

Nitro

Um servidor para qualquer runtime

Sobre Mim



Sobre Mim

- Bacharel em Ciências da Computação (UFSC)
- 7+ anos de experiência em JS e Web
- Carreira
 - Imaginarium/Grupo Uni.co (Franqueadora)
 - Tekoa/Zoly/Mutant (Agência de Marketing)
 - Tarmac (Consultoria)



Vários Mundos



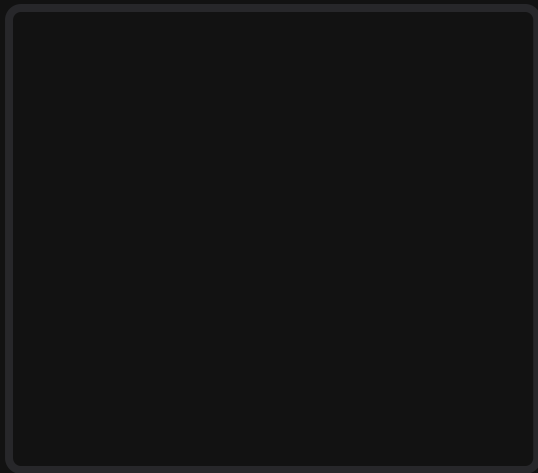
Vários Mundos

Vários Mundos

Ambiente A
Containers / VMs
(Kubernetes)

Vários Mundos

Ambiente A
Containers / VMs
(Kubernetes)



Vários Mundos

Ambiente A
Containers / VMs
(Kubernetes)

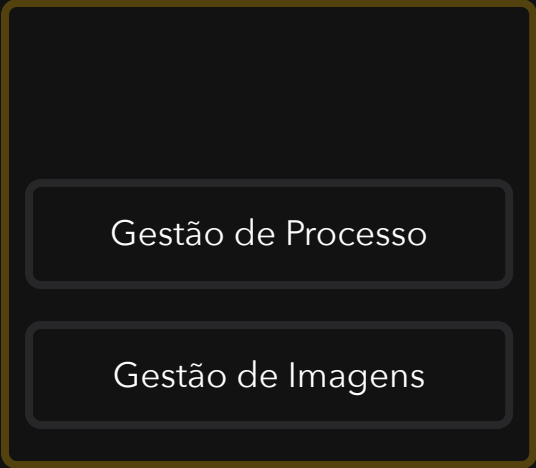


Gestão de Imagens

The diagram consists of a large, dark gray rounded rectangle representing 'Ambiente A'. Inside this rectangle, at the bottom-left corner, is a smaller, lighter gray rounded rectangle representing 'Containers / VMs'. The text 'Gestão de Imagens' is centered within this smaller rectangle. A thin green border outlines the large rectangle, and a thin gray border outlines the smaller rectangle.

Vários Mundos

Ambiente A
Containers / VMs
(Kubernetes)



Gestão de Processo

The diagram shows a large dark rectangle representing 'Ambiente A'. Inside this rectangle, there are two smaller, rounded rectangular boxes stacked vertically. The top box is labeled 'Gestão de Processo' and the bottom box is labeled 'Gestão de Imagens'. The entire structure is outlined with a thin gold border.

Gestão de Imagens

Vários Mundos

Ambiente A
Containers / VMs
(Kubernetes)

Sem Garantias

Gestão de Processo

Gestão de Imagens

Vários Mundos

Ambiente A

Containers / VMs

(Kubernetes)

Ambiente B

Serverless Node-Friendly

(AWS Lambda)

Sem Garantias

Gestão de Processo

Gestão de Imagens

Vários Mundos

Ambiente A

Containers / VMs

(Kubernetes)

Sem Garantias

Gestão de Processo

Gestão de Imagens

Ambiente B

Serverless Node-Friendly

(AWS Lambda)

Vários Mundos

Ambiente A

Containers / VMs

(Kubernetes)

Sem Garantias

Gestão de Processo

Gestão de Imagens

Ambiente B

Serverless Node-Friendly

(AWS Lambda)

Assinatura Proprietária

```
http(route, handler)  
app.http(route, config)  
handler(event, context)
```

Vários Mundos

Ambiente A

Containers / VMs

(Kubernetes)

Sem Garantias

Gestão de Processo

Gestão de Imagens

Ambiente B

Serverless Node-Friendly

(AWS Lambda)

Cold Start

Assinatura Proprietária

```
http(route, handler)  
app.http(route, config)  
handler(event, context)
```

Vários Mundos

Ambiente A

Containers / VMs

(Kubernetes)

Sem Garantias

Gestão de Processo

Gestão de Imagens

Ambiente B

Serverless Node-Friendly

(AWS Lambda)

Vendor Lock-in

Cold Start

Assinatura Proprietária

```
http(route, handler)  
app.http(route, config)  
handler(event, context)
```


Vários Mundos

Ambiente A

Containers / VMs

(Kubernetes)

Sem Garantias

Gestão de Processo

Gestão de Imagens

Ambiente B

Serverless Node-Friendly

(AWS Lambda)

Vendor Lock-in

Cold Start

Assinatura Proprietária

```
http(route, handler)  
app.http(route, config)  
handler(event, context)
```

Ambiente C

Serverless Custom Runtime

(Cloudflare Workers)

Vários Mundos

Ambiente A

Containers / VMs

(Kubernetes)

Sem Garantias

Gestão de Processo

Gestão de Imagens

Ambiente B

Serverless Node-Friendly

(AWS Lambda)

Vendor Lock-in

Cold Start

Assinatura Proprietária

```
http(route, handler)  
app.http(route, config)  
handler(event, context)
```

Ambiente C

Serverless Custom Runtime

(Cloudflare Workers)

Vários Mundos

Ambiente A

Containers / VMs

(Kubernetes)

Sem Garantias

Gestão de Processo

Gestão de Imagens

Ambiente B

Serverless Node-Friendly

(AWS Lambda)

Vendor Lock-in

Cold Start

Assinatura Proprietária

```
http(route, handler)  
app.http(route, config)  
handler(event, context)
```

Ambiente C

Serverless Custom Runtime

(Cloudflare Workers)

V8 Isolates (JS / WASM)

Vários Mundos

Ambiente A

Containers / VMs

(Kubernetes)

Sem Garantias

Gestão de Processo

Gestão de Imagens

Ambiente B

Serverless Node-Friendly

(AWS Lambda)

Vendor Lock-in

Cold Start

Assinatura Proprietária

```
http(route, handler)  
app.http(route, config)  
handler(event, context)
```

Ambiente C

Serverless Custom Runtime

(Cloudflare Workers)

Bibliotecas do Node não
funcionam

V8 Isolates (JS / WASM)

Vários Mundos

Ambiente A

Containers / VMs

(Kubernetes)

Sem Garantias

Gestão de Processo

Gestão de Imagens

Ambiente B

Serverless Node-Friendly

(AWS Lambda)

Vendor Lock-in

Cold Start

Assinatura Proprietária

```
http(route, handler)  
app.http(route, config)  
handler(event, context)
```

Ambiente C

Serverless Custom Runtime

(Cloudflare Workers)

Compatibilidade parcial com
Ecossistema JS

Bibliotecas do Node não
funcionam

V8 Isolates (JS / WASM)

Vários Mundos

Ambiente A

Containers / VMs

(Kubernetes)

Sem Garantias

Gestão de Processo

Gestão de Imagens

Ambiente B

Serverless Node-Friendly

(AWS Lambda)

Vendor Lock-in

Cold Start

Assinatura Proprietária

```
http(route, handler)  
app.http(route, config)  
handler(event, context)
```

Ambiente C

Serverless Custom Runtime

(Cloudflare Workers)

Compatibilidade parcial com
Ecossistema JS

Bibliotecas do Node não
funcionam

V8 Isolates (JS / WASM)

“Quanto mais a plataforma opera por você, mais ela dita como seu código é escrito”

*"Eu quero gastar meu tempo no
domínio do problema, não
nas especificidades de onde o
código roda"*

*"Eu quero gastar meu tempo no
domínio do problema, não
nas especificidades de onde o
código roda"*

Acoplamiento

```
1 // Ambiente A – Express (VM/k8s)
2 app.get("/products/:id", async (req, res) => {
3     const id = req.params.id
4     const product = await getProduct(id)
5     res.json(product)
6 })
```

