proxy entre a aplicação e o servidor

Cache API

Armazena os arquivos estáticos da aplicação

Fetch

Requisições HTTP ao servidor / Service Worker

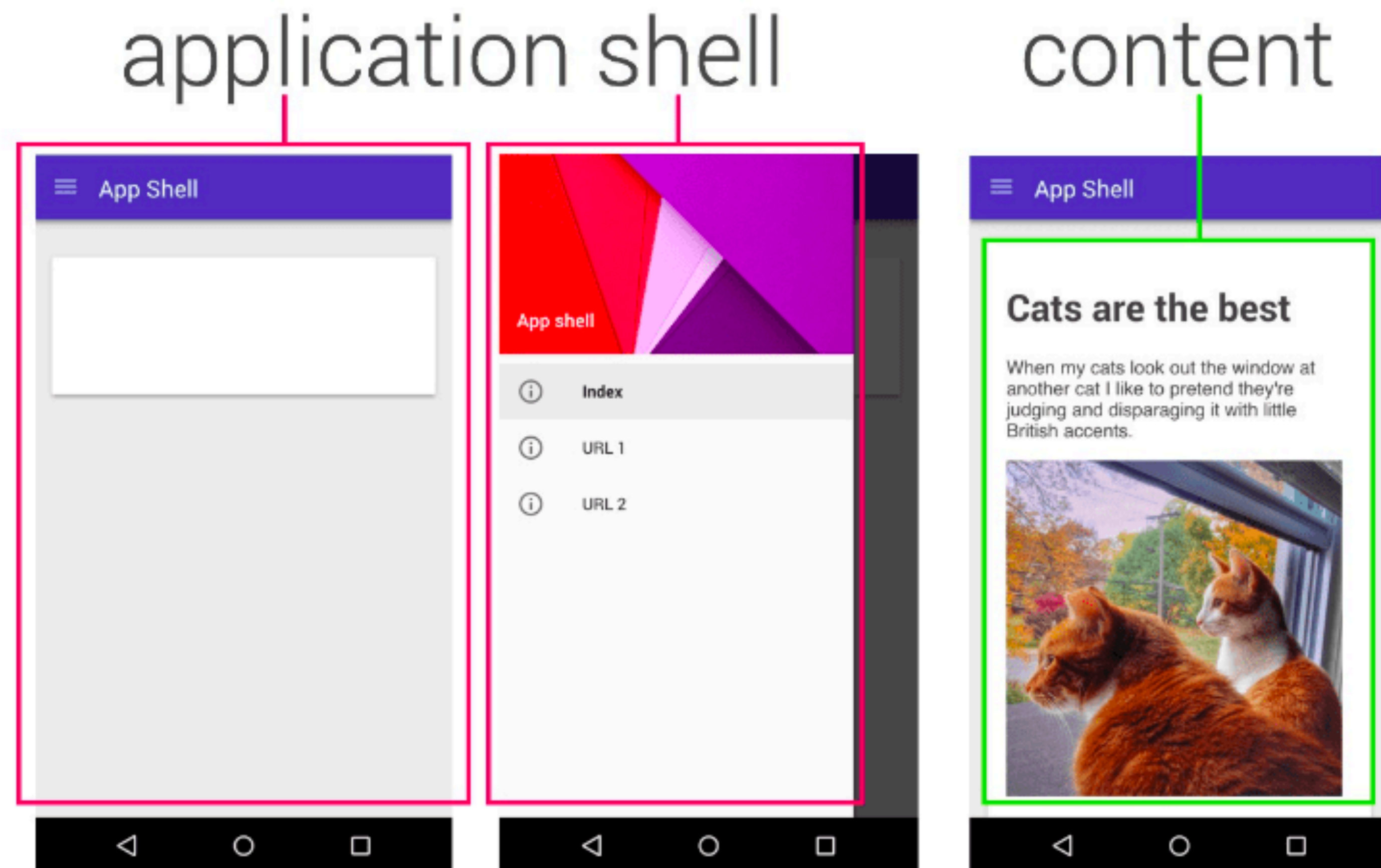
Push Notifications

Permite interagir com o usuário (AppLike)