Acoplamiento

```
1 // Ambiente A – Express (VM/k8s)
2 app.get("/products/:id", async (req, res) => {
3     const id = req.params.id
4     const product = await getProduct(id)
5     res.json(product)
6 })
```

Acoplamiento

```
1 // Ambiente A - Express (VM/k8s)
2 app.get("/products/:id", async (req, res) => {
3     const id = req.params.id
4     const product = await getProduct(id)
5     res.json(product)
6 })
```

Acoplamiento

```
1 // Ambiente B – AWS Lambda
2 export const handler = async (event) => {
3     const id = event.pathParameters.id
4     const product = await getProduct(id)
5     return {
6         statusCode: 200,
7         headers: {
8             "content-type": "application/json"
9         },
10        body: JSON.stringify(product),
11    }
12 }
```

Acoplamiento

```
1 // Ambiente B – AWS Lambda
2 export const handler = async (event) => {
3     const id = event.pathParameters.id
4     const product = await getProduct(id)
5     return {
6         statusCode: 200,
7         headers: {
8             "content-type": "application/json"
9         },
10        body: JSON.stringify(product),
11    }
12 }
```

Acoplamiento

```
1 // Ambiente B – AWS Lambda
2 export const handler = async (event) => {
3     const id = event.pathParameters.id
4     const product = await getProduct(id)
5     return {
6         statusCode: 200,
7         headers: {
8             "content-type": "application/json"
9         },
10        body: JSON.stringify(product),
11    }
12 }
```

Acoplamiento

```
1 // Ambiente C - Cloudflare Workers
2 export default {
3   async fetch(request, env) {
4     const url = new URL(request.url)
5     const id = url.pathname.split("/").pop()
6     const product = await getProduct(id, env.KV)
7     return Response.json(product)
8   },
9 }
```

Acoplamiento

```
1 // Ambiente C - Cloudflare Workers
2 export default {
3   async fetch(request, env) {
4     const url = new URL(request.url)
5     const id = url.pathname.split("/").pop()
6     const product = await getProduct(id, env.KV)
7     return Response.json(product)
8   },
9 }
```


Acoplamiento

```
1 // Ambiente C – Cloudflare Workers
2 export default {
3   async fetch(request, env) {
4     const url = new URL(request.url)
5     const id = url.pathname.split("/").pop()
6     const product = await getProduct(id, env.KV)
7     return Response.json(product)
8   },
9 }
```

Acoplamento

```
// Ambiente A – Express (VM/k8s)
app.get(
  "/products/:id",
  async (req, res) => {
    const id = req.params.id
    const product =
      await getProduct(id)
    res.json(product)
  }
)
```

```
// Ambiente B – AWS Lambda
export const handler = async (event) => {
  const id = event.pathParameters.id
  const product = await getProduct(id)
  return {
    statusCode: 200,
    headers: {
      "content-type":
        "application/json"
    },
    body: JSON.stringify(product),
  }
}
```

```
// Ambiente C – Cloudflare Workers
export default {
  async fetch(request, env) {
    const url =
      new URL(request.url)
    const id =
      url.pathname.split("/").pop()
    const product =
      await getProduct(id, env.KV)
    return Response.json(product)
  },
}
```

“O problema não está na AWS ou Cloudflare, mas no
boilerplate e no **acoplamento**”

Acoplamento

```
// Ambiente A – Express (VM/k8s)
app.get(
  "/products/:id",
  async (req, res) => {
    const id = req.params.id
    const product =
      await getProduct(id)
    res.json(product)
  }
)
```

```
// Ambiente B – AWS Lambda
export const handler = async (event) => {
  const id = event.pathParameters.id
  const product = await getProduct(id)
  return {
    statusCode: 200,
    headers: {
      "content-type":
        "application/json"
    },
    body: JSON.stringify(product),
  }
}
```

```
// Ambiente C – Cloudflare Workers
export default {
  async fetch(request, env) {
    const url =
      new URL(request.url)
    const id =
      url.pathname.split("/").pop()
    const product =
      await getProduct(id, env.KV)
    return Response.json(product)
  },
}
```

“O problema não está na AWS ou Cloudflare, mas no
boilerplate e no **acoplamento**”

Acoplamento

```
// Ambiente A – Express (VM/k8s)
app.get(
  "/products/:id",
  async (req, res) => {
    const id = req.params.id
    const product =
      await getProduct(id)
    res.json(product)
  }
)
```

```
// Ambiente B – AWS Lambda
export const handler = async (event) => {
  const id = event.pathParameters.id
  const product = await getProduct(id)
  return {
    statusCode: 200,
    headers: {
      "content-type":
        "application/json"
    },
    body: JSON.stringify(product),
  }
}
```

```
// Ambiente C – Cloudflare Workers
export default {
  async fetch(request, env) {
    const url =
      new URL(request.url)
    const id =
      url.pathname.split("/").pop()
    const product =
      await getProduct(id, env.KV)
    return Response.json(product)
  },
}
```

“O problema não está na AWS ou Cloudflare, mas no
boilerplate e no **acoplamento**”

A man with short dark hair, wearing a dark racing suit, is seated in the driver's seat of a car. He is looking out the window with a serious expression. The scene is dimly lit, with warm, orange light coming from outside, possibly from streetlights or other vehicles. The interior of the car is dark, and the man's face is partially illuminated by the external light. The word "Nitro" is overlaid in white text on the left side of the image.

Nitro

O que é Nitro?

O que é Nitro?

- ✗ “O servidor do Nuxt”

O que é Nitro?

- ✗ “O servidor do Nuxt”
- ✓ Um **Server Toolkit**



O que é Nitro?

- ✗ “O servidor do Nuxt”
- ✓ Um **Server Toolkit**
 - Orientado a Typescript



O que é Nitro?

- ✗ “O servidor do Nuxt”
- ✓ Um **Server Toolkit**
 - Orientado a Typescript
 - Baseado em Web Standards



O que é Nitro?

- ✗ “O servidor do Nuxt”
- ✓ Um **Server Toolkit**
 - Orientado a Typescript
 - Baseado em Web Standards
 - Zero Config



O que é Nitro?

- ✗ “O servidor do Nuxt”
- ✓ Um **Server Toolkit**
 - Orientado a Typescript
 - Baseado em Web Standards
 - Zero Config
 - Compatível com qualquer runtime



O que é Nitro?

- ✗ “O servidor do Nuxt”
- ✔ Um **Server Toolkit**
 - Orientado a Typescript
 - Baseado em Web Standards
 - Zero Config
 - Compatível com qualquer runtime
- Você escreve contra uma **abstração**



O que é Nitro?

- ✗ “O servidor do Nuxt”
- ✔ Um **Server Toolkit**
 - Orientado a Typescript
 - Baseado em Web Standards
 - Zero Config
 - Compatível com qualquer runtime
- Você escreve contra uma **abstração**
 - O Nitro **transforma** para o alvo



O que é Nitro?

- ✗ “O servidor do Nuxt”
- ✔ Um **Server Toolkit**
 - Orientado a Typescript
 - Baseado em Web Standards
 - Zero Config
 - Compatível com qualquer runtime
- Você escreve contra uma **abstração**
 - O Nitro **transforma** para o alvo



☆ 11K | 📦 47.8M/mo

Reescrevendo o Endpoint

```
1 // Ambiente B - AWS Lambda
2 export const handler = async (event) => {
3   const id = event.pathParameters.id
4   const product = await getProduct(id)
5   return {
6     statusCode: 200,
7     headers: {
8       "content-type": "application/json"
9     },
10    body: JSON.stringify(product),
11  }
12 }
```


Reescrevendo o Endpoint

```
1 // Ambiente B - AWS Lambda
2 export const handler = async (event) => {
3   const id = event.pathParameters.id
4   const product = await getProduct(id)
5   return {
6     statusCode: 200,
7     headers: {
8       "content-type": "application/json"
9     },
10    body: JSON.stringify(product),
11  }
12 }
```

Reescrevendo o Endpoint

```
1 // Ambiente B - AWS Lambda
2 export const handler = async (event) => {
3   const id = event.pathParameters.id
4   const product = await getProduct(id)
5   return {
6     statusCode: 200,
7     headers: {
8       "content-type": "application/json"
9     },
10    body: JSON.stringify(product),
11  }
12 }
```

Reescrevendo o Endpoint

```
1 // server/routes/products/[id].get.ts
2 export default defineHandler(async (event) => {
3   const { id } = event.context.params
4   const product = await getProduct(id)
5   return product
6 })
```

Reescrevendo o Endpoint

```
1 // server/routes/products/[id].get.ts
2 export default defineHandler(async (event) => {
3   const { id } = event.context.params
4   const product = await getProduct(id)
5   return product
6 })
```

Reescrevendo o Endpoint

```
1 // server/routes/products/[id].get.ts
2 export default defineHandler(async (event) => {
3   const { id } = event.context.params
4   const product = await getProduct(id)
5   return product
6 })
```

Reescrevendo o Endpoint

```
1 // Ambiente C - Cloudflare Workers
2 export default {
3   async fetch(request, env) {
4     const url = new URL(request.url)
5     const id = url.pathname.split("/").pop()
6     const product = await getProduct(id, env.KV)
7     return Response.json(product)
8   },
9 }
```

Reescrevendo o Endpoint

```
1 // server/routes/products/[id].get.ts
2 export default defineHandler(async (event) => {
3   const { id } = event.context.params
4   const product = await getProduct(id)
5   return product
6 })
```

Reescrevendo o Endpoint

```
1 // server/routes/products/[id].get.ts
2 export default defineHandler(async (event) => {
3   const { id } = event.context.params
4   const product = await getProduct(id)
5   return product
6 })
```


Reescrevendo o Endpoint

```
1 // server/routes/products/[id].get.ts
2 export default defineHandler(async (event) => {
3   const { id } = event.context.params
4   const product = await getProduct(id)
5   return product
6 })
```

O boilerplate?

Reescrevendo o Endpoint

```
1 // server/routes/products/[id].get.ts
2 export default defineHandler(async (event) => {
3   const { id } = event.context.params
4   const product = await getProduct(id)
5   return product
6 })
```

O boilerplate?

- O arquivo é a rota e o método HTTP

Reescrevendo o Endpoint

```
1 // server/routes/products/[id].get.ts
2 export default defineHandler(async (event) => {
3   const { id } = event.context.params
4   const product = await getProduct(id)
5   return product
6 })
```

O boilerplate?

- O arquivo é a rota e o método HTTP
- Retornar o objeto é responder em JSON

Reescrevendo o Endpoint

```
1 // server/routes/products/[id].get.ts
2 export default defineHandler(async (event) => {
3   const { id } = event.context.params
4   const product = await getProduct(id)
5   return product
6 })
```

O boilerplate?

- O arquivo é a rota e o método HTTP
- Retornar o objeto é responder em JSON
- Sem `app`

Reescrevendo o Endpoint

```
1 // server/routes/products/[id].get.ts
2 export default defineHandler(async (event) => {
3   const { id } = event.context.params
4   const product = await getProduct(id)
5   return product
6 })
```

O boilerplate?

- O arquivo é a rota e o método HTTP
- Retornar o objeto é responder em JSON
- Sem `app`
- Sem `res`

Reescrevendo o Endpoint

```
1 // server/routes/products/[id].get.ts
2 export default defineHandler(async (event) => {
3   const { id } = event.context.params
4   const product = await getProduct(id)
5   return product
6 })
```

O boilerplate?

- O arquivo é a rota e o método HTTP
- Retornar o objeto é responder em JSON
- Sem `app`
- Sem `res`
- Sem *parsing* de URL

Reescrevendo o Endpoint

```
1 // server/routes/products/[id].get.ts
2 export default defineHandler(async (event) => {
3   const { id } = event.context.params
4   const product = await getProduct(id)
5   return product
6 })
```

~~O boilerplate?~~

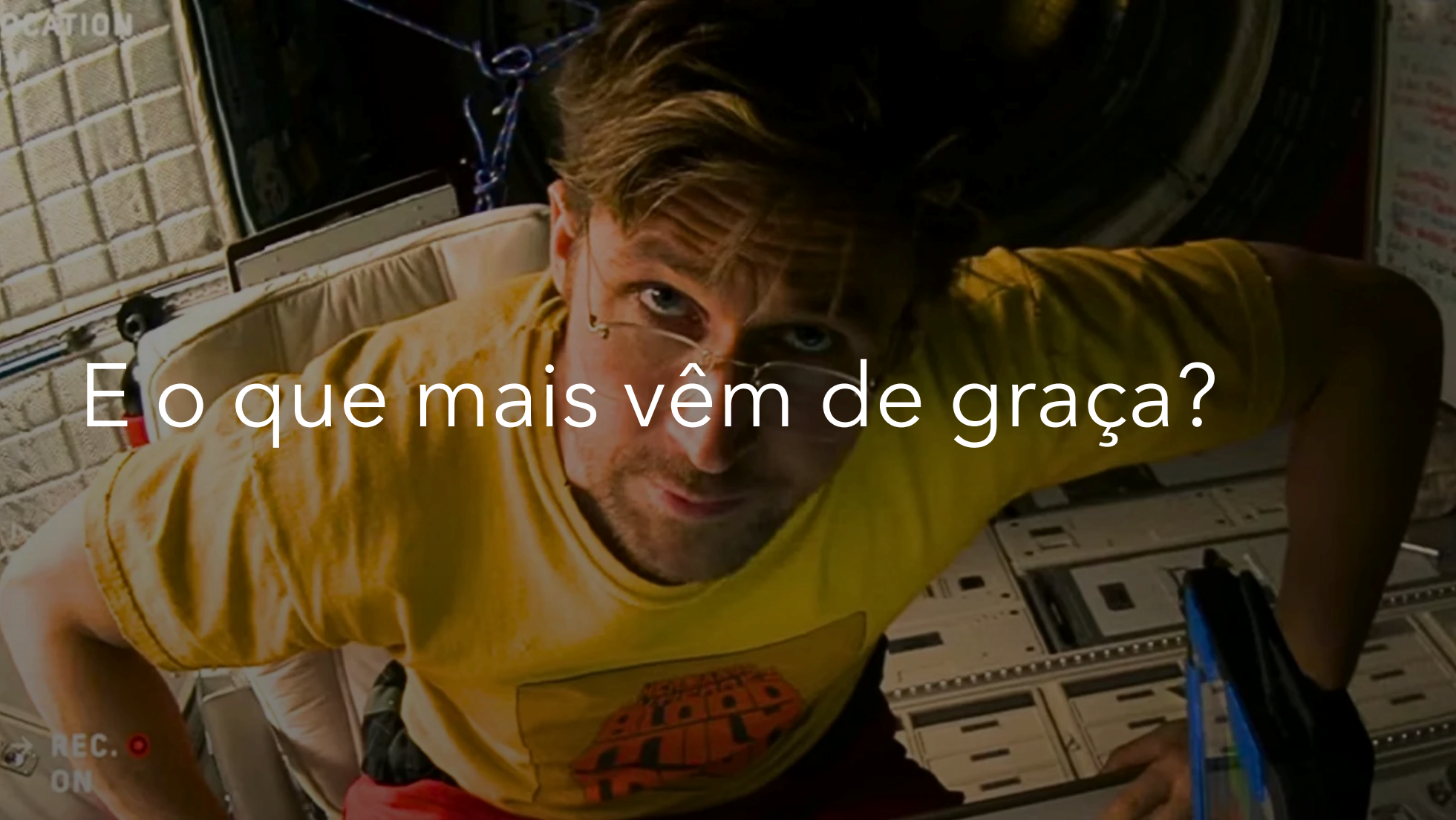
- O arquivo é a rota e o método HTTP
- Retornar o objeto é responder em JSON
- Sem `app`
- Sem `res`
- Sem *parsing* de URL

Reescrevendo o Endpoint

```
1 // server/routes/products/[id].get.ts
2 export default defineHandler(async (event) => {
3   const { id } = event.context.params
4   const product = await getProduct(id)
5   return product
6 })
```

~~O boilerplate?~~ Sumiu!

- O arquivo é a rota e o método HTTP
- Retornar o objeto é responder em JSON
- Sem `app`
- Sem `res`
- Sem *parsing* de URL



E o que mais vêm de graça?