Indexed DB

Estrutura de dados no browser

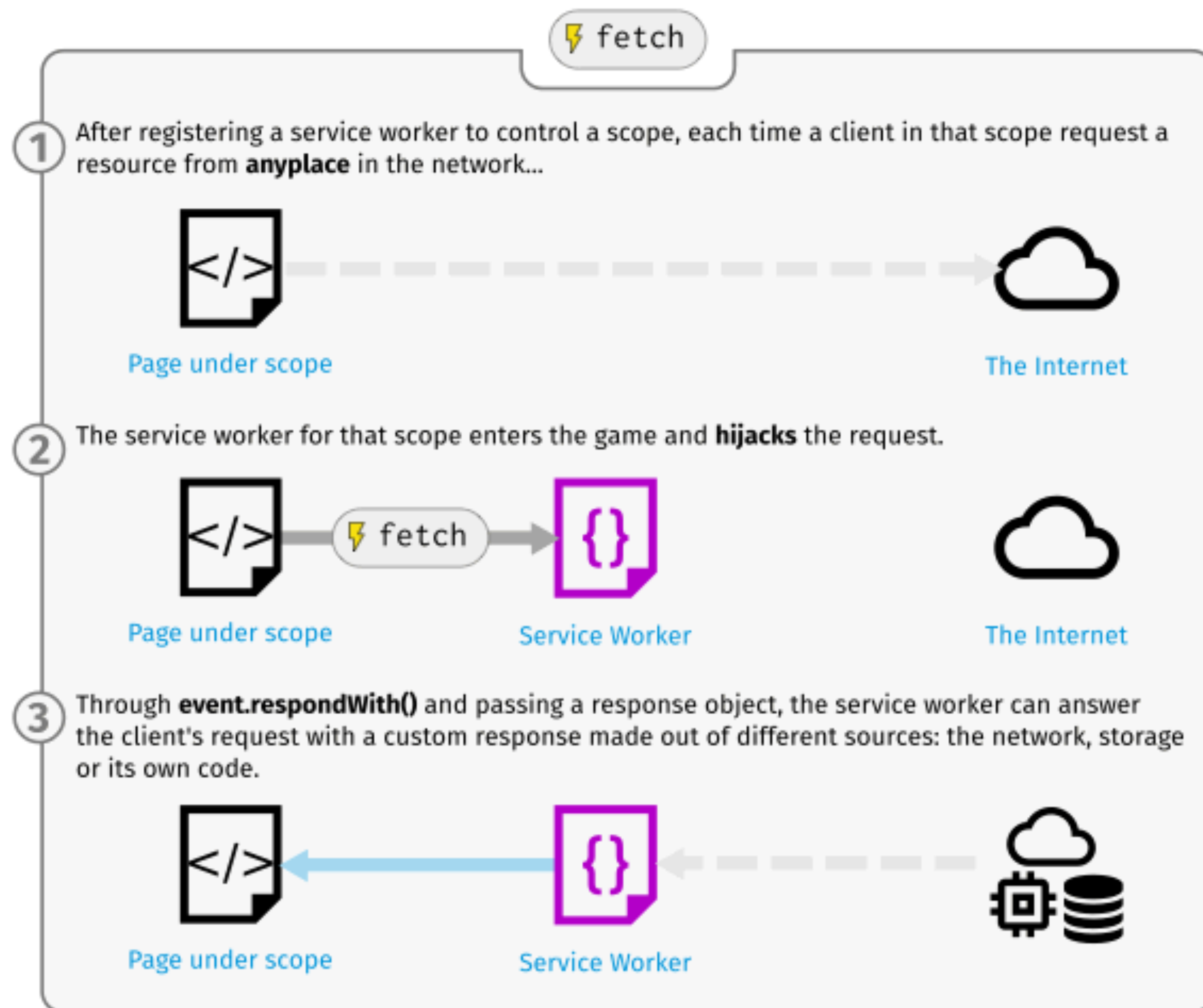
Padrão arquitetural - App Shell



Cached shell loads **instantly** on repeat visits.

Dynamic content then
populates the view

Padrão arquitetural - App Shell



Web Assembly

Samuel Martins

Web assembly (WASM)

- Tipo de código diferente do JavaScript que consegue rodar nos browsers modernos;
- Possibilidade de escrever código web em múltiplas linguagens (C++, C#, Rust...);
- Linguagem de baixo nível, similar ao Assembly, em um formato binário compacto que roda em cima da plataforma nativa;
 - Permite que outras linguagens de baixo nível com baixa utilização de memória (C++, Rust) consigam utilizar recursos da web.