REC. 
ON

Caching

server/routes/products/[id].get.ts

```
1 export default defineHandler(  
2   async (event) => {  
3     const { id } = event.context.params  
4     const product = await getProduct(id)  
5     return product  
6   });
```

Caching

- Cache de Rotas e Funções

server/routes/products/[id].get.ts

```
1 export default defineHandler(  
2   async (event) => {  
3     const { id } = event.context.params  
4     const product = await getProduct(id)  
5     return product  
6   });
```

Caching

- Cache de Rotas e Funções
- Suporte múltiplos drivers

server/routes/products/[id].get.ts

```
1 export default defineHandler(  
2   async (event) => {  
3     const { id } = event.context.params  
4     const product = await getProduct(id)  
5     return product  
6   });
```

Caching

- Cache de Rotas e Funções
- Suporte múltiplos drivers
 - Memória

server/routes/products/[id].get.ts

```
1  export default defineHandler(  
2    async (event) => {  
3      const { id } = event.context.params  
4      const product = await getProduct(id)  
5      return product  
6    });
```

Caching

- Cache de Rotas e Funções
- Suporte múltiplos drivers
 - Memória
 - Redis

server/routes/products/[id].get.ts

```
1 export default defineHandler(  
2   async (event) => {  
3     const { id } = event.context.params  
4     const product = await getProduct(id)  
5     return product  
6   });
```

Caching

- Cache de Rotas e Funções
- Suporte múltiplos drivers
 - Memória
 - Redis
 - S3

server/routes/products/[id].get.ts

```
1  export default defineHandler(  
2    async (event) => {  
3      const { id } = event.context.params  
4      const product = await getProduct(id)  
5      return product  
6    });
```

Caching

- Cache de Rotas e Funções
- Suporte múltiplos drivers
 - Memória
 - Redis
 - S3
 - Vercel

server/routes/products/[id].get.ts

```
1 export default defineHandler(  
2   async (event) => {  
3     const { id } = event.context.params  
4     const product = await getProduct(id)  
5     return product  
6   });
```


Caching

- Cache de Rotas e Funções
- Suporte múltiplos drivers
 - Memória
 - Redis
 - S3
 - Vercel
 - Cloudflare KV

server/routes/products/[id].get.ts

```
1 export default defineHandler(  
2   async (event) => {  
3     const { id } = event.context.params  
4     const product = await getProduct(id)  
5     return product  
6   });
```

Caching

- Cache de Rotas e Funções
- Suporte múltiplos drivers
 - Memória
 - Redis
 - S3
 - Vercel
 - Cloudflare KV

server/routes/products/[id].get.ts

```
1 export default defineHandler(  
2   async (event) => {  
3     const { id } = event.context.params  
4     const product = await getProduct(id)  
5     return product  
6   });
```

Caching

- Cache de Rotas e Funções
- Suporte múltiplos drivers
 - Memória
 - Redis
 - S3
 - Vercel
 - Cloudflare KV

server/routes/products/[id].get.ts

```
1  export default defineCachedHandler(  
2    async (event) => {  
3      const { id } = event.context.params  
4      const product = await getProduct(id)  
5      return product  
6    });
```

Caching

- Cache de Rotas e Funções
- Suporte múltiplos drivers
 - Memória
 - Redis
 - S3
 - Vercel
 - Cloudflare KV

server/routes/products/[id].get.ts

```
1 export default defineCachedHandler(  
2   async (event) => {  
3     const { id } = event.context.params  
4     const product = await getProduct(id)  
5     return product  
6   },  
7   {  
8     maxAge: 60 * 60 // 1 hora  
9   }  
10 );
```

Caching

- Cache de Rotas e Funções
- Suporte múltiplos drivers
 - Memória
 - Redis
 - S3
 - Vercel
 - Cloudflare KV

server/routes/products/[id].get.ts

```
1  export default defineCachedHandler(  
2    async (event) => {  
3      const { id } = event.context.params  
4      const product = await getProduct(id)  
5      return product  
6    },  
7    {  
8      maxAge: 60 * 60 // 1 hora  
9    }  
10 );
```

KV Storage

KV Storage

- Mesmo conceito de cache aplicado à persistência

server/routes/products/[id].get.ts

```
1 export default defineHandler(async (event) => {  
2   const { id } = event.context.params  
3   const product = await getProduct(id)  
4   return product  
5 })
```

KV Storage

- Mesmo conceito de cache aplicado à persistência

server/routes/products/[id].get.ts

```
1 export default defineHandler(async (event) => {  
2   const { id } = event.context.params  
3   const product = await getProduct(id)  
4   return product  
5 })
```


KV Storage

- Mesmo conceito de cache aplicado à persistência

server/routes/products/[id].get.ts

```
1 export default defineHandler(async (event) => {  
2   const { id } = event.context.params  
3   const storage = useStorage("products");  
4   const product = await getProduct(id)  
5   return product  
6 })
```

KV Storage

- Mesmo conceito de cache aplicado à persistência

server/routes/products/[id].get.ts

```
1 export default defineHandler(async (event) => {  
2   const { id } = event.context.params  
3   const storage = useStorage("products");  
4   const product = await getProduct(id)  
5   return product  
6 })
```

KV Storage

- Mesmo conceito de cache aplicado à persistência

server/routes/products/[id].get.ts

```
1 export default defineHandler(async (event) => {  
2   const { id } = event.context.params  
3   const storage = useStorage("products");  
4   const product = await storage.getItem(id)  
5   return product  
6 })
```

KV Storage

- Mesmo conceito de cache aplicado à persistência

server/routes/products/[id].get.ts

```
1 export default defineHandler(async (event) => {  
2   const { id } = event.context.params  
3   const storage = useStorage("products");  
4   const product = await storage.getItem(id)  
5   return product  
6 })
```

KV Storage

- Mesmo conceito de cache aplicado à persistência

server/routes/products/[id].get.ts

```
1 export default defineHandler(async (event) => {  
2   const { id } = event.context.params  
3   const storage = useStorage("products");  
4   const product = await storage.getItem(id)  
5   return product  
6 })
```

KV Storage

- Mesmo conceito de cache aplicado à persistência
- Permite observar mudanças

server/routes/products/[id].get.ts

```
1 export default defineHandler(async (event) => {
2   // SSE nativo do Nitro
3   const stream = createEventStream(event);
4
5   const storage = useStorage("products");
6   await storage.watch(async (kind, key) => {
7     const value = await storage.getItem(key);
8     await stream.push({
9       // ... serialização
10    });
11  });
12
13  return stream.send();
14 });
```

KV Storage

- Mesmo conceito de cache aplicado à persistência
- Permite observar mudanças

server/routes/products/[id].get.ts

```
1 export default defineHandler(async (event) => {
2   // SSE nativo do Nitro
3   const stream = createEventStream(event);
4
5   const storage = useStorage("products");
6   await storage.watch(async (kind, key) => {
7     const value = await storage.getItem(key);
8     await stream.push({
9       // ... serialização
10    });
11  });
12
13  return stream.send();
14 });
```

KV Storage

- Mesmo conceito de cache aplicado à persistência
- Permite observar mudanças

server/routes/products/[id].get.ts

```
1 export default defineHandler(async (event) => {
2   // SSE nativo do Nitro
3   const stream = createEventStream(event);
4
5   const storage = useStorage("products");
6   await storage.watch(async (kind, key) => {
7     const value = await storage.getItem(key);
8     await stream.push({
9       // ... serialização
10    });
11  });
12
13  return stream.send();
14 });
```


KV Storage

- Mesmo conceito de cache aplicado à persistência
- Permite observar mudanças

server/routes/products/[id].get.ts

```
1 export default defineHandler(async (event) => {
2   // SSE nativo do Nitro
3   const stream = createEventStream(event);
4
5   const storage = useStorage("products");
6   await storage.watch(async (kind, key) => {
7     const value = await storage.getItem(key);
8     await stream.push({
9       // ... serialização
10    });
11  });
12
13  return stream.send();
14 });
```