Web assembly (WASM)

- Javascript pode ser problemático ao executar aplicações de altíssima complexidade, exemplo:
 - Jogos 3D;
 - Realidade aumentada e virtual;
 - Edição de imagem/video;
 - Caso de uso: [WebAssembly cut Figma's load time by 3x](#);
 - WebAssembly: [Get Started](#).
 - [Outros casos de uso para WebAssembly](#).

Blazor

Build client web apps with C#

Get Started

Download

To learn more, visit the [Blazor documentation](https://dotnet.microsoft.com/apps/aspnet/web-apps/blazor).

<https://dotnet.microsoft.com/apps/aspnet/web-apps/blazor>

Web assembly - blazor

- Projeto da Microsoft para rodar C# para construção de interfaces ao invés do JavaScript;
- C# com .NET Core compila para WebAssembly e roda diretamente no browser;
- Compartilha algumas APIs do JavaScript para uso de funções no browser;
- Get Started => <https://docs.microsoft.com/pt-br/aspnet/core/tutorials/build-a-blazor-app>

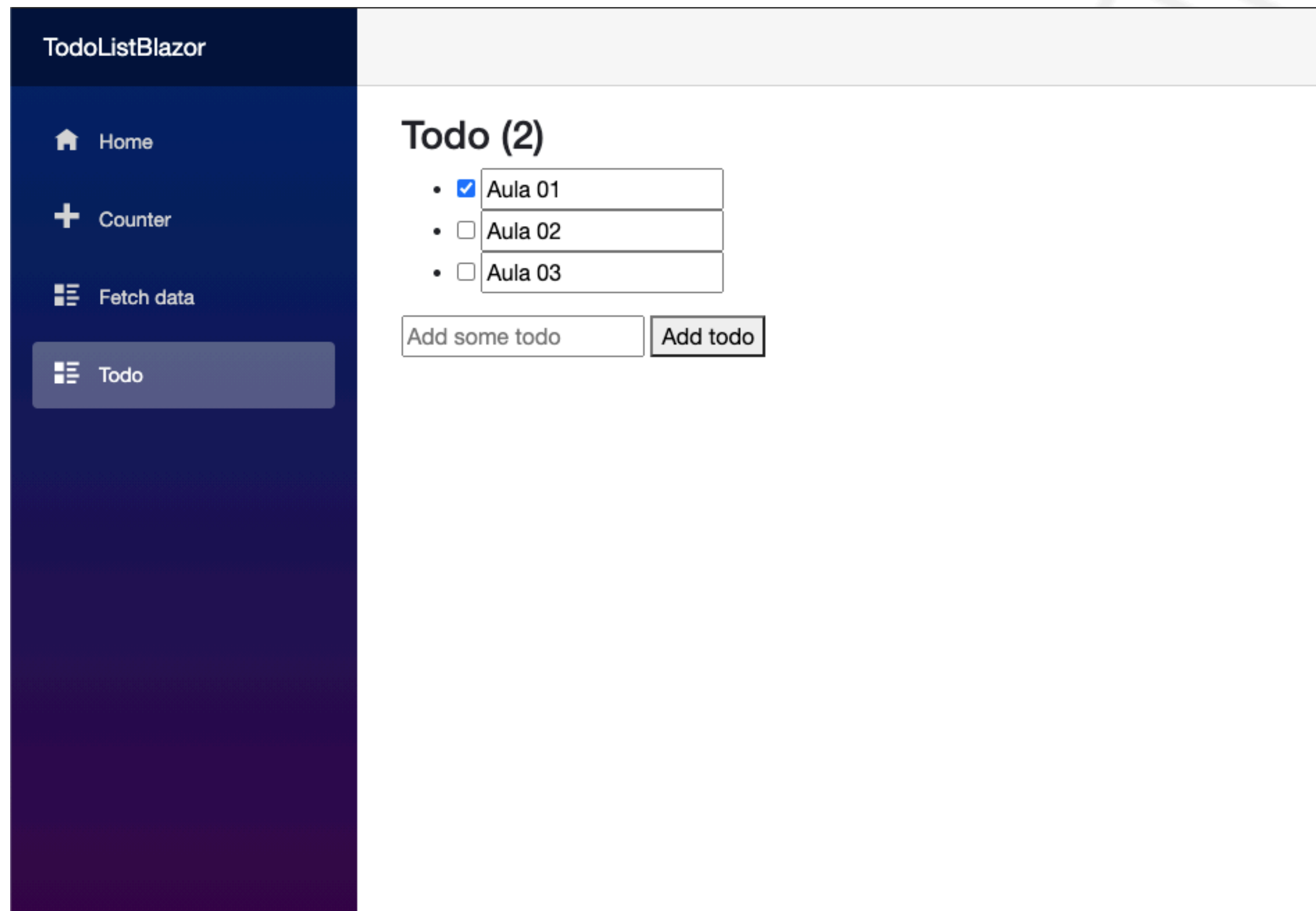
Blazor - component

```
@page "/"
<form action="/" @onsubmit:stopPropagation="true" @onsubmit="AddTodo">
    <input type="text" @bind="newTodo" placeholder="Add some todo" />
    <button @onclick="AddTodo">Add todo</button>
</form>

@code {
    3 references
    private IList<TodoItem> todos = new List<TodoItem>();
    4 references
    private string newTodo;

    2 references
    private void AddTodo()
    {
        if (!string.IsNullOrEmpty(newTodo))
        {
            todos.Add(new TodoItem { Title = newTodo });
            newTodo = string.Empty;
        }
    }
}
```