KV Storage

- Mesmo conceito de cache aplicado à persistência
- Permite observar mudanças
 - Sincronização dependente do driver

server/routes/products/[id].get.ts

```
1 export default defineHandler(async (event) => {
2   // SSE nativo do Nitro
3   const stream = createEventStream(event);
4
5   const storage = useStorage("products");
6   await storage.watch(async (kind, key) => {
7     const value = await storage.getItem(key);
8     await stream.push({
9       // ... serialização
10    });
11  });
12
13  return stream.send();
14 });
```

KV Storage

- Mesmo conceito de cache aplicado à persistência
- Permite observar mudanças
 - Sincronização dependente do driver

server/routes/products/[id].get.ts

```
1 export default defineHandler(async (event) => {
2   // SSE nativo do Nitro
3   const stream = createEventStream(event);
4
5   const storage = useStorage("products");
6   await storage.watch(async (kind, key) => {
7     const value = await storage.getItem(key);
8     await stream.push({
9       // ... serialização
10    });
11  });
12
13  return stream.send();
14 });
```

KV Storage

- Mesmo conceito de cache aplicado à persistência
- Permite observar mudanças
 - Sincronização dependente do driver

server/routes/products/[id].get.ts

```
1 export default defineHandler(async (event) => {
2   // SSE nativo do Nitro
3   const stream = createEventStream(event);
4
5   const storage = useStorage("products");
6   await storage.watch(async (kind, key) => {
7     const value = await storage.getItem(key);
8     await stream.push({
9       // ... serialização
10    });
11  });
12
13  return stream.send();
14 });
```

Tasks

Tasks

- Rodar operações pontuais

Tasks

- Rodar operações pontuais
 - Exemplos:

server/tasks/products/sync.ts

```
1 export default defineTask({  
2  
3 });
```

Tasks

- Rodar operações pontuais
 - Exemplos:
 - Migrar um banco de dados

server/tasks/products/sync.ts

```
1 export default defineTask({
2   meta: { name: "db:migrate" },
3   run({ payload, context }) {
4     // .... executa as migrações do banco de dados
5     return { result: "Sucesso" };
6   },
7 });
```


Tasks

- Rodar operações pontuais
 - Exemplos:
 - Migrar um banco de dados
 - Sincronizar produtos de um ERP

server/tasks/products/sync.ts

```
1 export default defineTask({
2   meta: { name: "products:sync" },
3   run({ payload, context }) {
4     // ... revalida o catálogo de produtos
5     return { result: "Sucesso" };
6   },
7 });
```

Tasks

- Rodar operações pontuais
 - Exemplos:
 - Migrar um banco de dados
 - Sincronizar produtos de um ERP
 - Acionamento via:

server/tasks/products/sync.ts

```
1 export default defineTask({
2   meta: { name: "products:sync" },
3   run({ payload, context }) {
4     // ... revalida o catálogo de produtos
5     return { result: "Sucesso" };
6   },
7 });
```

Tasks

- Rodar operações pontuais
 - Exemplos:
 - Migrar um banco de dados
 - Sincronizar produtos de um ERP
 - Acionamento via:
 - HTTP (Dev-Only)

server/tasks/products/sync.ts

```
1 export default defineTask({
2   meta: { name: "products:sync" },
3   run({ payload, context }) {
4     // ... revalida o catálogo de produtos
5     return { result: "Sucesso" };
6   },
7 });
```

http /_nitro/tasks/db:migrate

Tasks

- Rodar operações pontuais
 - Exemplos:
 - Migrar um banco de dados
 - Sincronizar produtos de um ERP
 - Acionamento via:
 - HTTP (Dev-Only)
 - CLI

server/tasks/products/sync.ts

```
1 export default defineTask({
2   meta: { name: "products:sync" },
3   run({ payload, context }) {
4     // ... revalida o catálogo de produtos
5     return { result: "Sucesso" };
6   },
7 });
```

```
npx nitro task run db:migrate
```

Tasks

- Rodar operações pontuais
 - Exemplos:
 - Migrar um banco de dados
 - Sincronizar produtos de um ERP
 - Acionamento via:
 - HTTP (Dev-Only)
 - CLI
 - CRON

server/tasks/products/sync.ts

```
1 export default defineTask({
2   meta: { name: "products:sync" },
3   run({ payload, context }) {
4     // ... revalida o catálogo de produtos
5     return { result: "Sucesso" };
6   },
7 });
```

nitro.config.ts

```
export default defineConfig({
  experimental: { tasks: true },
  scheduledTasks: {
    // Sincroniza produtos a cada 5 minutos
    '*/*5 * * * *': ['products:sync'],
  }
});
```

Tasks

- Rodar operações pontuais
 - Exemplos:
 - Migrar um banco de dados
 - Sincronizar produtos de um ERP
 - Acionamento via:
 - HTTP (Dev-Only)
 - CLI
 - CRON
- Suporta CRON em diversas plataformas

server/tasks/products/sync.ts

```
1 export default defineTask({
2   meta: { name: "products:sync" },
3   run({ payload, context }) {
4     // ... revalida o catálogo de produtos
5     return { result: "Sucesso" };
6   },
7 });
```

nitro.config.ts

```
export default defineConfig({
  experimental: { tasks: true },
  scheduledTasks: {
    // Sincroniza produtos a cada 5 minutos
    '*/*5 * * * *': ['products:sync'],
  }
});
```

Tasks

- Rodar operações pontuais
 - Exemplos:
 - Migrar um banco de dados
 - Sincronizar produtos de um ERP
 - Acionamento via:
 - HTTP (Dev-Only)
 - CLI
 - CRON
- Suporta CRON em diversas plataformas
 - Node | Deno | Bun

server/tasks/products/sync.ts

```
1 export default defineTask({
2   meta: { name: "products:sync" },
3   run({ payload, context }) {
4     // ... revalida o catálogo de produtos
5     return { result: "Sucesso" };
6   },
7 });
```

nitro.config.ts

```
export default defineConfig({
  experimental: { tasks: true },
  scheduledTasks: {
    // Sincroniza produtos a cada 5 minutos
    '*/*5 * * * *': ['products:sync'],
  }
});
```

Tasks

- Rodar operações pontuais
 - Exemplos:
 - Migrar um banco de dados
 - Sincronizar produtos de um ERP
 - Acionamento via:
 - HTTP (Dev-Only)
 - CLI
 - CRON
- Suporta CRON em diversas plataformas
 - Node | Deno | Bun
 - Cloudflare (Cron Triggers)

server/tasks/products/sync.ts

```
1 export default defineTask({
2   meta: { name: "products:sync" },
3   run({ payload, context }) {
4     // ... revalida o catálogo de produtos
5     return { result: "Sucesso" };
6   },
7 });
```

nitro.config.ts

```
export default defineConfig({
  experimental: { tasks: true },
  scheduledTasks: {
    // Sincroniza produtos a cada 5 minutos
    '* / 5 * * * *': ['products:sync'],
  }
});
```


Tasks

- Rodar operações pontuais
 - Exemplos:
 - Migrar um banco de dados
 - Sincronizar produtos de um ERP
 - Acionamento via:
 - HTTP (Dev-Only)
 - CLI
 - CRON
- Suporta CRON em diversas plataformas
 - Node | Deno | Bun
 - Cloudflare (Cron Triggers)
 - Vercel (Vercel Cron Jobs)

server/tasks/products/sync.ts

```
1 export default defineTask({
2   meta: { name: "products:sync" },
3   run({ payload, context }) {
4     // ... revalida o catálogo de produtos
5     return { result: "Sucesso" };
6   },
7 });
```

nitro.config.ts

```
export default defineConfig({
  experimental: { tasks: true },
  scheduledTasks: {
    // Sincroniza produtos a cada 5 minutos
    '* / 5 * * * *': ['products:sync'],
  }
});
```

Tasks

- Rodar operações pontuais
 - Exemplos:
 - Migrar um banco de dados
 - Sincronizar produtos de um ERP
 - Acionamento via:
 - HTTP (Dev-Only)
 - CLI
 - CRON
- Suporta CRON em diversas plataformas
 - Node | Deno | Bun
 - Cloudflare (Cron Triggers)
 - Vercel (Vercel Cron Jobs)
 - Mais a caminho ...