Blazor - Tela



Fundamentos de segurança em front-end

Samuel Martins

Segurança em front-end

Duas esferas de análises

- Boas práticas de desenvolvimento;
- Ferramentas.

Segurança em front-end - boas práticas

- **OWASP (*Open Web Application Security Project*)**: organização sem fins lucrativos referência em segurança mundialmente reconhecida por disseminar conhecimentos sobre segurança em projetos web.
- **OWASP TOP TEN**: documento de padronizado para desenvolvedores pessoas ligadas a segurança de aplicações web. Representa um vasto consenso sobre os riscos de segurança mais críticos para aplicações web. Documentação completa.

Segurança em front-end

Boas práticas - Frontend

- Injection;
- Cross-Site Scripting;
- Using Components with Known Vulnerabilities;
- Insufficient Logging & Monitoring;

Segurança em front-end

Injection

- Inserção de scripts SQL maliciosos através de campos de formulários mal validados;
- **Prevenção:**
 - Validação dos campos de formulário e dupla validação com o schema da API de destino.

Segurança em front-end

Cross-site scripting

- Execução de scripts maliciosos com o objetivo de roubar informações sensíveis do usuário;
- Prevenção:
 - Sanitização de HTML em campos de formulário ([DOMPurify](#));

Segurança em front-end

Using Components with Known Vulnerabilities

- Utilização de componentes ou bibliotecas na aplicação com vulnerabilidades conhecidas;
- **Prevenção:**
 - Atualização dos pacotes npm;
 - Utilização do comando **npm audit** para checagem de pacotes instalados com vulnerabilidades conhecidas;
 - Utilização de ferramentas para detecção de vulnerabilidades em dependências.

Segurança em front-end

Using Components with Known Vulnerabilities

```
~/projects/my-movies master ● ? 10067 ✓  
$ npm audit  
  
=== npm audit security report ===  
  
found 0 vulnerabilities  
in 1858 scanned packages
```

Segurança em front-end

Insufficient Logging & Monitoring

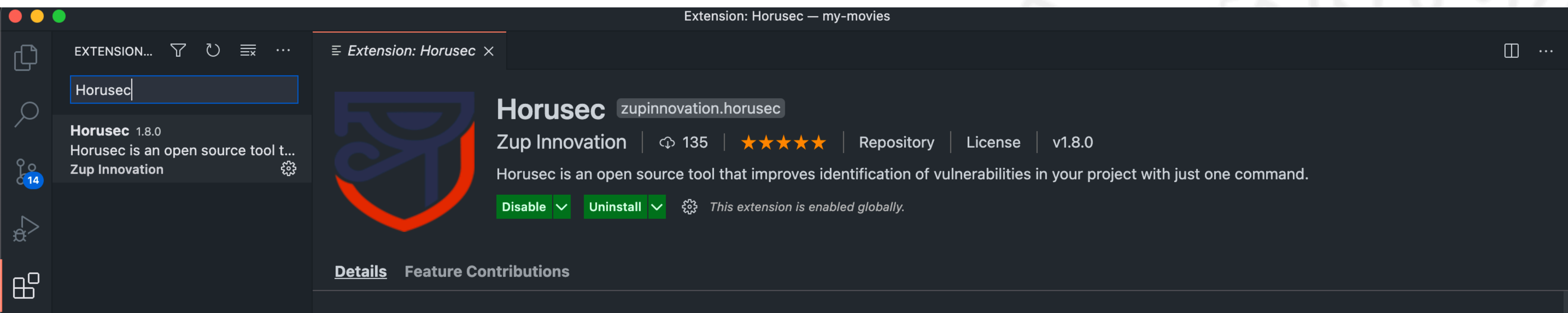
- Impossibilidade de monitorar a aplicação, uma vez que ataques ou comportamentos suspeitos possam estar acontecendo sem o consentimento do time de desenvolvimento;
- **Prevenção:**
 - Utilização de ferramentas de log em tempo real:
 - New Relic;
 - Sentry.

Segurança em front-end

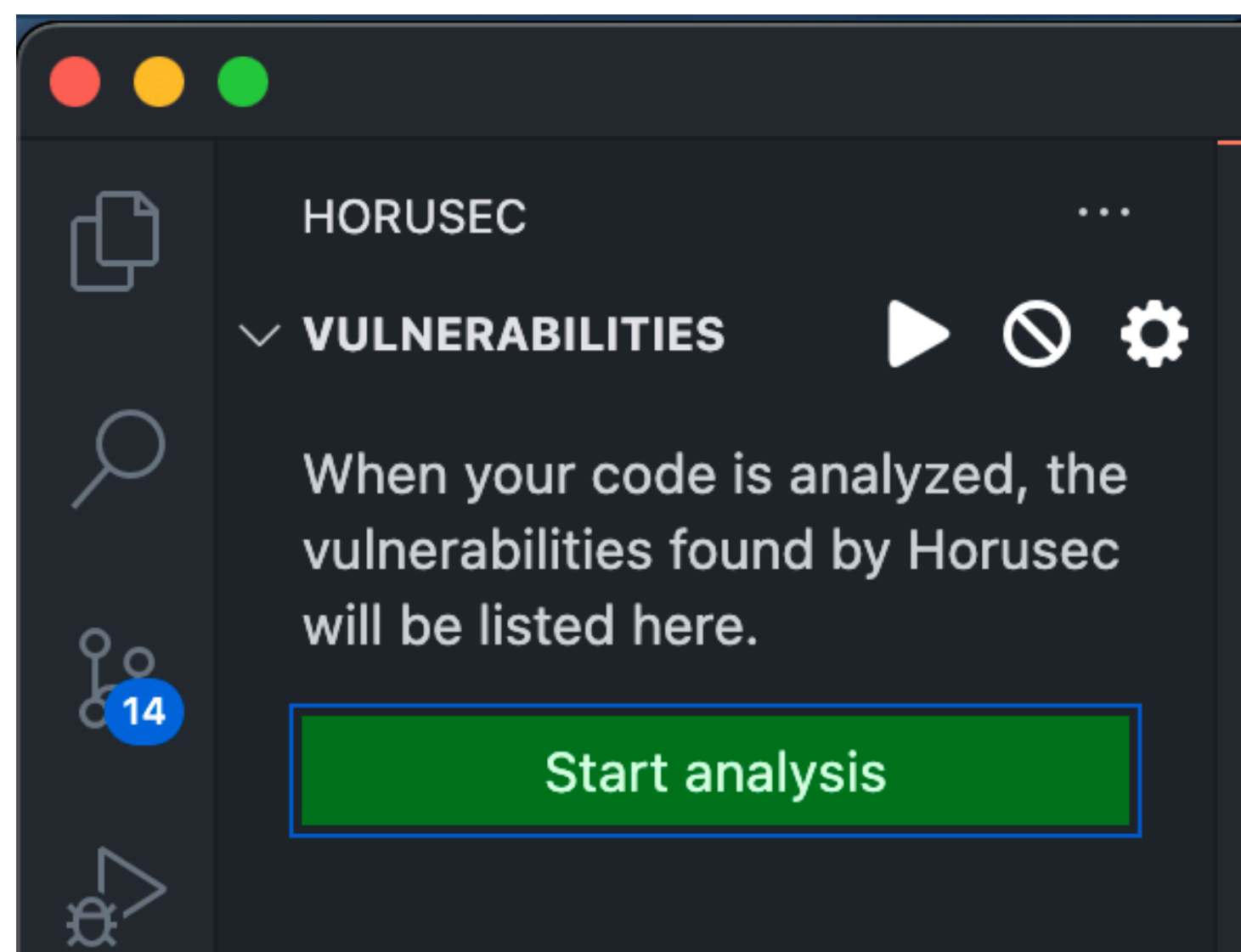
Ferramentas

- Horusec: ferramenta integrada ao VS Code para detecção de vulnerabilidades no código;
- Npm audit: comando para rodar no terminal com objetivo de identificar pacotes instalados com vulnerabilidades conhecidas;
- Verdaccio: hospedagem própria de pacotes npm;