server/tasks/products/sync.ts

```
1 export default defineTask({
2   meta: { name: "products:sync" },
3   run({ payload, context }) {
4     // ... revalida o catálogo de produtos
5     return { result: "Sucesso" };
6   },
7 });
```

nitro.config.ts

```
export default defineConfig({
  experimental: { tasks: true },
  scheduledTasks: {
    // Sincroniza produtos a cada 5 minutos
    '* / 5 * * * *': ['products:sync'],
  }
})
```

Plugins

Plugins

- Permite extender funcionalidades

Plugins

- Permite estender funcionalidades
- Integração com o ciclo de vida através de hooks

Plugins

- Permite estender funcionalidades
- Integração com o ciclo de vida através de hooks
- Exemplos:

server/plugins/examples.ts

```
1 export default definePlugin((nitro) => {  
2  
3 });
```

Plugins

- Permite estender funcionalidades
- Integração com o ciclo de vida através de hooks
- Exemplos:
 - Graceful Shutdown

server/plugins/examples.ts

```
1 export default definePlugin((nitroApp) => {
2   const resource = initResource();
3
4   nitroApp.hooks.hook("close", async () => {
5     // Limpar recursos, fechar conexões, etc.
6     await resource.close();
7   });
8 });
```

Plugins

- Permite estender funcionalidades
- Integração com o ciclo de vida através de hooks
- Exemplos:
 - Graceful Shutdown
 - Logging

```
server/plugins/examples.ts

1  export default definePlugin(nitro => {
2    const logtape = configure({
3      // ...
4    })
5
6    const logger = getLogger("nitro");
7    nitro.hooks.hook("request", async (event) => {
8      await logtape;
9      logger.info(`${event.req.method} ${event.req.url}`)
10    })
11
12    nitro.hooks.hook("close", async () => {
13      await logtape;
14      await dispose();
15    })
16  })
```


Plugins

- Permite estender funcionalidades
- Integração com o ciclo de vida através de hooks
- Exemplos:
 - Graceful Shutdown
 - Logging
 - Alterar Resposta HTTP

server/plugins/examples.ts

```
1 export default definePlugin((nitroApp) => {
2   nitroApp.hooks.hook("response", (res, event) => {
3     const { pathname } = new URL(event.req.url);
4     if (pathname.endsWith(".css") || pathname.endsWith(".js")) {
5       res.headers.append("Vary", "Origin");
6     }
7   });
8 });
```

E muito mais

E muito mais



Open API

E muito mais



Open API



WebSockets

E muito mais



Open API



WebSockets



Banco de Dados

E muito mais



Open API



WebSockets



Banco de Dados



Renderizador

E muito mais



Open API



WebSockets



Banco de Dados



Renderizador



Framework HTTP
customizável

E muito mais



Open API



WebSockets



Banco de Dados



Renderizador



Framework HTTP
customizável



Configurações
Avançadas

*“ Mas como o Nitro consegue
suportar diferentes ambientes? ”*

Presets



Presets

Presets

- Adapters em ***build time***

Presets

- Adapters em ***build time***
- Sem precisar de `node_modules` no `output`

Presets

- Adapters em ***build time***
- Sem precisar de `node_modules` no `output`
- Suporte a diversos provedores

Presets

- Adapters em ***build time***
- Sem precisar de `node_modules` no `output`
- Suporte a diversos provedores
- O entrypoint do servidor é gerado automaticamente

Presets

- Adapters em ***build time***
 - Sem precisar de `node_modules` no `output`
 - Suporte a diversos provedores
 - O entrypoint do servidor é gerado automaticamente
-

-  `.output` *aws_lambda preset*
 -  `nitro.json` *Metadados*
 -  `server`
 -  `index.mjs` *Entrypoint do Lambda*
 -  `_routes`
 -  `api.mjs`
 -  `_libs`
 -  `public`

Presets

- Adapters em ***build time***
- Sem precisar de `node_modules` no `output`
- Suporte a diversos provedores
- O entrypoint do servidor é gerado automaticamente

▪ `.output` *aws_lambda preset*

- `nitro.json` *Metadados*
- `server`
 - `index.mjs` *Entrypoint do Lambda*
 - `_routes`
 - `api.mjs`
 - `_libs`
- `public`

▪ `.vercel` *vercel preset*

- `output`
 - `nitro.json` *Metadados*
 - `config.json` *Configuração da Vercel*
 - `functions`
 - `__server.func`
 - `index.mjs` *Entrypoint da Function*
 - `_routes`
 - `api.mjs`

Exemplo: Build Cron para Vercel

server/tasks/products_sync.ts

```
export default defineTask({
  meta: { name: "products:sync" },
  run({ payload, context }) {
    // ... revalida o catálogo de produtos
  },
});
```

Exemplo: Build Cron para Vercel

server/tasks/products_sync.ts

```
export default defineTask({
  meta: { name: "products:sync" },
  run({ payload, context }) {
    // ... revalida o catálogo de produtos
  },
});
```

nitro.config.ts

```
export default defineConfig({
  experimental: { tasks: true },
  serverDir: './server',
  scheduledTasks: {
    '*/*5 * * * *': 'products:sync'
  }
});
```

Exemplo: Build Cron para Vercel

server/tasks/products_sync.ts

```
export default defineTask({
  meta: { name: "products:sync" },
  run({ payload, context }) {
    // ... revalida o catálogo de produtos
  },
});
```

nitro.config.ts

```
export default defineConfig({
  experimental: { tasks: true },
  serverDir: './server',
  scheduledTasks: {
    '*/*5 * * * *': 'products:sync'
  }
});
```

```
npx nitro build --preset vercel
```


.vercel/output/nitro.json

```
1  {
2    "date": "2026-07-18T01:36:44.637Z",
3    "preset": "vercel",
4    "framework": {
5      "name": "nitro",
6      "version": "3.0.260610-beta"
7    },
8    "serverEntry": "functions/__server.func/index.mjs",
9    "publicDir": "static",
10   "commands": {
11     "preview": "npx srvx --static ../../static ./functions/__server.func/index.mjs",
12     "deploy": "npx vercel deploy --prebuilt"
13   },
14   "config": {
15     "vercel": {
16       "skewProtection": false,
17       "cronHandlerRoute": "/_vercel/cron",
18       "functions": {
19         "runtime": "nodejs24.x"
20       }
21     }
22   }
23 }
```


.vercel/output/nitro.json

```
1  {
2    "date": "2026-07-18T01:36:44.637Z",
3    "preset": "vercel",
4    "framework": {
5      "name": "nitro",
6      "version": "3.0.260610-beta"
7    },
8    "serverEntry": "functions/__server.func/index.mjs",
9    "publicDir": "static",
10   "commands": {
11     "preview": "npx srvx --static ../../static ./functions/__server.func/index.mjs",
12     "deploy": "npx vercel deploy --prebuilt"
13   },
14   "config": {
15     "vercel": {
16       "skewProtection": false,
17       "cronHandlerRoute": "/_vercel/cron",
18       "functions": {
19         "runtime": "nodejs24.x"
20       }
21     }
22   }
23 }
```


O mesmo código, mas para a Cloudflare

```
npx nitro build --preset cloudflare_module
```


.output/nitro.json

```
1  {
2    "date": "2026-07-18T01:46:10.046Z",
3    "preset": "cloudflare-module",
4    "framework": {
5      "name": "nitro",
6      "version": "3.0.260610-beta"
7    },
8    "versions": {
9      "nitro": "3.0.260610-beta"
10   },
11   "serverEntry": "server/index.mjs",
12   "publicDir": "public",
13   "commands": {
14     "preview": "npx wrangler --cwd ./ dev",
15     "deploy": "npx wrangler --cwd ./ deploy"
16   },
17   "config": {
18     "cloudflare": {
19       "deployConfig": true,
20       "nodeCompat": true
21     }
22   }
23 }
```

.output/nitro.json

```
1  {
2    "date": "2026-07-18T01:46:10.046Z",
3    "preset": "cloudflare-module",
4    "framework": {
5      "name": "nitro",
6      "version": "3.0.260610-beta"
7    },
8    "versions": {
9      "nitro": "3.0.260610-beta"
10   },
11   "serverEntry": "server/index.mjs",
12   "publicDir": "public",
13   "commands": {
14     "preview": "npx wrangler --cwd ./ dev",
15     "deploy": "npx wrangler --cwd ./ deploy"
16   },
17   "config": {
18     "cloudflare": {
19       "deployConfig": true,
20       "nodeCompat": true
21     }
22   }
23 }
```


.output/server/wrangler.json

```
1  {
2    "compatibility_date": "2026-07-14",
3    "main": "index.mjs",
4    "assets": {
5      "binding": "ASSETS",
6      "directory": "../public"
7    },
8    "name": "nitro-starter",
9    "compatibility_flags": [
10     "nodejs_compat"
11   ],
12   "no_bundle": true,
13   "rules": [
14     {
15       "type": "ESModule",
16       "globs": [
17         "**/*.mjs",
18         "**/*.js"
19       ]
20     }
21   ],
22   "triggers": {
23     "crons": [ "* / 5 * * * *" ]
24   }
25 }
```


Providers suportados

RUNTIMES



Node.js



Bun



Deno

PROVIDERS



Alwaysdata



AWS Lambda



AWS Amplify



Azure



Cleavr



Cloudflare



Deno Deploy



DigitalOcean



EdgeOne
Pages



Firebase



Flightcontrol



Genezio



GitHub Pages



GitLab Pages



Heroku



IIS



Koyeb



Netlify



Platform.sh



Render.com



StormKit



Vercel



Zeabur



Zephyr Cloud



Zerops

“E como resolver problemas como esse?”

```
import { Buffer } from "node:buffer"  
// Node: Ok  
// Cloudflare Workers (sem flag de compat): ✨
```


Unenv 



- Camada de Polyfill para APIs do Node.js



- Camada de Polyfill para APIs do Node.js
- Decisão em tempo de **build**



- Camada de Polyfill para APIs do Node.js
- Decisão em tempo de **build**
 - Node: Sem Polyfill



- Camada de Polyfill para APIs do Node.js
- Decisão em tempo de **build**
 - Node: Sem Polyfill
 - Workers: Importa Polyfill

Unenv

- Camada de Polyfill para APIs do Node.js
- Decisão em tempo de **build**
 - Node: Sem Polyfill
 - Workers: Importa Polyfill
- Cloudflare utiliza o Unenv

Unenv

- Camada de Polyfill para APIs do Node.js
- Decisão em tempo de **build**
 - Node: Sem Polyfill
 - Workers: Importa Polyfill
- Cloudflare utiliza o Unenv
 - `node_compat`

Unenv

- Camada de Polyfill para APIs do Node.js
- Decisão em tempo de **build**
 - Node: Sem Polyfill
 - Workers: Importa Polyfill
- Cloudflare utiliza o Unenv
 - `node_compat`
- Parte do ecossistema UnJS

*Cada feature que você viu é uma
biblioteca independente*

Ecosistema UnJS



Ecossistema UnJS

Ecosystema UnJS

- Criado por Pooya Parsa (pi0)

Ecossistema UnJS

- Criado por Pooya Parsa (pi0)
- Derivado dos trabalhos no Nuxt 3

Ecossistema UnJS

- Criado por Pooya Parsa (pi0)
- Derivado dos trabalhos no Nuxt 3
- Baseado na filosofia UNIX

Ecossistema UnJS

- Criado por Pooya Parsa (pi0)
- Derivado dos trabalhos no Nuxt 3
- Baseado na filosofia UNIX
 - Alta qualidade e propósito único

Ecossistema UnJS

- Criado por Pooya Parsa (pi0)
- Derivado dos trabalhos no Nuxt 3
- Baseado na filosofia UNIX
 - Alta qualidade e propósito único
 - Repositórios bem mantidos

Ecossistema UnJS

- Criado por Pooya Parsa (pi0)
- Derivado dos trabalhos no Nuxt 3
- Baseado na filosofia UNIX
 - Alta qualidade e propósito único
 - Repositórios bem mantidos
 - Ecossistema colaborativo

Ecossistema UnJS

- Criado por Pooya Parsa (pi0)
- Derivado dos trabalhos no Nuxt 3
- Baseado na filosofia UNIX
 - Alta qualidade e propósito único
 - Repositórios bem mantidos
 - Ecossistema colaborativo
 - Consistência e compatibilidade

Ecossistema UnJS

- Criado por Pooya Parsa (pi0)
- Derivado dos trabalhos no Nuxt 3
- Baseado na filosofia UNIX
 - Alta qualidade e propósito único
 - Repositórios bem mantidos
 - Ecossistema colaborativo
 - Consistência e compatibilidade
 - Desenvolvimento independente

Ecossistema UnJS

- Criado por Pooya Parsa (pi0)
- Derivado dos trabalhos no Nuxt 3
- Baseado na filosofia UNIX
 - Alta qualidade e propósito único
 - Repositórios bem mantidos
 - Ecossistema colaborativo
 - Consistência e compatibilidade
 - Desenvolvimento independente



Handler + Router: ⚡ *H3* + 🌳 *Rou3*

Handler + Router: ⚡ *H3* + 🌳 *Rou3*

- **H3**: Framework HTTP baseado em Web Standards

Handler + Router: ⚡ *H3* + 🌳 *Rou3*

- **H3**: Framework HTTP baseado em Web Standards
 - Orientado a *utilities* (tree-shaking)

Handler + Router: ⚡ *H3* + 🌳 *Rou3*

- **H3**: Framework HTTP baseado em Web Standards
 - Orientado a *utilities* (tree-shaking)
- **Rou3**: Roteador Compilável

Handler + Router: ⚡ *H3* + 🌳 *Rou3*

- **H3**: Framework HTTP baseado em Web Standards
 - Orientado a *utilities* (tree-shaking)
- **Rou3**: Roteador Compilável
 - Menos código inicial

Handler + Router: ⚡ *H3* + 🌳 *Rou3*

- **H3**: Framework HTTP baseado em Web Standards
 - Orientado a *utilities* (tree-shaking)
- **Rou3**: Roteador Compilável
 - Menos código inicial
 - Cold Start mais rápido

Handler + Router: ⚡ **H3** + 🌳 **Rou3**

- **H3**: Framework HTTP baseado em Web Standards
 - Orientado a *utilities* (tree-shaking)
- **Rou3**: Roteador Compilável
 - Menos código inicial
 - Cold Start mais rápido
- Carregamento do handler sob demanda

Handler + Router: ⚡ **H3** + 🌳 **Rou3**

- **H3**: Framework HTTP baseado em Web Standards
 - Orientado a *utilities* (tree-shaking)
- **Rou3**: Roteador Compilável
 - Menos código inicial
 - Cold Start mais rápido
- Carregamento do handler sob demanda

```
1 // 1. Você escreve - H3
2 // server/routes/products/[id].get.ts
3 export default defineHandler((event) => {
4   const { id } = event.context.params
5   return getProduct(id)
6 })
```

Handler + Router: ⚡ **H3** + 🌳 **Rou3**

- **H3**: Framework HTTP baseado em Web Standards
 - Orientado a *utilities* (tree-shaking)
- **Rou3**: Roteador Compilável
 - Menos código inicial
 - Cold Start mais rápido
- Carregamento do handler sob demanda

```
1 // 2. Nitro lê a pasta e registra - Rou3
2 // [id].get.ts → GET /products/:id
3 const router = createRouter()
4
5 addRoute(router, "GET", "/products/:id", {
6   handler: () ⇒ import("../chunks/products-id.mjs")
7 })
```

Handler + Router: ⚡ **H3** + 🌳 **Rou3**

- **H3**: Framework HTTP baseado em Web Standards
 - Orientado a *utilities* (tree-shaking)
- **Rou3**: Roteador Compilável
 - Menos código inicial
 - Cold Start mais rápido
- Carregamento do handler sob demanda

```
1 // 3. Rou3 compila o router em código
2 const code =
3   compileRouterToString(router, "findRoute")
4 // → "const findRoute = (m, p) => { ... }"
5 // matcher estático, emitido no bundle
```

Handler + Router: ⚡ **H3** + 🌳 **Rou3**

- **H3**: Framework HTTP baseado em Web Standards
 - Orientado a *utilities* (tree-shaking)
- **Rou3**: Roteador Compilável
 - Menos código inicial
 - Cold Start mais rápido
- Carregamento do handler sob demanda

```
1 // 4. 0 que roda em produção
2 // .output/server/index.mjs
3 const findRoute = (m, p) => { /* ... */ }
4
5 const route = findRoute("GET", "/products/42")
6 const mod = await route.data.handler()
7 return mod.default(event) // handler H3 original
```