Segurança em front-end - Horusec

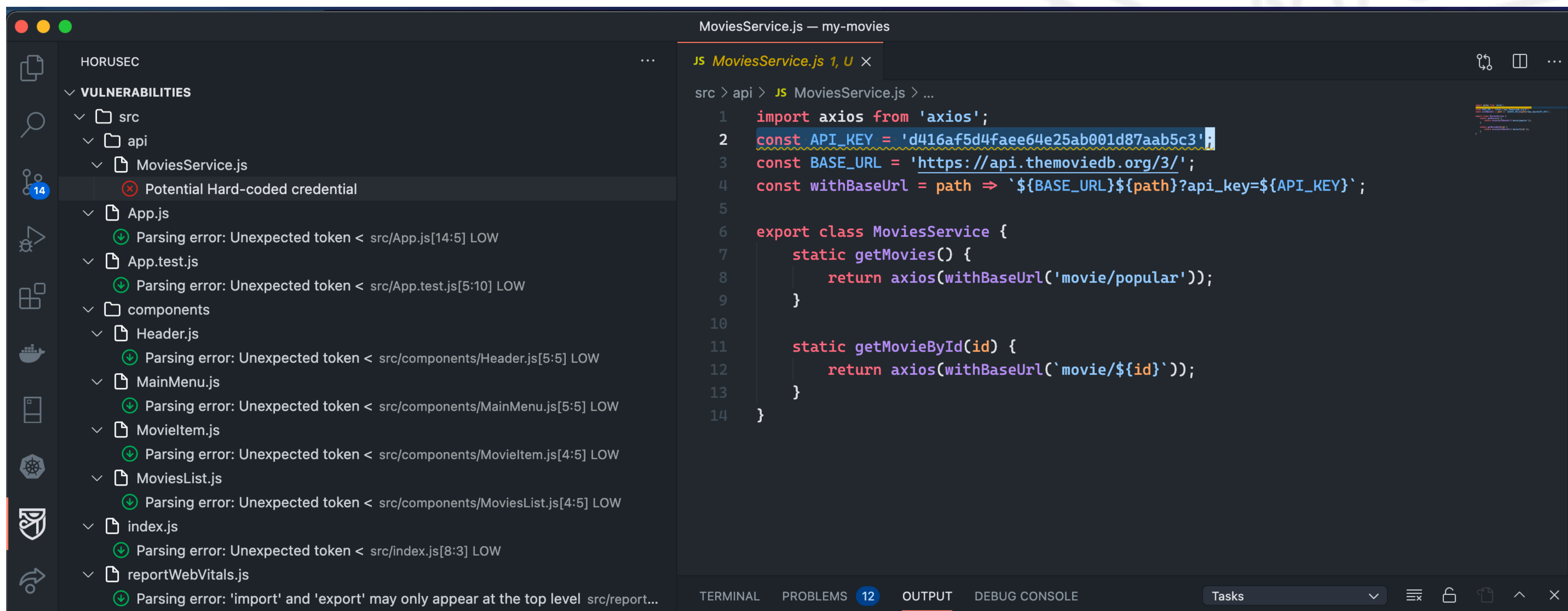


Segurança em front-end - Horusec








```
master* 0 0 Live Share Horusec: Security analysis running -- NORMAL --
```

Segurança em front-end - Horusec



Segurança em front-end - Horusec

-  **INFO**
-  **AUDIT**
-  **LOW**
-  **MEDIUM**
-  **HIGH**



PUC Minas

Virtual