Servidor Universal:  **srvx**

Servidor Universal: *srvx*

- Servidor HTTP Web Standard

Servidor Universal: *srvx*

- Servidor HTTP Web Standard
 - Request → Response (Fetch API)

Servidor Universal: *srvx*

- Servidor HTTP Web Standard
 - Request → Response (Fetch API)
- Um adaptador dedicado por runtime

Servidor Universal: *srvx*

- Servidor HTTP Web Standard
 - `Request` → `Response` (Fetch API)
- Um adaptador dedicado por runtime
 - `srvx/node` · `srvx/bun` · `srvx/deno`

Servidor Universal: *srvx*

- Servidor HTTP Web Standard
 - `Request` → `Response` (Fetch API)
- Um adaptador dedicado por runtime
 - `srvx/node` · `srvx/bun` · `srvx/deno`
 - `srvx/cloudflare` · `srvx/aws-lambda`

Servidor Universal: *srvx*

- Servidor HTTP Web Standard
 - Request → Response (Fetch API)
- Um adaptador dedicado por runtime
 - `srvx/node` · `srvx/bun` · `srvx/deno`
 - `srvx/cloudflare` · `srvx/aws-lambda`
- Nitro serve o app **H3** pelo `srvx`

Servidor Universal: *srvx*

- Servidor HTTP Web Standard
 - Request → Response (Fetch API)
- Um adaptador dedicado por runtime
 - `srvx/node` · `srvx/bun` · `srvx/deno`
 - `srvx/cloudflare` · `srvx/aws-lambda`
- Nitro serve o app **H3** pelo `srvx`
 - O preset escolhe o adaptador do target

Servidor Universal: *srvx*

- Servidor HTTP Web Standard
 - Request → Response (Fetch API)
- Um adaptador dedicado por runtime
 - `srvx/node` · `srvx/bun` · `srvx/deno`
 - `srvx/cloudflare` · `srvx/aws-lambda`
- Nitro serve o app **H3** pelo `srvx`
 - O preset escolhe o adaptador do target
 - Mesmo handler, entrypoint específico

Servidor Universal: *srvx*

- Servidor HTTP Web Standard
 - Request → Response (Fetch API)
- Um adaptador dedicado por runtime
 - `srvx/node` · `srvx/bun` · `srvx/deno`
 - `srvx/cloudflare` · `srvx/aws-lambda`
- Nitro serve o app **H3** pelo `srvx`
 - O preset escolhe o adaptador do target
 - Mesmo handler, entrypoint específico

srvx

```
import { serve } from "srvx"

// Mesma assinatura em Node, Deno e Bun
serve({
  port: 3000,
  fetch(request) {
    return new Response("Olá do srvx!")
  },
})
```

nitro

```
// .output/server/index.mjs – preset node-server
import { serve } from "srvx/node"
import { handler } from "../chunks/nitro.mjs"

// mesmo handler H3; o adaptador vem do target
serve({ fetch: handler })
```

KV?  *unstorage*

KV? *unstorage*

- Armazenamento de Chave-Valor

KV? *unstorage*

- Armazenamento de Chave-Valor
 - Inclui manipulação de dados binários

KV? *unstorage*

- Armazenamento de Chave-Valor
 - Inclui manipulação de dados binários
 - Upload de Imagens

KV? *unstorage*

- Armazenamento de Chave-Valor
 - Inclui manipulação de dados binários
 - Upload de Imagens
- Permite inicialização assíncrona

KV? *unstorage*

- Armazenamento de Chave-Valor
 - Inclui manipulação de dados binários
 - Upload de Imagens
- Permite inicialização assíncrona
- Multiplos drivers

KV? *unstorage*

- Armazenamento de Chave-Valor
 - Inclui manipulação de dados binários
 - Upload de Imagens
- Permite inicialização assíncrona
- Multiplos drivers

unstorage

```
import { createStorage } from "unstorage"
import redisDriver from "unstorage/drivers/redis"

const storage = createStorage({ driver: redisDriver({ url }) })
await storage.getItem("product:42")
```

nitro

```
export default defineHandler(async (event) => {
  const { id } = event.context.params
  const storage = await useStorage();
  const product = await storage.getItem(id)
  return product
})
```


E tem muito mais no UnJS...

E tem muito mais no UnJS...



Scule

Manipulação de
Strings

E tem muito mais no UnJS...



Scule

Manipulação de
Strings



Crossws

WebSocket Cross-
Platform

E tem muito mais no UnJS...



Scule

Manipulação de
Strings



Crossws

WebSocket Cross-
Platform



Hookable

Awaitable Hooks

E tem muito mais no UnJS...



Sculc

Manipulação de
Strings



Crossws

WebSocket Cross-
Platform



Hookable

Awaitable Hooks



Ofetch

Fetch Melhorado

E tem muito mais no UnJS...



Sculc

Manipulação de
Strings



Crossws

WebSocket Cross-
Platform



Hookable

Awaitable Hooks



Ofetch

Fetch Melhorado



Consola

Console Elegante

E tem muito mais no UnJS...



Sculc

Manipulação de
Strings



Crossws

WebSocket Cross-
Platform



Hookable

Awaitable Hooks



Ofetch

Fetch Melhorado



Consola

Console Elegante



+60 outras
bibliotecas

Onde o Nitro ainda apanha



Onde o Nitro ainda apanha

- Falta de suporte à plugins *async*



Onde o Nitro ainda apanha

- Falta de suporte à plugins *async*
 - Alguns providers não tem suporte à *top-level await*



Onde o Nitro ainda apanha

- **Falta de suporte à plugins *async***
 - Alguns providers não tem suporte à *top-level await*
 - Integrações que dependem de inicialização assíncrona podem falhar



Onde o Nitro ainda apanha

- **Falta de suporte à plugins *async***
 - Alguns providers não tem suporte à *top-level await*
 - Integrações que dependem de inicialização assíncrona podem falhar
- **Inicialização de *Secrets***



Onde o Nitro ainda apanha

- **Falta de suporte à plugins *async***
 - Alguns providers não tem suporte à *top-level await*
 - Integrações que dependem de inicialização assíncrona podem falhar
- **Inicialização de *Secrets***
 - Algumas funcionalidades dependem de secrets em *build time*



Onde o Nitro ainda apanha

- **Falta de suporte à plugins *async***
 - Alguns providers não tem suporte à *top-level await*
 - Integrações que dependem de inicialização assíncrona podem falhar
- **Inicialização de *Secrets***
 - Algumas funcionalidades dependem de secrets em *build time*
- **V3 em beta**



Onde o Nitro ainda apanha

- **Falta de suporte à plugins *async***
 - Alguns providers não tem suporte à *top-level await*
 - Integrações que dependem de inicialização assíncrona podem falhar
- **Inicialização de *Secrets***
 - Algumas funcionalidades dependem de secrets em *build time*
- **V3 em beta**
 - API relativamente estável



Onde o Nitro ainda apanha

- **Falta de suporte à plugins *async***
 - Alguns providers não tem suporte à *top-level await*
 - Integrações que dependem de inicialização assíncrona podem falhar
- **Inicialização de *Secrets***
 - Algumas funcionalidades dependem de secrets em *build time*
- **V3 em beta**
 - API relativamente estável
 - Lançamento oficial sincronizado com o Nuxt 5





E quem usa o Nitro?

Quem usa o Nitro?



Nuxt

☆ 60K | 📄 6.8M/mo



Solid Start

☆ 5.9K | 📄 267K/mo

Como começar?

Como começar?

```
npx create-nitro-app  
npm run nitro dev
```

*“Da próxima vez que o cliente disser 'aqui
a gente roda em X' não se assuste.
Basta mudar o **preset**”*

A man with short brown hair, wearing a white lab coat and clear safety glasses with a red strap, is working in a laboratory. He is holding a glass tube with his right hand, which is wearing a blue nitrile glove. The background is a blurred laboratory setting with various glassware and equipment.

Obrigado!

Dúvidas?

 enzo-albornoz